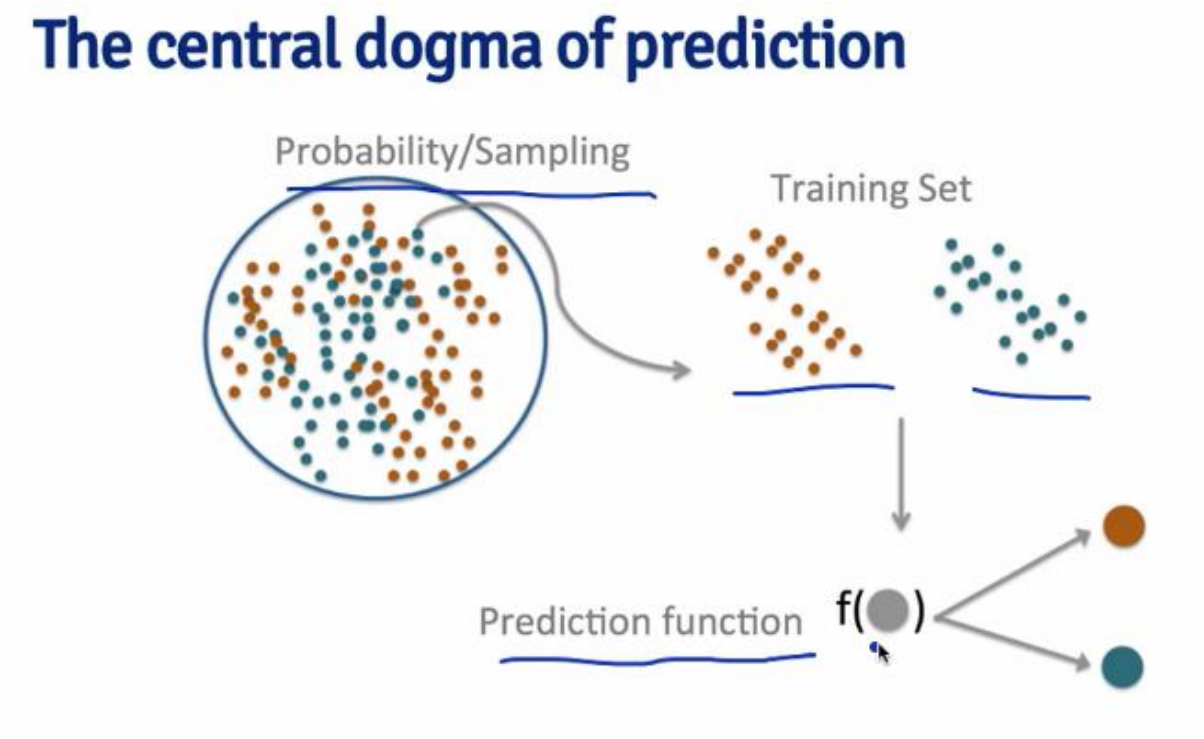
PREDICTION:



Suppose we have a dataset of red and blue balls. Now we take a sample of these red and blue balls each and train the sample and create a prediction function. Now, what this prediction function does is that it classifies a random ball whose color is not known into whether it is red or blue .

Now, we create a very basic prediction model , we want to predict spam or non – spam emails on the basis of frequency of the word 'your' in an email:

If frequency is > 0.5 , email is a spam and not otherwise.

```
prediction<-ifelse(spam$your>0.5,"spam","nonspam")
```

Now , we compare our prediction model to the actual prediction made in the data set:

```
xtabs(~prediction+spam$type)

          spam$type
prediction nonspam spam
   nonspam    2112  468
   spam        676 1345
```

We see that out of our total predicted non spam emails i.e. 2580 , 2112 are in actual non-spam and 468 are spam. Similarly for spam emails , out of the total 2021 emails , 676 are in actual non-spam and 1345 are spam.

Hence our prediction is correct 2112/2580 i.e. 81.8% correct for non spam email and 1345/2021 i.e. 66.5% correct for spam emails.

INSAMPLE AND OUTSAMPLE ERRORS:

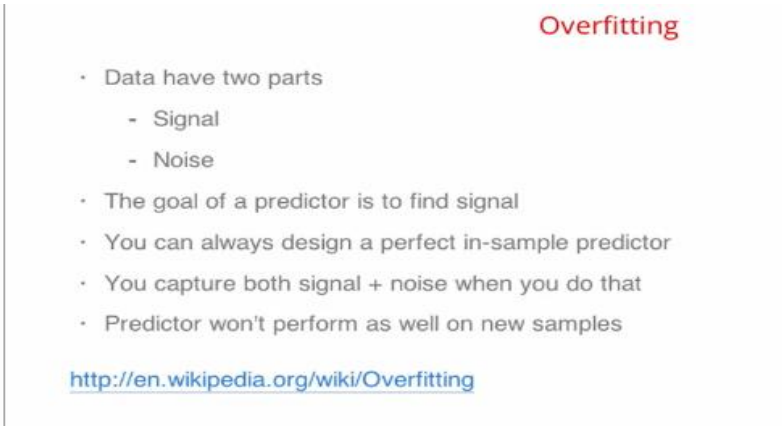
1.In sample error is in the same data which we have used to train our predictor , also called resubstitution error.

In Sample error is always going to be a little bit optimistic , from what the error is that you would get from a new sample. And the reason why is, in your specific sample, sometimes your prediction algorithm will tune itself a little bit to the noise that you collected in that particular data set. And so when you get a new data set, there'll be different noise, and so the accuracy will go down a little bit.

Hence , In sample error< out of sample error always.

2.Out of sample error : The error we get when we perform our prediction on some other sample data set . We usually want to test our prediction using another sample data to see how it would actually work on the real dataset.

OVERFITTING: When we train our model 'too perfectly' i.e. it performs very accurately on the sample chosen to train the predictor, there can be some errors when it is predicting for some our sample or for the population itself , since it is built in a way to tackle the noise there in the sample used to train it.



Rules of thumb for prediction study design

- If you have a large sample size
  - 60% training
  - 20% test
  - 20% validation
- If you have a medium sample size
  - 60% training
  - 40% testing
- If you have a small sample size
  - Do cross validation
  - Report caveat of small sample size

Some principles to remember

- Set the test/validation set aside and *don't look at it*
- In general *randomly* sample training and test
- Your data sets must reflect structure of the problem
  - If predictions evolve with time split train/test in time chunks (calledbacktesting in finance)
- All subsets should reflect as much diversity as possible
  - Random assignment does this
  - You can also try to balance by features - but this is tricky

TYPES OF ERROR:

Basic Terms:

In general, **Positive** = identified and **negative** = rejected. Therefore:

True positive = correctly identified

False positive = incorrectly identified

True negative = correctly rejected

False negative = incorrectly rejected

Explaining the above with an example:

Medical testing example:

True positive = Sick people correctly diagnosed as sick

False positive= Healthy people incorrectly identified as sick

True negative = Healthy people correctly identified as healthy

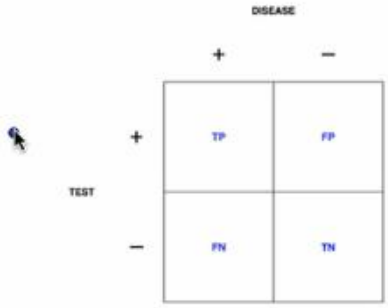
False negative = Sick people incorrectly identified as healthy.

You were healthy but we predicted you to be sick.

You were sick but we predicted you to be healthy.

Key Quantities:

Key quantities



Sensitivity	→	Pr (positive test | disease)
Specificity	→	Pr (negative test | no disease)
Positive Predictive Value	→	Pr (disease | positive test)
Negative Predictive Value	→	Pr (no disease | negative test)
Accuracy	→	Pr (correct outcome)

1.Sensitivity means diseased if a person is diseased , what is the probability that we get it right from our model?

2.Specificity means if a person is healthy , what is the probability we get it right from our model.

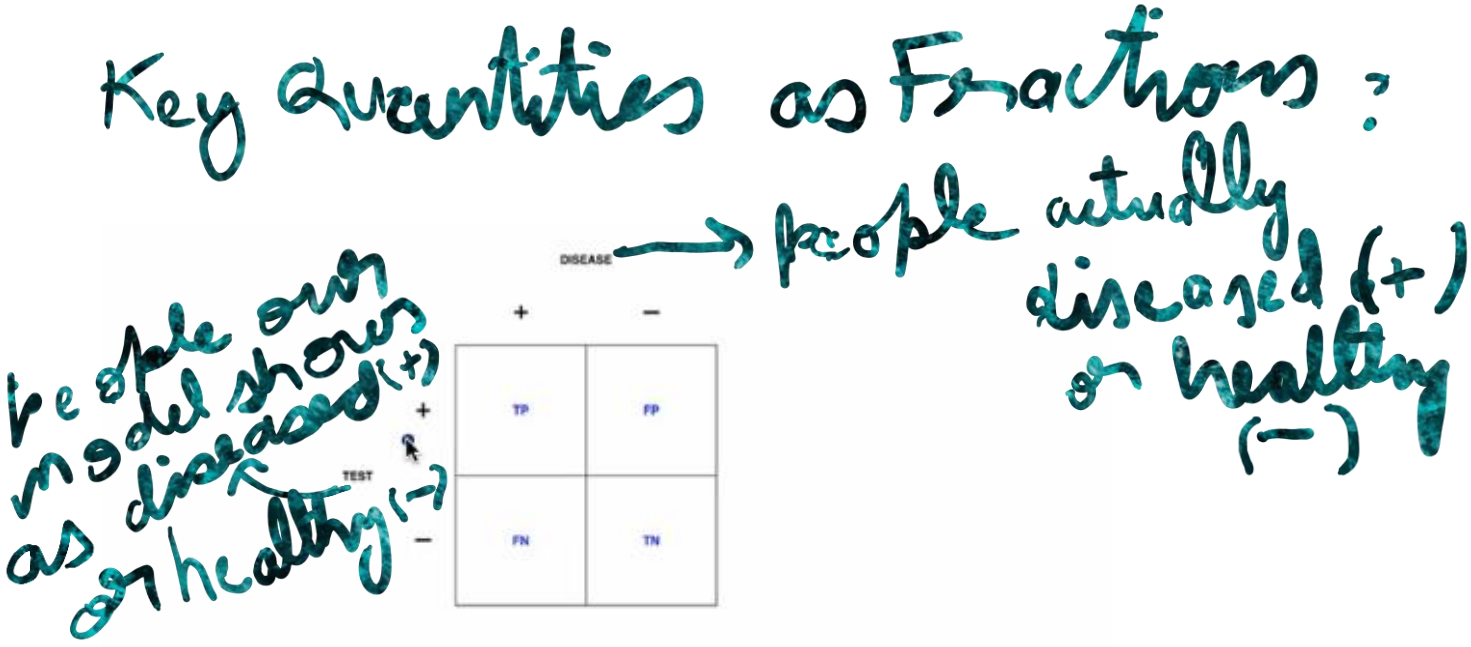
3. Positive predictive value means if our test shows that you are diseased, what is the probability that you are actually diseased or if we look at all the people we call diseased , what fraction of them are actually diseased.

4.Negative predictive value means that if our test/model shows that you are healthy, what is the probability that you are actually healthy.

5. Accuracy is the probability of the correct outcomes, in the table below , it is the true positives (TP) and the true negatives(TN) added up,

NOTE: In the Above table:
1.Say our prediction model predicted TP + FP people as diseased, out of that , from the lab results , we saw that TP were actually diseased whereas FP where healthy so TP was the correct prediction whereas FP was the wrong prediction.

2.Similarly Say our prediction model predicted FN + TN people as Healthy, out of that , from the lab results , we saw that TN were actually healthy whereas FN where sick so TN was the correct prediction whereas FN was the wrong prediction.



Sensitivity	→ TP / (TP+FN)
Specificity	→ TN / (FP+TN)
Positive Predictive Value	→ TP / (TP+FP)
Negative Predictive Value	→ TN / (FN+TN)
Accuracy	→ (TP+TN) / (TP+FP+FN+TN)

Question:

Screening tests

Assume that some disease has a 0.1% prevalence in the population. Assume we have a test kit for that disease that works with 99% sensitivity and 99% specificity. What is the probability of a person having the disease given the test result is positive, if we randomly select a subject from

- the general population?
- a high risk sub-population with 10% disease prevalence?

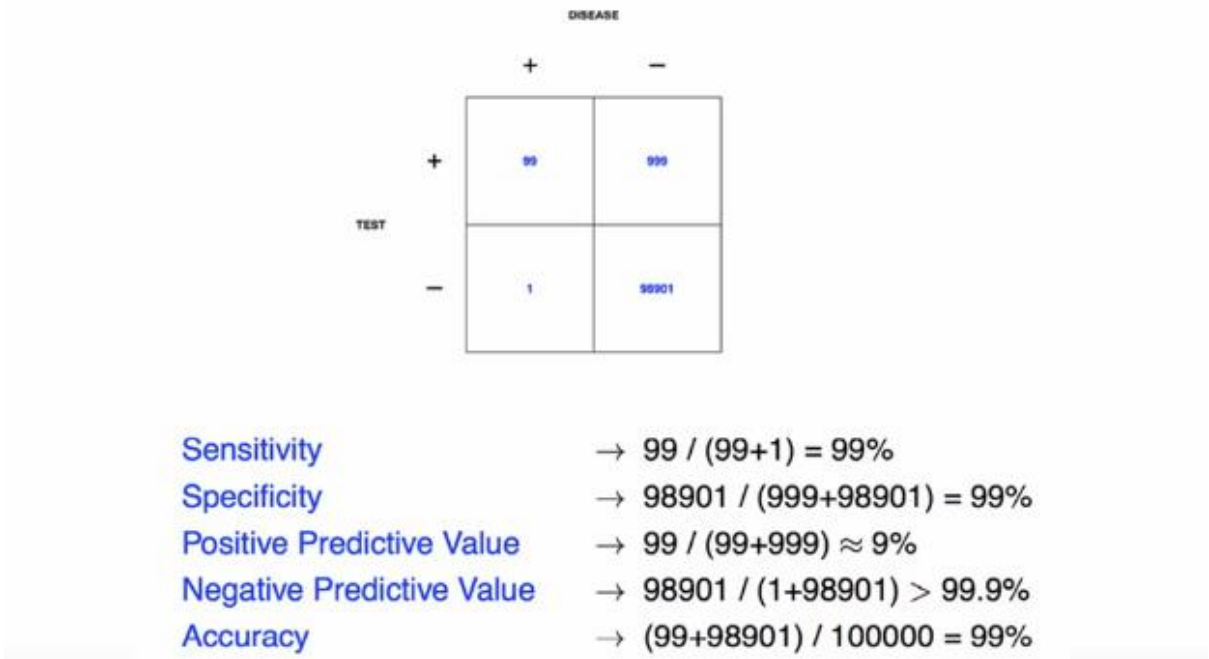
Ans: Let us assume that our populat
-ion is 100000

(1) Now, as given in the question, only 0.1%
of these 100000 have the disease, i.e.
only 100 have disease.

(2) Thus 99900 are healthy.

(3) sensitivity = 99.1., specificity = 99.1.
a) hence out of the 100 diseased people
we predict 99 i.e $\frac{99}{100} \times 100 = 99$ correct
-ly.

e.g. And, out of the 99900 healthy people, we predict 99% i.e. $\frac{99}{100} \times 99908$ to be healthy i.e 98901



Sensitivity	→ 99 / (99+1) = 99%
Specificity	→ 98901 / (999+98901) = 99%
Positive Predictive Value	→ 99 / (99+999) ≈ 9%
Negative Predictive Value	→ 98901 / (1+98901) > 99.9%
Accuracy	→ (99+98901) / 100000 = 99%

In above picture we see that positive predicted value is very less, it is only 9.1%, hence there is only a 9.1% chance that a person actually has a disease, given that we predict him/her to be sick.

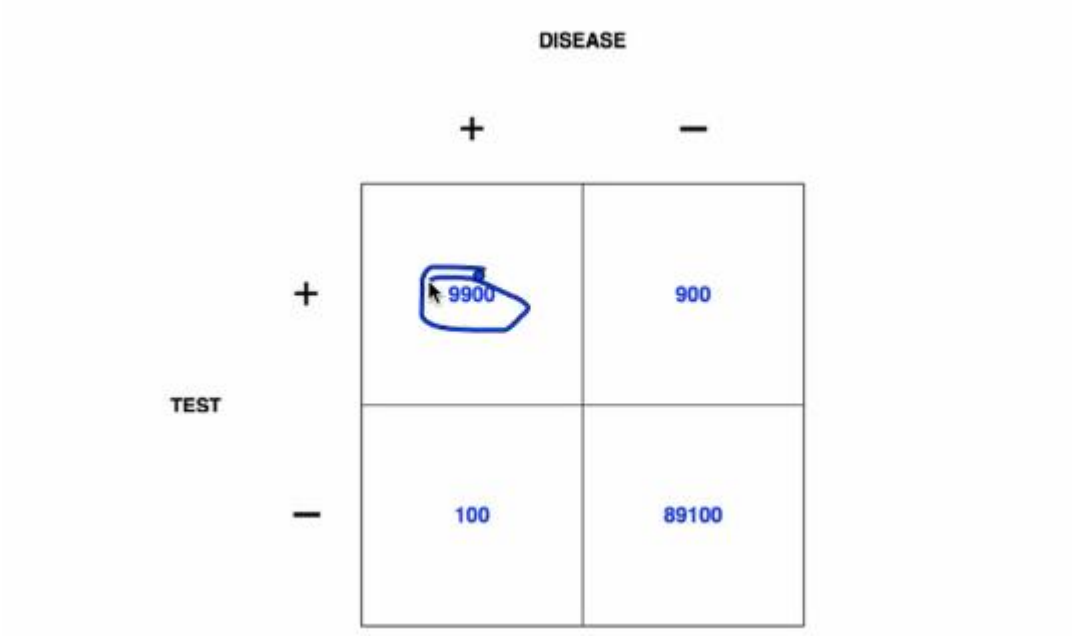
This can be attributed to the fact that only 0.1% of the people in our population have a disease, so the probability of a person having a disease decreases considerably.

X — X — X — X —

#9) Instead we assume that 10% of our population has a particular disease:

No. of diseased people = 10% of 100000
= 10,000

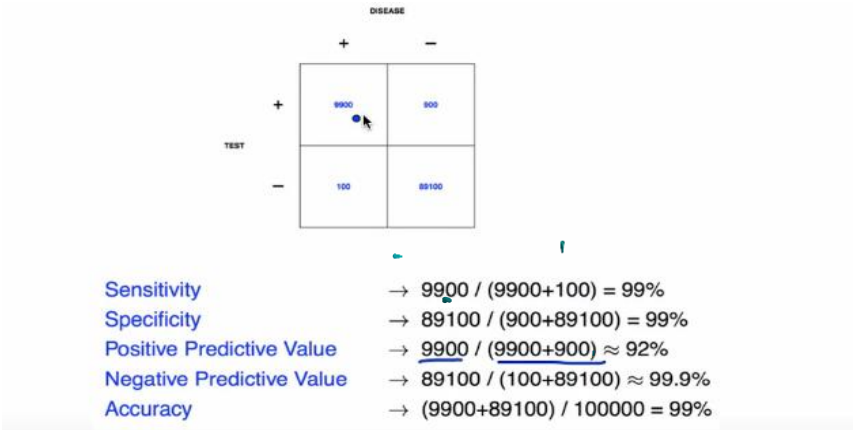
No. of healthy people = 100000 - 10000
= 90,000



positive predicted value $= \frac{9900}{9900+900}$

$= \frac{9900}{10800} = 91.6\%$

hence, 91.6% of our predicted diseased patients are actually diseased.



Sensitivity	→ 9900 / (9900+100) = 99%
Specificity	→ 89100 / (900+89100) = 99%
Positive Predictive Value	→ 9900 / (9900+900) ≈ 92%
Negative Predictive Value	→ 89100 / (100+89100) ≈ 99.9%
Accuracy	→ (9900+89100) / 100000 = 99%

ERROR IN PREDICTED VS TRUE VALUE:

For continuous data

1. Mean squared error (MSE):

$$\frac{1}{n}\sum_{i=1}^{n}(Prediction_i - Truth_i)^2$$

2. Root mean squared error (RMSE):

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(Prediction_i - Truth_i)^2}$$

#Formula (1) given above is just like calculating average residuals in Regression.

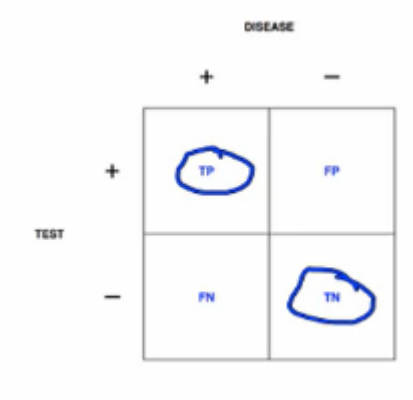
#In formula 2, we take the sq. root to make the units back to what they were.

