Week 1 Notes

Video 1 – Model Representation

Notation

1. M = # of training examples
2. X’s = input to the model
3. Y’s = output of predicted value
4. (x,y) = 1 training example
5. = ith training example

Example 1

|  |  |
| --- | --- |
| **Size in ft^2 (x)** | **Price in 000’s (y)** |
| 2104 | 460 |
| 1416 | 252 |
| 1534 | 315 |
| 852 | 178 |
| … | … |

1. Here m=47

What is a hypothesis (h)?

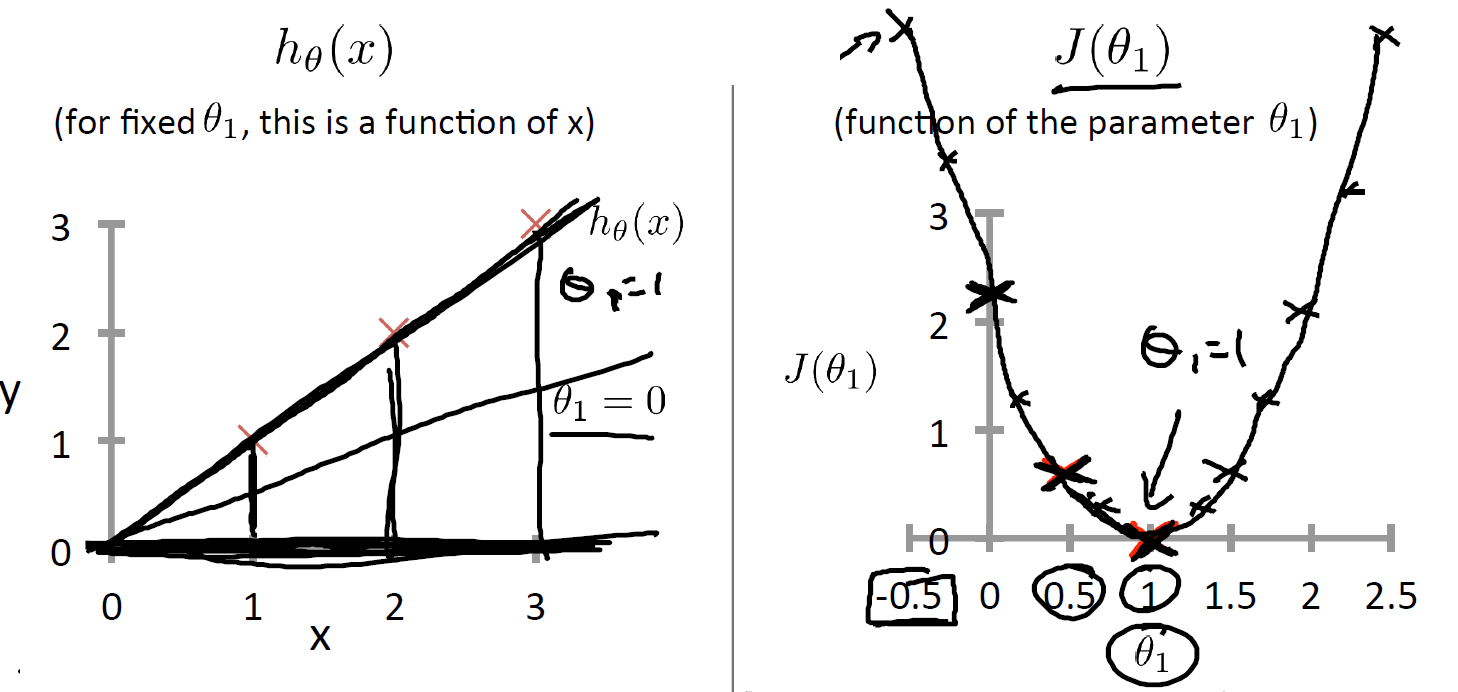
1. It is a function that maps x’s to y’s
2. How would the hypothesis be represented in Example 1?

Video 2 – Cost Function

1. We have where are the parameters
2. How do we choose ?
   1. **Idea:** Choose so h(x) is close to y from the training set
   2. **Formalization:**
      1. We want
      2. By convention we define the cost function to be
      3. Goal : We want to minimize
   3. **Note:** This cost function is known as the squared error function

Video 3 – Cost Function Intuition I

1. We will start off with the simple case by letting
2. We now get => =
3. Now we compare our hypothesis to our cost function
4. Running through an example suppose that
   1. Below are diagrams of plots of both the hypothesis and cost function



1. Calculating the cost function for the example above

Video 4 – Cost Function Inution II

1. Here just showed different contour plots of

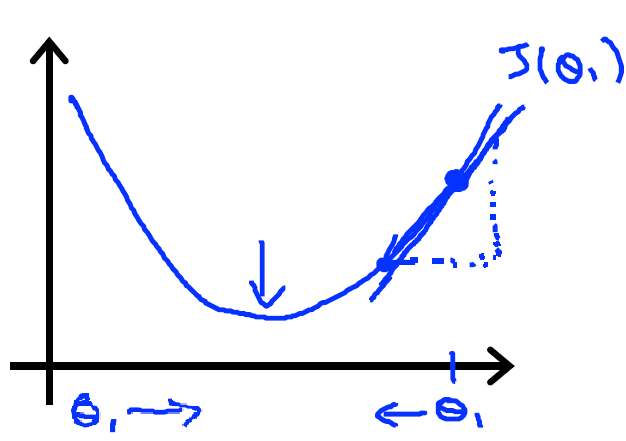
Video 5 – Gradient Descent

1. We have a function and we want to
2. How do we minimize the cost function?
   1. Outline
      1. Initialize to some arbitrary values
      2. Travel to get to a local minimum
3. Gradient Descent Algorithm
   * 1. Repeat until convergence {
     3. }
     4. **Note 1**:
4. Gradient Descent requires simultaneous updating
   1. Simultaneous Updating
      1. Temp0:=
      2. Temp1:=
      3. Temp0
      4. Temp1
   2. Note 1: and need to be updated after Temp0 and Temp1
   3. Note 2: Doing the update any other way is wrong

Video 6 - Gradient Descent Intuition

1. Here we are going to try to understand what Gradient Descent would on

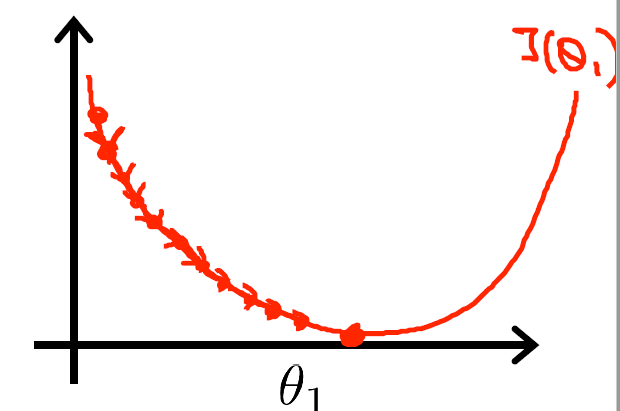
We start the initialization here.

****

Here moves left if the update has a positive value and moves right if update has a negative value.

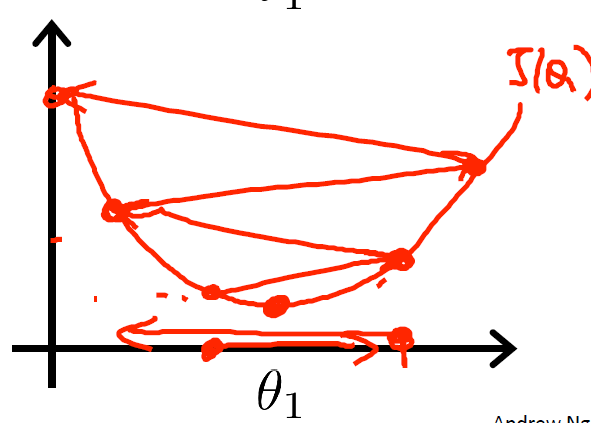
What about the learning rate?

