Week 1

Table of Contents

Introduction

Introduce the core idea of teaching a computer to learn concepts using data - without being explicitly programmed.

linear regression with one variable. Discuss the application of linear regression to housing price prediction, present the notion of a cost function, also introduce the gradient descent method for learning.

The course require the knowledge of l**inear algebra** concepts.

What's Machine Learning?

Andrew NG: Machine Learning is the science of getting computers to learn, without being explicitly programmed.

- Grew out of work in AI
- New capability for computers

Examples:

- Database mining
    - Large datasets from growth of automation/web
    - E.g., Web click data, medical records, biology, engineering
- Applications can't program by hand.
    - E.g., Autonomous helicopter, handwriting recognition, most of Natural Language Processing(NLP), Computer Vision.
- Self-customizing programs

  E.g., Amazon, Netflix product recommendations.
- Understanding human learning(brain, real AI).

What's Machine Learning?[Definition]

Two definitions of Machine Learning are offered.

- Arthur Samuel described it as: **"the field of study that gives computers the ability to learn without being explicitly programmed."** This is an older, informal definition.
- Tom Mitchell provides a more modern definition: **"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."**

Examples:

playing checkers.

E = the experience of playing many games of checkers

T = the task of playing checkers.

P = the probability that the program will win the next game.

Suppose your email program watches which emails you do or do not mark as spam, and based on that learns how to better filter spam. Explain the E T P according to Tom Mitchell's definiation.

E = Watching you label emails as spam or not spam.

T = Classify emails as spam or not spam.

P = The number(or fraction) of emails correctly classified as spam/not spam.

In general, any machine learning problem can be assigned to one of two broad classifications:

Supervised learning and Unsupervised learning.

Machine Learning algorithms

- Supervised Learning
- Unsupervised Learning

Others: Reinforcement Learning, recommender systems.

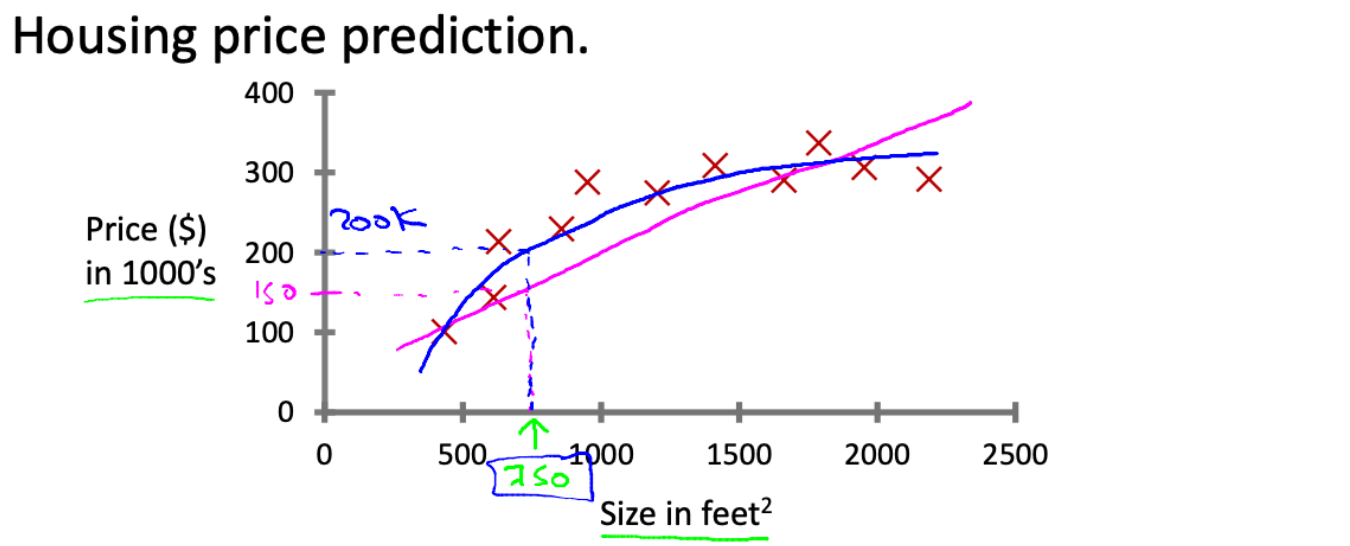
Supervised Learning

In supervised learning, we are given a data set and already know what our correct output should look like, **having the idea that there is a relationship between the input and the output.**

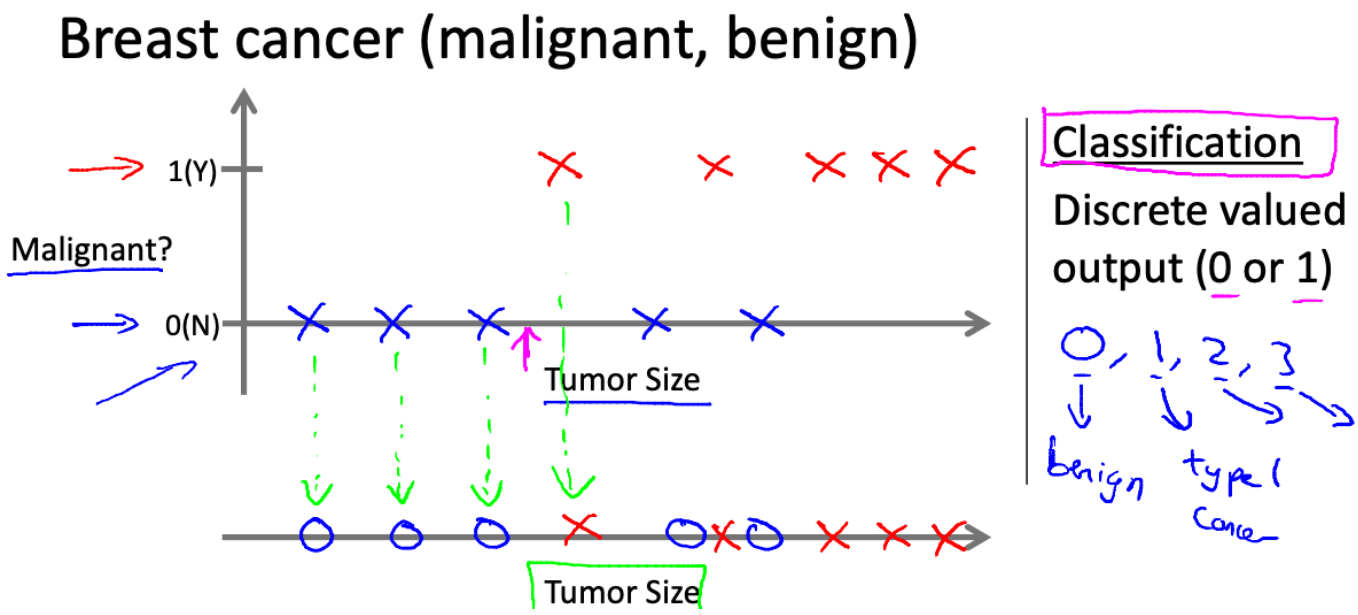
Supervised learning problems are categorized into "**regression**" and "**classification**" problems.

In a regression problem, we are trying to predict results within a **continuous output**, meaning that we are trying to map input variables to some continuous function.

In a classification problem, we are instead trying to predict results in a **discrete output[离散输出]**. In other words, we are trying to map input variables into discrete categories.



Regression exp



Classification

Example 1:

Given data about the size of houses on the real estate market, try to predict their price. Price as a function of size is a continuous output, so this is a regression problem.

We could turn this example into a classification problem by instead making our output about whether the house "sells for more or less than the asking price." Here we are classifying the

houses based on price into two discrete categories.

Example 2:

(a) Regression - Given a picture of a person, we have to predict their age on the basis of the given picture

(b) Classification - Given a patient with a tumor, we have to predict whether the tumor is malignant or benign.

Practice：

You’re running a company, and you want to develop learning algorithms to address each of two problems.

Problem 1: You have a large inventory of identical items. You want to predict how many of these items will sell over the next 3 months.

Problem 2: You'd like software to examine individual customer accounts, and for each account decide if it has been hacked/compromised. Should you treat these as classification or as regression problems?

Answer:

Treat problem 1 as a regression problem, problem 2 as a classification problem.

Explain:

First one : we have thousand of items [Continuous]

Second one: 0 - not hacked, 1 - hacked.[Hacked or not, Discrete]

Unsupervised Learning

Unsupervised learning is a type of machine learning that looks for previously undetected patterns in a data set with no pre-existing labels and with a minimum of human supervision. —-Wikipedia

Unsupervised learning allows us to approach problems with little or no idea what our results should look like. We can derive structure from data where we don't necessarily know the effect of the variables.

We can derive this structure by clustering the data based on relationships among the variables in the data.

With unsupervised learning there is no feedback based on the prediction results.

Example:

Clustering: Take a collection of 1,000,000 different genes, and find a way to automatically group these genes into groups that are somehow similar or related by different variables, such as lifespan, location, roles, and so on.

Non-clustering: The "Cocktail Party Algorithm", allows you to find structure in a chaotic environment. (i.e. identifying individual voices and music from a mesh of sounds at a cocktail party).