Receiver Operating Characteristic curve OR The

ROC CURVE

In ROC curve, we plot P(TP) vs P(FP). P(TP) tells us out of the actual total diseased people, how many we correctly predicted to be diseased.

Refer to this table:



P(TP) = No. of people correctly predicted to be diseased / Total no. of people actually diseased

$= \frac{TP}{TP+FN}$

P(FP) tells us out of all people who are actually healthy, how many did we predict to be sick.

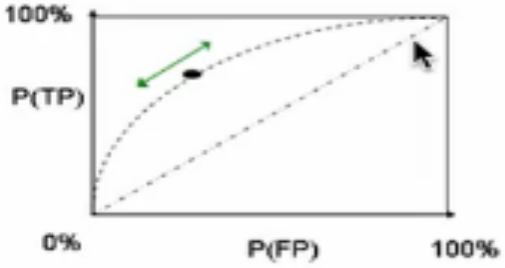
P(FP) = No. of people we wrongly predicted to be sick / Total healthy people

$= \frac{FP}{FP+TN}$

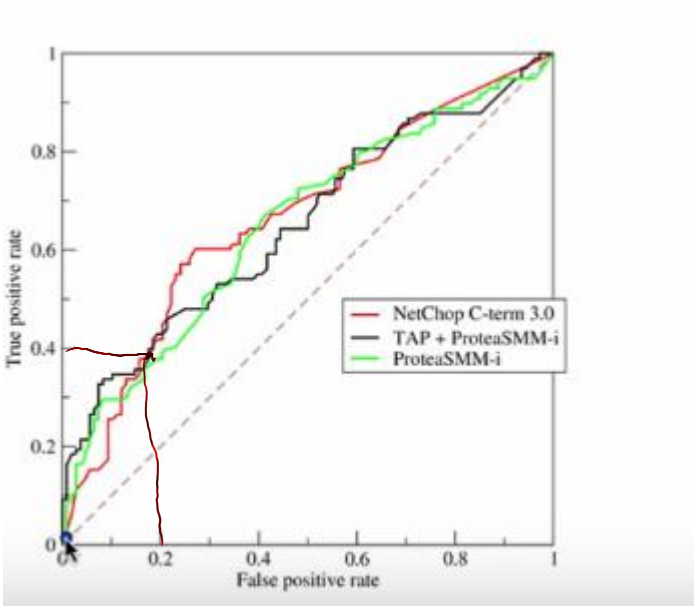
Note: P(TP) = sensitivity, where P(FP) = 1 − specificity

$= 1 - \frac{TN}{FP+TN}$

ROC CURVE:



An actual eg of ROC curve:



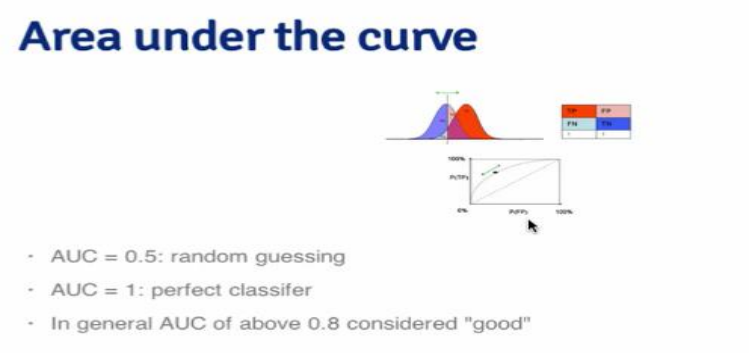
Here in the above curve , a false positive rate of 0.2 means a specificity of 0.8 whereas the point 0.4 => a sensitivity of 0.4 . Hence, this point has a high specificity but a low sensitivity.

COMPARING TWO OR MORE ROC's:

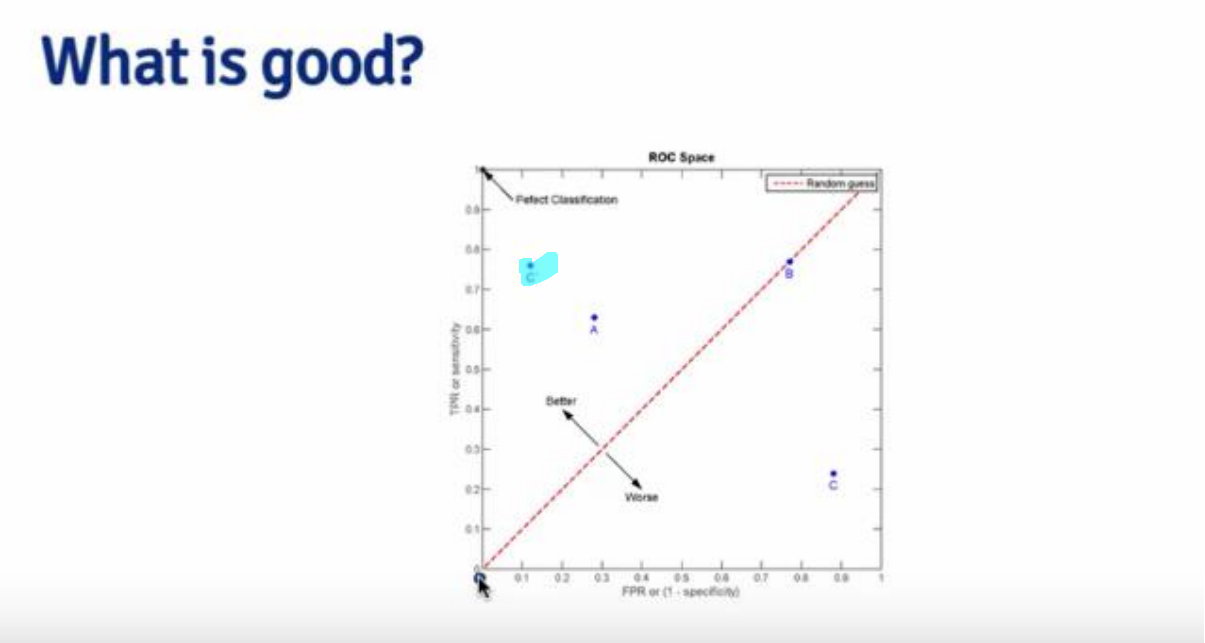
What we want in our ROC is that it should have low False positive rate and high true positive rate both of which indicate high sensitivity and high specificity. Thus, we want our points to be on the upper left side of the curve. The point on the upper left corner has the highest specificity and the highest sensitivity.

We compare two or more ROC's by looking at the area under the curve. High area indicates that the curve will be more towards the upper left side and hence the points lying on it will have high sensitivity and high specificity.

GENEREALLY:



AUC = 0.5 is the dotted line and it indicates that is prediction is just random guessing, whereas AUC <0.5 indicates our prediction is worse than random guessing and the points on such a curve will be towards the lower right corner.



In the above picture, the highlighted point C is the best predictor followed by point A . Point B is just a random guessing predictor whereas point C below the dotted line is the worst predictor.

CROSS VALIDATION:

We subset our sample data which we are going to use to train the prediction model into a training set and a testing set. We do this because we can't use another sample data as a test set while building the model since, this new sample dataset can then become a part of the training set and model can become acquainted with the noise there in this new dataset .

Hence we do all our testing while building the model on the training data set only by sub-setting it.

Cross-validation

Approach:

1. Use the training set
2. Split it into training/test sets
3. Build a model on the training set
4. Evaluate on the test set
5. Repeat and average the estimated errors

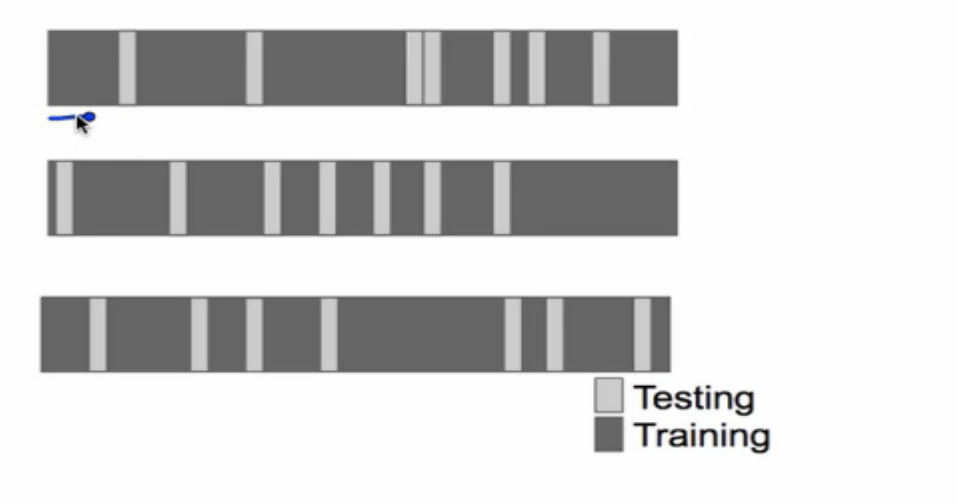
Used for:

1. Picking variables to include in a model
2. Picking the type of prediction function to use
3. Picking the parameters in the prediction function
4. Comparing different predictors

4/8

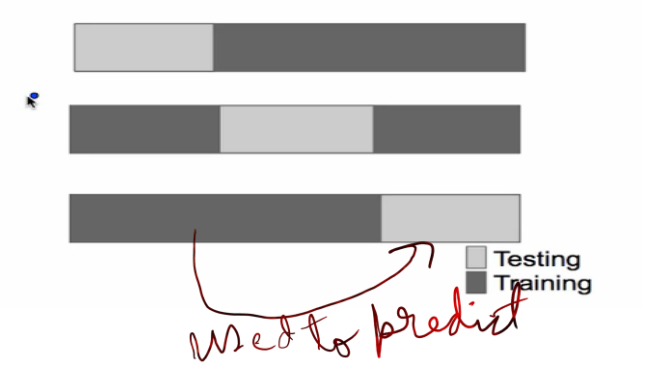
METHODS OF SUBSETTING:

1. RANDOM SUBSAMPLING:



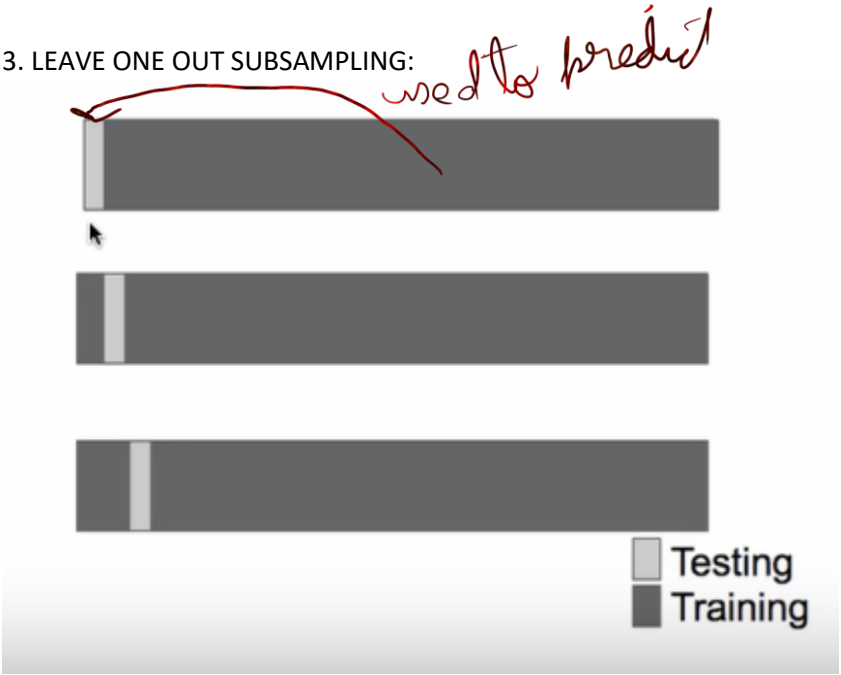
#We randomly divide the data into testing and training datasets.
#We use the black coloured values to train our model and try to predict the grey coloured values to test the accuracy of our model.

2. K-FOLD SUBSAMPLING:



Above is an example of 3 fold testing. We divide our data into 1/3 testing , 2/3 training then into 1/3 training ,1/3 testing ,1/3 training and then into 2/3 training , 1/3 testing and do the training of black parts and testing on grey parts.

3. LEAVE ONE OUT SUBSAMPLING:



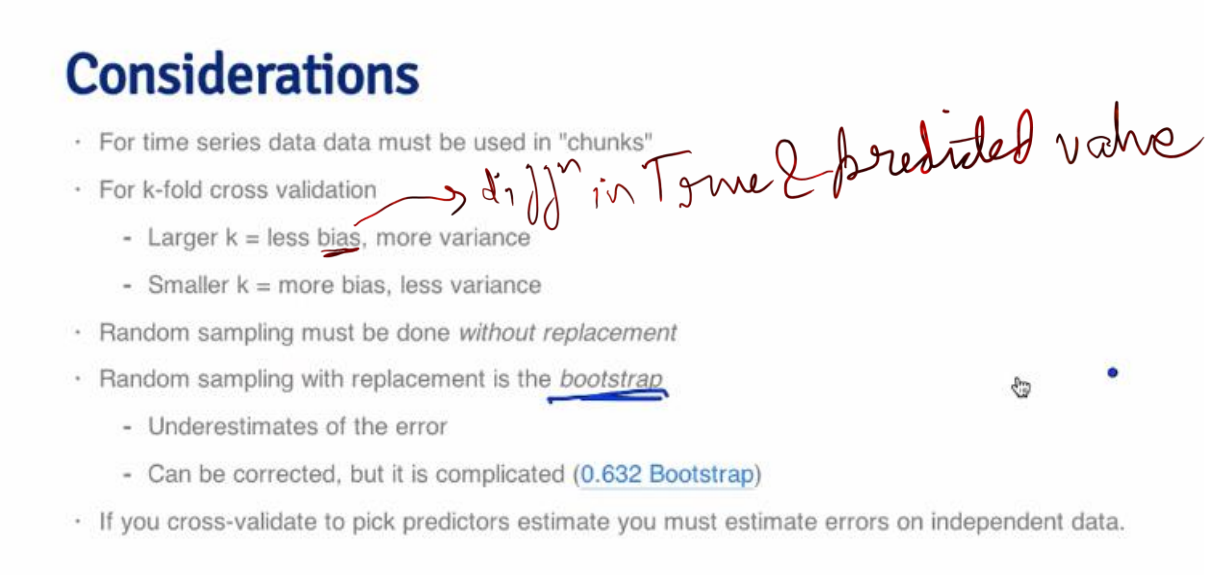
a) We leave one sample out each time for testing and do the training using the other samples.

b) In the first picture, we leave the first sample out for testing and use other for training.

c) In the second picture, we leave the second sample out for testing and use other samples for training.

d) We do this until we reach the end of the dataset.

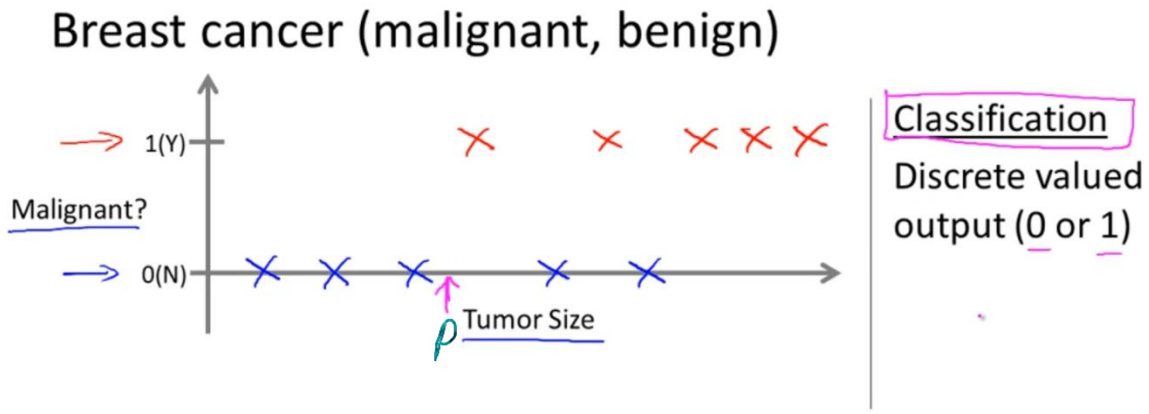
SOME THINGS ABOUT CROSS VALIDATION TO KEEP IN MIND



Supervised learning:

1.In supervised learning we give the programme certain amount of information and it trains itself it using and tries to predict further things.

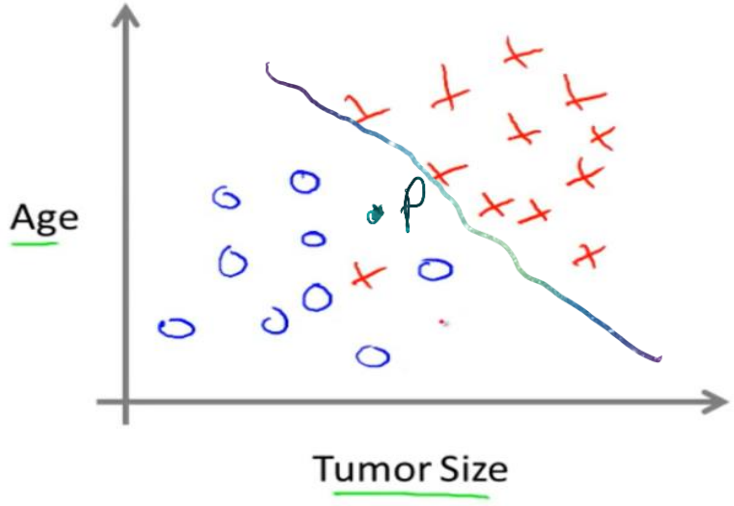
Eg:



Here the x axis corresponds to certain tumour sizes and y axis tells that whether a tumour is malignant(1 on the y line)i.e. harmful or benign(0 on the y line) i.e. harmless.

We have inputted some pre-existing values pertaining to tumours -identified in the graph by blue and red crosses into the programme. The programme now tries to classify the point p as malignant or benign on the basis of the information we have inputted.

Eg2:



Here we try to find the relation between age and tumor size. The blue dots represent benign tumors and the red crosses represent malignant tumors.

If we ask the programme to classify point p as malignant or benign , it would first try and separate crosses and dots from each other and then try to classify p.

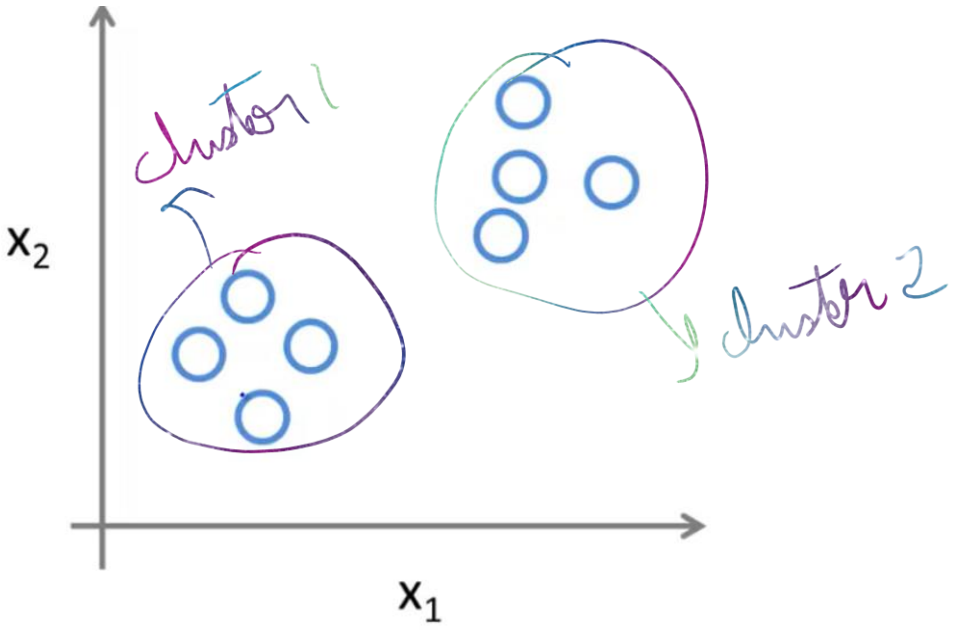
Classification:

It means grouping data into discreet values like 0 , 1 ,2 etc.

Eg: In eg 1 given above, benign tumors are represented by 0 and malignant tumors by 1.

UNSUPERVSIED LEARNING:

In unsupervised learning we present the programme with just the data , which has not been grouped or classified. We ask the programme to classify it.