1. If is too small gradient descent case be slow. (Figure 1)



**Figure 1**

1. If is too big, gradient descent fail to converge (Figure 2)



**Figure 2**

1. Tricky if is already at a local minimum so the partial deritivative=0 so in the update we get
2. Note – As we approach a local minimum, gradient descent will automatically take smaller steps so no need to decrease as it takes it steps

Video 7 – Gradient Descent for Linear Regression

1. Recall that the gradient descent algorithm is
   * 1. Repeat until convergence {
     3. }
2. How to derive
3. This cost function is a “convex function” so it always has a global minimum

Week 2 – Linear Regression in Multiple Variables

Videos 1-2- Multiple Features

1. New Notations
   1. n = # of features
   2. = input features of the ith training example
   3. = value of feature j in the ith training example
2. Example going over the new notation. Suppose we got the dataset below

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Size** | **# of Rooms** | **# of Floors** | **Age of Home** | **Price (y)** |
| 2104 | 5 | 1 | 45 | 460 |
| 1416 | 3 | 2 | 40 | 232 |
| 1534 | 3 | 2 | 30 | 315 |
| 852 | 2 | 1 | 36 | 178 |
| … | … | … | … | … |

* 1. Here n=4

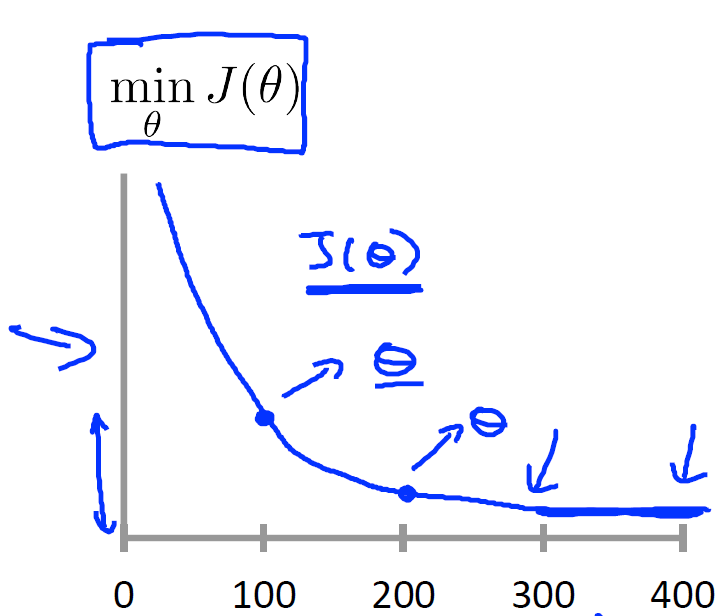
1. How does the hypothesis look like now?
   1. and define
2. Gradient Descent for Multiple Linear Regression follows the same algorithm as simple LR
3. For a mental check remember

Video 3 – Feature Scaling

1. In practice we use feature scaling when we are using gradient descent
2. Ideally every feature should be between -1 and 1, however it’s fine to have them between -3 and 3
3. Try to avoid small ranges such as -0.0001 and 0.0001
4. Mean Normalization
   1. Another scaling where is replaced with and we don’t apply it to out of convention
   2. Note that is the standard deviation of column i

Video 4 – Learning Rate

1. Here we cover two things
   1. How to debug gradient descent to make sure its working right?
   2. How to choose the learning rate **α**
2. How to make sure gradient descent is working right?
   1. We plot the cost function vs # of iteration and you should notice the plot descrease overtime as below



1. How to choose the right learning rate **α?**
   1. For a sufficiently small **α,** the cost function should decrease on every iteration
   2. If **α** is too small, gradient descent can be slow to converge
   3. If **α** is too large then the cost function may not decrease on every iteration so it wont converge.
   4. For **α** we can try it in 3 fold increments as such below
      1. **α=**0.001
      2. **α=**0.003
      3. **α=**0.01
      4. **α=**0.03
      5. **…**
      6. **α=**0.3

Video 5 – Normal Equations

1. In matrix form we have and
2. Using normality we have
3. What is the intuition behind this ?
4. In octave in order to calculate use pinv(X’ \* X)\*X’\*Y and feature scaling is not needed
5. Gradient Descent vs Normal Eqs

|  |  |
| --- | --- |
| Gradient Descent | Normality |
| Need to choose **α** | No need to choose **α** |
| Needs many iterations | Only one iteration |
| Works well with large n | Need to compute an inverse |
| O(n^2) | O(n^3) |
|  | Slow If n is very large |

Video 6 – What is the normal equation is noninvertible?

1. What if is noninvertible?
   1. The dataset has redundant features such as size in ft and size in ft^2
   2. To many features (m<<n) so we have to delete some features or use regularization
2. Vectorization
   1. Important to vectorize as much as you can as it is more efficient then using loops

Assignment 1 – Linear Regression Notes

1. When normalizing features its important to store the values used for normalization which is the mean/sd.
2. Given a new **x** value we have to first normalize it before using the model to predict a value
3. The cost function for Gradient Descent can be written in vectorized form as shown below

where and

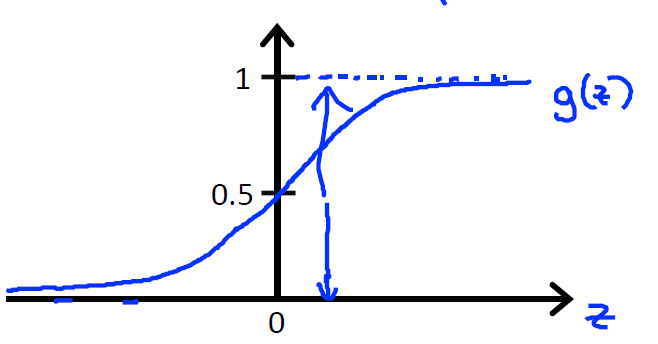
Week 3 – Logistic Regression

Video 1 – Classification

1. Example of Logistic Regression use
   1. Classifying email as Spam/ Not Spam
   2. Online Transactions as Fradulent or not
2. Description of target variable **y:**  where 0= negative class and 1=positive class
3. Threshold Classifier
   1. If
   2. If
   3. Threshold is at

Video 2- Hypothesis Representation

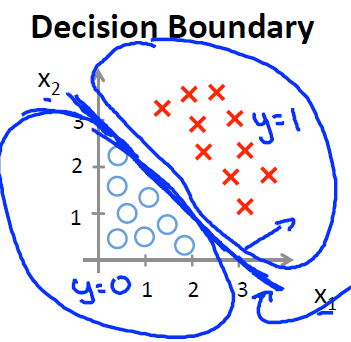
1. We want
2. Hypothesis representation: where
3. is known as the sigmoid function and a plot of it below



1. Hypothesis Interpretation
   1. estimated probability **y=1** on input **x**
   2. So 🡨 Conditional on the parameter theta
   3. Properties

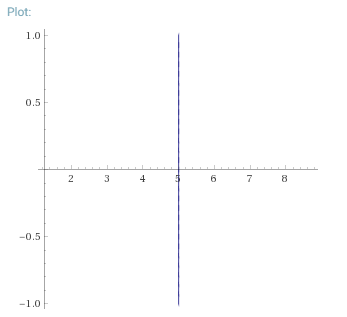
Video 3 – Decision Boundary

1. Below shows the decision boundary we will implement for logistic regression
2. Example 1
   1. Suppose that ) where
   2. It follows that we predict y=1 only If
   3. Thus is our decision boundary as plotted below



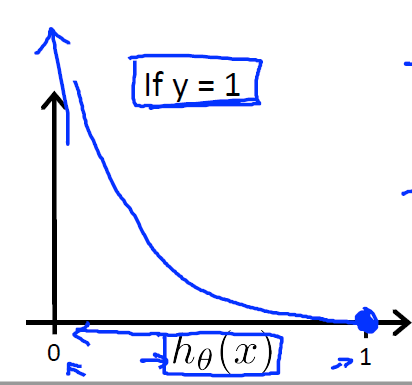
Quiz Question

1. Consider the logistic regression of 2 features x1 and x2 and so that . We want to plot the decision boundary of



Video 4-Cost Function

1. We have the following
   1. m examples where
2. How to choose parameters **θ?**
   1. If we define
      1. This function is non-convex so it wont work
   2. If we define then this is what we want
   3. Intution behind the cost function (b)



* + 1. Here we see if that the hypothesis exactly y=1 then the cost=0 but as

then cost🡪 infinity so we penalize the cost function very large if and vice versa for the case when y=0

Video 5- Simplified Cost Function and Gradient Descent

1. We write an equivalent but simplified cost function for Logistic regression as defined in 2b in the previous section.
2. Simplified:
3. Thus running the m examples the cost function becomes

1. Now we minimize via gradient descent
2. The gradient descent algorithm then becomes
   * 1. Repeat until convergence {
     3. }
     4. Simultaneous for all
3. What is the vectorized implementation for gradient descent algorithm?