The Following examples using an unsupervised learning algorithm:

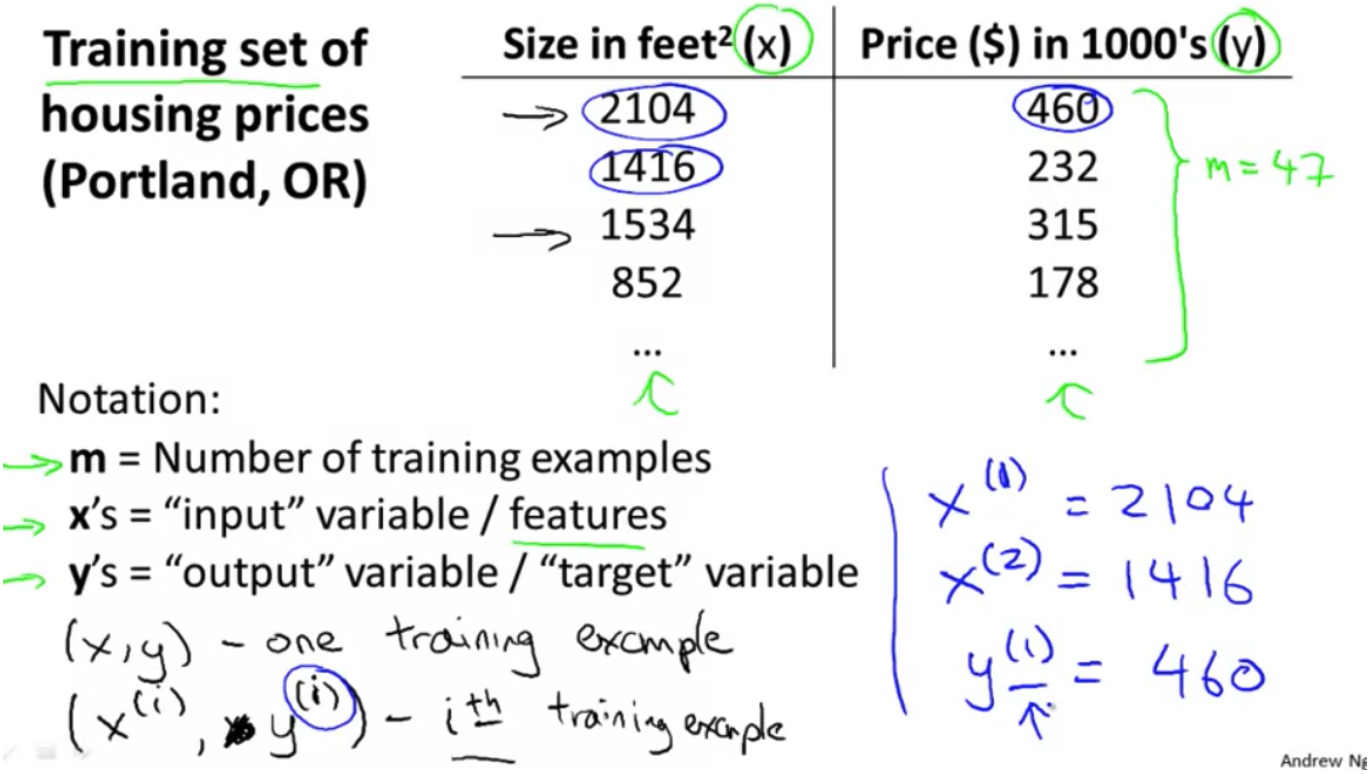
Given a set of news articles found on web, group them into sets of articles about the same stories.

Given a database of customer data, automatically discover market segments and group customers into different market segments.

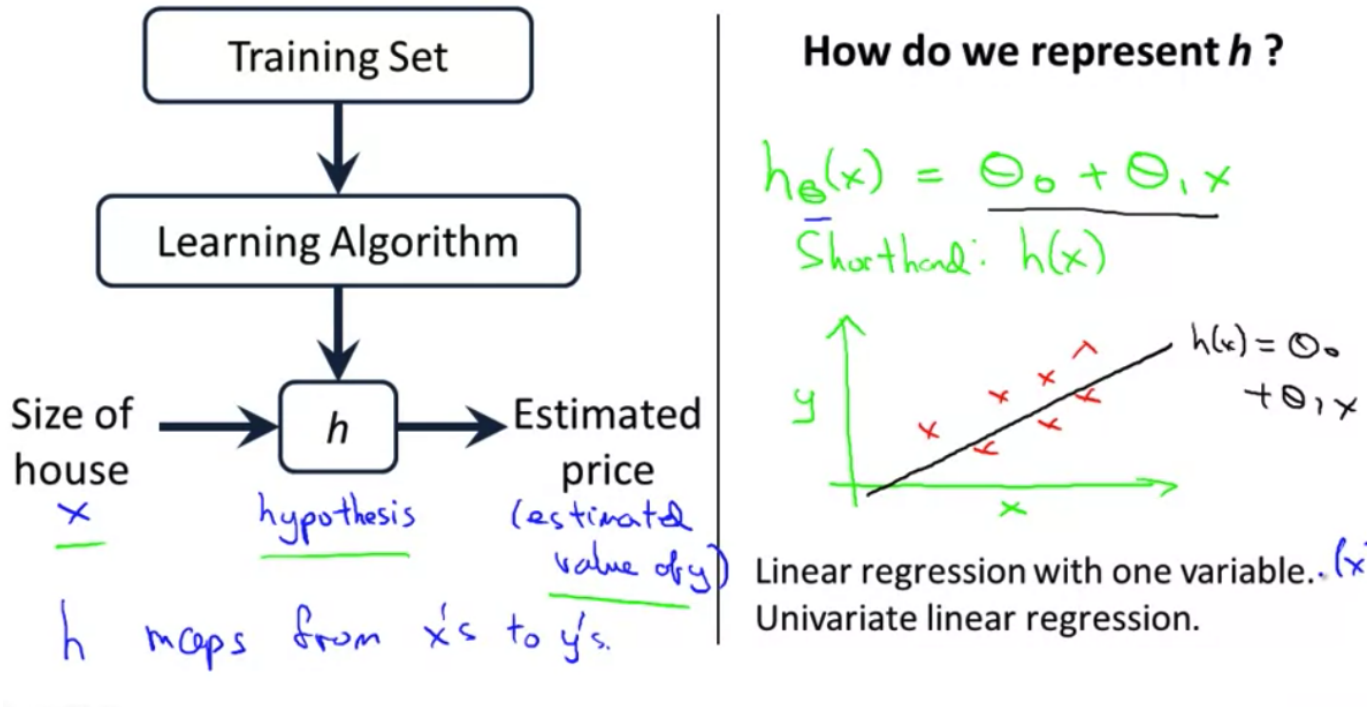
Model and Cost Function

Model Representation

Notes:



Linear regression model with only one variable[Also called univariate linear regression]



Material:

To describe the supervised learning problem slightly more formally, our goal is, given a training set, to learn a function h : X → Y so that h(x) is a "good" predictor for the corresponding value of

y. For historical reasons, this function h is called a **hypothesis**. Seen pictorially, the process is therefore like this:

When the target variable that we're trying to predict is continuous, such as in our housing example, we call the learning problem a regression problem. When y can take on only a small number of discrete values (such as if, given the living area, we wanted to predict if a dwelling is a house or an apartment, say), we call it a classification problem.

Cost Function

Notes:

How to choose the parameters?

Training Set

Size in feet² (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
...	...

m = 47

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

θ_i's: Parameters

How to choose θ_i's ?

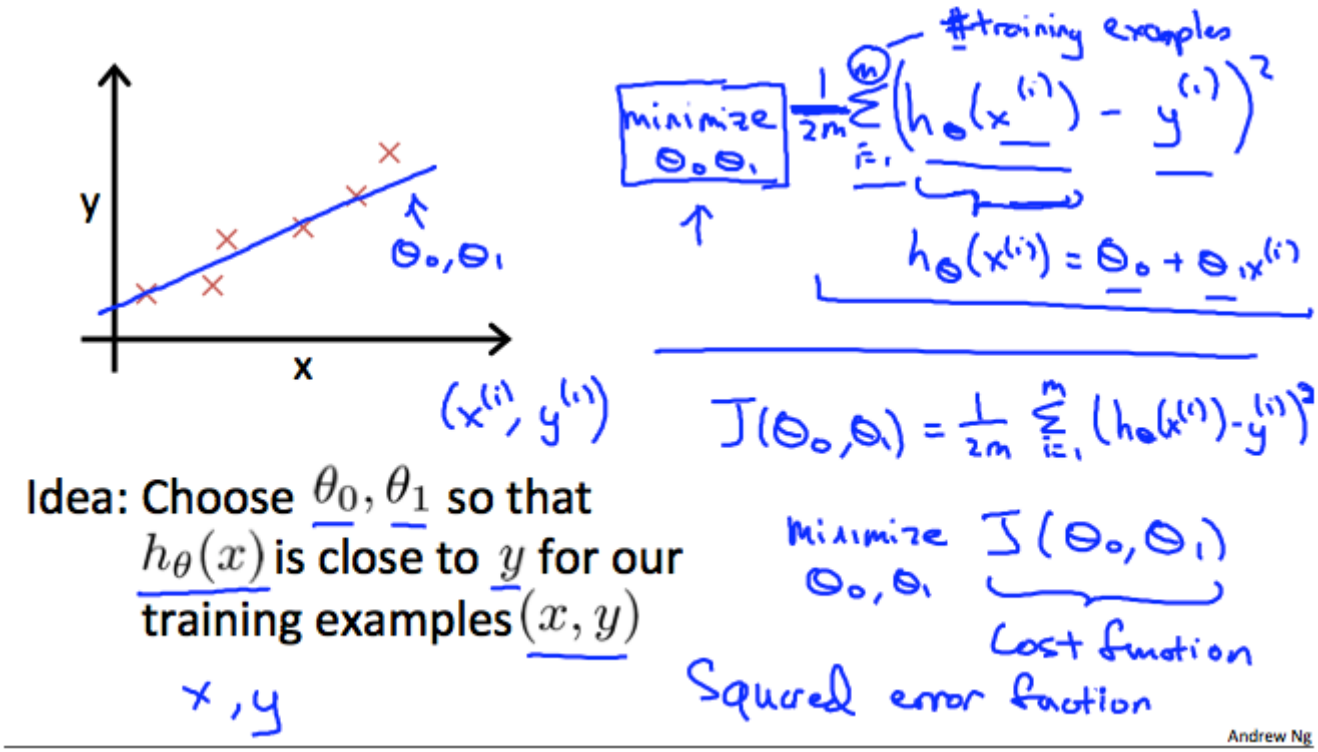
Material:

We can measure the accuracy of our hypothesis function by using a **cost function**. This takes an average difference (actually a fancier version of an average) of all the results of the hypothesis with inputs from x's and the actual output y's.

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y}_i - y_i)^2 = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2$$

To break it apart, it is $\frac{1}{2}\ \bar{x}$ where $\bar{x}$ is the mean of the squares of $h_\theta(x_i) - y_i$, or the difference between the predicted value and the actual value.