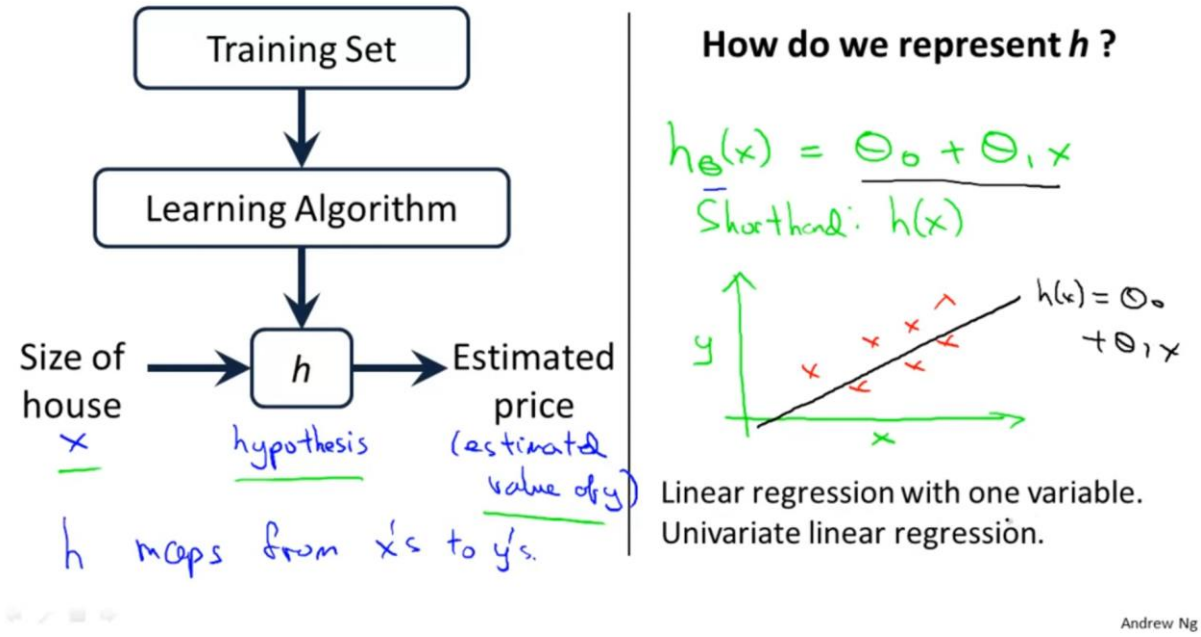


Here we ask the programme to classify the points plotted on the graph . The programme may classify the points as cluster 1 and cluster 2.

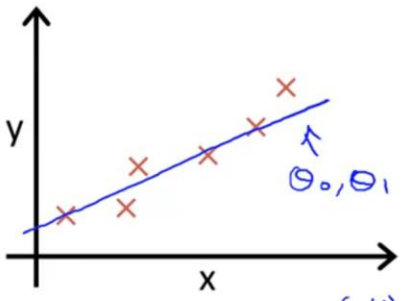
Another example may be that we may have two audio recordings with two overlapping voices. Suppose voice 1 is more significant in audio 1 and voice 2 in audio 2. Now we can ask the

programme to separate voice 1 and 2 from both the recordings . Now after the separation recording 1 has only voice 1 and recording 2 has only voice 2. This is an example of unsupervised learning.

MODEL REPRESENTATION:



COST FUNCTION:



#training examples

$$\text{minimize}_{\theta_0,\theta_1} \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

$$h_\theta(x^{(i)}) = \theta_0 + \theta_1 x^{(i)}$$

$$J(\theta_0,\theta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

Idea: Choose θ_0, θ_1 so that $h_\theta(x)$ is close to y for our training examples (x, y)

x, y

$$\text{minimize}_{\theta_0,\theta_1} J(\theta_0,\theta_1)$$

Cost function

Squared error function

Andrew Ng

Cost function is also known as the squared error function:

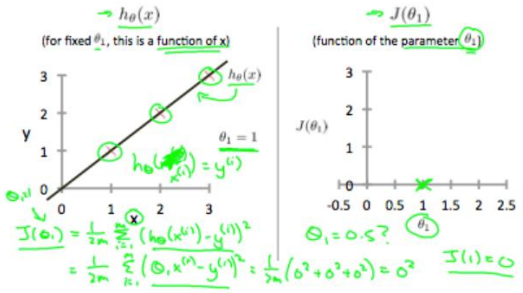
##sLet us consider our constant to be 0 , so the equation becomes:

$h_\theta(x) = \theta_1 x$

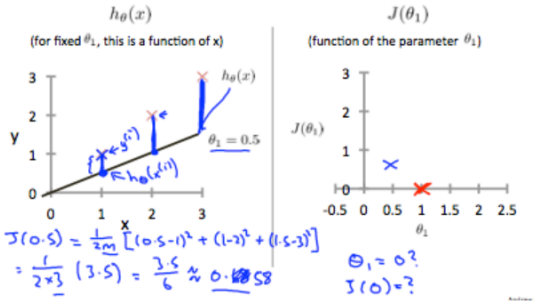
Cost Function - Intuition I

If we try to think of it in visual terms, our training data set is scattered on the x-y plane. We are trying to make a straight line (defined by $h_\theta(x)$) which passes through these scattered data points.

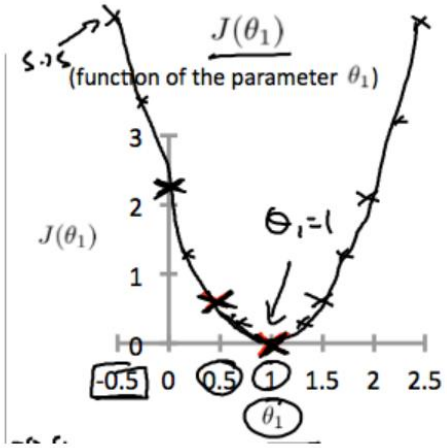
Our objective is to get the best possible line. The best possible line will be such so that the average squared vertical distances of the scattered points from the line will be the least. Ideally, the line should pass through all the points of our training data set. In such a case, the value of $J(\theta_0, \theta_1)$ will be 0. The following example shows the ideal situation where we have a cost function of 0.



When $\theta_1 = 1$, we get a slope of 1 which goes through every single data point in our model. Conversely, when $\theta_1 = 0.5$, we see the vertical distance from our fit to the data points increase.



This increases our cost function to 0.58. Plotting several other points yields to the following graph:



Thus as a goal, we should try to minimize the cost function. In this case, $\theta_1 = 1$ is our global minimum.

And hence $h_\theta(x) = 1 \times x$

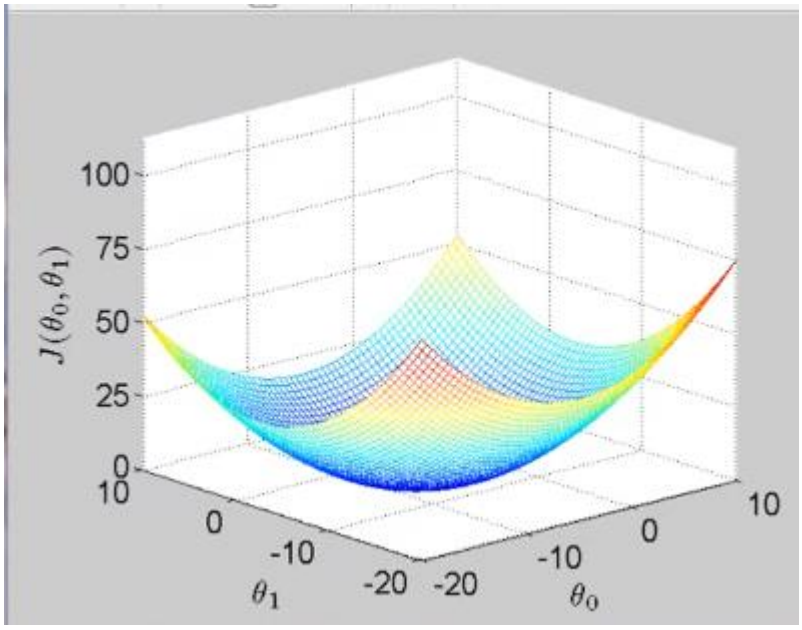
$h_\theta(x) = x$ is our final equation for which cost function is minimum.

Now let us assume our equation to be:

$$h_\theta(x) = \theta_0 + \theta_1 x$$

Our aim to find the value of θ_0 & θ_1 for which the cost function is minimum.

Graph of cost function vs θ_0 & θ_1



We infer from the above graph that a particular value of the cost function can be achieved through different combinations of different values of θ_0 & θ_1

For eg , the value 25 of the cost function may be achieved through various combinations of values of θ_0 & θ_1 .

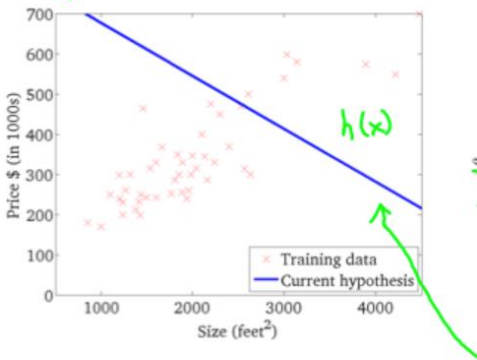
Above visualization in 2-d using contour plots:

Cost Function - Intuition II

A contour plot is a graph that contains many contour lines. A contour line of a two variable function has a constant value at all points of the same line. An example of such a graph is the one to the right below.

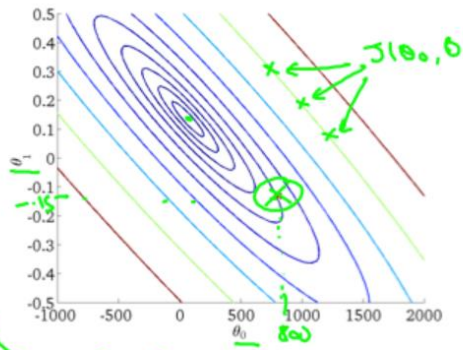
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



$J(\theta_0, \theta_1)$

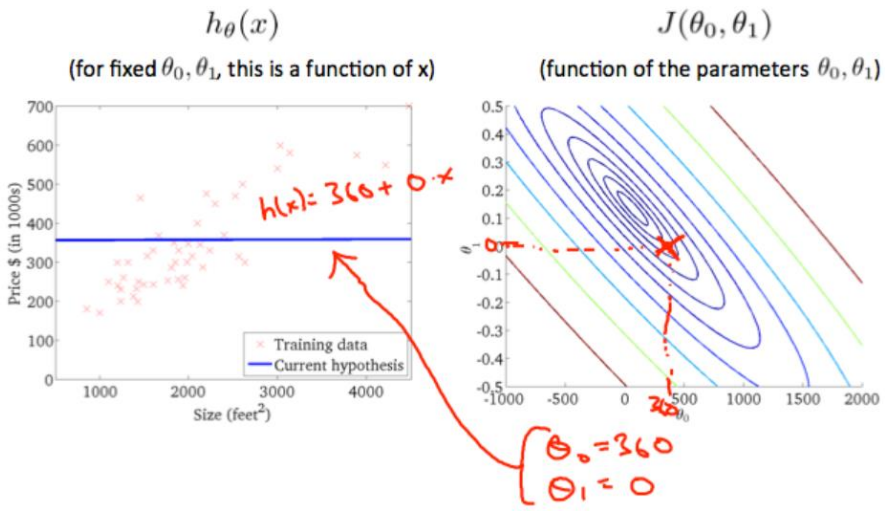
(function of the parameters θ_0, θ_1)



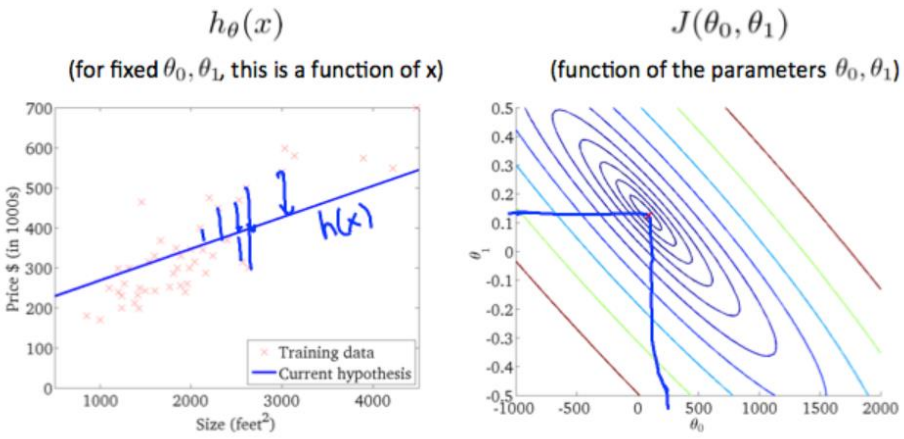
θ_0, θ_1

(all 3 points have same value of cost function)

Taking any color and going along the 'circle', one would expect to get the same value of the cost function. For example, the three green points found on the green line above have the same value for $J(\theta_0, \theta_1)$ and as a result, they are found along the same line. The circled x displays the value of the cost function for the graph on the left when θ_0 = 800 and θ_1= -0.15. Taking another h(x) and plotting its contour plot, one gets the following graphs:



When θ_0 = 360 and θ_1 = 0, the value of $J(\theta_0, \theta_1)$ in the contour plot gets closer to the center thus reducing the cost function error. Now giving our hypothesis function a slightly positive slope results in a better fit of the data.



The graph above minimizes the cost function as much as possible and consequently, the result of θ_1 and θ_0 tend to be around 0.12 and 250 respectively. Plotting those values on our graph to the right seems to put our point in the center of the inner most 'circle'.

GRADIENT DESCENT

Gradient descent is a technique used to find minimum value of any function $J(\theta_0, \theta_1, \theta_2 \ldots \theta_n)$

Gradient descent algorithm

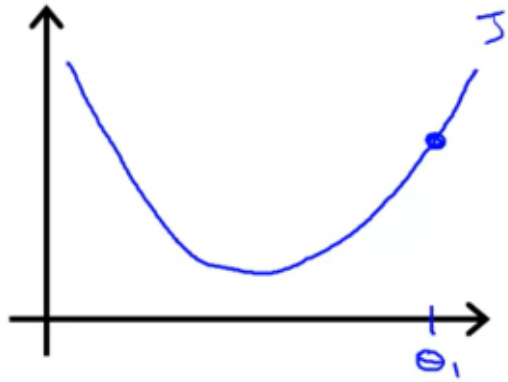
repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \quad \text{(for } j = 0 \text{ and } j = 1\text{)}$$

}

First we look at a very simple example where we have a function $J(\theta_1)$.

Now, we have to minimize the value of this function.

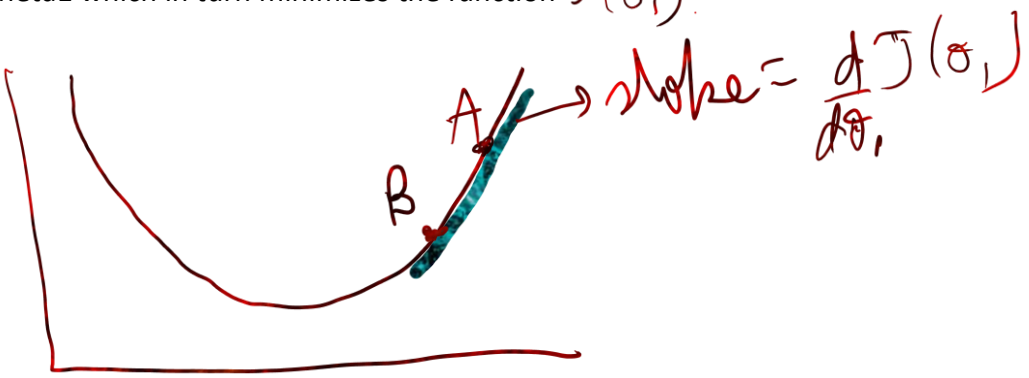


$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$

here α is the learning rate of the algorithm.

$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$, := means we are updating value of θ_1 to $\theta_1 - \alpha \frac{dJ(\theta_1)}{d\theta_1}$.

Updating theta1 means that we are changing value of theta 1 till we reach the minimum value of theta1 which in turn minimizes the function $J(\theta_1)$.



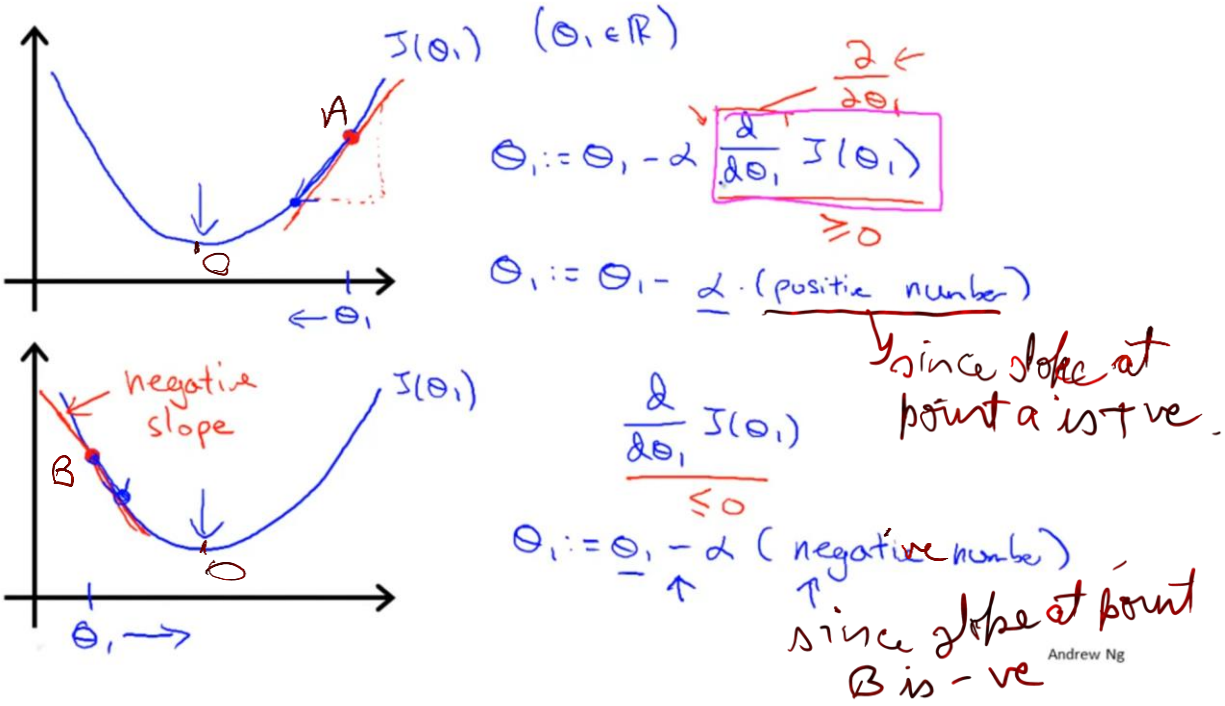
Let slope at point A = a. Hence, $\text{slope} = \frac{d}{d\theta_1}(J(\theta_{1A})) = a$

As we move from point A to point B, θ_1 changes:

$\theta_{1B} = \theta_{1A} - \alpha \frac{d}{d\theta_1}(J\theta_{1A})$

here θ_{1B} represents change as we move from A to B.

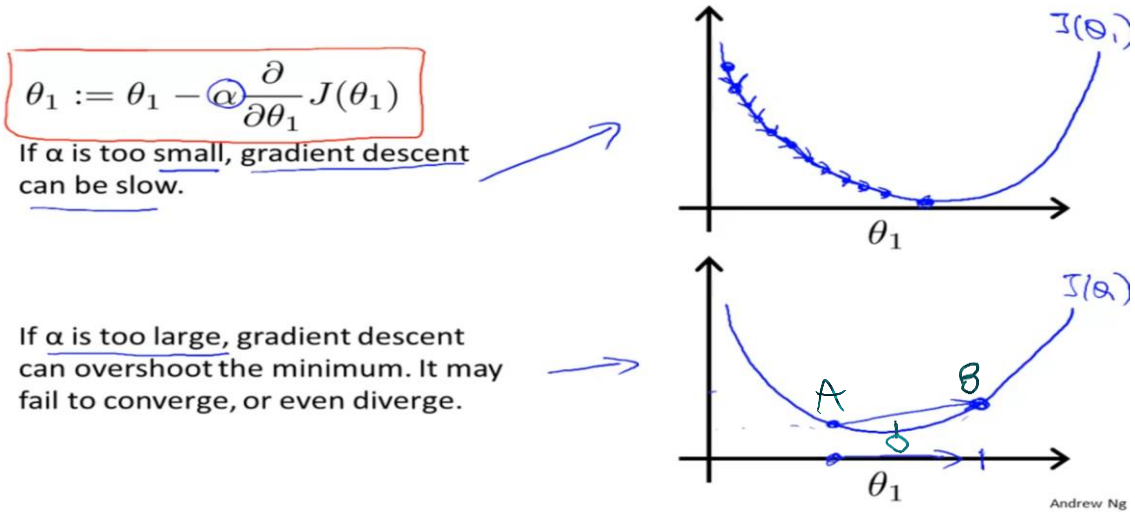
NOTE: Higher the value of slope alpha, faster the change in theta1.