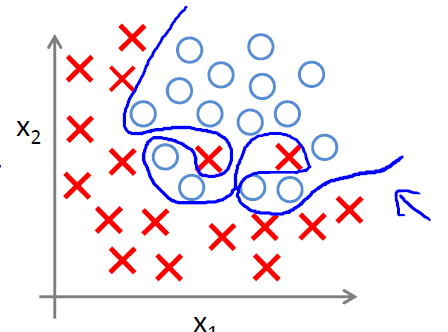
Video 6 – Advanced Optimization (Skip)

Video 7- Multi Class classification ( 1 vs All)

1. We want to fit n classifiers for n>2 classes so
2. For a new input **x** we pick the class **i** that maxes

Video 8 – Problem with Overfitting

1. Underfit – causes more bias (more on this later)
2. Overfitting – This causes higher variance and happens when
   1. We have too many features and failes to generalize to new examples
   2. Example



1. Addressing Overfitting
   1. Plot the hypothesis but we cant when there are lots of features
   2. If we have lots of features and small **m** (more columns then rows) then we have an overfitting issue
   3. Solutions
      1. Manually select what features to keep
      2. Model selection algorithm
      3. Regularization
         1. Keep all features but reduce magnitude/values of **θ**
         2. Works well when we have lots of features each of which contribute to predicting **y**

Video 9 – Cost function with regularization

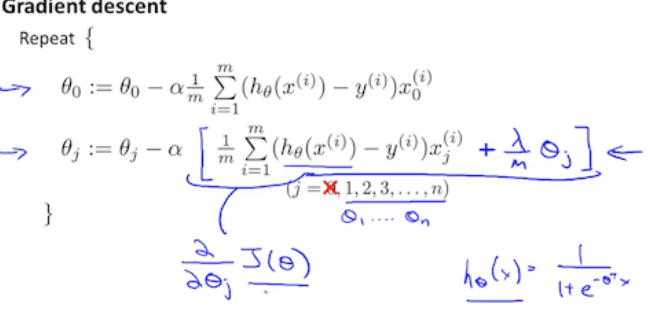
1. Smaller values for paramters for leads to two things
   1. Simple hypothesis by penalizing all parameters
   2. Less prone to overfitting
2. Modified Cost Function
3. Achieve 2 goals
   1. Fit the training dataset well
   2. Keep the parameters small
4. Regularization parameters balances goals a and b.
   1. If is set to a very high value then the model may underfit the data
   2. If is set to too small the model may overfit the data

Video 10 – Regularized Linear Regression

1. The modified gradient descent algorithm becomes
   * 1. Repeat until convergence {

     4. }
     5. Simultaneous for all
2. If we are using normal equations then
3. where the matrix is (n+1) x (n+1)
   1. This matrix is invertible if

Video 11 – Regularized Logistic Regression



1. Gradient descent for logistic regression is the same format as linear regression.

Assignment 2 Notes

1. How to vectorize the cost function?
   1. Note that
2. Fminc() function
   1. A function in octave that finds the minimum of an unconstrained function
   2. Here we want to optimize J(θ) with respect to θ
   3. Parameters
      1. In the function @ symbol prefixes the definition of an anonymous function
      2. In Octave anonymous functions are defined using the syntax @(arg-list) expression

Week 4- Neural Networks (NN)

Video 1- Non-Linear Hypothesis

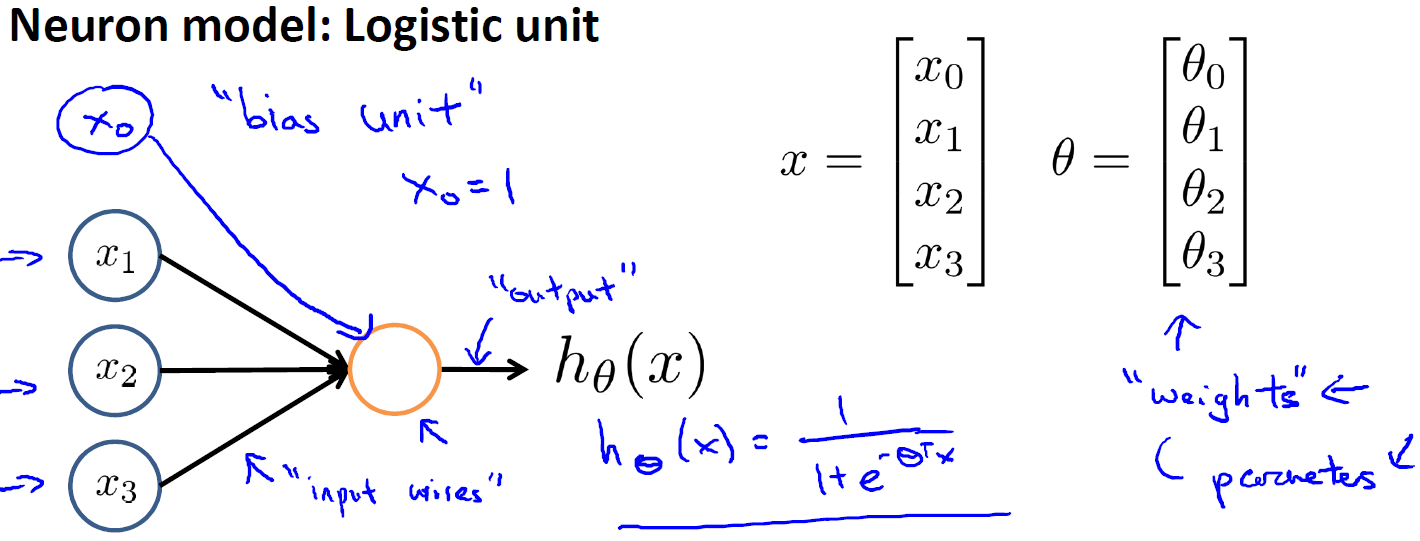
1. We will skip this as this part just goes over what a nonlinear hypothesis looks like

Video 2- Neurons and Brain

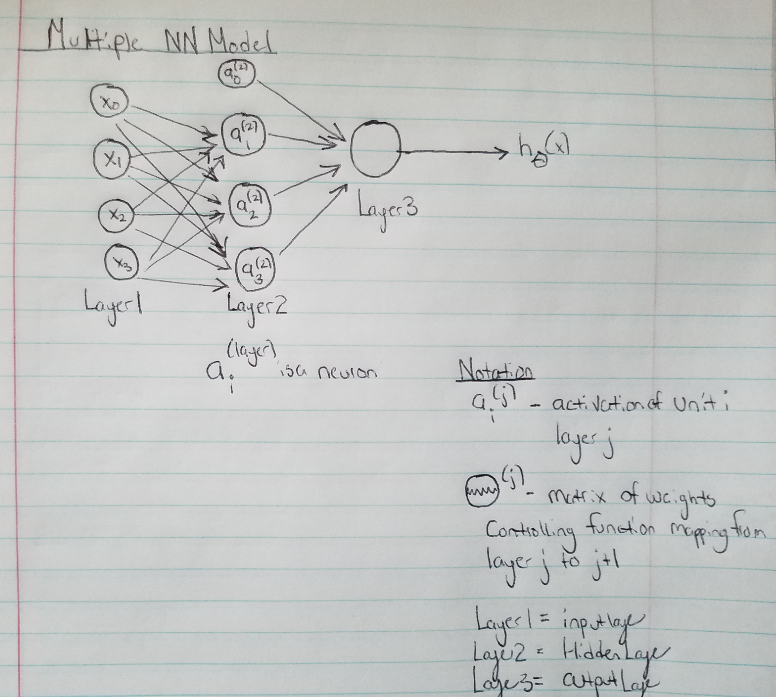
1. We will skip this part as it goes over NNs and their correlation to how a brain’s neuron works