This function is otherwise called the "Squared error function", or "Mean squared error". The mean is halved $\left(\frac{1}{2}\right)$ as a convenience for the computation of the gradient descent, as the derivative term of the square function will cancel out the $\frac{1}{2}$ term. The following image summarizes what the cost function does:

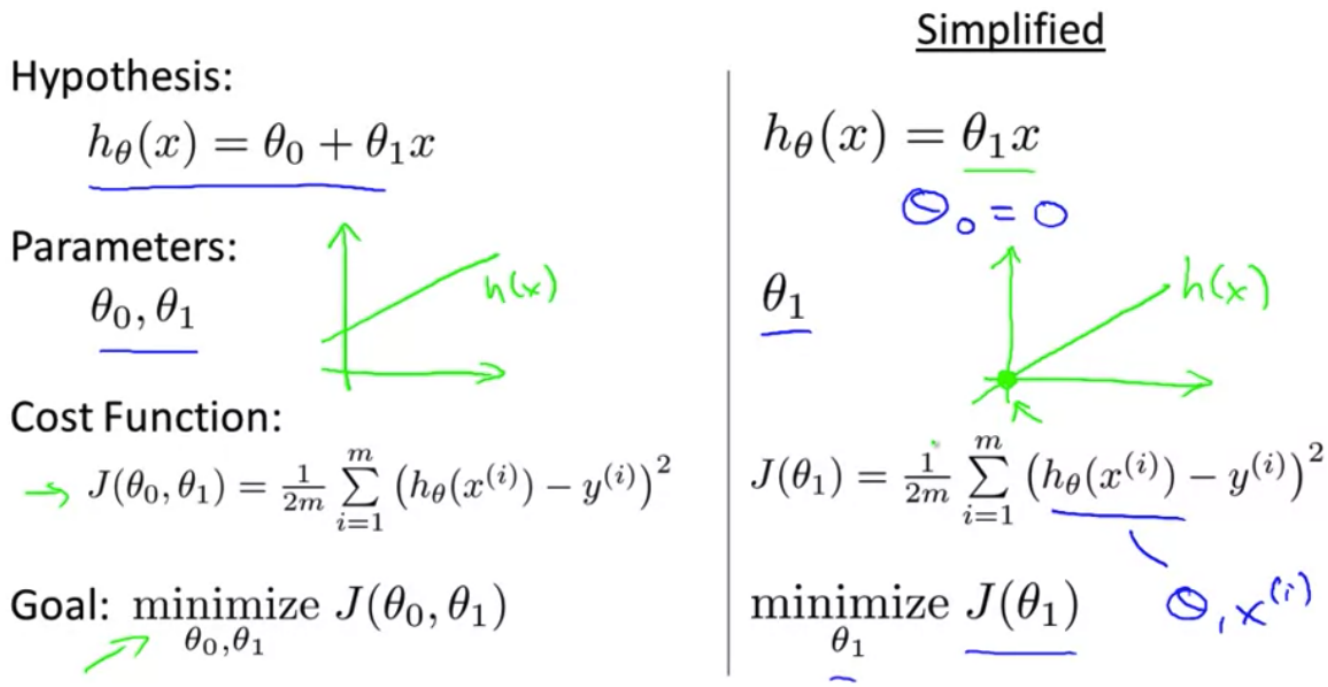


MSE: the most common one used for regression problems.

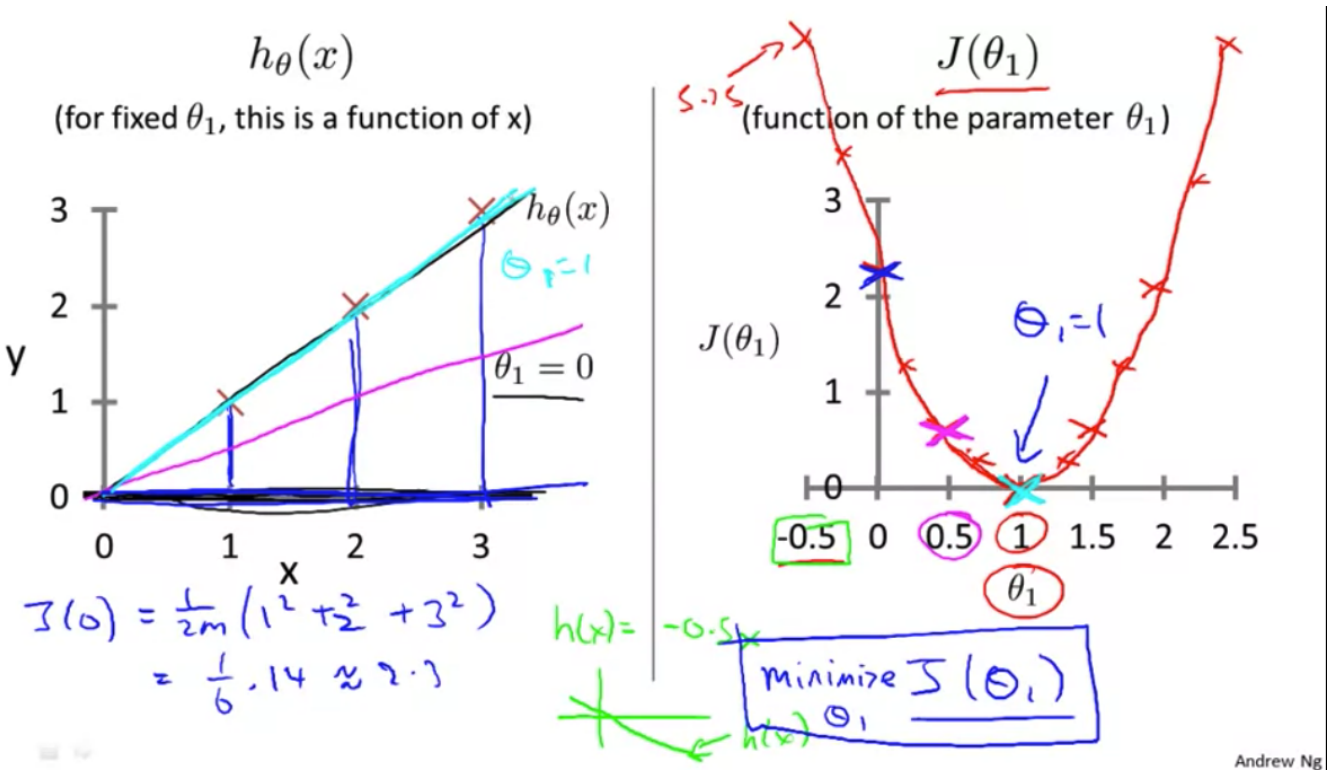
Mean squared error

Cost Function Intuition I

1.Set the theta(0) to zero.



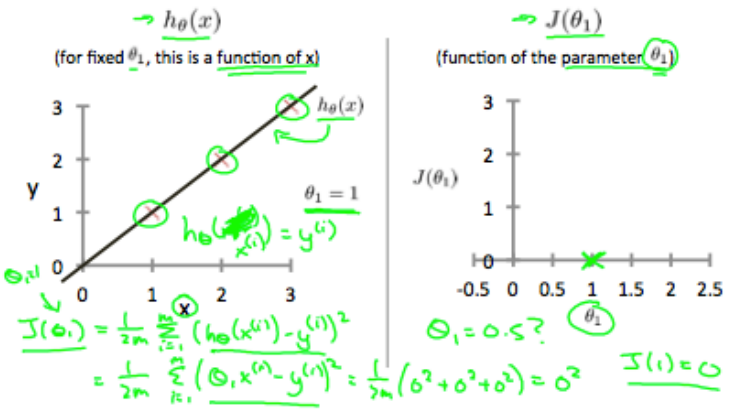
2.The optimization objective for our learning algorithm is we want to choose the value of theta (1). The Minimizes J(theat(1)).



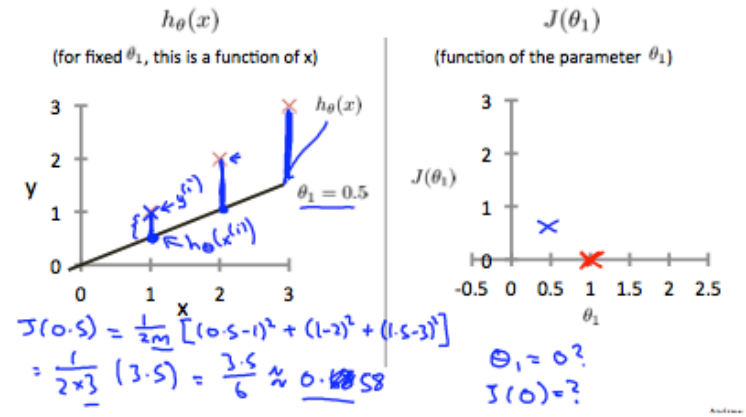
Material:

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.

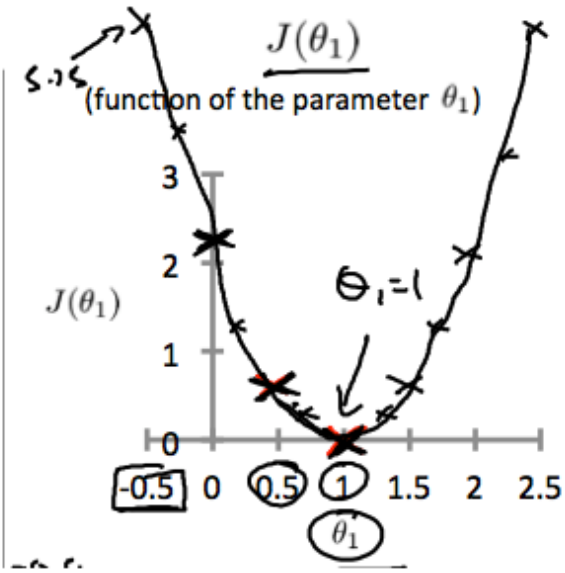
Our objective is to get the best possible line. The best possible line will be such so that the average squared vertical distances of the scattered points from the line will be the least. Ideally, the line should pass through all the points of our training data set. In such a case, the value of $J(\theta_0,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