In graph 1 , θ_1 value keeps on decreasing as we move from point A to point o (point of minima).

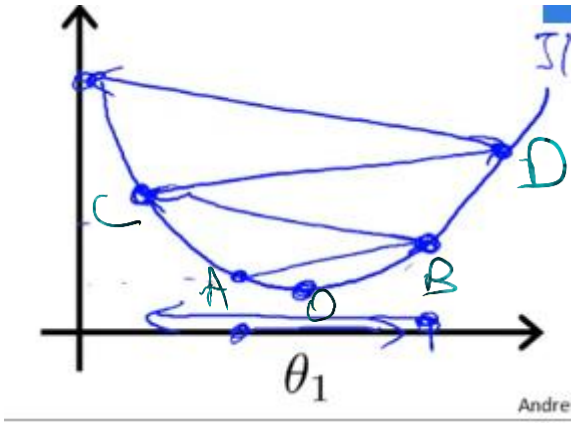
In graph 2, θ_1 value keeps on increasing as we move from point B to point o.

Some imp. points pertaining to alpha:

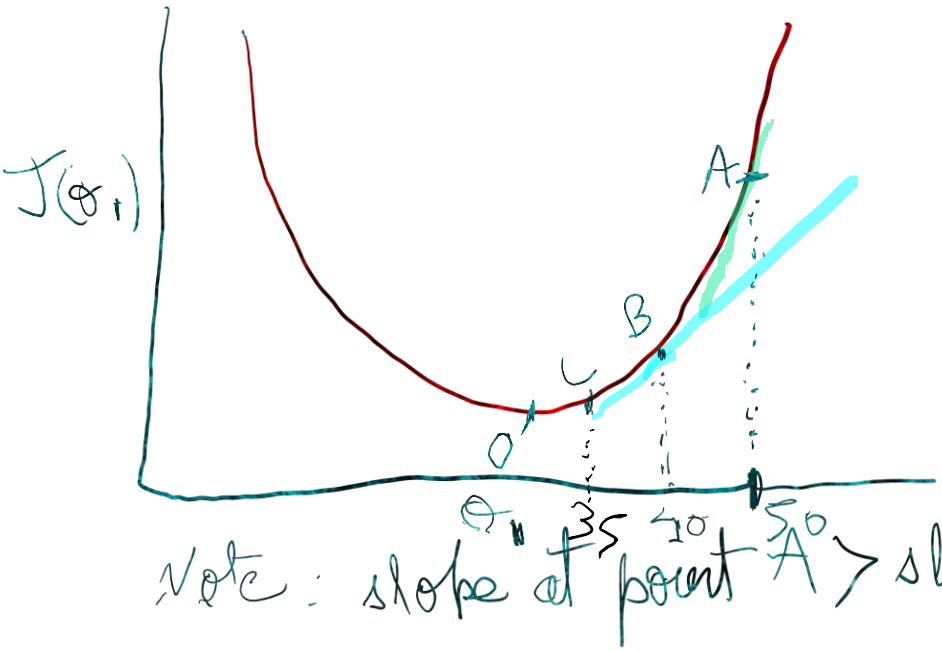


In graph 2 , as we change try to go from point A to point O (point of minima), there is positive change in value of θ_1 i.e. value of θ_1 increases yet we do not reach point O

and reach point B since value of alpha is so large , then we try to come back to point O but since alpha is so large , we reach a point C , and then point D and so on. Eventually we see we keep getting farther and farther from point O instead of getting closer to it.



Divergence of gradient descent.



Note : slope at point A > slope at point B.

As we move from point A to point B, value of θ_1 changes from 50 to 40, i.e. change of 10 units.

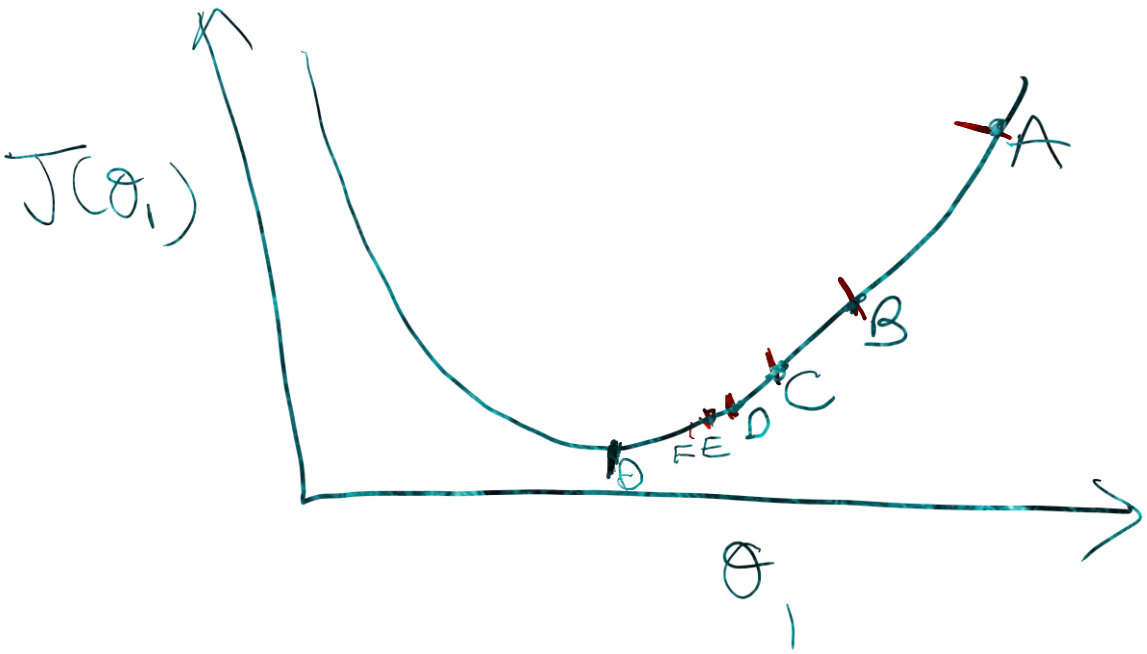
As we move from point B to point C, value of θ_1 changes from 40 to 35, i.e. change of 5 units.

We see less change in value of θ_1, because the slope at B is < slope at point A.

$$\theta := \theta_1 - \frac{d}{d\theta_1}\left(J(\theta_1)\right)$$

Hence as we keep getting closer to point of minima θ, the leaps we take from one point to another keep decreasing in magnitude.

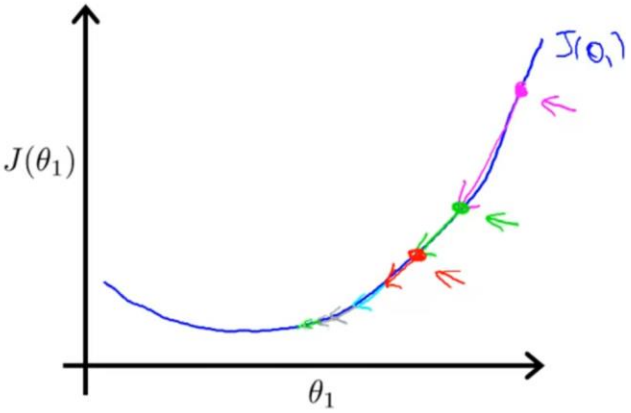
As we get very get very close to θ, the steps taken from one point to another become extremely small.



Gradient descent can converge to a local minimum, even with the learning rate α fixed.

$$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$$

As we approach a local minimum, gradient descent will automatically take smaller steps. So, no need to decrease α over time.



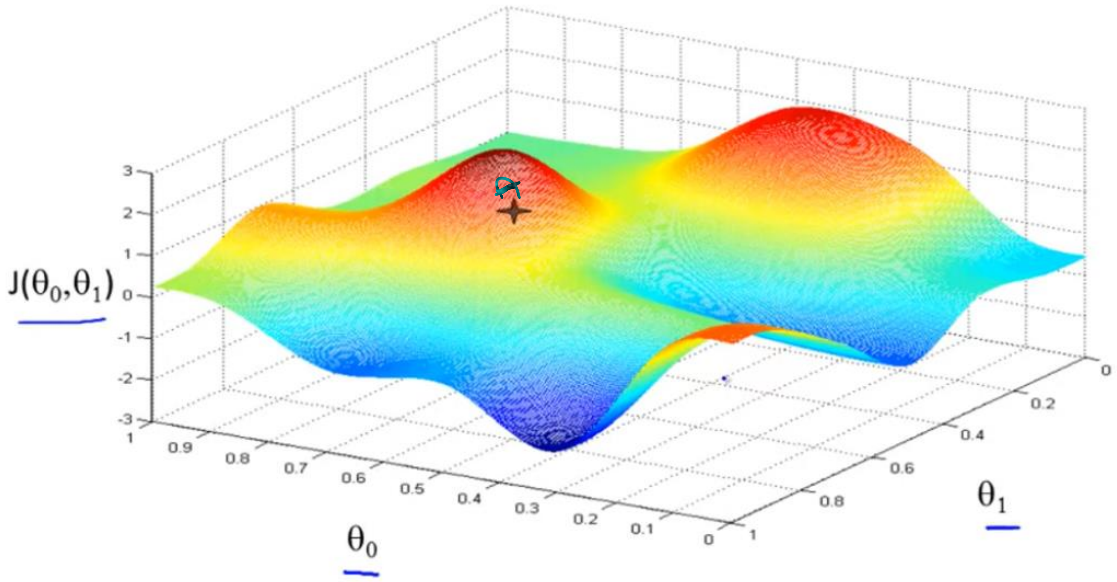
Andrew Ng

s

Now , We see another example in which we have the function $J(\theta_0, \theta_1)$

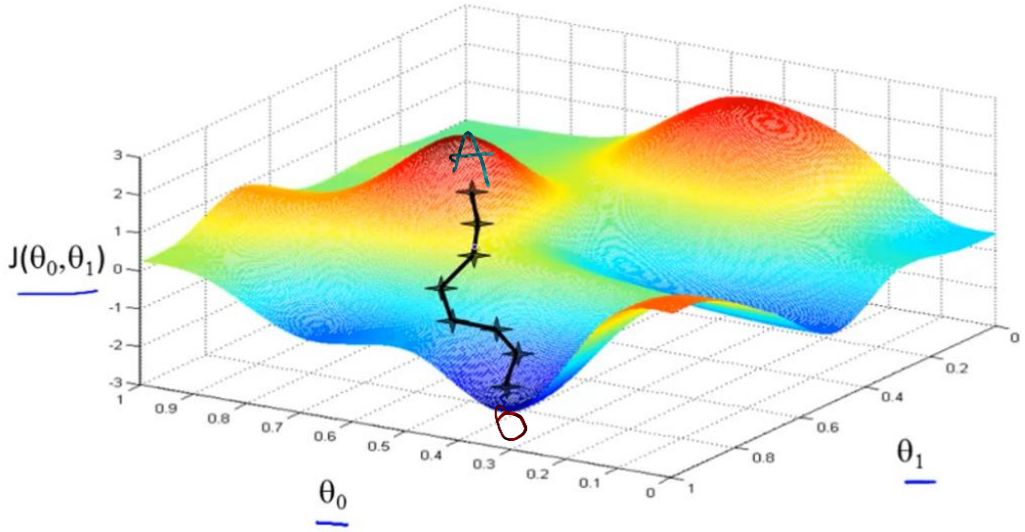
We want to reach point of minima for which the function has minimum value for the given θ_0, θ_1

Let us assume we are at point A which is represent by the cross on the graph of the function:



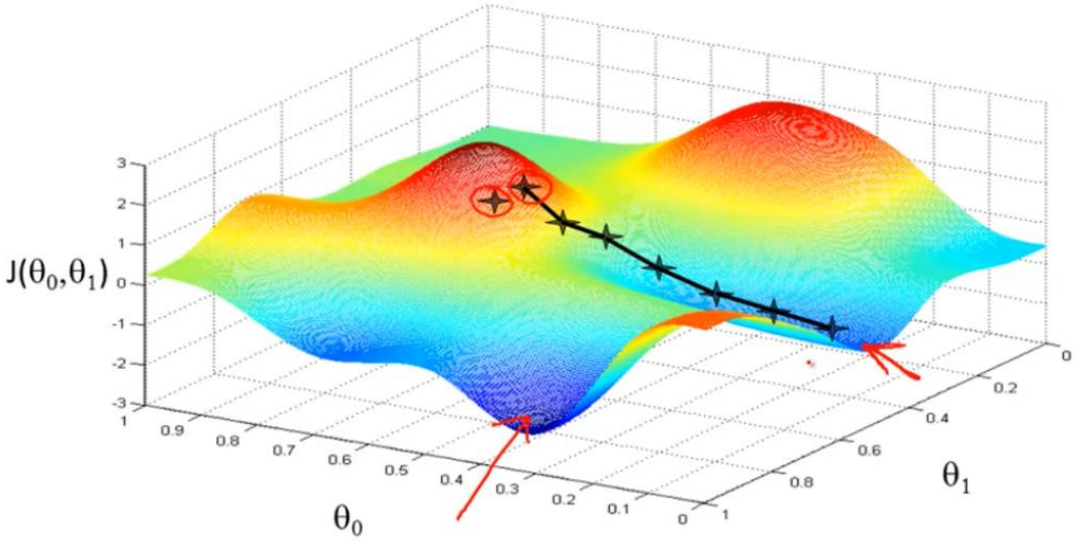
Now, we want to reach the point of local minima. Let's suppose that in this pursuit of reaching the point of minima, we start taking baby steps down the slope from point A .

We keep going until we reach a point O.



As we take each step starting from point A , value of θ_1, θ_0 changes with each step till we reach the point of minima.

##If we would have moved a bit towards right from A and then started moving downwards , we would have reached a different local minima.



Gradient descent algorithm

Assignment: $a := b$, $a := a+1$ | Truth assertion: $a = b$, $a = a+1$ ✗

repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \quad (\text{for } j = 0 \text{ and } j = 1)$$

}

learning rate

Simultaneously update θ_0 and θ_1

Correct: Simultaneous update

$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$

$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$

$\theta_0 := \text{temp0}$

$\theta_1 := \text{temp1}$

Incorrect:

$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$

$\theta_0 := \text{temp0}$

$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$

$\theta_1 := \text{temp1}$

wrong because value of θ_0 has been updated in this equation

Andrew Ng

Getting updated value of θ_0 & θ_1 at each step as we move from any point to the point of minima:

$$\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) = \frac{\partial}{\partial \theta_j} \cdot \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

$$= \frac{\partial}{\partial \theta_j} \frac{1}{2m} \sum_{i=1}^{m} \left(\theta_0 + \theta_1 x^{(i)} - y^{(i)} \right)^2$$

θ_0 $j = 0$: $\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)$

θ_1 $j = 1$: $\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) \cdot x^{(i)}$

Explanation:

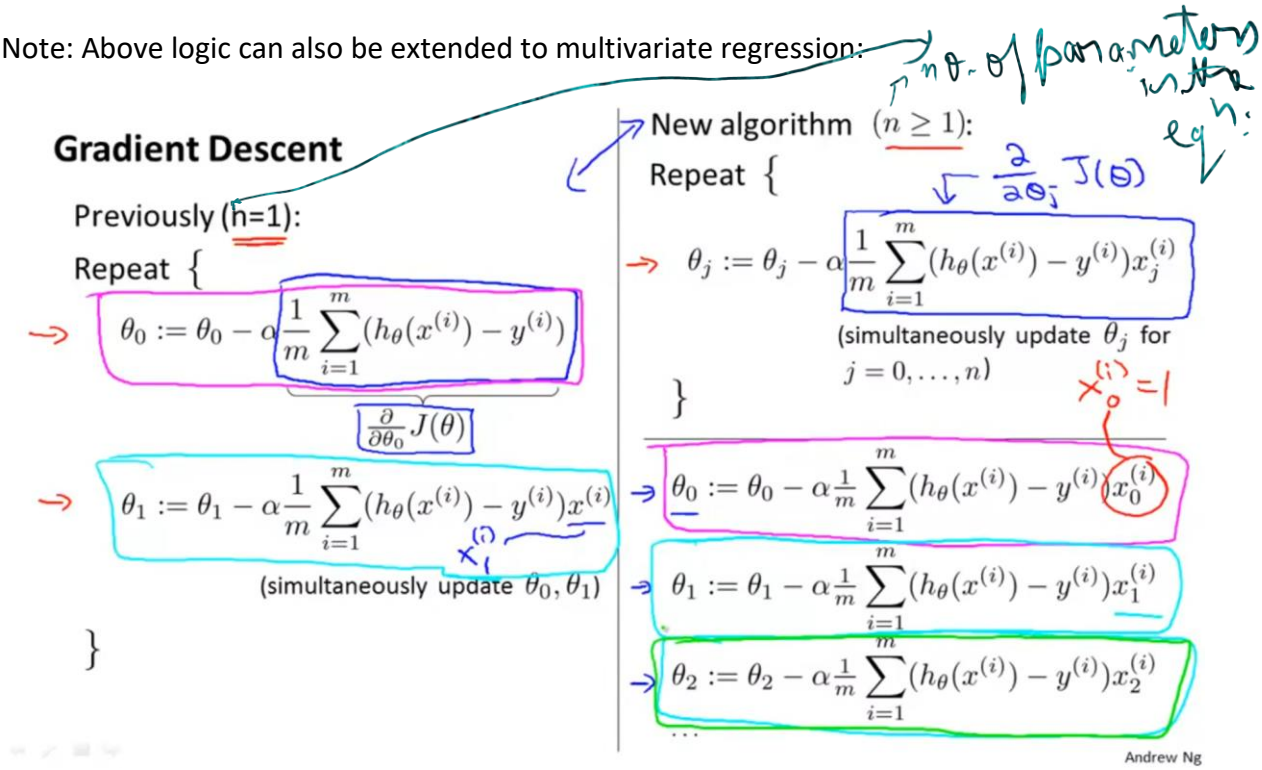
$$\frac{\partial}{\partial \theta_0} \frac{1}{2m} \sum_{i=1}^{m} \left(\theta_0 + \theta_1 x^{(i)} - y^{(i)} \right)^2$$

$$= \frac{1}{2m} \times 2 \sum_{i=1}^{m} (\theta_0 + \theta_1 x^{(i)} - y^{(i)}) \times \frac{\partial}{\partial \theta_0} (\theta_0 + \theta_1 x^{(i)} - y^{(i)})$$

$$= \frac{1}{m} \sum_{i=1}^{m} (\theta_0 + \theta_1 x^{(i)} - y^{(i)}) \times 1$$

Similar goes for $\frac{\partial}{\partial \theta_1}$

Note: Above logic can also be extended to multivariate regression:



Note: Superscript (i), is not usually not used to denote rows or columns but simply refers to the i^{th} training example. For eg: $x^{(2)}$, refers to training 2^{nd} example.

Let us assume the equation has 2 variables i.e. n=2.

$h(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2$ where $x_0 = 1$

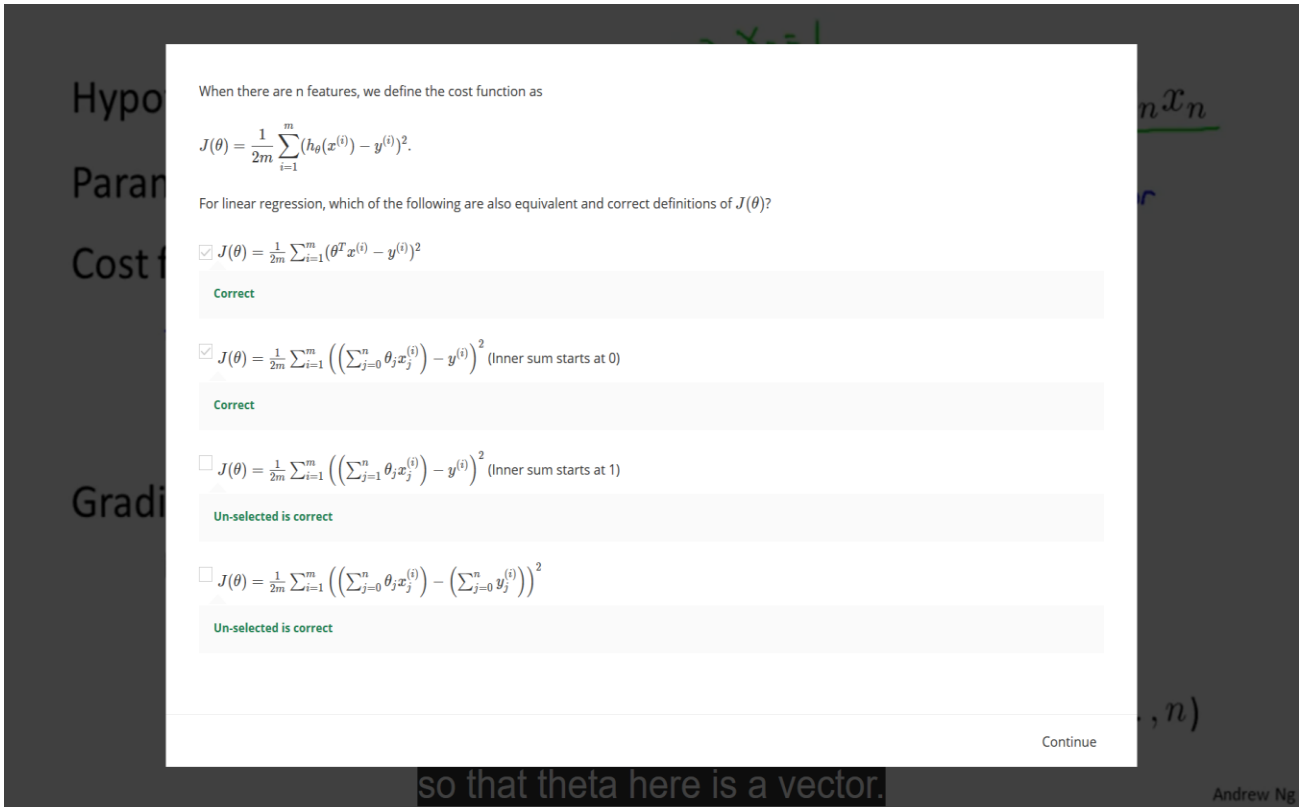
$$\frac{\partial}{\partial \theta_j} \left[\frac{1}{2m} \sum_{i=1}^{m} (\theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2)^2 \right] \rightarrow (1)$$

Let $j = 2$

(1) becomes: $\frac{\partial}{\partial \theta_2} (1) \Rightarrow \frac{1}{2m} \times 2 \sum_{i=1}^{m} (\theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2) \times (0 + 0 + x_2)$

$$= \frac{1}{m} \sum_{i=1}^{m} (\theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2) x_2$$

QUESTION:



When there are n features, we define the cost function as

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2.$$

For linear regression, which of the following are also equivalent and correct definitions of $J(\theta)$?