Video 3 – Model Representation Part I

1. Model 1: Logistic Unit or single neuron(Simple 2 layer network)



1. Model 2: Multiple Neuron Model



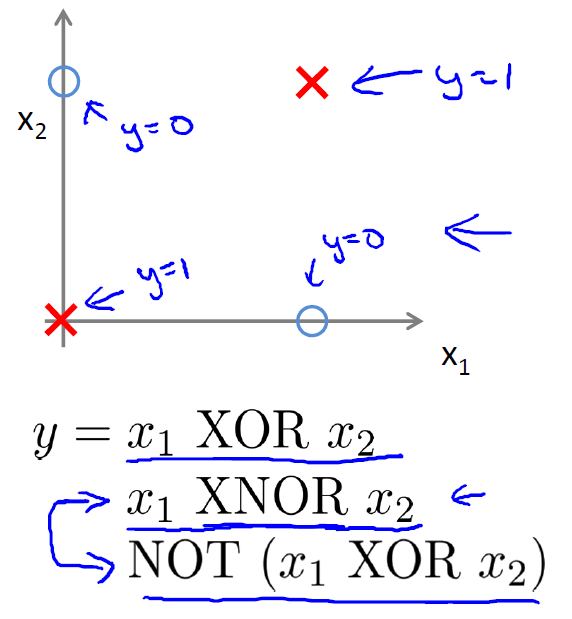
1. For model 2 we have the following equations
   1. Note – If a network has units in layer j and units in layer j+1 then the dimension of will be (
   2. In this example is 3x 4 as there are 3 units in layer 1 and 3 units in layer 2 ( we don’t count the hidden layer in this calculation as that is already accounted for in the formula in part e.

Video 5- Model Representation Part II

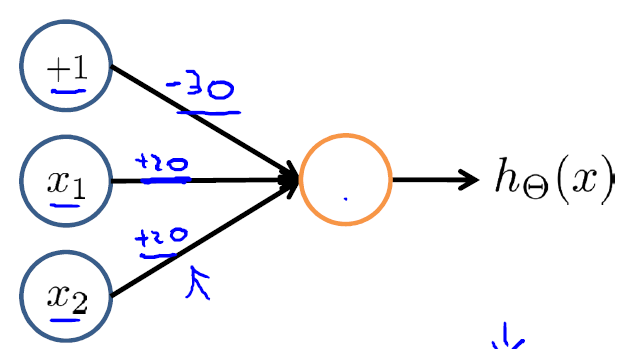
1. We are going to use shorter notation shown below
2. Let and so on
3. for i=1,2,3
4. The process of going forward to calculate the activation is called forward propogation
5. NN works using logistic regression except the inputs are the activations
6. In general for layer j=2 and node k the variable z is
   1. We set
7. In general except for the output later
   1. Here is
8. Last step is the calculation in the output layer which is represented by
   1. which is (1x n+1) and (n+1 x 1) resulting in 1x1 (Remember we add in the bias unit
9. Remember that

Video 6 - NN example 1

1. Suppose we are classifying XOR/NOR
   1. are binary 0,1 and we have the following target variable



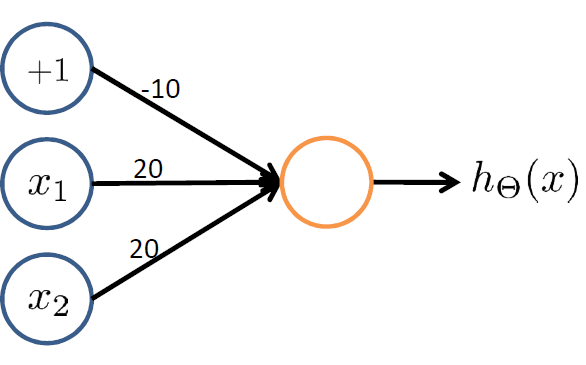
* 1. Below will show the NN diagram with weights



* 1. It follows that
     1. Remember – if g(x>0)=1

Video 7 – NN example 2

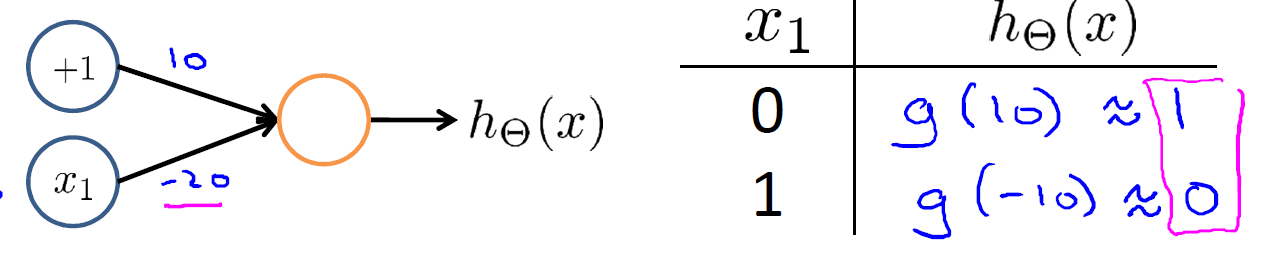
1. OR operator
   1. Below is the NN diagram



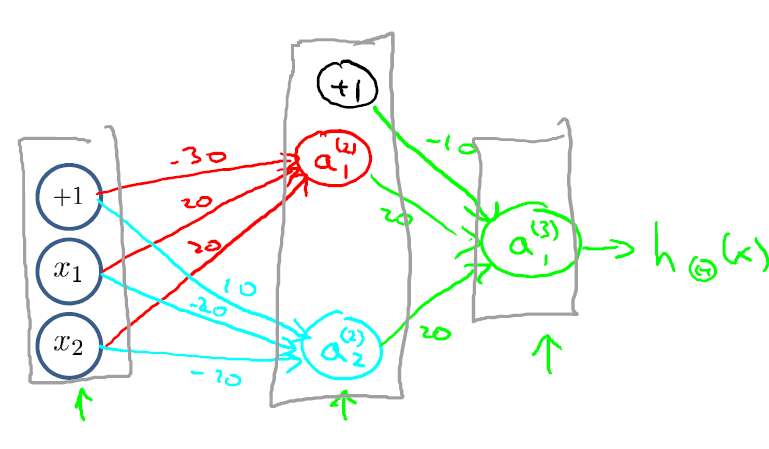
* 1. Below is the output

|  |  |  |
| --- | --- | --- |
| X1 | X2 |  |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

1. Not Operator
   1. Below is the NN diagram & calculation



1. XNOR operator

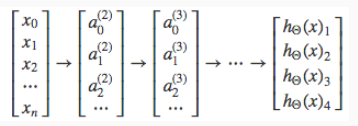


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **X1** | **X2** |  |  |  |
| 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 |

* 1. Here we will go over some of the matrix representations for this NN

Video 8 – Multiclass Classification

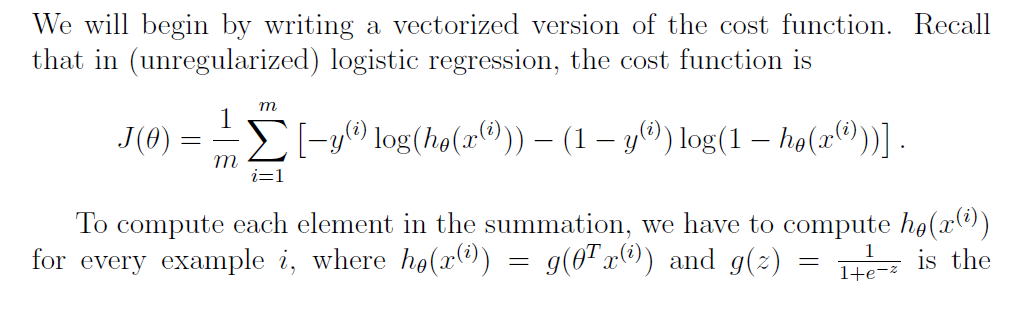
1. Here we use the 1 vs All method
2. Example
   1. Suppose we want to idenfify
      1. Pedestrian
      2. Car
      3. Motorcyle
      4. Truck
   2. Then we want the following
   3. The setup will then look like below