Cost function - Intuition II

1. Keep both two parameters of the cost function.

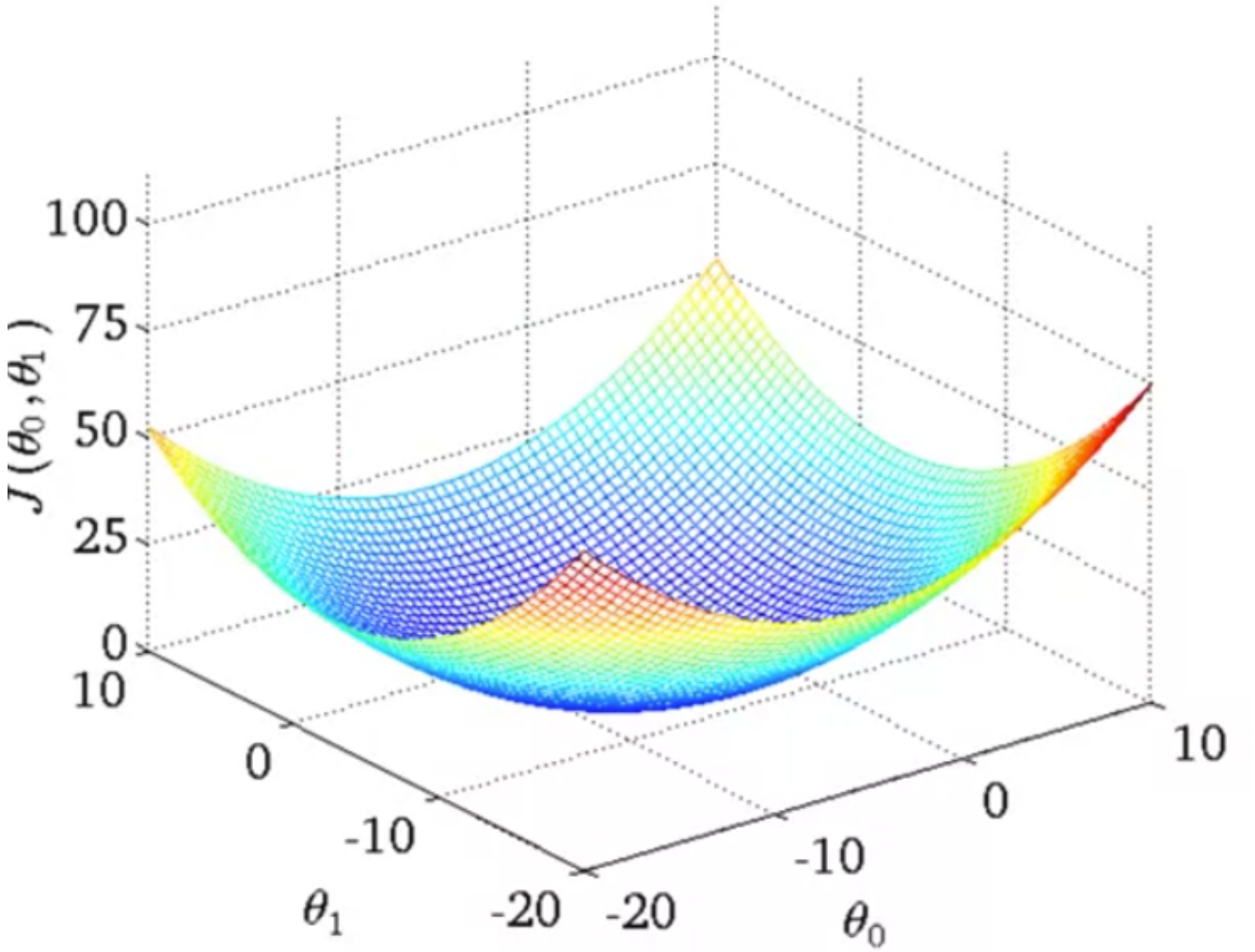
Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

Parameters: θ_0, θ_1

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

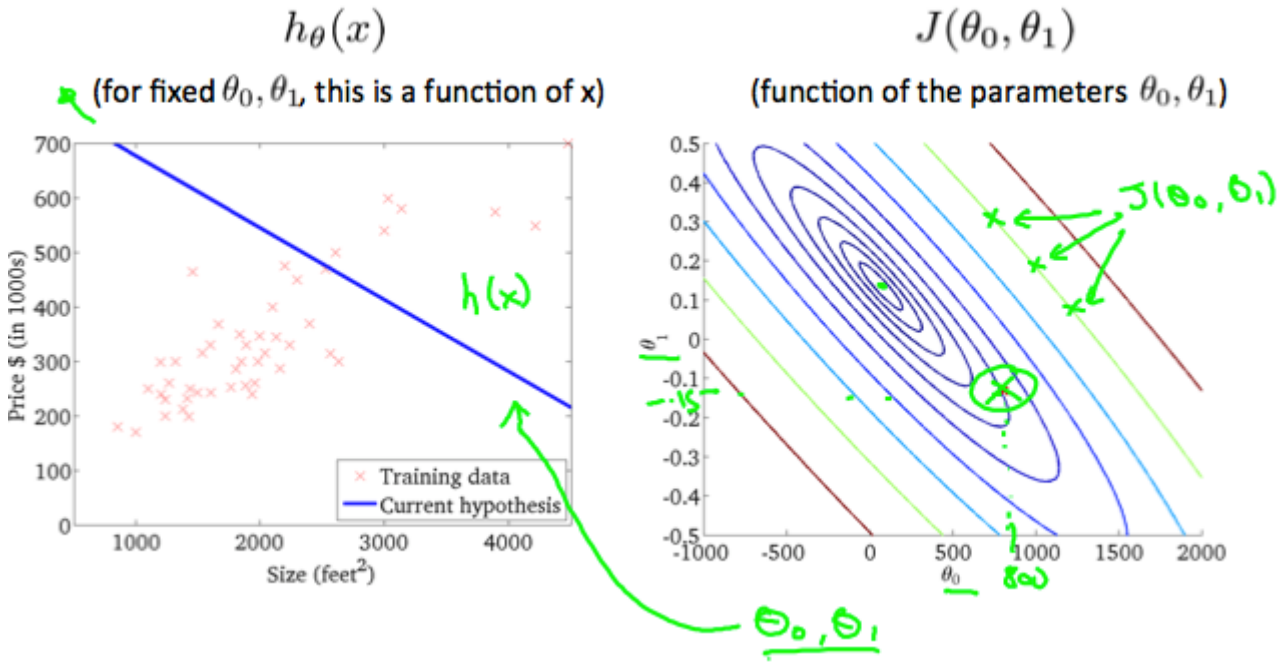
Goal: $\underset{\theta_0,\theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$

The cost function may be look like below:



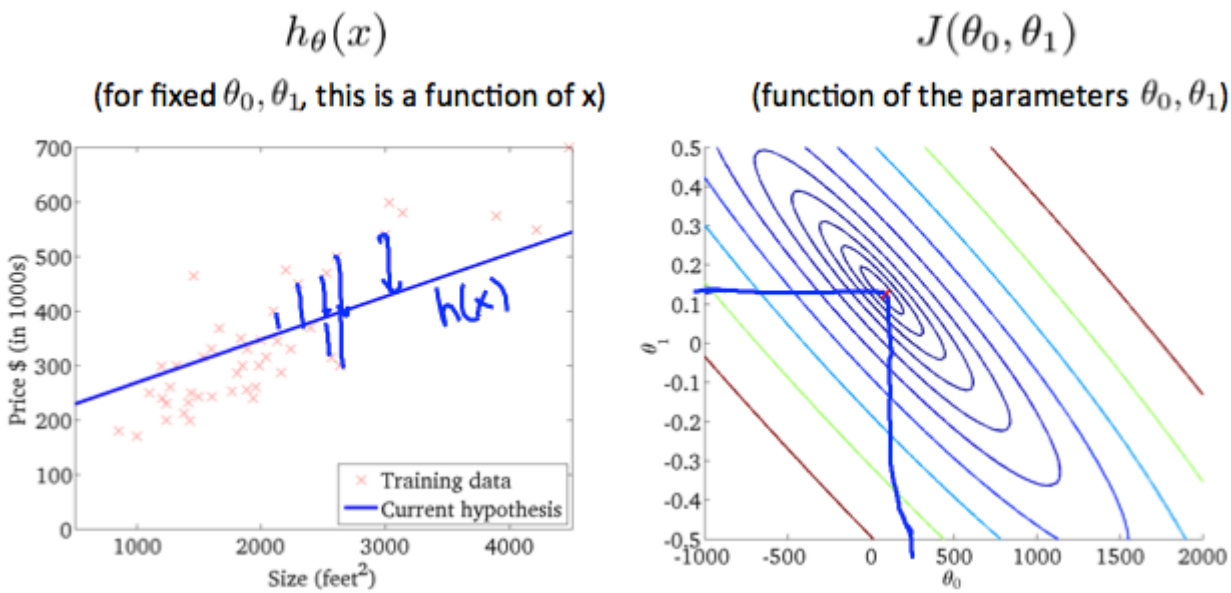
Contour plots / figure

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



3 green x have the same cost.

When θ0=360 and θ1 = 0, the value of J(θ0, θ1) in the contour plot gets closer to the center thus reducing the cost function error. Now Giving our hypothesis function a slightly positive slope result in a better fit of the data.