- [x] $J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(\theta^T x^{(i)} - y^{(i)})^2$

  Correct

- [x] $J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(\left(\sum_{j=0}^{n}\theta_j x_j^{(i)}\right) - y^{(i)}\right)^2$ (Inner sum starts at 0)

  Correct

- [ ] $J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(\left(\sum_{j=1}^{n}\theta_j x_j^{(i)}\right) - y^{(i)}\right)^2$ (Inner sum starts at 1)

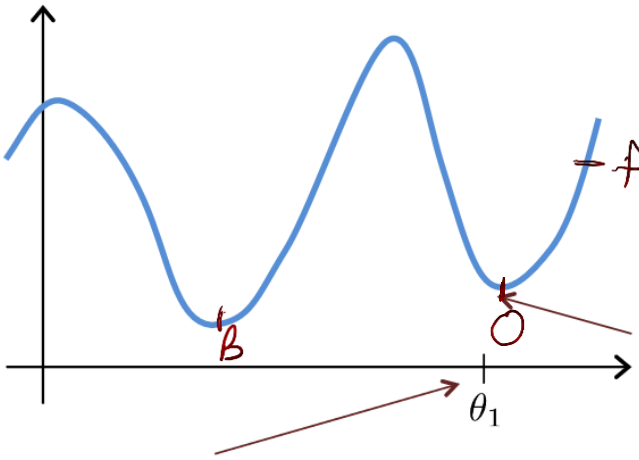
  Un-selected is correct

- [ ] $J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(\left(\sum_{j=0}^{n}\theta_j x_j^{(i)}\right) - \left(\sum_{j=0}^{n} y_j^{(i)}\right)\right)^2$

  Un-selected is correct

Continue

Theta always converges to the point of local minima due to gradient descent:



θ_1 at local optima

Current value of θ_1

$$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$$

0 when we are at point O.

Let us assume that we start from point A and proceed towards point O. We know O is the local minima , while point B is the global minima. As we reach point O , there can be no further change in value of theta1 this is because slope at point O is = 0 , hence in accordance with the above given formula , change in theta 1 = 0. Since

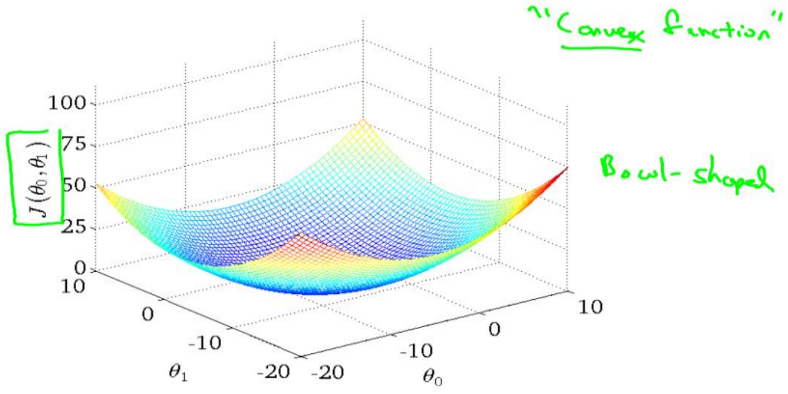
$\theta_1 := \theta_1 - \alpha \times 0$

$\theta_1 := \theta_1 - 0$

$\theta_1 := \theta_1$

COST FUNCTION FOR LINEAR REGRESSION:

It turns out that that the cost function for linear regression is always going to be a bow shaped function like this:

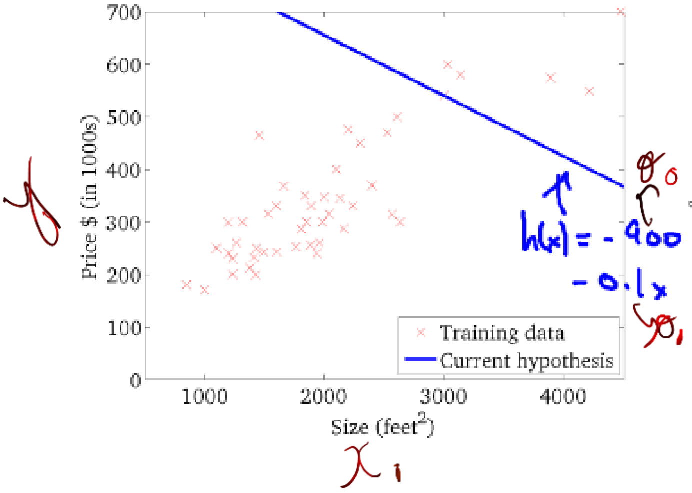


The technical term for this is that this is called a convex function.

AN EXAMPLE SHOWING GRADIENT DESCENT IN WORK:

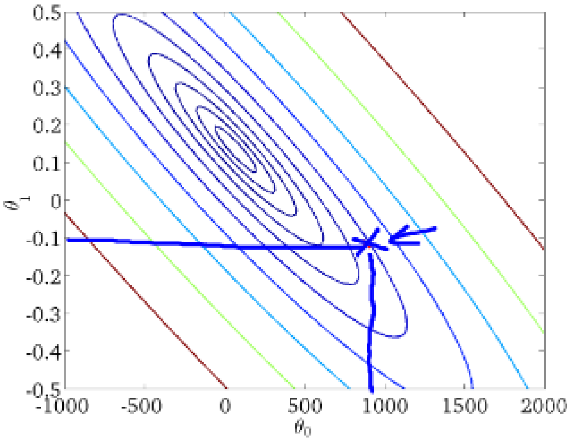
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



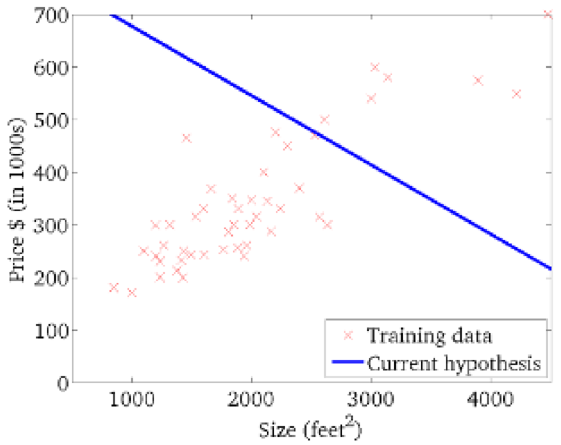
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



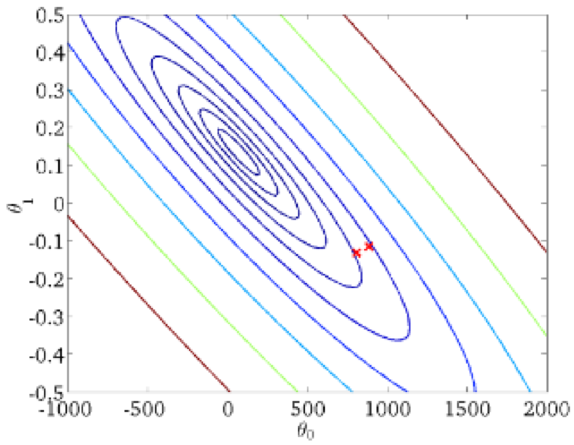
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



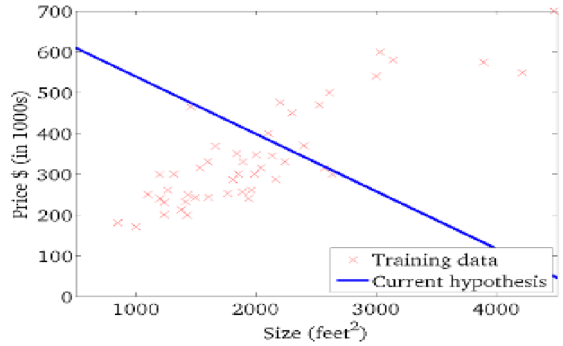
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



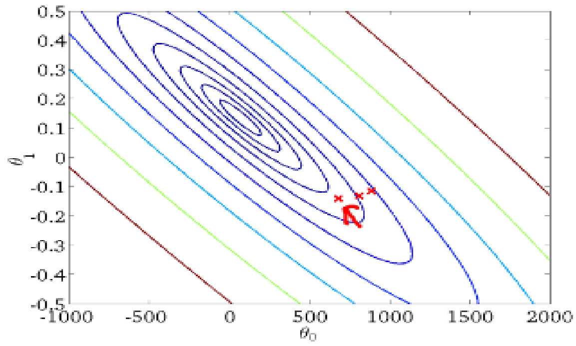
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



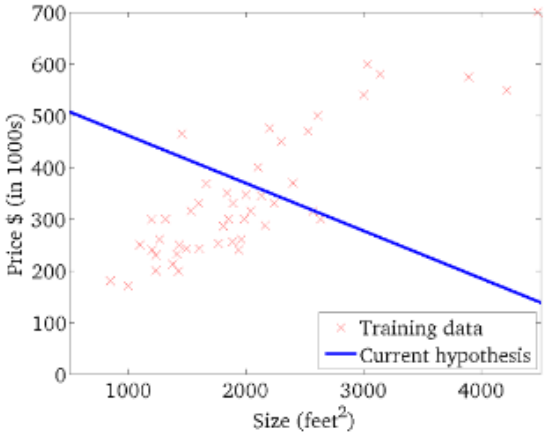
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



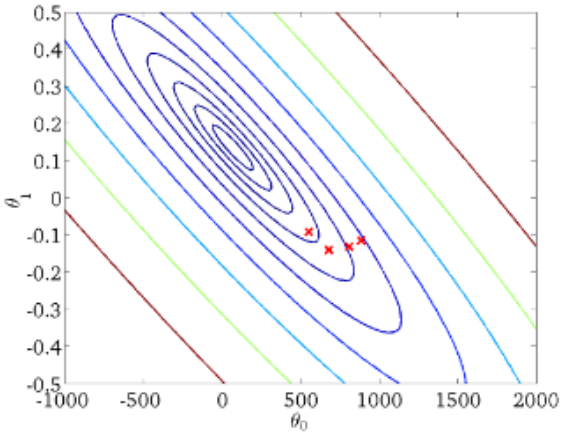
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



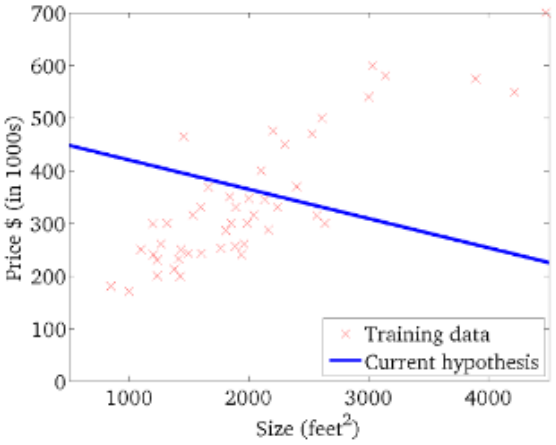
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



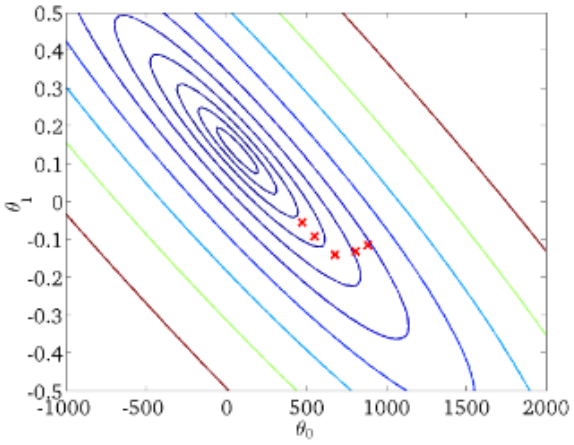
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



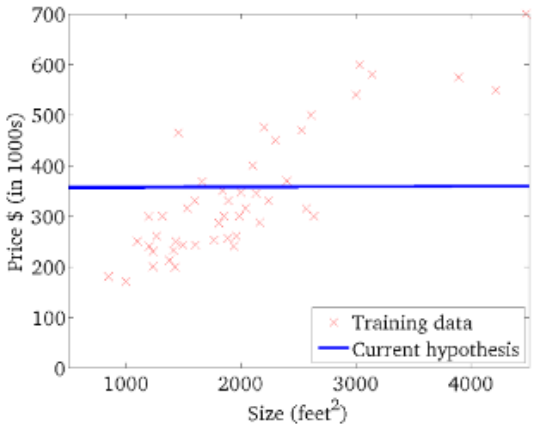
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



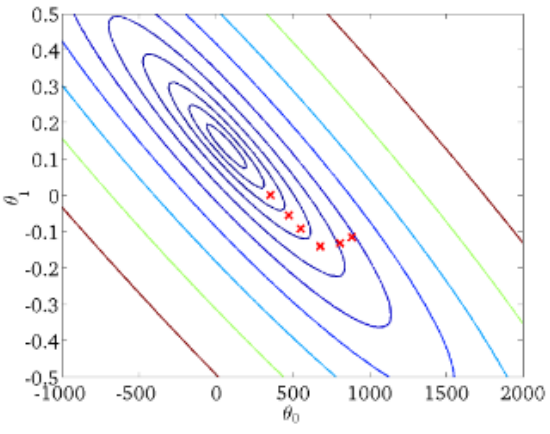
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



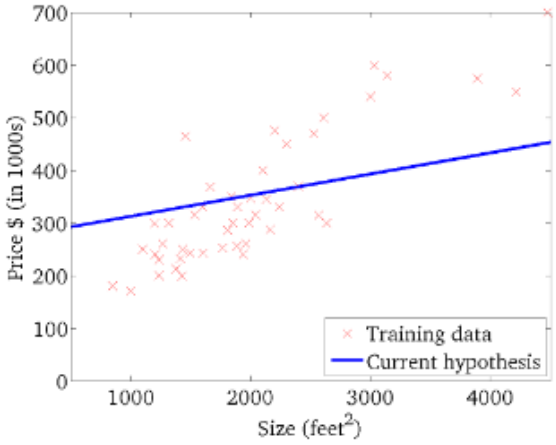
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



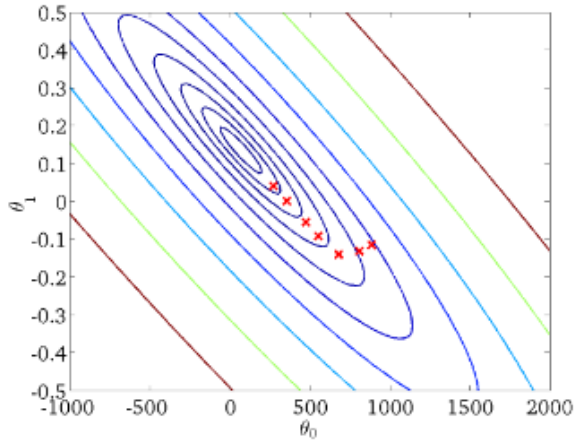
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



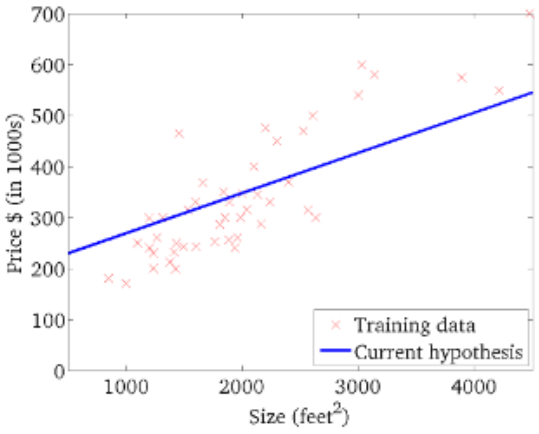
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



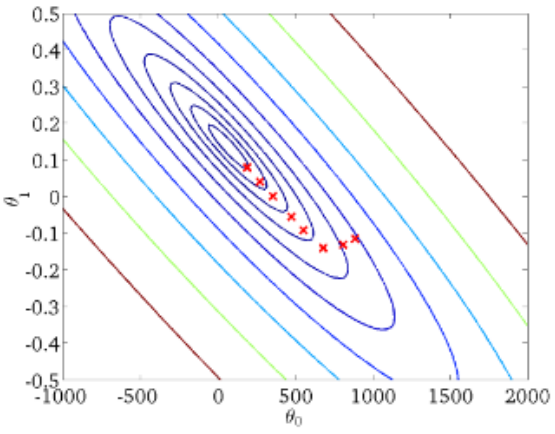
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



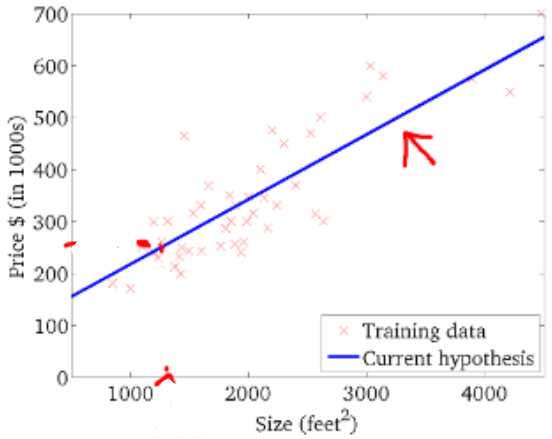
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



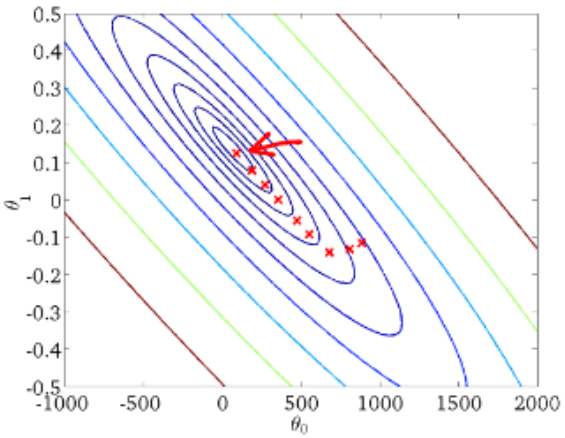
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