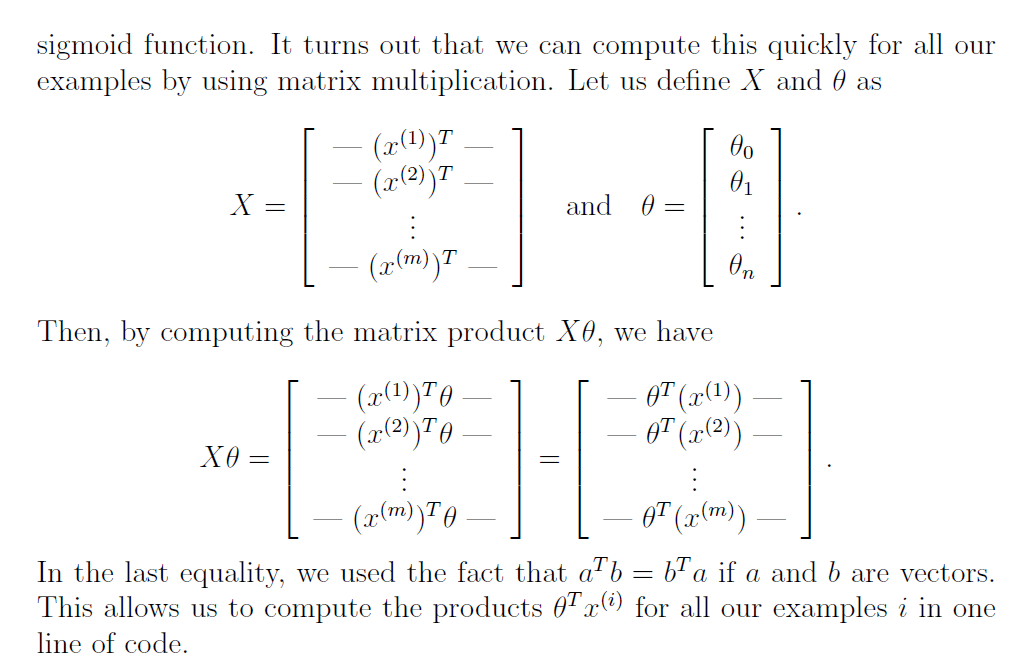


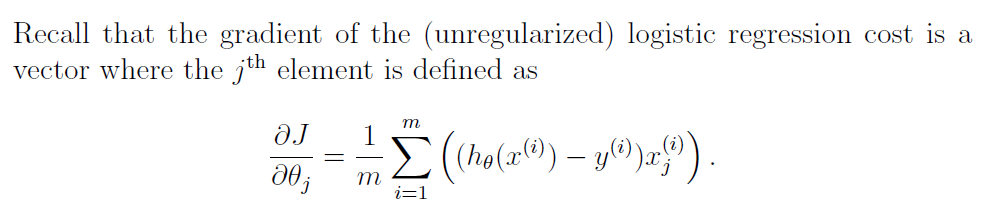
Quiz Question

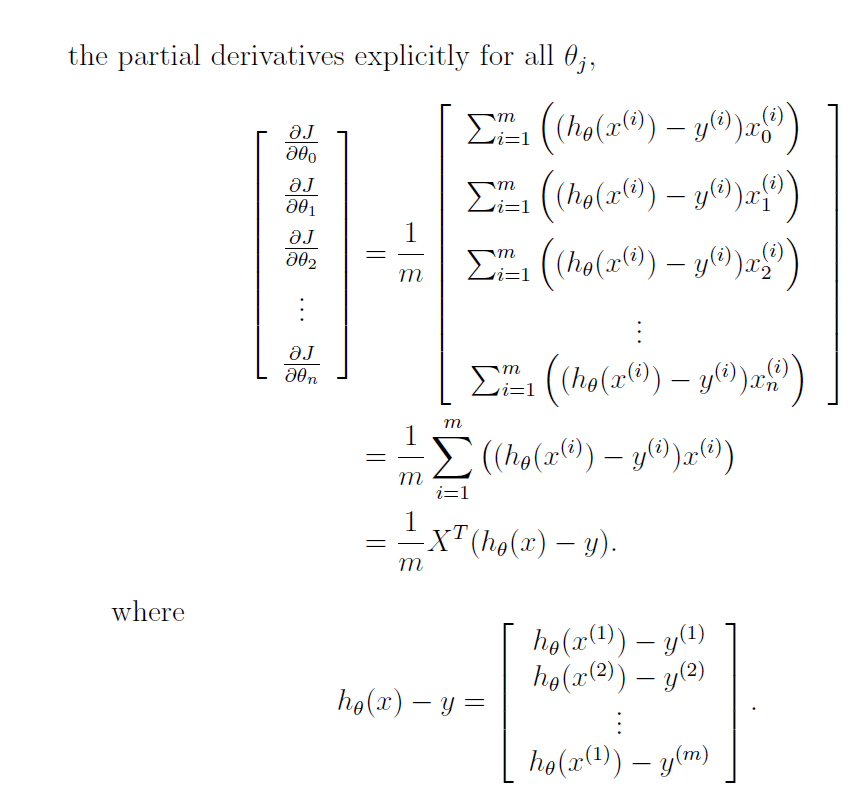
1. Suppose you have a classification problem with 10 classes. The NN has 3 layers and hidden layer has 5 units. Using the 1 vs All method how many elements are in ?
   1. We have S2=5 and S3=10. Thus