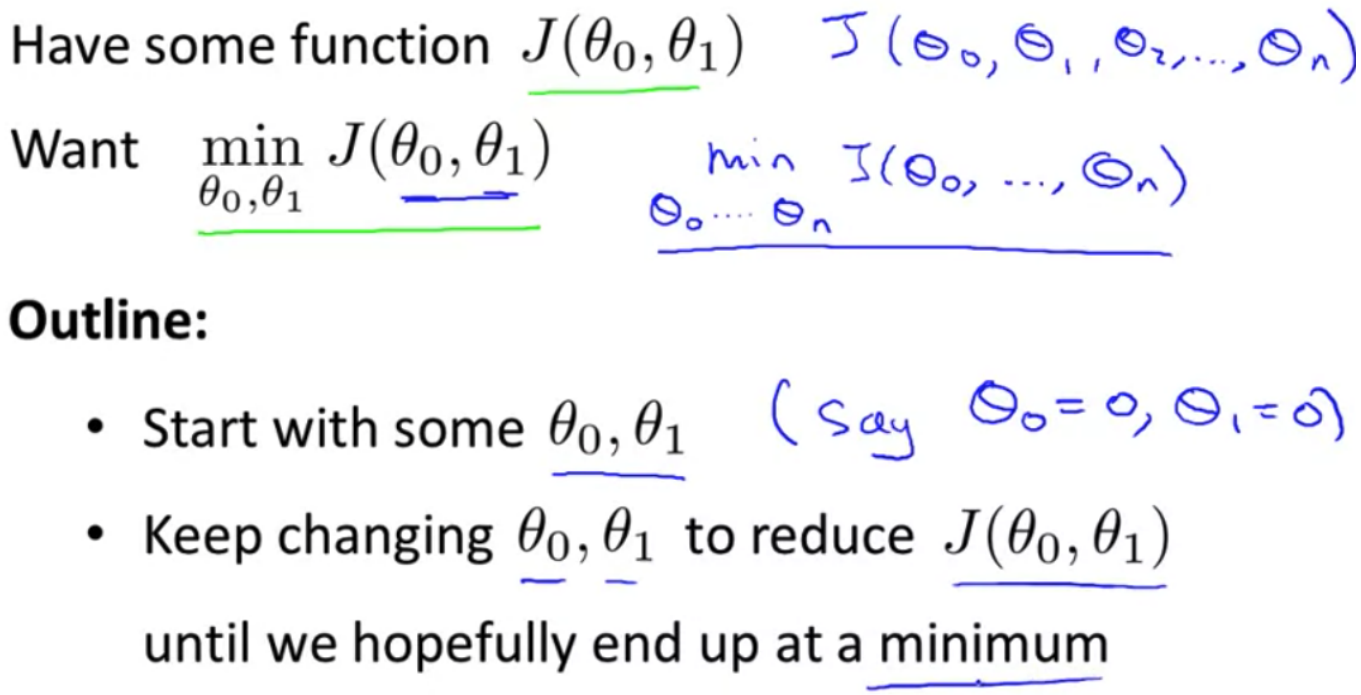
The graph above minimizes the cost function as much as possible and consequently, the result of θ1 and θ0 tend to be around 0.12 and 250 respectively. Plotting those values on our graph to the right seems to put our point in the center of the inner most 'circle'.

Parameter Learning

Gradient Descent

The algorithm called gradient descent for **minimizing the cost function j().**

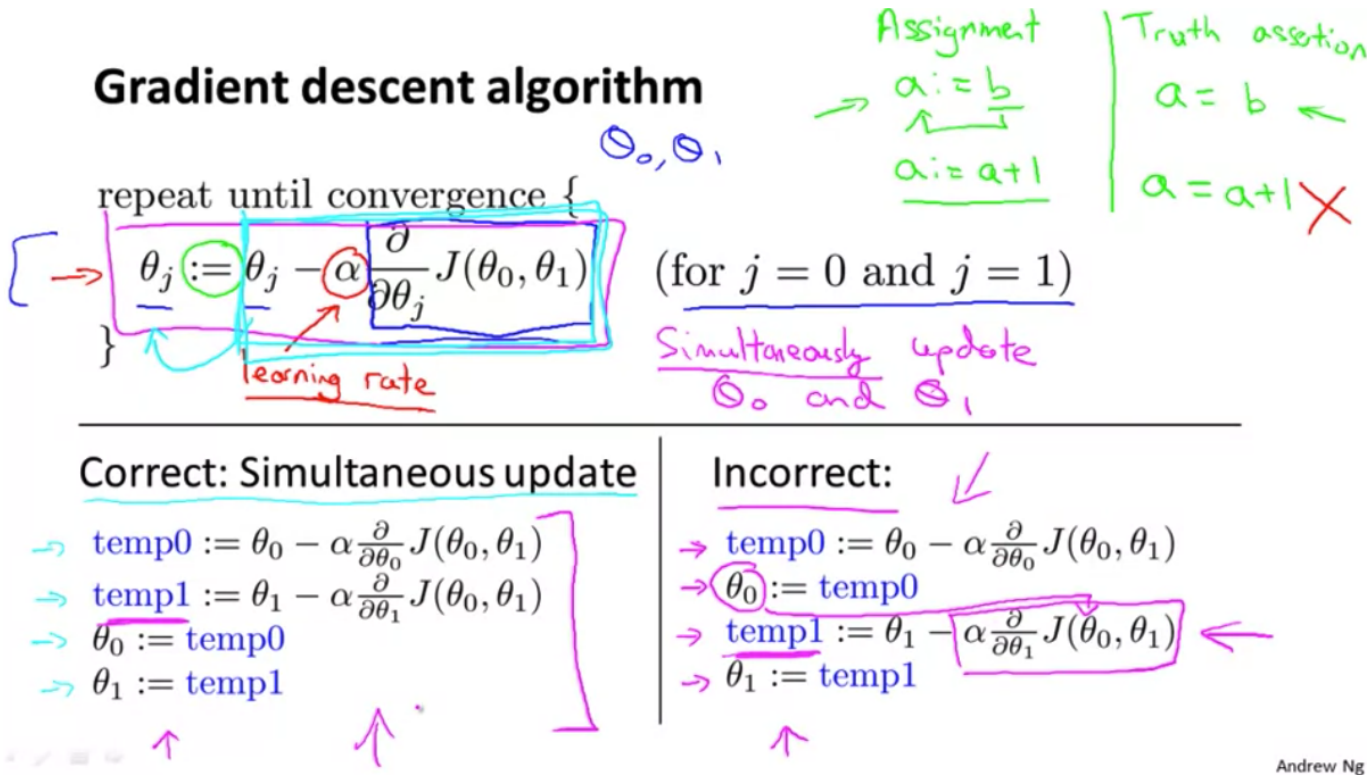
Gradient is a general algorithm.



The outline of Gradient descent algorithm:

Understand the concept between Assignment and Truth assertion.

Simultaneous update both parameters.

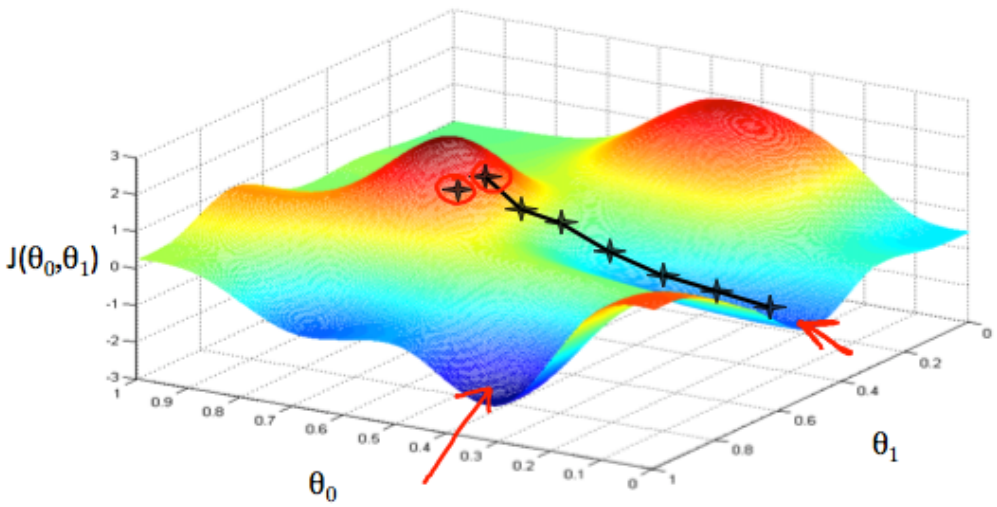


Material:

So we have our hypothesis function and we have a way of measuring how well it fits into the data. Now we need to estimate the parameters in the hypothesis function. That's where gradient descent comes in.

Imagine that we graph our hypothesis function based on its fields θ_0 and θ_1 (actually we are graphing the cost function as a function of the parameter estimates). We are not graphing x and y itself, but the parameter range of our hypothesis function and the cost resulting from selecting a particular set of parameters.

We put θ_0 on the x axis and θ_1 on the y axis, with the cost function on the vertical z axis. The points on our graph will be the result of the cost function using our hypothesis with those specific theta parameters. The graph below depicts such a setup.



We will know that we have succeed when our cost function is at the very bottom of the pits in our graph. when its values is the minimum. The red arrows show the minimum points in the graph.

The way we do is by taking the derivative of our cost function. The slope of the tangent is the derivative at the point and it will give us a direction move towards. We make steps down the cost function in the direction with the steepest descent. The size of each step is determined by the parameter α, which is called the learning rate.

We will know that we have succeeded when our cost function is at the very bottom of the pits in our graph, when its value is the minimum. The red arrows show the minimum points in the graph.

The way we do this is by taking the derivative (the tangential line to a function) of our cost function. The slo of the tangent is the derivative at that point and it will give us a direction to move towards. We make steps down the cost function in the direction with the steepest descent. The size of each step is determined by th parameter α, which is called the learning rate.

For example, the distance between each 'star' in the graph above represents a step determined by our parameter α. A smaller α would result in a smaller step and a larger α results in a larger step. The direction which the step is taken is determined by the partial derivative of $J(\theta_0, \theta_1)$. Depending on where one starts the graph, one could end up at different points. The image above shows us two different starting points th end up in two different places.

The gradient descent algorithm is:

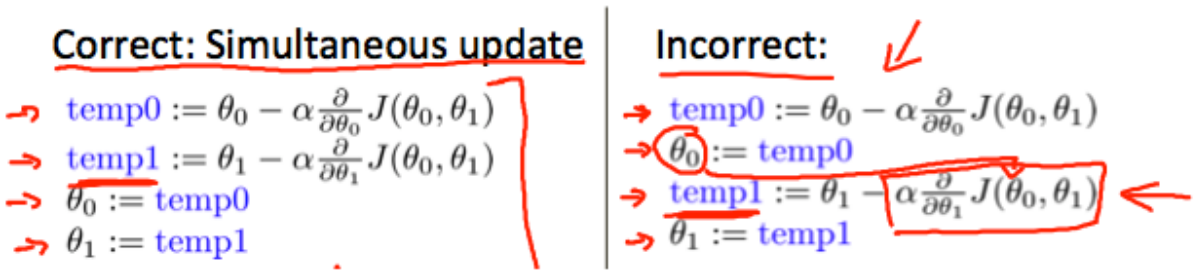
repeat until convergence:

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

where

j=0,1 represents the feature index number.

At each iteration j, one should simultaneously update the parameters $\theta_1, \theta_2, ..., \theta_n$. Updating a specific parameter prior to calculating another one on the $j^{(th)}$ iteration would yield to a wrong implementation.