$J(\theta_0, \theta_1)$

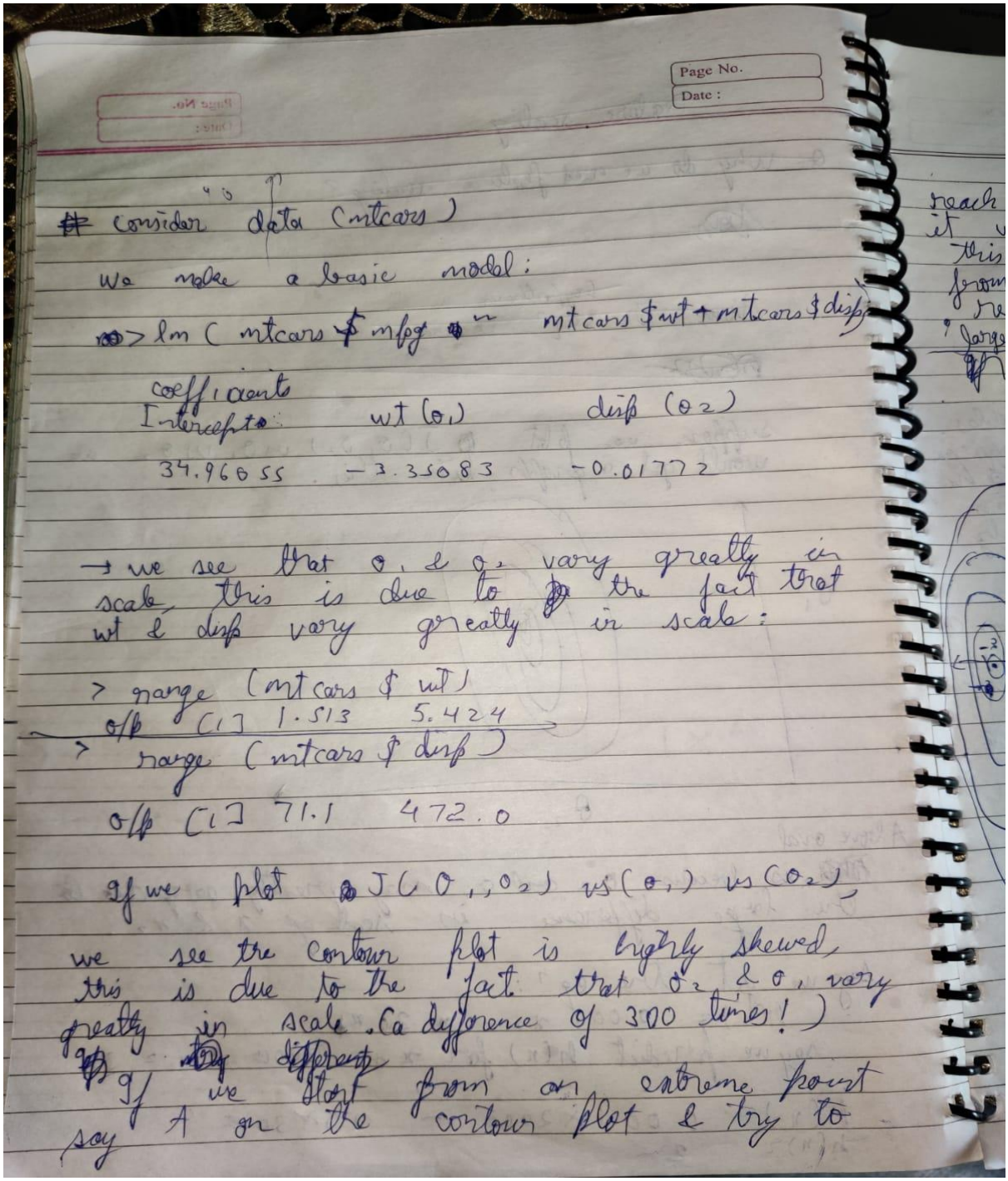
(function of the parameters θ_0, θ_1)



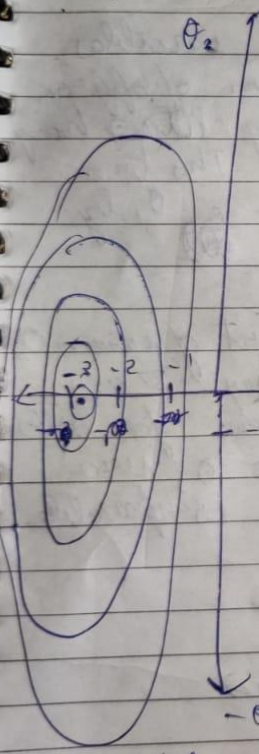
In the above pictures as we keep getting closer to the perfect fit we start getting closer to the centre of the contour plot i.e. , the point of minima.

FEATURE SCALING:

Why do we need feature scaling?



reach point O, i.e. the point of minima it would take a lot of time to reach this point since reaching (-3.35, -0.01) from a point say A = (37, 42) is very longer reaching 0.01 is especially tough. θ_2 will take a large time to descend from A to O & hence gradient descent will become slow. The graph looks like this:



Thus, we say huge difference in scale, say 0.01772 i.e. θ_1 can vary from 0.003 to 0.0478 and θ_2 can vary from 0.1 to 8.4, if we plot corresponding θ_2 vs θ_1, we get a very skewed graph due to diff^n in values.

If on the other hand we normalise the variables:

```
> w <- (mtcars$wt - mean(mtcars$wt))/sd(mtcars$wt)

> d <- (mtcars$disp - mean(mtcars$disp))/sd(mtcars$disp)
```

> lm(mpg ~ wt + d)

coefficients:

(Intercept)	w (θ_1)	d (θ_2)
20.091	-3.279	-2.197

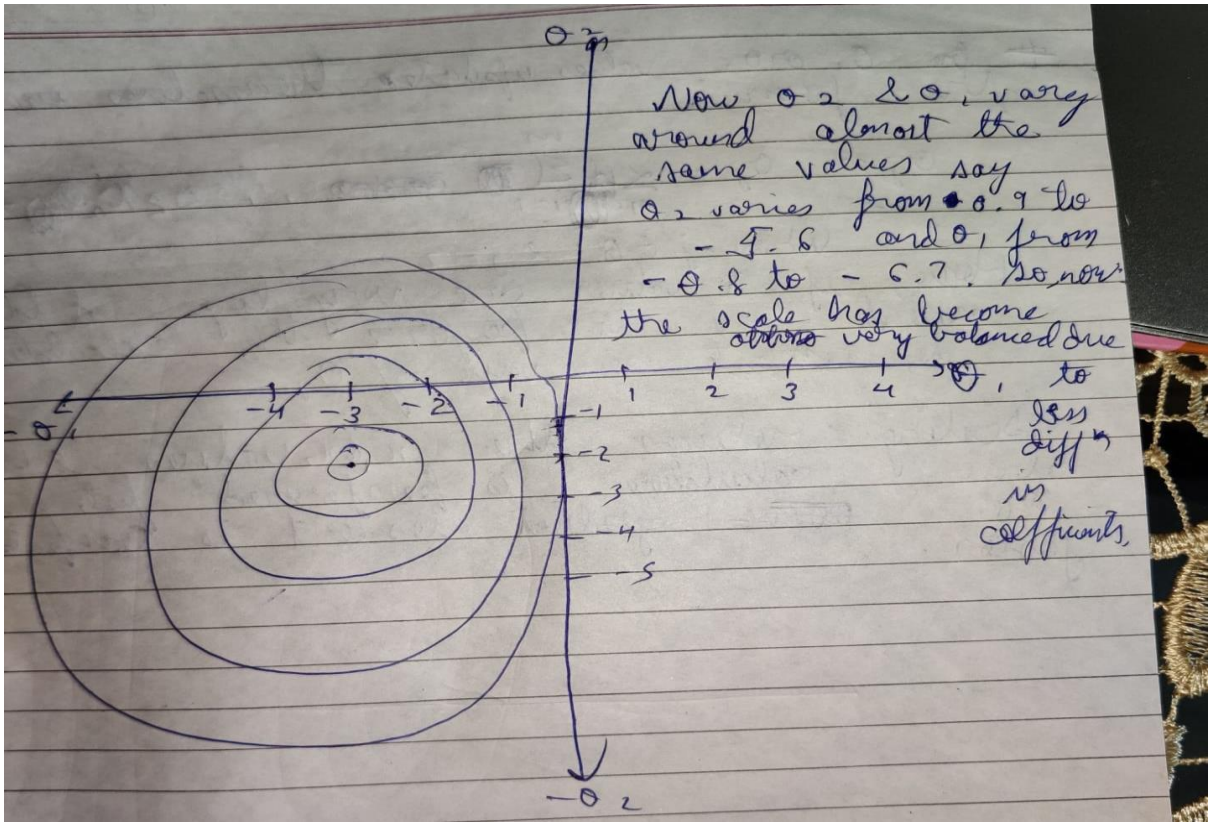
Now, as we have normalised the variables, the variables which had a difference of approx. 300 times, are now only having a difference of 1.49249 times. This times difference is obtained by dividing θ_1 by θ_2.

Now, if we have to reach the point of minima θ_1 (-3.279, -2.197) from the point A say (37, 42), it would take approximately same time for both θ_1 & θ_2 to descend from A to 0, since the coeff. of θ_1 & θ_2 are now comparable.

This makes gradient descent run faster.

The graph looks like this:

P.T.O



Now θ_2 & θ_1 vary around almost the same values say θ_2 varies from 0.9 to -5.6 and θ_1 from -0.8 to -6.7. So now the scale has become very balanced due to less diff^{n} in coefficients.

Page No.
Date :

the θ_1 or θ_2 values update themselves using the formula:

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (\theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} - y^{(i)}) x_j^{(i)}$$

where $j \in 0, 1$ & $j \in 2$

Here, if x values are very large it takes more time to perform the calculation to compute new value of θ_j

Scaling reduces the x_1 & x_2 values & hence above calculation is performed faster & the gradient descent is computed faster.

Feature Scaling

Get every feature into approximately a $-1 \leq x_i \leq 1$ range.

$x_0 = 1$

$0 \leq x_1 \leq 3$ ✓

$-2 \leq x_2 \leq 0.5$ ✓

$-100 \leq x_3 \leq 100$ ✗

$-0.0001 \leq x_4 \leq 0.0001$ ✗

-3 to 3 ✓

$-\frac{1}{3}$ to $\frac{1}{3}$ ✓

These are also correct scales based on requirement

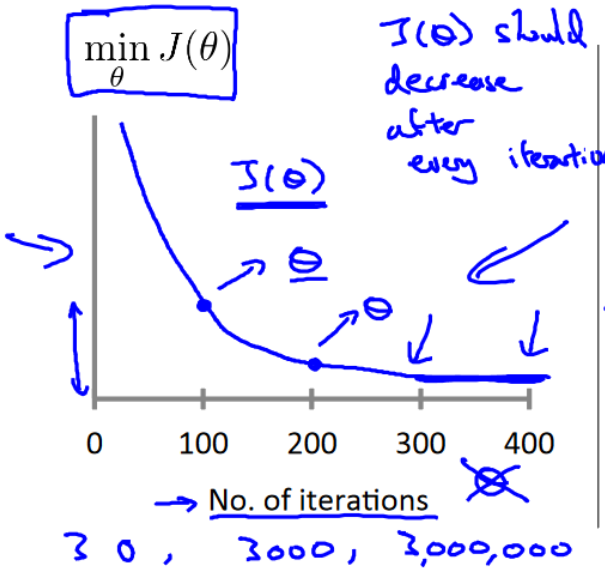
Andrew Ng

Methods for scaling:

1. Divide the data by upper limit of range – lower limit of range
2. mean normalisation i.e. $\frac{data - mean(data)}{std(data)}$

GRADIENT DESCENT AND ALPHA

Making sure gradient descent is working correctly.



→ Example automatic convergence test:

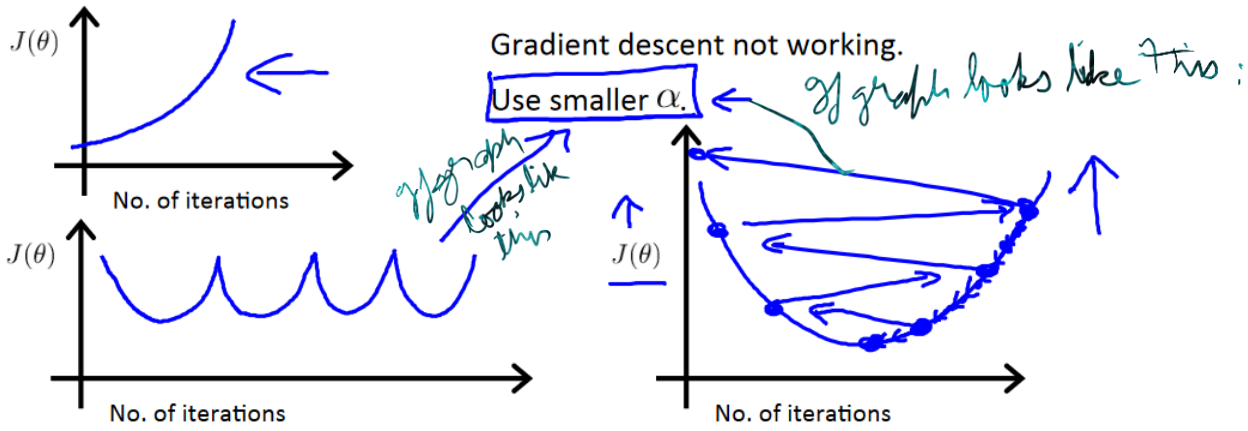
→ Declare convergence if $J(\theta)$ decreases by less than 10^{-3} in one iteration.

If it is not always easy to declare such a threshold value.

ε

Andrew Ng

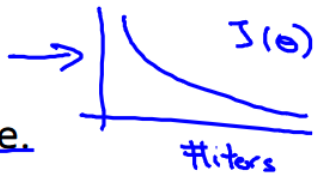
Making sure gradient descent is working correctly.



- For sufficiently small α, $J(\theta)$ should decrease on every iteration.
- But if α is too small, gradient descent can be slow to converge.

Andrew Ng

Summary:



- If α is too small: slow convergence.
- If α is too large: $J(\theta)$ may not decrease on every iteration; may not converge. (Slow converge also possible)

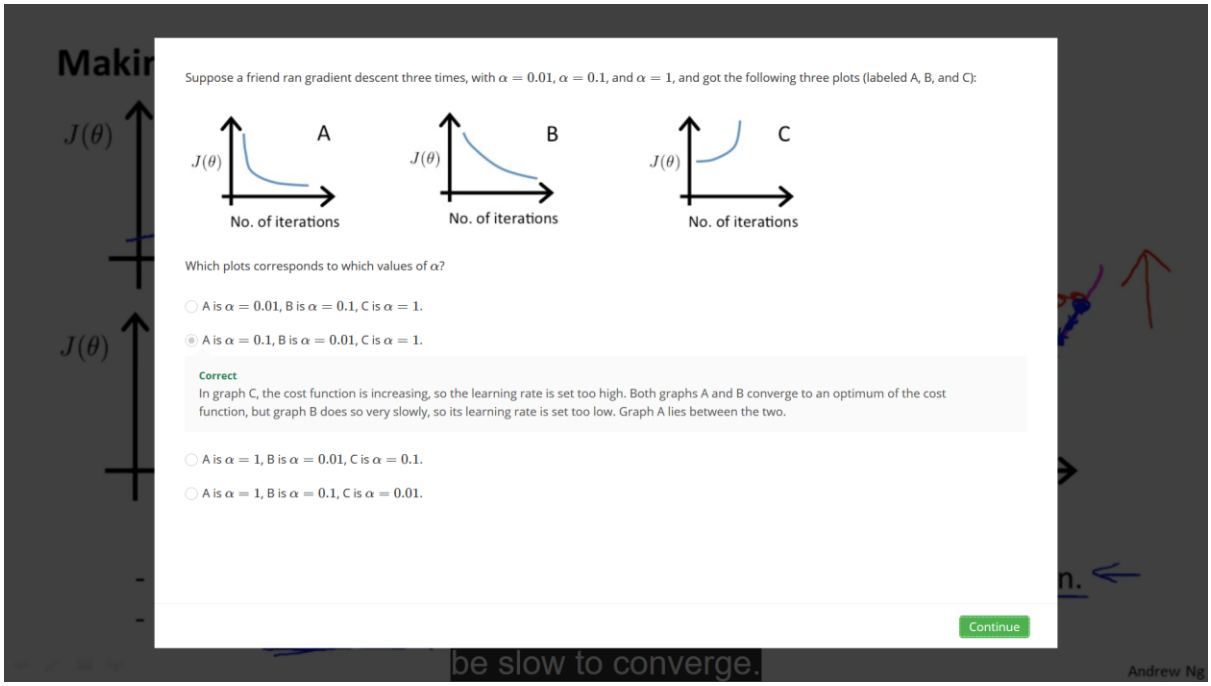
To choose α, try

$$\ldots, 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1, \ldots$$

Andrew Ng

Above line means we can first assume alpha to be 0.001 and then increase alpha in multiples of 3 and see the no. of iterations.

Question:



POLYNOMIAL REGRESSION:

Sometimes fitting a straight line through the data may not help, we may need to fit a line that corresponds to degree 2 , degree 3 , degree ½ ,etc . equation .

Features and Polynomial Regression

We can improve our features and the form of our hypothesis function in a couple different ways.

We can **combine** multiple features into one. For example, we can combine x_1 and x_2 into a new feature x_3 by taking $x_1 \cdot x_2$.

Polynomial Regression

Our hypothesis function need not be linear (a straight line) if that does not fit the data well.

We can **change the behavior or curve** of our hypothesis function by making it a quadratic, cubic or square root function (or any other form).

For example, if our hypothesis function is $h_\theta(x) = \theta_0 + \theta_1 x_1$ then we can create additional features based on x_1, to get the quadratic function $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2$ or the cubic function $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^3$

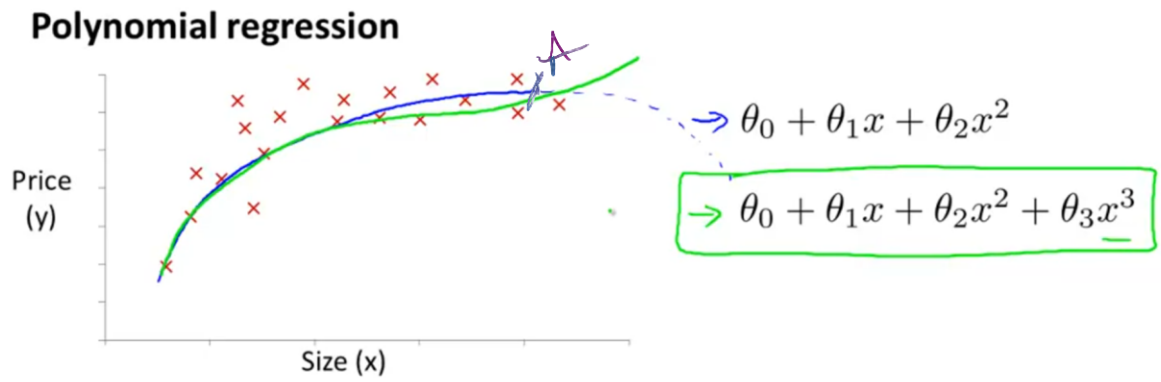
In the cubic version, we have created new features x_2 and x_3 where $x_2 = x_1^2$ and $x_3 = x_1^3$.

To make it a square root function, we could do: $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 \sqrt{x_1}$

One important thing to keep in mind is, if you choose your features this way then feature scaling becomes very important.

eg. if x_1 has range 1 - 1000 then range of x_1^2 becomes 1 - 1000000 and that of x_1^3 becomes 1 - 1000000000

As an example:

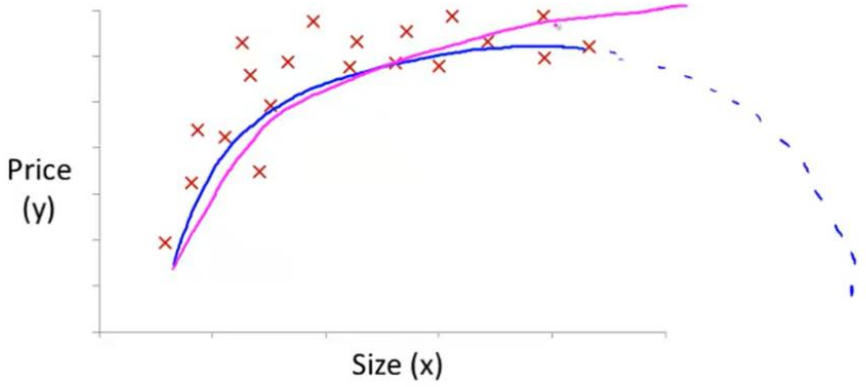


Andrew Ng

We first fit a degree 2 equation line through the data(represented by blue) , but this curve starts going down from point A and hence it is wrong.

We then fit a degree 3 equation line through the data (represented by green)

Choice of features



$$h_\theta(x) = \theta_0 + \theta_1(size) + \theta_2(size)^2$$

$$h_\theta(x) = \theta_0 + \theta_1(size) + \theta_2\sqrt{(size)}$$

We can also fit a degree ½ equation (represented by pink).

Normal Equation:

Gradient descent gives one way of minimizing J. Let's discuss a second way of doing so, this time performing the minimization explicitly and without resorting to an iterative algorithm. In the "Normal Equation" method, we will minimize J by explicitly taking its derivatives with respect to the θj 's, and setting them to zero. This allows us to find the optimum theta without iteration. The normal equation formula is given below:

$$\theta = (X^T X)^{-1} X^T y$$

Examples: $m = 4$.

x_0	Size (feet²) x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price ($1000) y
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix} \qquad y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}$$

m x (n+1)

m-dimensional vector

$$\theta = (X^T X)^{-1} X^T y$$

There is **no need** to do feature scaling with the normal equation.