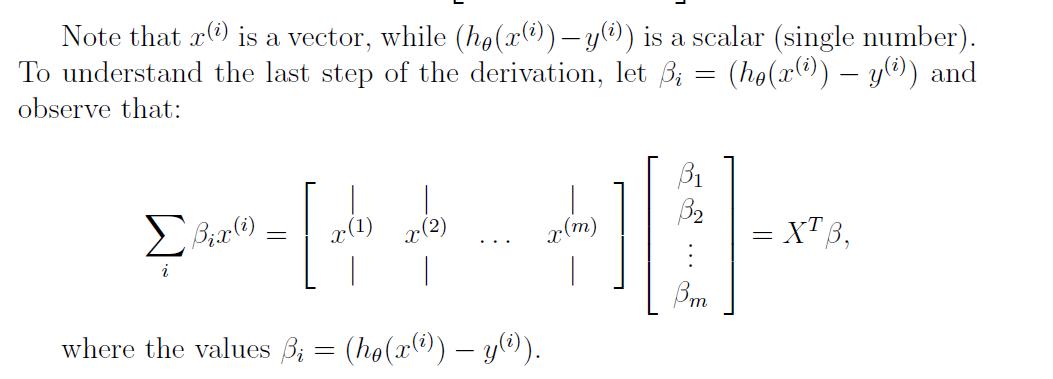
Assignment 3 Notes









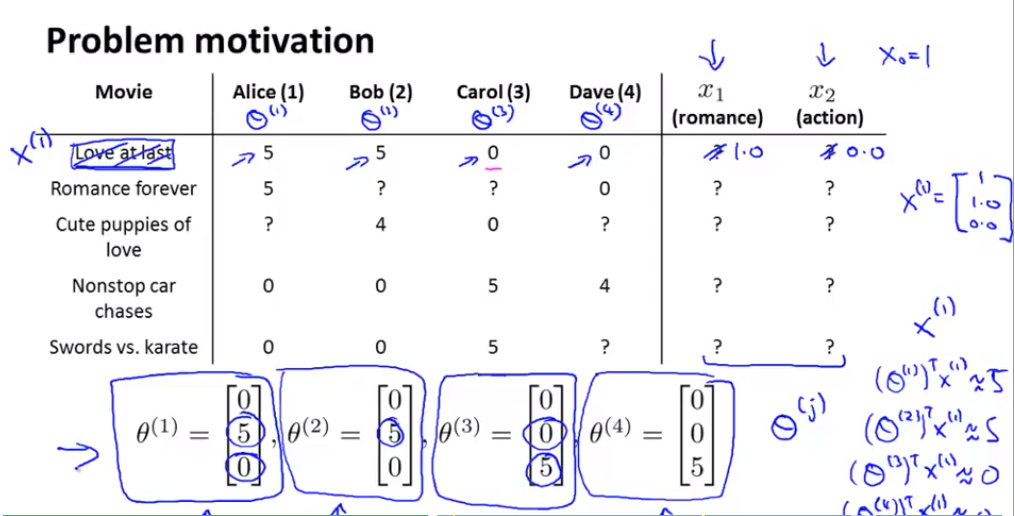


Week 5 – Stopped here

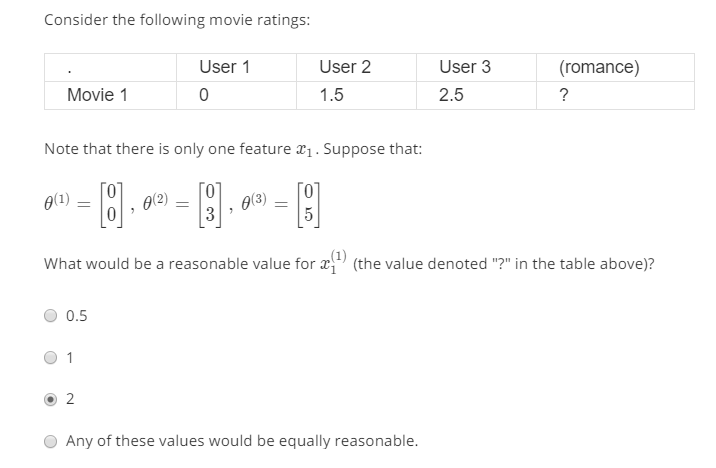
Week 9

Video 9 – Collaborative Filtering

1. Suppose the features are not known but we are given the weights by user showing us how much they like romatic movies and action movies

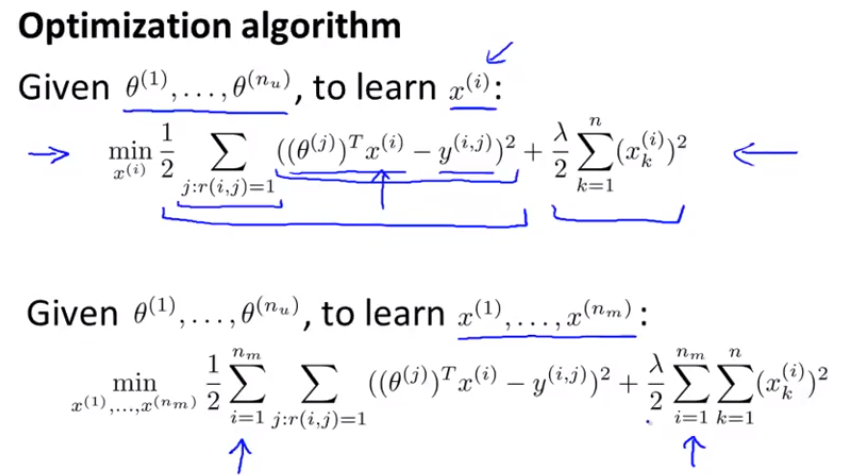


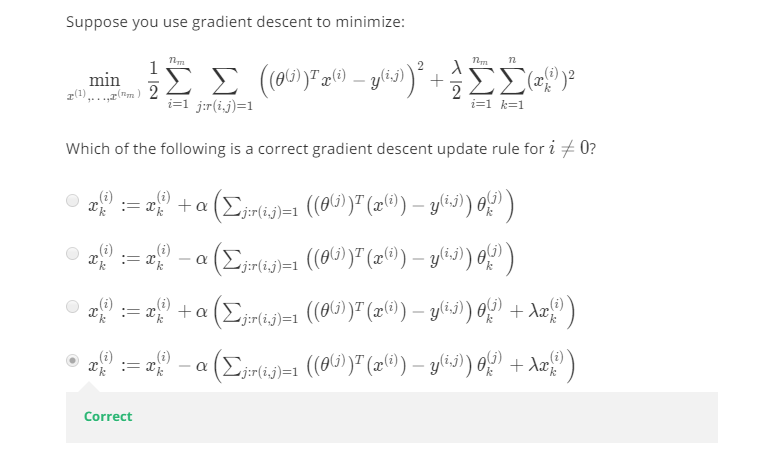
1. Above we are trying to figure out the movie in the 1st row is classified as the having a romance feature and not an action feature since Alice & Bob like romance and not the other two.
2. Quiz Question

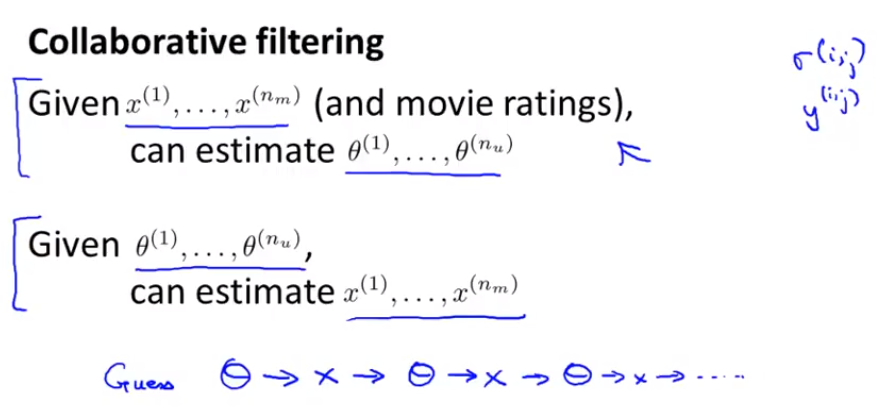


Answer: Since User 1 hates it, user 2 voted for half of it and same with user 3 it be 0.5

1. Problem Formalization



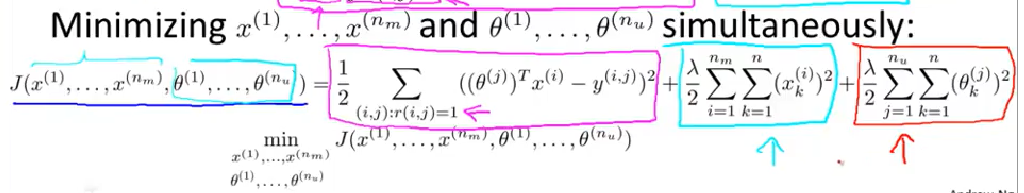
1. Here we want to minimize all the features
2. 