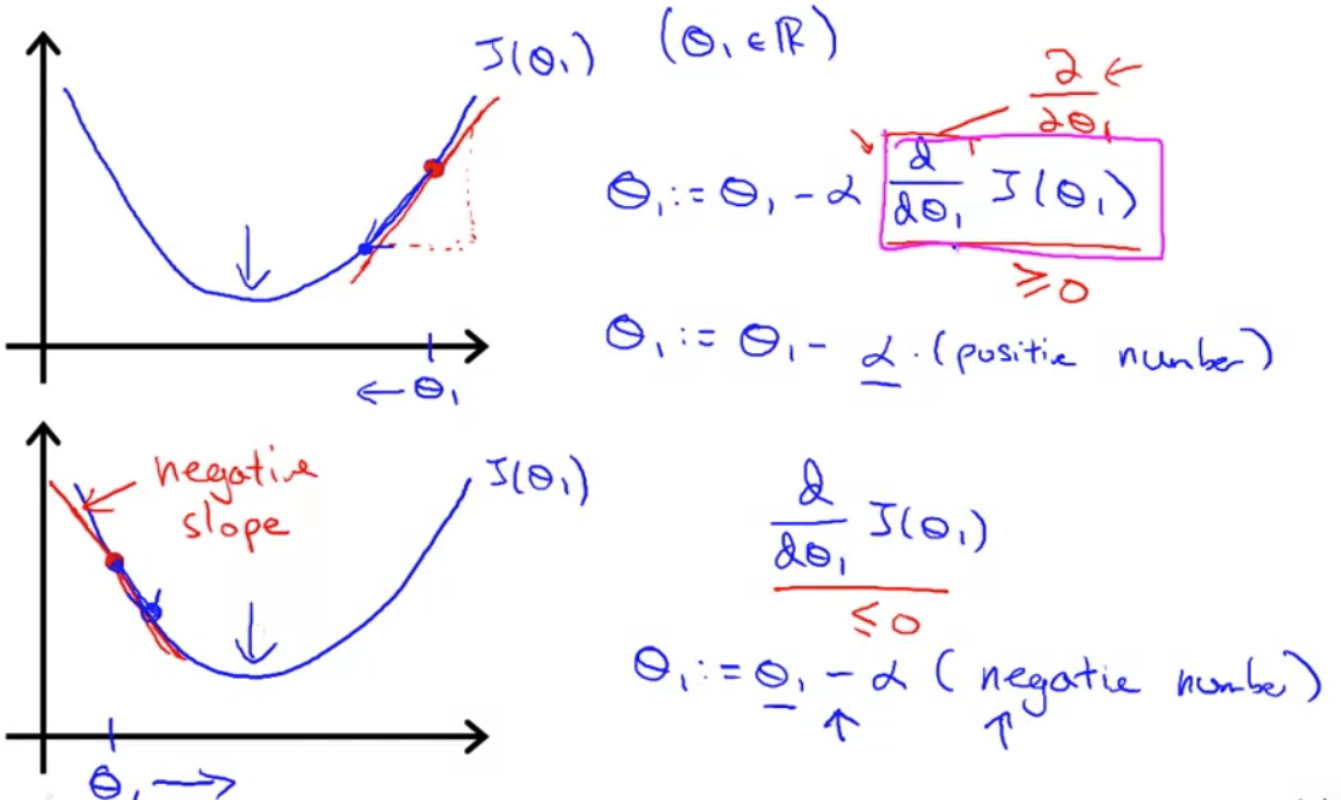
Gradient Descent Intuition

Apply gradient descent to minimize our squared cost function.

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

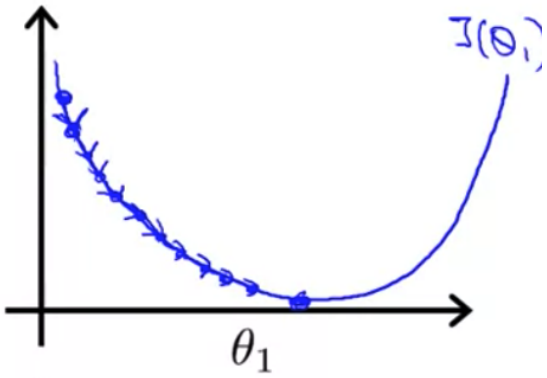
don't need to change the a.

Explain the Gradient descent with one variable:

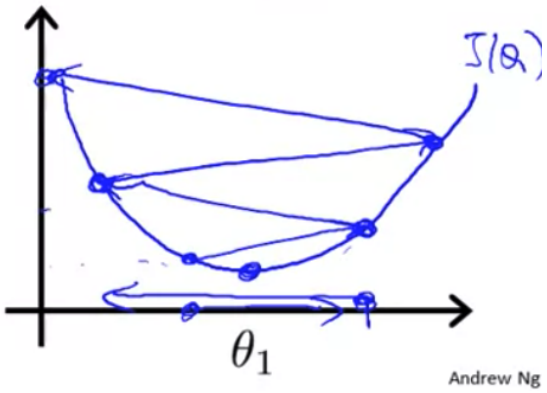


$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

If α is too small, gradient descent can be slow.



If α is too large, gradient descent can overshoot the minimum. It may fail to converge, or even diverge.



Andrew Ng

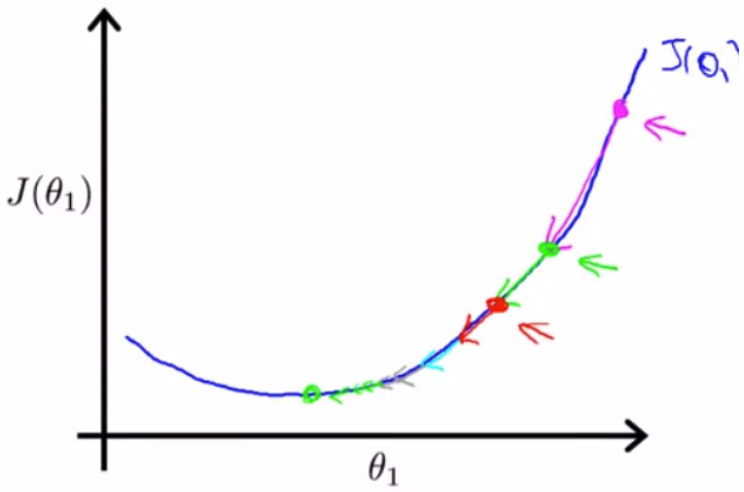
Features of Gradient descent:

When reach the bottom of our convex function, the derivative will always be 0, thus we get the settled *θ1.*

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.