Note: X^T is a (n+1) * m matrix , while X is a m * (n+1) matrix. Thus X^T*X is a (n+1)*(n+1) matrix. . Thus $(X^T*X)^{-1}$ is also a (n+1)*(n+1) matrix. $(X^T*X)^{-1} * X^T$ is a vector of size (n+1)*m . y is a vector of size m or

a m*1 matrix and hence ($(X^T*X)^{-1} * X^T$)* y is a vector of size (n+1)*1, which also happens to be the dimensions of theta. Where $\theta = (\theta_0, \theta_1, \theta_2 \ldots \theta_n)$.

From the above formula we obtain values of all respective thetas($\theta_0, \theta_1, \theta_2 \ldots \theta_n$) for which the cost function is minimum.

The following is a comparison of gradient descent and the normal equation:

Gradient Descent	Normal Equation
Need to choose alpha	No need to choose alpha
Needs many iterations	No need to iterate
O (kn^2)	O (n^3), need to calculate inverse of $X^T X$
Works well when n is large	Slow if n is very large

With the normal equation, computing the inversion has complexity $\mathcal{O}(n^3)$. So if we have a very large number of features, the normal equation will be slow. In practice, when n exceeds 10,000 it might be a good time to go from a normal solution to an iterative process.

The normal equation gets slow with many samples due to the reason that it takes time for performing computation of matrix related operations for large values of n.

Normal Equation Noninvertibility

When implementing the normal equation in octave we want to use the 'pinv' function rather than 'inv.' The 'pinv' function will give you a value of θ even if $X^T X$ is not invertible.

If $X^T X$ is **noninvertible,** the common causes might be having :

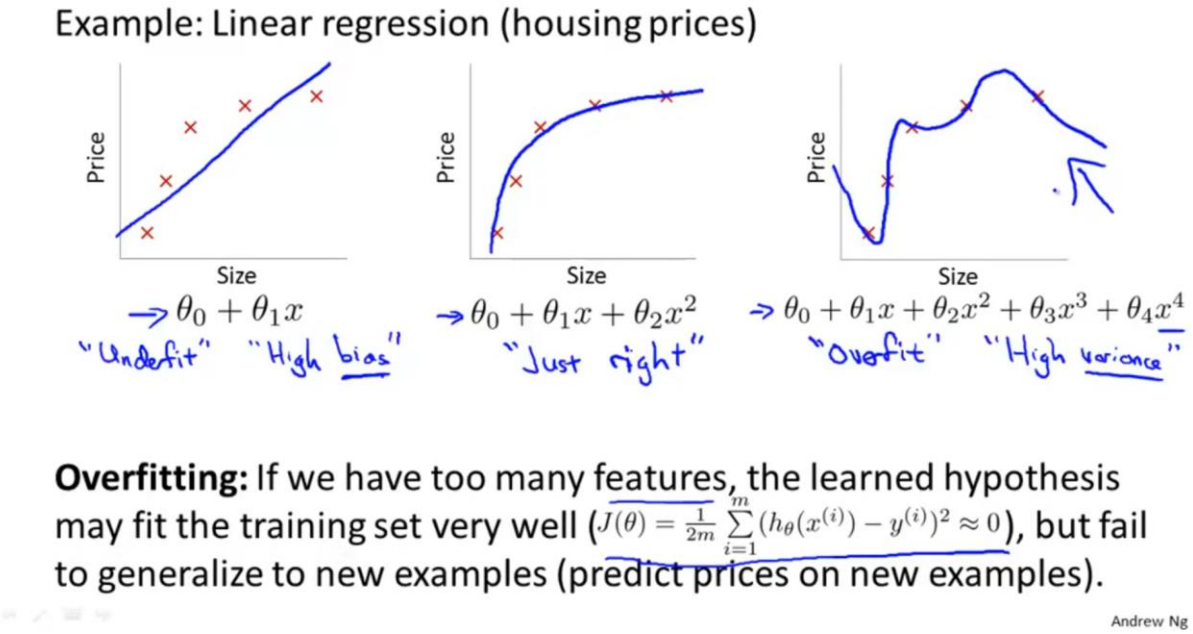
- Redundant features, where two features are very closely related (i.e. they are linearly dependent)
- Too many features (e.g. m ≤ n). In this case, delete some features or use "regularization" (to be explained in a later lesson).

Solutions to the above problems include deleting a feature that is linearly dependent with another or deleting one or more features when there are too many features.

What if $X^T X$ is non-invertible?

- Redundant features (linearly dependent).
  E.g. x_1 = size in feet^2
  x_2 = size in m^2
  $1m = 3.28$ feet
  $x_1 = (3.28)^2 x_2$
- Too many features (e.g. $m \leq n$).
  $\rightarrow m = 10$
  $\rightarrow n = 100$
  $\theta \in \mathbb{R}^{101}$
  - Delete some features, or use regularization.
    ↓ later

Overfitting:

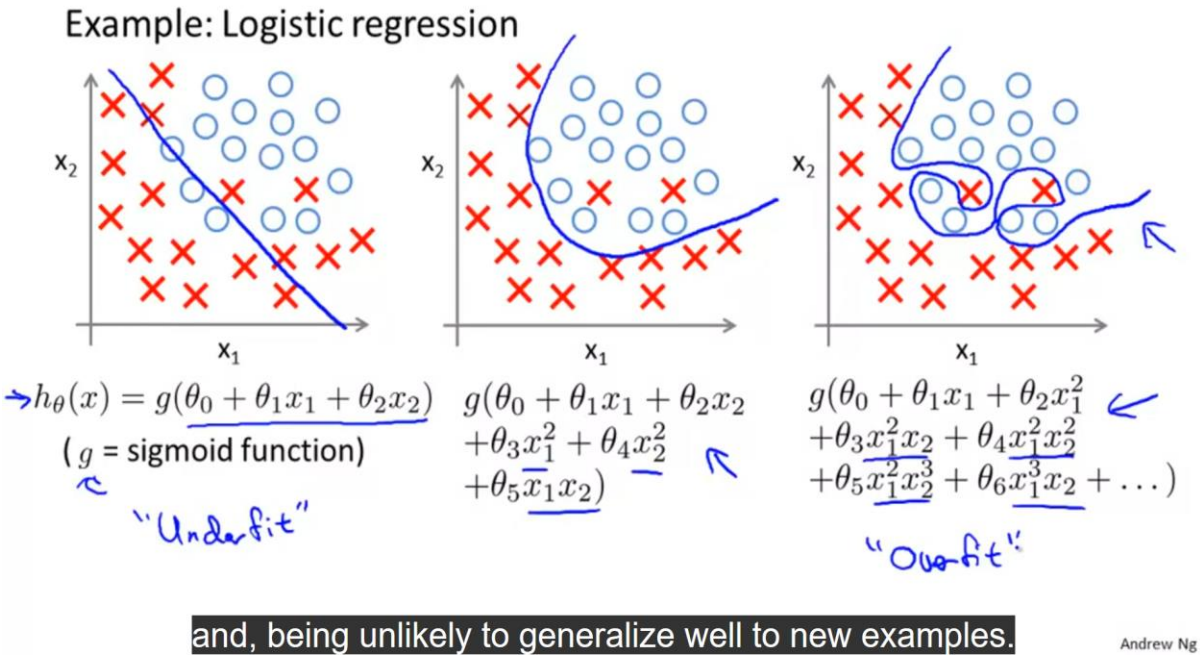


Consider the problem of predicting y from x ∈ R. The leftmost figure below shows the result of fitting a $y = \theta 0 + \theta 1 x$ to a dataset. We see that the data doesn't really lie on straight line, and so the fit is not very good.

Instead, if we had added an extra feature x^2, and fit $y=\theta_0+\theta_1 x+\theta_2 x^2$,then we obtain a slightly better fit to the data (See middle figure). Naively, it might seem that the more features we add, the better. However, there is also a danger in adding too many features: The rightmost figure is the result of fitting a 5^{th} order polynomial $y=\sum_{j=0}^{5} \theta_j x^j$. We see that even though the fitted curve passes through the data perfectly, we would not expect this to be a very good predictor of, say, housing prices (y) for different living areas (x). Without formally defining what these terms mean, we'll say the figure on the left shows an instance of **underfitting**—in which the data clearly shows structure not captured by the model—and the figure on the right is an example of **overfitting**.

Underfitting, or high bias, is when the form of our hypothesis function h maps poorly to the trend of the data. It is usually caused by a function that is too simple or uses too few features. At the other extreme, overfitting, or high variance, is caused by a hypothesis function that fits the available data but does not generalize well to predict new data. It is usually caused by a complicated function that creates a lot of unnecessary curves and angles unrelated to the data.

Above terminology is applied to both linear and logistic regression



There are two main options to address the issue of overfitting:

1) Reduce the number of features:

- Manually select which features to keep.
- Use a model selection algorithm (studied later in the course).

2) Regularization

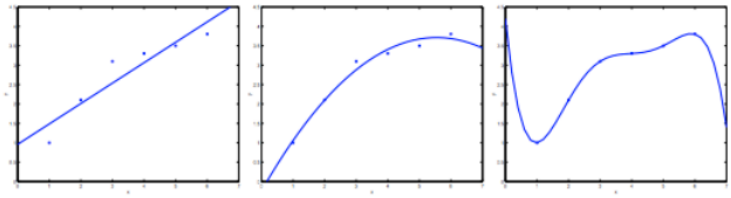
- Keep all the features, but reduce the magnitude of parameters θ_j.
- Regularization works well when we have a lot of slightly useful features.

Regularisation:

When our model is underfitting the data i.e. not able to predict the trend there in the training data properly, we may think of adding more parameters in it to increase the accuracy of prediction. Although this may seem to solve the problem , we have to realise that this can lead to overfitting that is , our model can adjusts itself according to the loopholes there in the training data and hence predicts the training data very accurately , but can't do the same on new data. In the example given as we introduce x^3 and x^4 term in our equation y= $theta_0$ + $theta1_1$ *x +$theta_2x^2$, though the predictions become more accurate , the problem of overfitting also arises.

The Problem of Overfitting

Consider the problem of predicting y from x ∈ R. The leftmost figure below shows the result of fitting a $y = \theta_0 + \theta_1 x$ to a dataset. We see that the data doesn't really lie on straight line, and so the fit is not very good.



Instead, if we had added an extra feature x^2 , and fit $y = \theta_0 + \theta_1 x + \theta_2 x^2$, then we obtain a slightly better fit to the data (See middle figure). Naively, it might seem that the more features we add, the better. However, there is also a danger in adding too many features: The rightmost figure is the result of fitting a 5^{th} order polynomial $y = \sum_{j=0}^{5} \theta_j x^j$. We see that even though the fitted curve passes through the data perfectly, we would not expect this to be a very good predictor of, say, housing prices (y) for different living areas (x). Without formally defining what these terms mean, we'll say the figure on the left shows an instance of **underfitting**—in which the data clearly shows structure not captured by the model—and the figure on the right is an example of **overfitting**.

Underfitting, or high bias, is when the form of our hypothesis function h maps poorly to the trend of the data. It is usually caused by a function that is too simple or uses too few features. At the other extreme, overfitting, or high variance, is caused by a hypothesis function that fits the available data but does not generalize well to predict new data. It is usually caused by a complicated function that creates a lot of unnecessary curves and angles unrelated to the data.

So what we do is that we introduce a regularisation term in our cost function equation. What this term does is that 'keeps a track' on the new features we add in the equation for better fitting of the model. The way it keeps a track is by increasing the value of the cost function if the theta values corresponding to the features/regressors is large. As we want our cost function to be less, the values of theta corresponding to regressors are kept balanced. This also prevents assignment of unnecessary weight to any of the regressors.

##Some screenshots from an article on regularisation:

Find the article here: https://towardsdatascience.com/regularization-in-machine-learning-76441ddcf99a

Regularization

This is a form of regression, that constrains/ regularizes or shrinks the coefficient estimates towards zero. In other words, ***this technique discourages learning a more complex or flexible model, so as to avoid the risk of overfitting.***

A simple relation for linear regression looks like this. Here Y represents the learned relation and *β represents the coefficient estimates for different variables or predictors(X).*

$$Y \approx \beta 0 + \beta 1X1 + \beta 2X2 + \ldots + \beta pXp$$

The fitting procedure involves a loss function, known as residual sum of squares or RSS. The coefficients are chosen, such that they minimize this loss function.

$$\mathrm{RSS} = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 .$$

Now, this will adjust the coefficients based on your training data. *If there is noise in the training data, then the estimated coefficients won't generalize well to the future data. This is where regularization comes in and shrinks or regularizes these learned estimates towards zero.*

Ridge Regression

$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{p}\beta_j^2=\mathrm{RSS}+\lambda\sum_{j=1}^{p}\beta_j^2$$

Above image shows ridge regression, where the ***RSS is modified by adding the shrinkage quantity.*** Now, the coefficients are estimated by minimizing this function. Here, ***λ is the tuning parameter that decides how much we want to penalize the flexibility of our model.*** The increase in flexibility of a model is represented by increase in its coefficients, and if we want to minimize the above function, then these coefficients need to be small. This is how the Ridge regression technique prevents coefficients from rising too high. Also, notice that we shrink the estimated association of each variable with the response, except the intercept β0, This intercept is a measure of the mean value of the response when xi1 = xi2 = …= xip = 0.

When λ = 0, the penalty term has no effect, and the estimates produced by ridge regression will be equal to least squares. However, ***as λ→∞, the impact of the shrinkage penalty grows, and the ridge regression coefficient estimates will approach zero***. As can be seen, selecting a good value of λ is critical. Cross validation comes in handy for this purpose. The coefficient estimates produced by this method are ***also known as the L2 norm***.

The coefficients that are produced by the standard least squares method are scale equivariant, i.e. if we multiply each input by c then the corresponding coefficients are scaled by a factor of 1/c. Therefore, regardless of how the predictor is scaled, the multiplication of predictor and coefficient(Xjβj) remains the same. ***However, this is not the case with ridge regression, and therefore, we need to standardize the predictors or bring the predictors to the same scale before performing ridge regression***. The formula used to do this is given below.

$$\tilde{x}_{ij} = \frac{x_{ij}}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_{ij} - \overline{x}_j)^2}},$$

NOTES FROM COURSERA:

If we have overfitting from our hypothesis function, we can reduce the weight that some of the terms in our function carry by increasing their cost.

Say we wanted to make the following function more quadratic:

$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

We'll want to eliminate the influence of $\theta_3 x^3$ and $\theta_4 x^4$. Without actually getting rid of these features or changing the form of our hypothesis, we can instead modify our **cost function**:

$min_\theta \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + 1000 \cdot \theta_3^2 + 1000 \cdot \theta_4^2$

We've added two extra terms at the end to inflate the cost of θ_3 and θ_4. Now, in order for the cost function to get close to zero, we will have to reduce the values of θ_3 and θ_4 to near zero. This will in turn greatly reduce the values of $\theta_3 x^3$ and $\theta_4 x^4$ in our hypothesis function. As a result, we see that the new hypothesis (depicted by the pink curve) looks like a quadratic function but fits the data better due to the extra small terms $\theta_3 x^3$ and $\theta_4 x^4$.

If we have overfitting from our hypothesis function, we can reduce the weight that some of the terms in our function carry by increasing their cost.

Say we wanted to make the following function more quadratic:

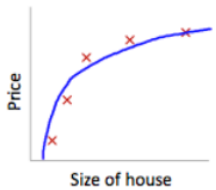
$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

We'll want to eliminate the influence of $\theta_3 x^3$ and $\theta_4 x^4$. Without actually getting rid of these features or changing the form of our hypothesis, we can instead modify our **cost function**:

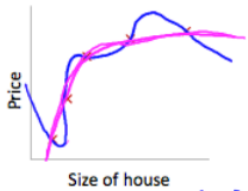
$min_\theta \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + 1000 \cdot \theta_3^2 + 1000 \cdot \theta_4^2$

We've added two extra terms at the end to inflate the cost of θ_3 and θ_4. Now, in order for the cost function to get close to zero, we will have to reduce the values of θ_3 and θ_4 to near zero. This will in turn greatly reduce the values of $\theta_3 x^3$ and $\theta_4 x^4$ in our hypothesis function. As a result, we see that the new hypothesis (depicted by the pink curve) looks like a quadratic function but fits the data better due to the extra small terms $\theta_3 x^3$ and $\theta_4 x^4$.

Intuition



$\theta_0 + \theta_1 x + \theta_2 x^2$



$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

Suppose we penalize and make θ_3, θ_4 really small.

$$\min_\theta \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + 1000\,\theta_3^2 + 1000\,\theta_4^2$$

$\theta_3 \approx 0 \quad \theta_4 \approx 0$

We could also regularize all of our theta parameters in a single summation as:

$$min_\theta \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} \theta_j^2$$

The λ, or lambda, is the **regularization parameter**. It determines how much the costs of our theta parameters are inflated.

Using the above cost function with the extra summation, we can smooth the output of our hypothesis function to reduce overfitting. If lambda is chosen to be too large, it may smooth out the function too much and cause underfitting. Hence, what would happen if $\lambda = 0$ or is too small ?

When lambda is too large we obtain a trend line parallel to the x axis. This is because to minimize the cost function $theta_1$,$theta_2$….$theta_n$ will assume values that tend to 0 and hence our model will become y= $theta_0$ + 0 , which explains the horizontal line parallel to x axis.

In regularized linear regression, we choose θ to minimize

$$J(\theta) = \frac{1}{2m} \left[\sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} \theta_j^2 \right]$$

What if λ is set to an extremely large value (perhaps for too large for our problem, say $\lambda = 10^{10}$)?

$\theta_1, \theta_2, \theta_3, \theta_4$

$\theta_1 \approx 0 \quad \theta_2 \approx 0$

$\theta_3 \approx 0 \quad \theta_4 \approx 0$

$h_\theta(x) = \theta_0$

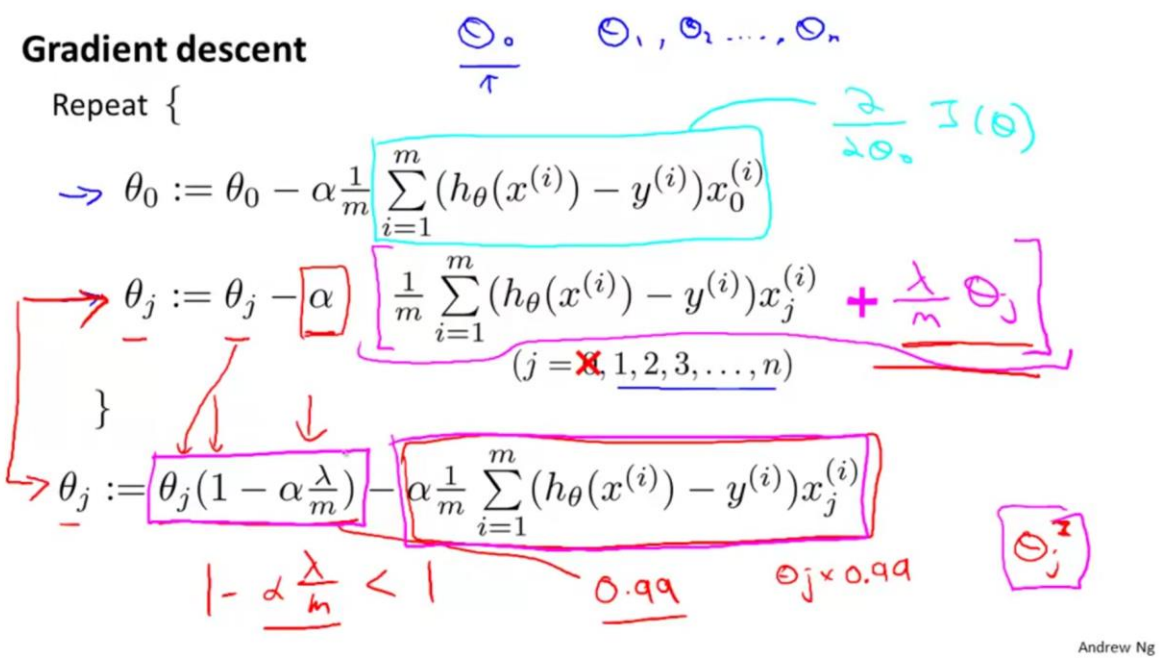
$h_\theta(x)$

$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

Andrew Ng

Regularisation and linear regression:

How the formula changes on adding the regularisation term:



Gradient Descent

We will modify our gradient descent function to separate out θ_0 from the rest of the parameters because we do not want to penalize θ_0.