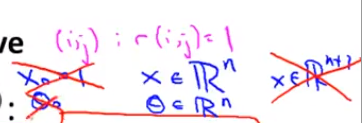


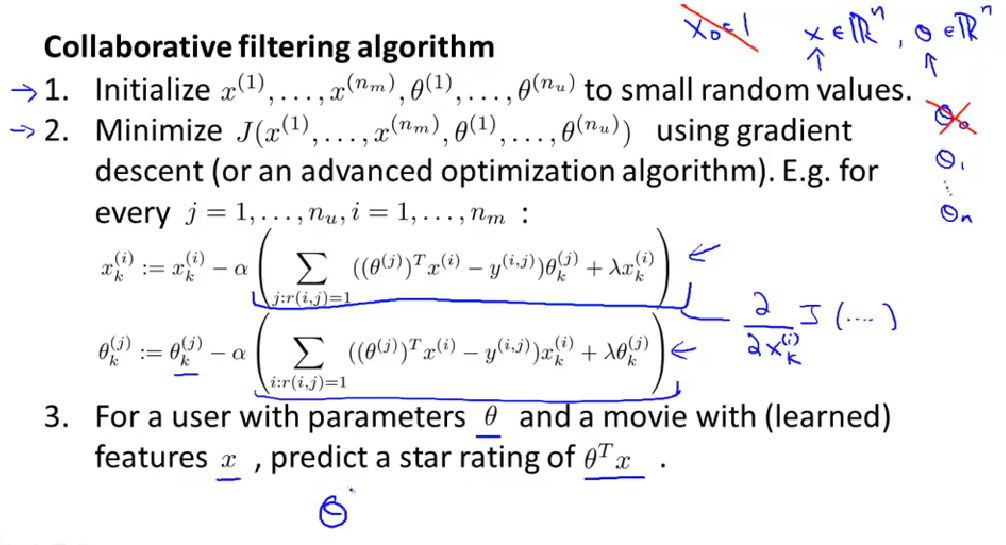
* 1. Eventually we converge to a given set a parameters

Video 10 – Collaborative Filtering ALgoritm II

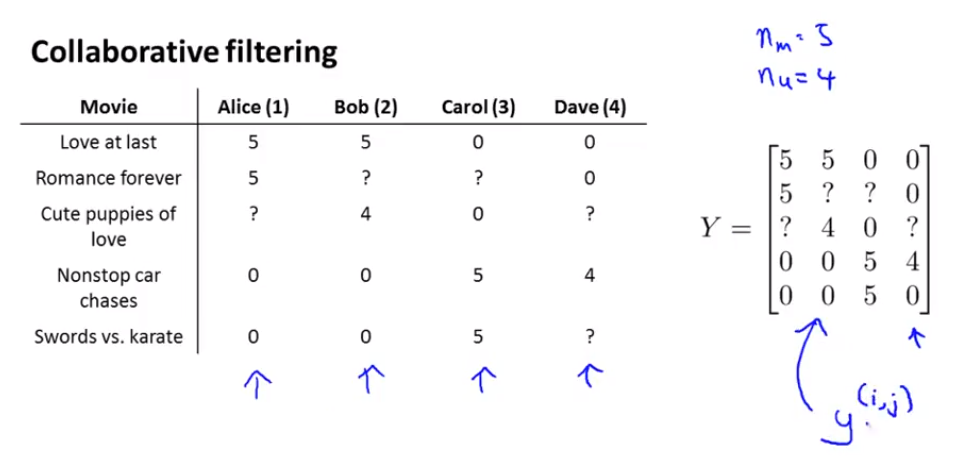
1. More efficient algorithm

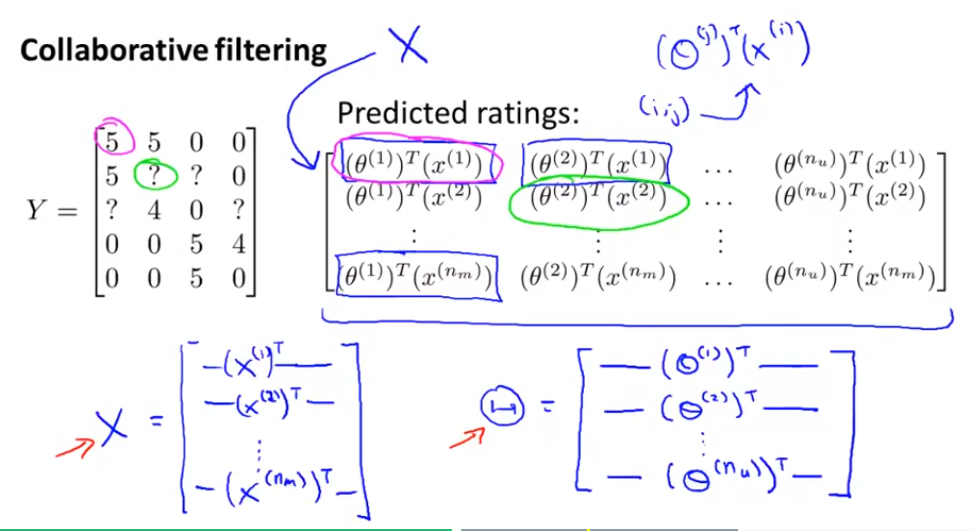




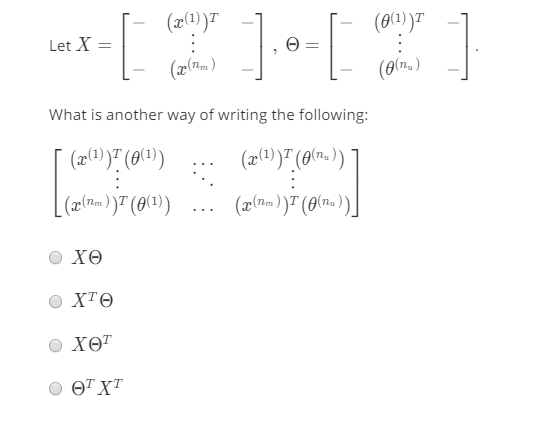


Video 11- Low Rank Matrix Factorization I

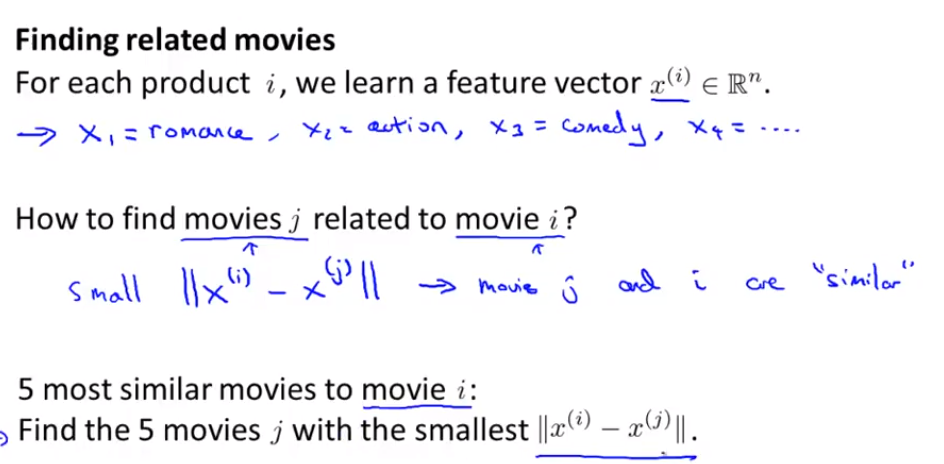
1. Work out an alternative way of writing out the data
2. 



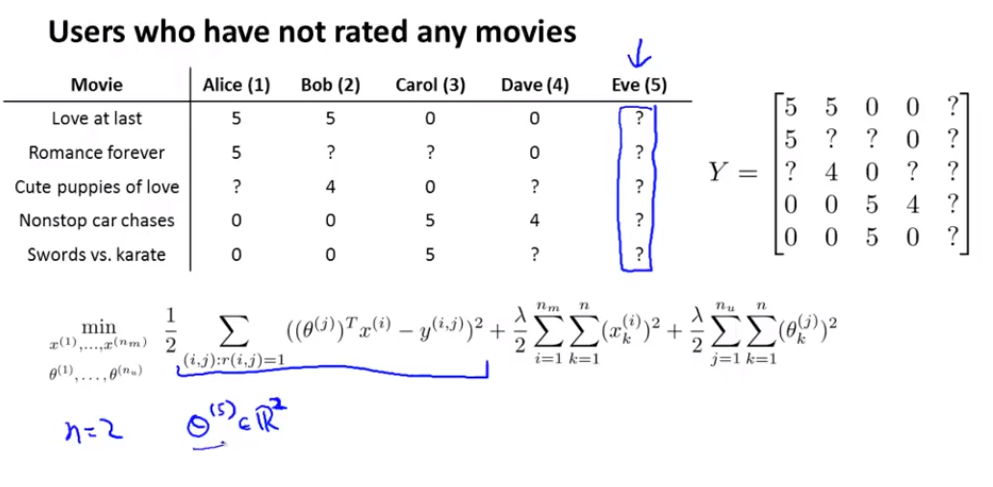
* 1. Below we want to write X and Theta
  2. This is called low rabj factor matrix factorizaition