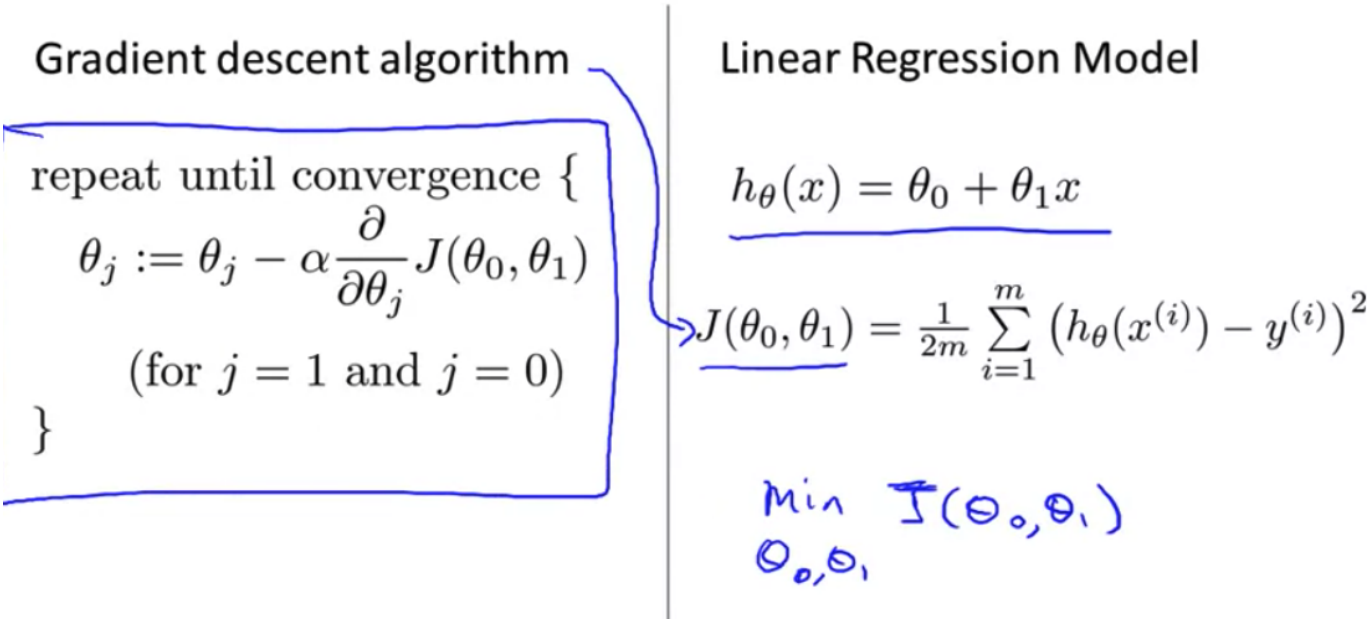


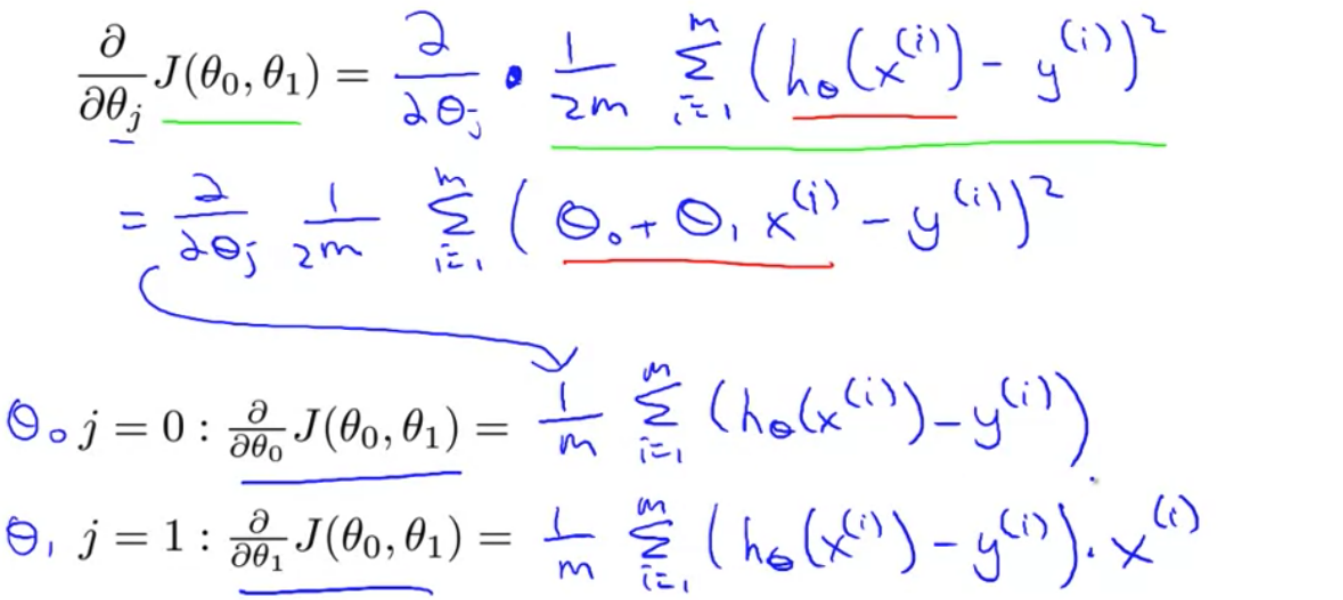
Andrew Ng

Gradient Descent For Linear Regression

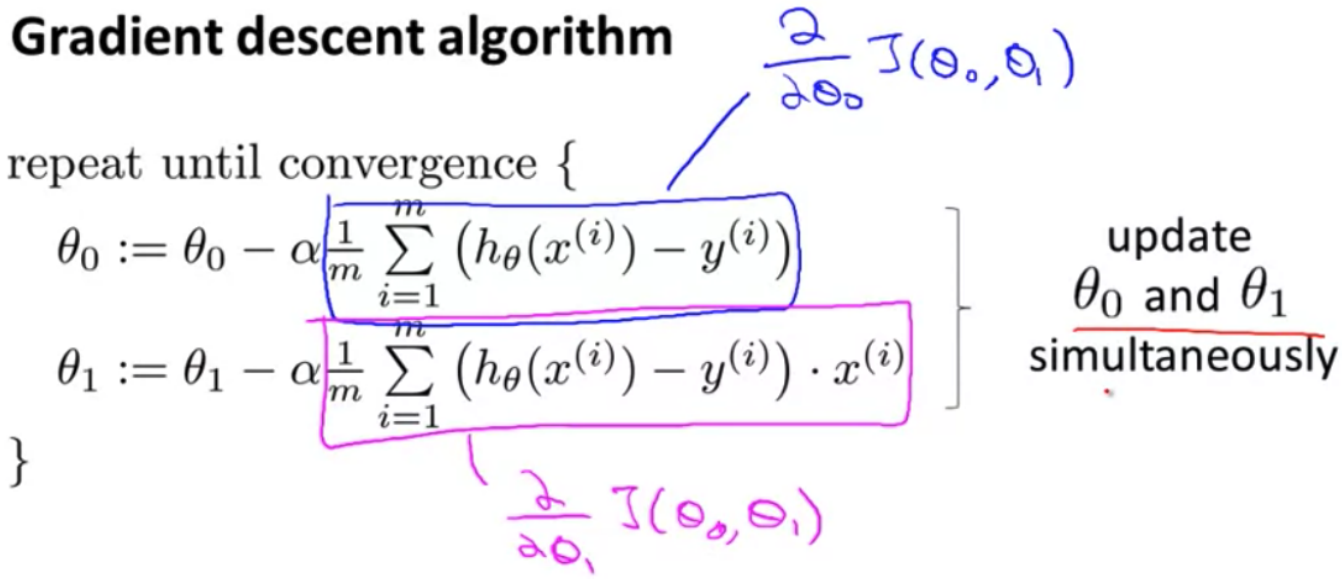
Simplify the function:

When we apply the Gradient descent algorithm with Linear Regression Model, we will derived a new form of the gradient descent equation.





Andrew N



Convex function equals to Bowl shaped

"Batch" Gradient Descent.

Which of the following are true statements? Select all that apply.

- [ ] To make gradient descent converge, we must slowly decrease α over time.
- [ ] Gradient descent is guaranteed to find the global minimum for any function $J(\theta_0, \theta_1)$.
- [x] Gradient descent can converge even if α is kept fixed. (But α cannot be too large, or else it may fail to converge.)

✓ 正确

- [x] For the specific choice of cost function $J(\theta_0, \theta_1)$ used in linear regression, there are no local optima (other than the global optimum).

✓ 正确

Material:

A new form of the gradient descent equation:

repeat until convergence: {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left((h_\theta(x_i) - y_i) x_i \right)$$

}

where m is the size of the training set, θ_0 a constant that will be changing simultaneously with θ_1 and x_i, y_i are values of the given training set (data).

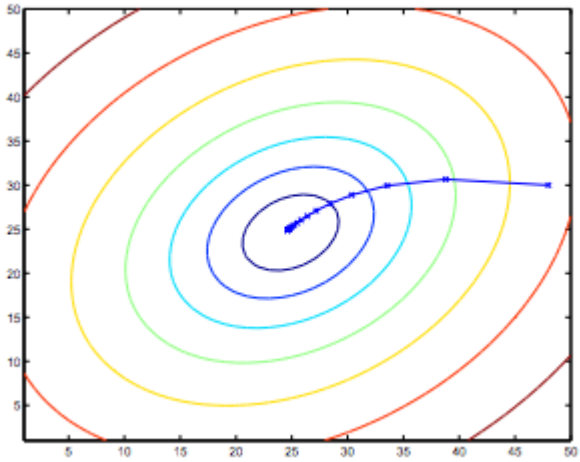
A single example of derivative.

Note that we have separated out the two cases for θ_j into separate equations for θ_0 and θ_1; and that for θ_1 we are multiplying x_i at the end due to the derivative. The following is a derivation of $\frac{\partial}{\partial \theta_j} J(\theta)$ for a single example :

$$\begin{aligned}
\frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{2} (h_\theta(x) - y)^2 \\
&= 2 \cdot \frac{1}{2} (h_\theta(x) - y) \cdot \frac{\partial}{\partial \theta_j} (h_\theta(x) - y) \\
&= (h_\theta(x) - y) \cdot \frac{\partial}{\partial \theta_j} \left(\sum_{i=0}^{n} \theta_i x_i - y \right) \\
&= (h_\theta(x) - y) \, x_j
\end{aligned}$$

So this is simply gradient descent on the original cost function j. This method looks at every example in the entire training set on every step, which is called **batch gradient descent.【批量梯度下降】** Note that, while gradient descent can be susceptible to **local minima** in general. the optimization problem we have posed here for linear regression has only on global, and no other local, optima; thus gradient descent always converges to the global minimum.

Indeed, j is a **convex quadratic function**. Here is an example of gradient descent as it run to minimize a quadratic function.



The ellipses shown above are the contours of a quadratic function. Also shown is the trajectory taken by gradient descent, which was initialized at (48,30). The x's in the figure (joined by straight lines) mark the successive values of θ that gradient descent went through as it converged to its minimum.

Rating Test:

1. Consider the problem of predicting how well a student does in her second year of college/university, given how well she did in her first year. 1 分

   Specifically, let x be equal to the number of "A" grades (including A-. A and A+ grades) that a student receives in their first year of college (freshmen year). We would like to predict the value of y, which we define as the number of "A" grades they get in their second year (sophomore year).

   Here each row is one training example. Recall that in linear regression, our hypothesis is $h_\theta(x) = \theta_0 + \theta_1 x$, and we use m to denote the number of training examples.

x	y
3	2
1	2
0	1
4	3

   For the training set given above (note that this training set may also be referenced in other questions in this quiz), what is the value of m? In the box below, please enter your answer (which should be a number between 0 and 10).

   4

2. For this question, assume that we are 1 分

using the training set from Q1. Recall our definition of the

cost function was $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$.

What is $J(0, 1)$? In the box below,

please enter your answer (Simplify fractions to decimals when entering answer, and '.' as the decimal delimiter e.g., 1.5).

0.5

3. Suppose we set $\theta_0 = 0, \theta_1 = 1.5$ in the linear regression hypothesis from Q1. What is $h_\theta(2)$? 1 分

3

4. Let f be some function so that

$f(\theta_0, \theta_1)$ outputs a number. For this problem,

f is some arbitrary/unknown smooth function (not necessarily the

cost function of linear regression, so f may have local optima).

Suppose we use gradient descent to try to minimize $f(\theta_0, \theta_1)$

as a function of θ_0 and θ_1. Which of the

following statements are true? (Check all that apply.)

- [ ] Setting the learning rate α to be very small is not harmful, and can only speed up the convergence of gradient descent.
- [ ] No matter how θ_0 and θ_1 are initialized, so long as α is sufficiently small, we can safely expect gradient descent to converge to the same solution.
- [ ] If the first few iterations of gradient descent cause $f(\theta_0, \theta_1)$ to **increase** rather than decrease, then the most likely cause is that we have set the learning rate α to too large a value.
- [ ] If θ_0 and θ_1 are initialized at the global minimum, then one iteration will not change their values.

Choose C and D

5. Suppose that for some linear regression problem (say, predicting housing prices as in the lecture), we have some training set, and for our training set we managed to find some θ_0, θ_1 such that $J(\theta_0, \theta_1) = 0$. 1 分

Which of the statements below must then be true? (Check all that apply.)

- [x] For these values of θ_0 and θ_1 that satisfy $J(\theta_0, \theta_1) = 0$,

  we have that $h_\theta(x^{(i)}) = y^{(i)}$ for every training example $(x^{(i)}, y^{(i)})$
- [x] This is not possible: By the definition of $J(\theta_0, \theta_1)$, it is not possible for there to exist

  θ_0 and θ_1 so that $J(\theta_0, \theta_1) = 0$
- [ ] For this to be true, we must have $\theta_0 = 0$ and $\theta_1 = 0$

  so that $h_\theta(x) = 0$
- [ ] We can perfectly predict the value of y even for new examples that we have not yet seen.