Repeat {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} + \frac{\lambda}{m} \theta_j \qquad j \in \{1, 2 \ldots n\}$$

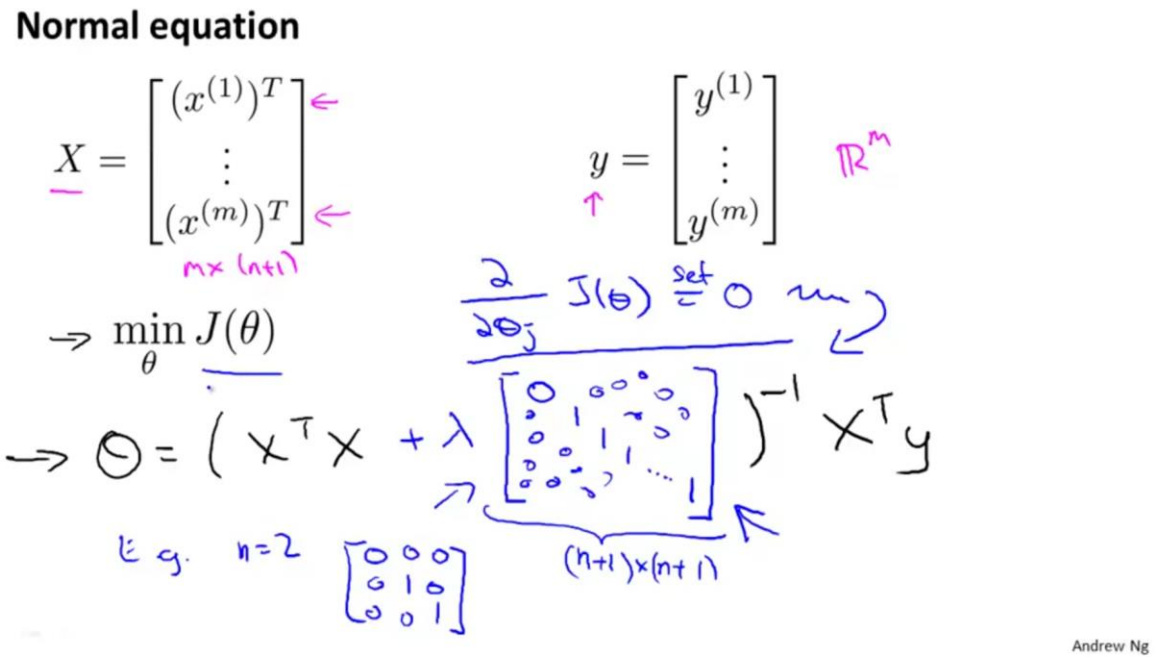
}

The term $\frac{\lambda}{m}\theta_j$ performs our regularization. With some manipulation our update rule can also be represented as:

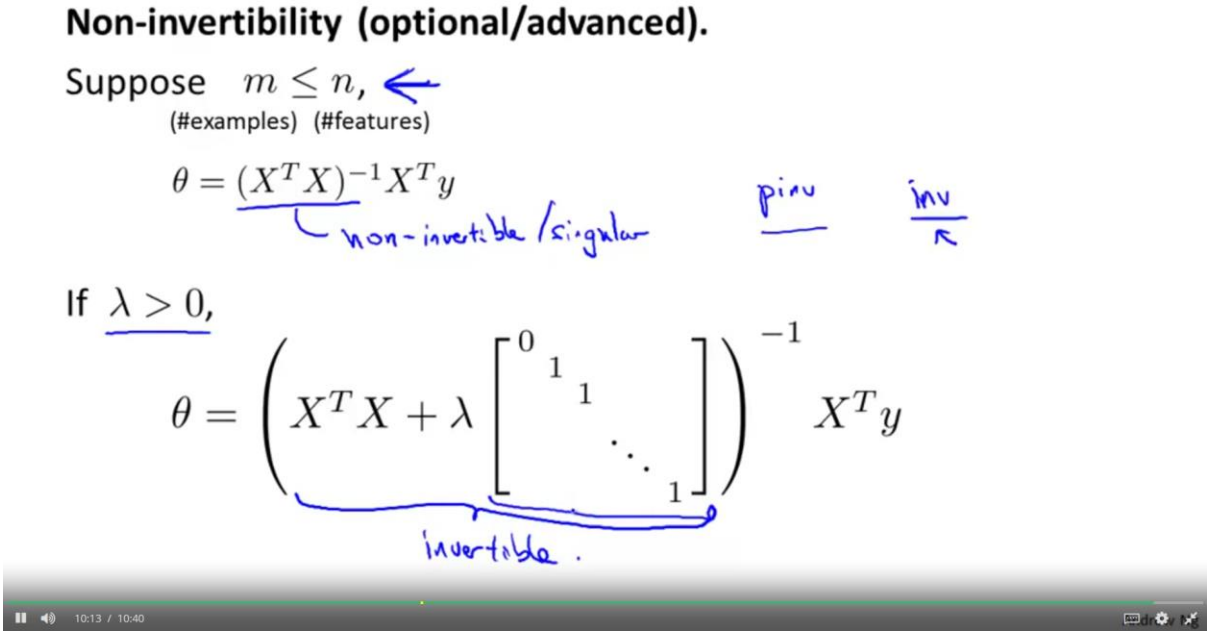
$\theta_j := \theta_j(1 - \alpha\frac{\lambda}{m}) - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$

The first term in the above equation, $1 - \alpha\frac{\lambda}{m}$ will always be less than 1. Intuitively you can see it as reducing the value of θ_j by some amount on every update. Notice that the second term is now exactly the same as it was before.

How the normal equation formula changes:



Note: If m<=n , then the matrix (X^TX) becomes non – invertable. The regularisation term lamba*the matrix shown in the figure below make this (X^TX) invertable even when m<=n. So adding this regularisation has one more advantage other than regularisation itself.



Coursera notes:

Normal Equation

Now let's approach regularization using the alternate method of the non-iterative normal equation.

To add in regularization, the equation is the same as our original, except that we add another term inside the parentheses:

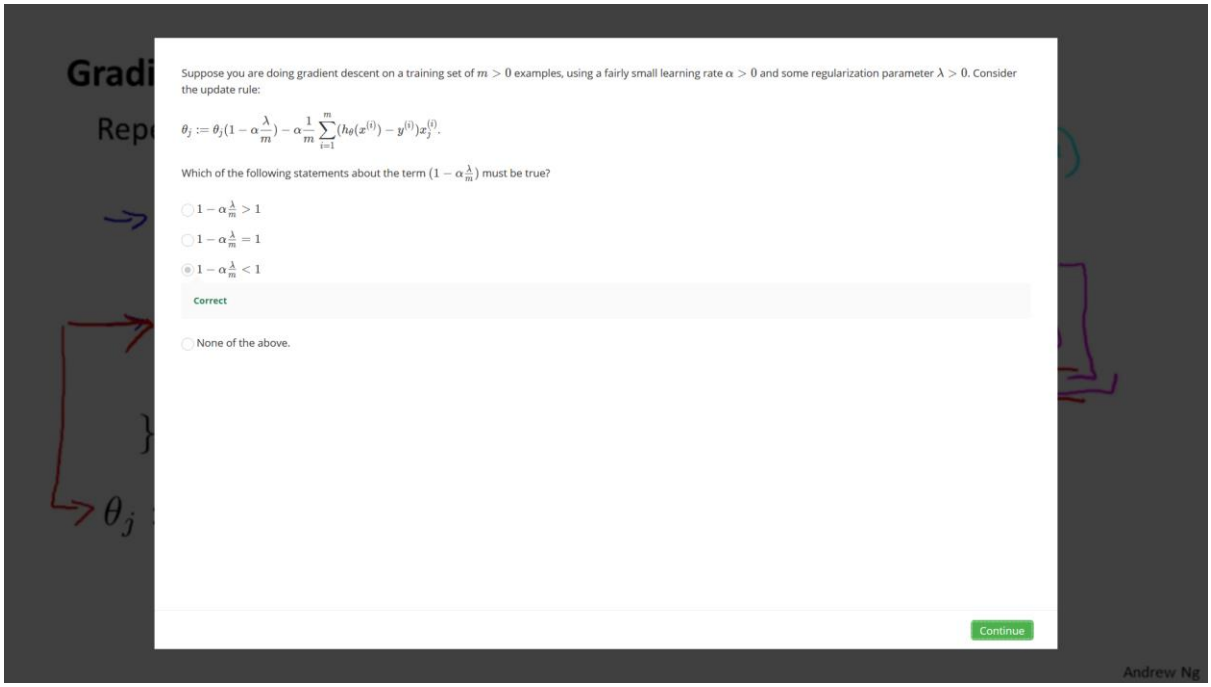
$$\theta = X^T X + \lambda \cdot L^{-1} X^T y$$

$$\text{where } L = \begin{bmatrix} 0 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{bmatrix}$$

L is a matrix with 0 at the top left and 1's down the diagonal, with 0's everywhere else. It should have dimension (n+1)×(n+1). Intuitively, this is the identity matrix (though we are not including x_0), multiplied with a single real number λ.

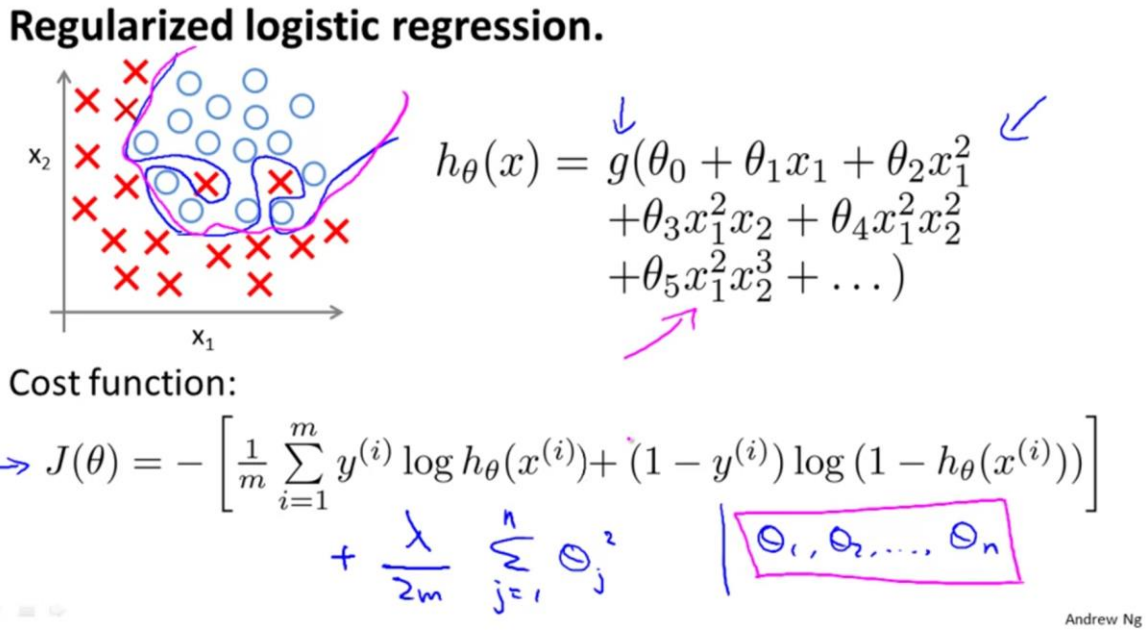
Recall that if m < n, then $X^T X$ is non-invertible. However, when we add the term λ·L, then $X^T X$ + λ·L becomes invertible.

Question:

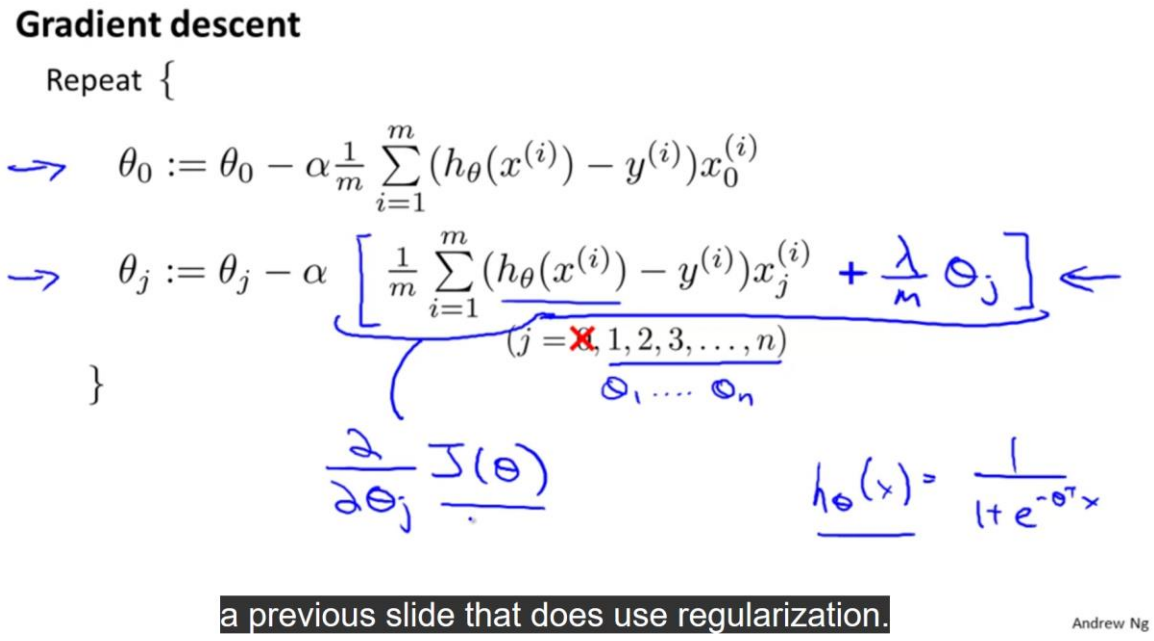


Regularisation and logistic regression:

Even when we fit high order terms like $x_1^2 * x_2^3$ in our equation , we don't overfit the model since due to regularisation , the coefficients like $theta_5$ corresponding to these terms decrease their magnitude and hence adjust their effect. The blue line shows an instance of overfitting and the line in pink is the regularized equation.



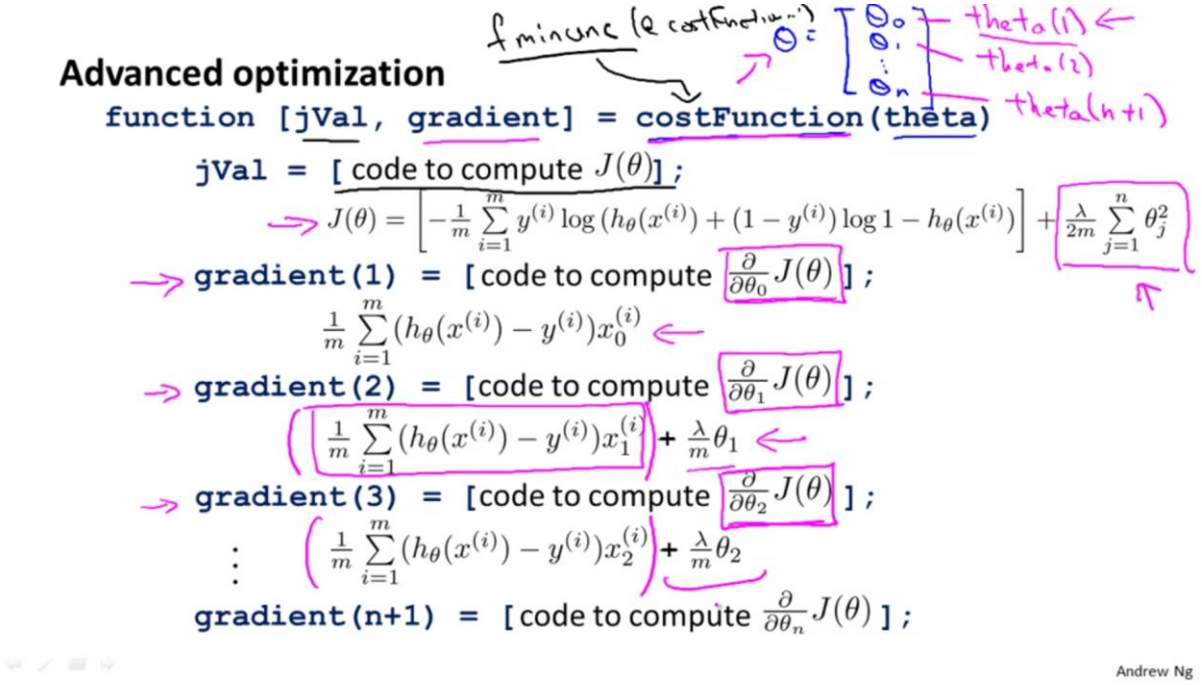
Changes in the formula :



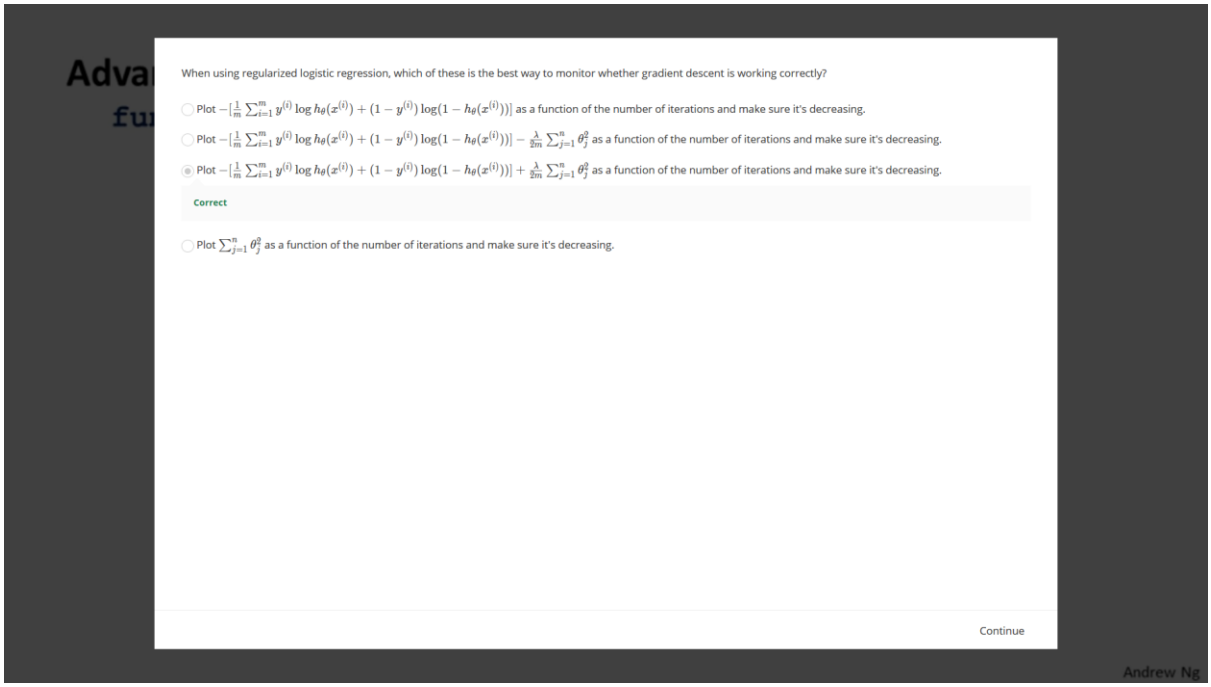
a previous slide that does use regularization.

Andrew Ng

The fminunc function and regularisation in octave:



Question:



Quiz 1

TOTAL POINTS 5

1. Which of the following are components in building a machine learning algorithm? 1 point

- Statistical inference
- Machine learning
- Artificial intelligence
- Training and test sets
- ollecting data to answer the question.

Quiz 1

Graded Quiz • 10 min

Due Mar 23, 12:29 PM IST

Collecting data to answer the question.

2. Suppose we build a prediction algorithm on a data set and it is 100% accurate on that data set. Why might the algorithm not work well if we collect a new data set? 1 point

- We have used neural networks which has notoriously bad performance.
- We may be using bad variables that don't explain the outcome.v
- We have too few predictors to get good out of sample accuracy.
- Our algorithm may be overfitting the training data, predicting both the signal and the noise.

3. What are typical sizes for the training and test sets? 1 point

Quiz 1
Graded Quiz • 10 min

Due Mar 23, 12:29 PM IST

3. What are typical sizes for the training and test sets? **1 / 1 point**

- (•) 60% in the training set, 40% in the testing set.
- ( ) 90% training set, 10% test set
- ( ) 20% training set, 80% test set.
- ( ) 100% training set, 0% test set.

✓ **Correct**

4. What are some common error rates for predicting binary variables (i.e. variables with two possible values like yes/no, disease/normal, clicked/didn't click)? Check the correct answer(s). **1 / 1 point**

Quiz 1
Graded Quiz • 10 min

Due Mar 23, 12:29 PM IST

✓ **Correct**

4. What are some common error rates for predicting binary variables (i.e. variables with two possible values like yes/no, disease/normal, clicked/didn't click)? Check the correct answer(s). **1 / 1 point**

- ( ) Median absolute deviation
- ( ) R^2
- ( ) Root mean squared error
- ( ) Correlation
- (•) Predictive value of a positive

✓ **Correct**

Correct

5. Suppose that we have created a machine learning algorithm that predicts whether a link will be clicked with 99% sensitivity and 99% specificity. The rate the link is clicked is 1/1000 of visits to a website. If we predict the link will be clicked on a specific visit, what is the probability it will actually be clicked? **1 / 1 point**

- 99.9%
- 50%
- 99%
- 9% (selected)

Correct