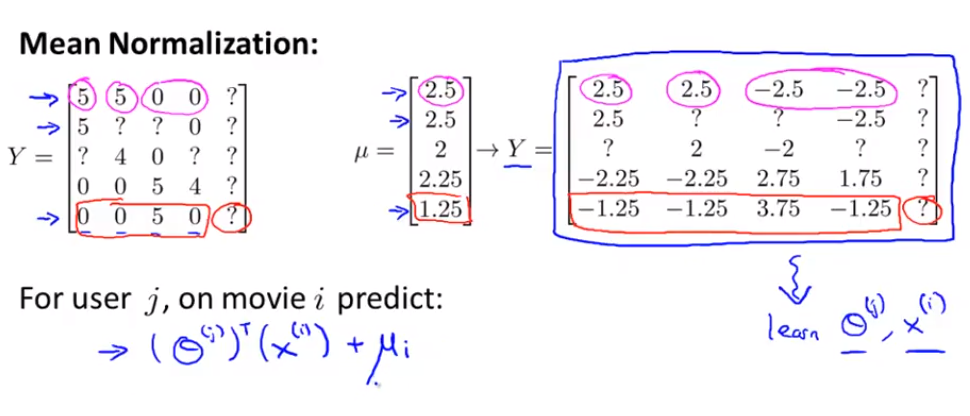
1. The answer to above is ( Note the picture above has a typo in the theta matrix should have a T for the last theta. ) third one

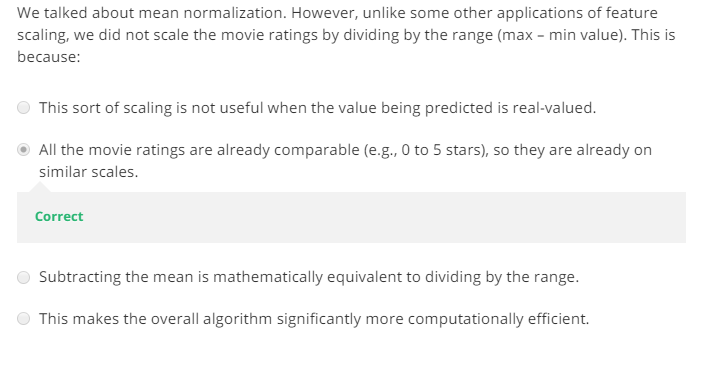


Video 12 – Mean Normalization



1. Predicting everything 0 for eve would not be good
2. So what we do is use mean normalization so we compute mu and subtract mu off the dataset Y and then learn x(i) and theta(i) off of it and then add back on mu when predicting

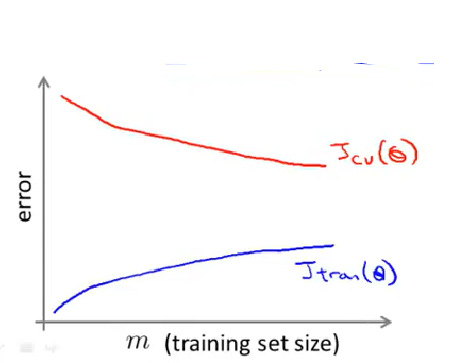




Week 10

Video 1- Learning with large datasets

1. Suppose we are working with a large training set where m=100 MM and your fitting linear regression with gradient descent then gradient descent would be pretty slow as you would have to sum over 100 MM examples.
2. Suppose if m=100MM and we train using a small subset say 1000 how can you tell if using all the data is likely to perform better then using the smaller set?
   1. Plot a learning curve for a range of values of m and verify that the algorithm has high variance when m is small AKA learning curves (Shown below)



Video 2- Stochastic Gradient Descent

1. Modification to the original gradient descent to allow for scaling
2. Suppose we are fitting linear regression via Stochastic Gradient Descent the algorithm becomes

|  |  |
| --- | --- |
| **Batch Gradient Descent** | **Stochastic Gradient Descent** |
|  |  |
| Then |
| **Algorithm Steps** |
| 1. Randomly shuffle dataset |
| 1. Repeat {   For i= 1: m {    }  } |

1. Stochastic gradient descent is just using 1 training example to fit it just a bit better rather then looping over all the examples each time and taking 1 small step as in batch gradeient descent.
2. You probably want to run **repeat** in the algorithm around 1-10 times and if your dataset is very large it may only take once

Video 3 – Mini-Batch Gradient Descent

1. Use b examples in each iterations where b=mini-batch size
2. Typical value for b is between 2-100
3. Example 1
   1. Get b=10 examples and say m=1000
   2. Repeat {
   3. For i=1,11,21,31,…,991 {
   4. }
   5. }
4. Mini- Batch gradient descent works well when vectorized

Video 6- Stochastic Gradient Descent Convergence

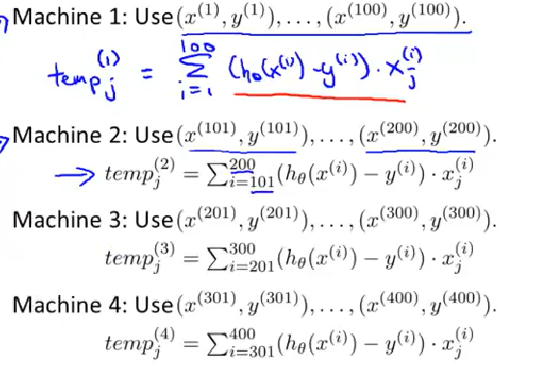
1. While computing compute the cost before updating theta using
2. Every 1000 iterations plot averaged over the last 1000 examples processed by the algorithm
3. Plot # of iterations vs averaged cost. If you notice the cost increasing using a smaller alpha
4. Learning rate is typically held constact so we can slowly decrease alpha overtime if we want theta to converge IE

Video 7 – Online Learning

1. Algorithms analyze a continuous stream of online data and automatically adapt the data making recommendations tailored to each user
2. Example 1
   1. Product Search (learning to search)
   2. Suppose user searches Android Phone 1080p and we have 100 phones in store. We want to return 10 results
   3. Here we can capture in our feature vector things such as features of phone, how many words in user query match name of phone, etc.
   4. Y=1 if user clicks on link and Y=0 otherwise
   5. Learn p(Y=1| x; theta) 🡪 predicted CTR

Video 8 – Map Reduce and Parallelism

1. Suppose we have several machines and a very large dataset
2. We first split the training set to each machine and running batch gradient descent say over 4 machines in this example



1. Then we combine and now
2. In this example you would get around a 4x speed up by parallelizing
3. Usually ask yourself can your algorithm be expressed as a sum of a traning set so mapreduce can then be used
4. You can also split the training set and send each part to a CPU core on same machine

**11.11.2 Anonymous Functions**

Anonymous functions are defined using the syntax

@(*argument-list*) *expression*

Any variables that are not found in the argument list are inherited from the enclosing scope. Anonymous functions are useful for creating simple unnamed functions from expressions or for wrapping calls to other functions to adapt them for use by functions like *quad*. For example,

f = @(x) x.^2;

quad (f, 0, 10)

⇒ 333.33

creates a simple unnamed function from the expression *x.^2* and passes it to *quad*,

quad (@(x) sin (x), 0, pi)

⇒ 2

wraps another function, and

a = 1;

b = 2;

quad (@(x) betainc (x, a, b), 0, 0.4)

⇒ 0.13867

adapts a function with several parameters to the form required by *quad*. In this example, the values of *a*and *b* that are passed to *betainc* are inherited from the current environment.

Note that for performance reasons it is better to use handles to existing Octave functions, rather than to define anonymous functions which wrap an existing function. The integration of *sin (x)* is 5X faster if the code is written as

quad (@sin, 0, pi)

rather than using the anonymous function *@(x) sin (x)*. There are many operators which have functional equivalents that may be better choices than an anonymous function. Instead of writing

f = @(x, y) x + y

this should be coded as

f = @plus

See [Operator Overloading](https://octave.org/doc/interpreter/Operator-Overloading.html#Operator-Overloading), for a list of operators which also have a functional form.