  (e.g., we can perfectly predict prices of even new houses that we have not yet seen.)

5. Suppose that for some linear regression problem (say, predicting housing prices as in the lecture), we have some training set, and for our training set we managed to find some θ_0, θ_1 such that $J(\theta_0, \theta_1) = 0$. 1 分

Which of the statements below must then be true? (Check all that apply.)

- [ ] Gradient descent is likely to get stuck at a local minimum and fail to find the global minimum.
- [ ] For this to be true, we must have $y^{(i)} = 0$ for every value of $i = 1, 2, \ldots, m$.
- [ ] For this to be true, we must have $\theta_0 = 0$ and $\theta_1 = 0$

  so that $h_\theta(x) = 0$
- [x] Our training set can be fit perfectly by a straight line,

  i.e., all of our training examples lie perfectly on some straight line.

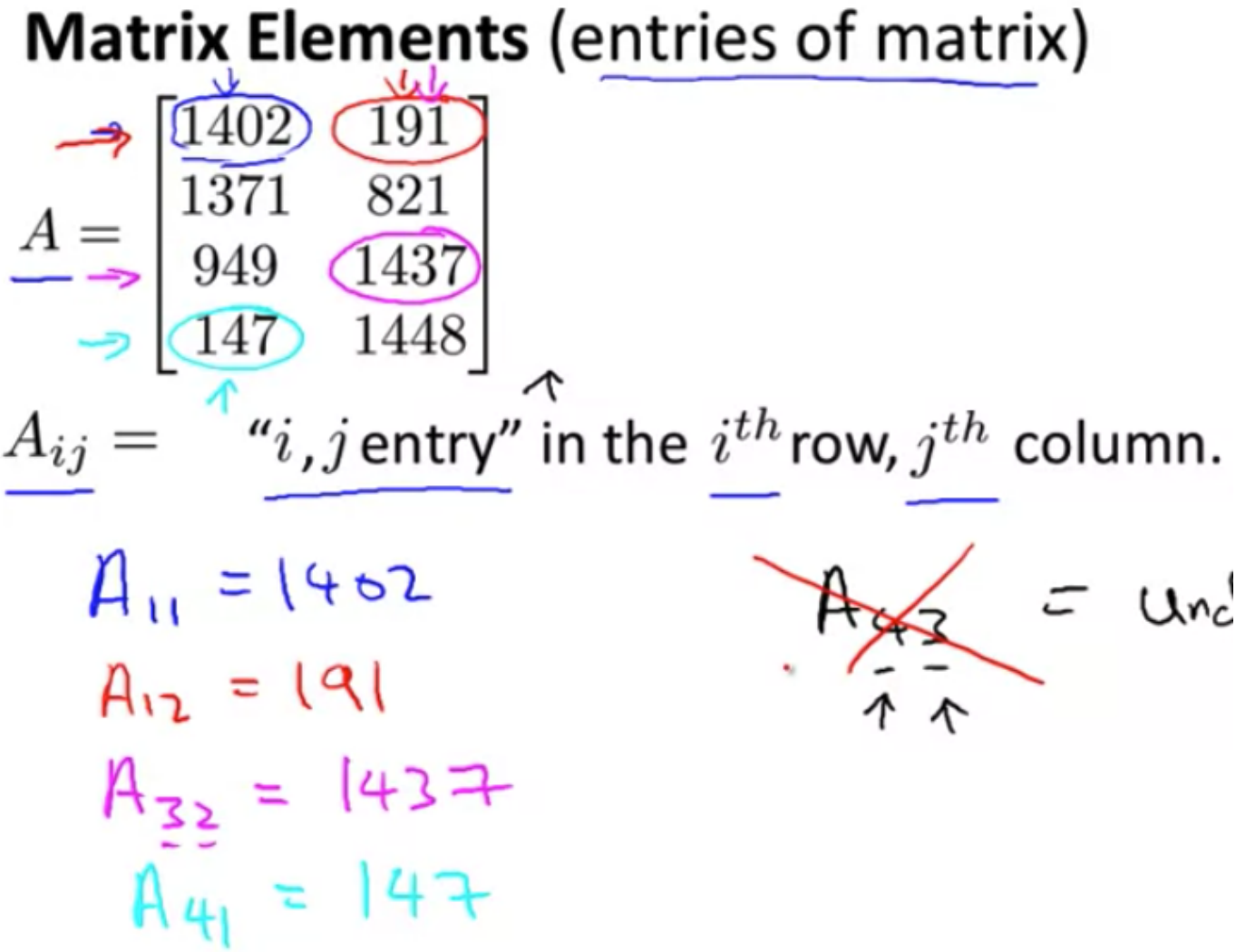
Answers

Linear Algebra Review

Matrices and Vectors

Matrix: Rectangular array of numbers.

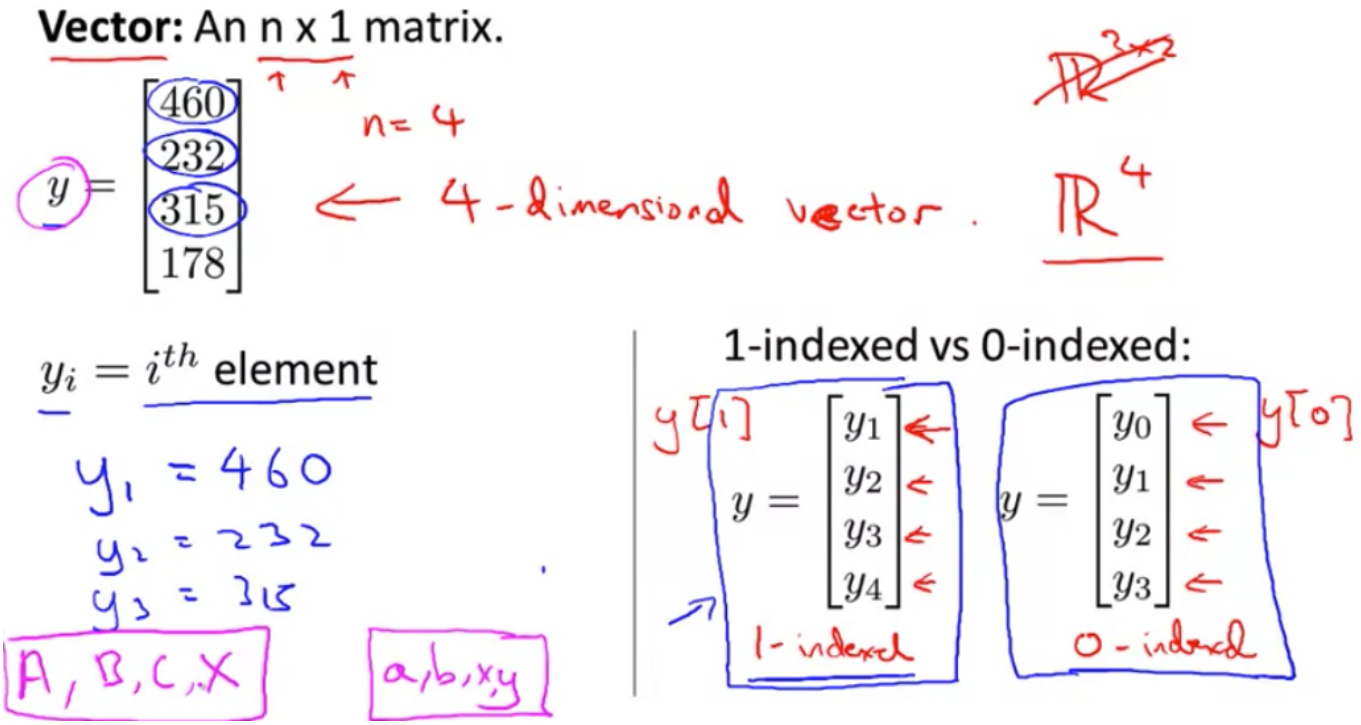
Matrices are 2-dimensional arrays.



Vector: An n x 1 matrix.

A vector is a matrix with one column and many rows.

So vectors are a **subset** of matrices.



Notation and terms[Essential]

Notation and terms:

- A_{ij} refers to the element in the ith row and jth column of matrix A.
- A vector with 'n' rows is referred to as an 'n'-dimensional vector.
- v_i refers to the element in the ith row of the vector.
- In general, all our vectors and matrices will be 1-indexed. Note that for some programming languages, the arrays are 0-indexed.
- Matrices are usually denoted by uppercase names while vectors are lowercase.
- "Scalar" means that an object is a single value, not a vector or matrix.
- $\mathbb{R}$ refers to the set of scalar real numbers.
- $\mathbb{R}^n$ refers to the set of n-dimensional vectors of real numbers.

Matlab commands below:

```
% The ; denotes we are going back to a new row.
A = [1, 2, 3; 4, 5, 6; 7, 8, 9; 10, 11, 12]

% Initialize a vector
v = [1;2;3]

% Get the dimension of the matrix A where m = rows and n = columns
[m,n] = size(A)

% You could also store it this way
dim_A = size(A)

% Get the dimension of the vector v
dim_v = size(v)

% Now let's index into the 2nd row 3rd column of matrix A
A_23 = A(2,3)
```

Addition and Scalar Multiplication

Outline:

Matrix addition and subtraction, and how to multiply a matrix by a number. SO called Scalar Multiplication.

Matrix Addition:

Matrix Addition

$$\begin{bmatrix} 1 & 0 \\ 2 & 5 \\ 3 & 1 \end{bmatrix} + \begin{bmatrix} 4 & 0.5 \\ 2 & 5 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 5 & 0.5 \\ 4 & 10 \\ 3 & 2 \end{bmatrix}$$

3 × 2 matrix

3 × 2

3 × 2

$$\begin{bmatrix} 1 & 0 \\ 2 & 5 \\ 3 & 1 \end{bmatrix} + \begin{bmatrix} 4 & 0.5 \\ 2 & 5 \end{bmatrix} = \text{error}$$

3 × 2

2 × 2

Scalar Multiplication:

Scalar Multiplication

real number

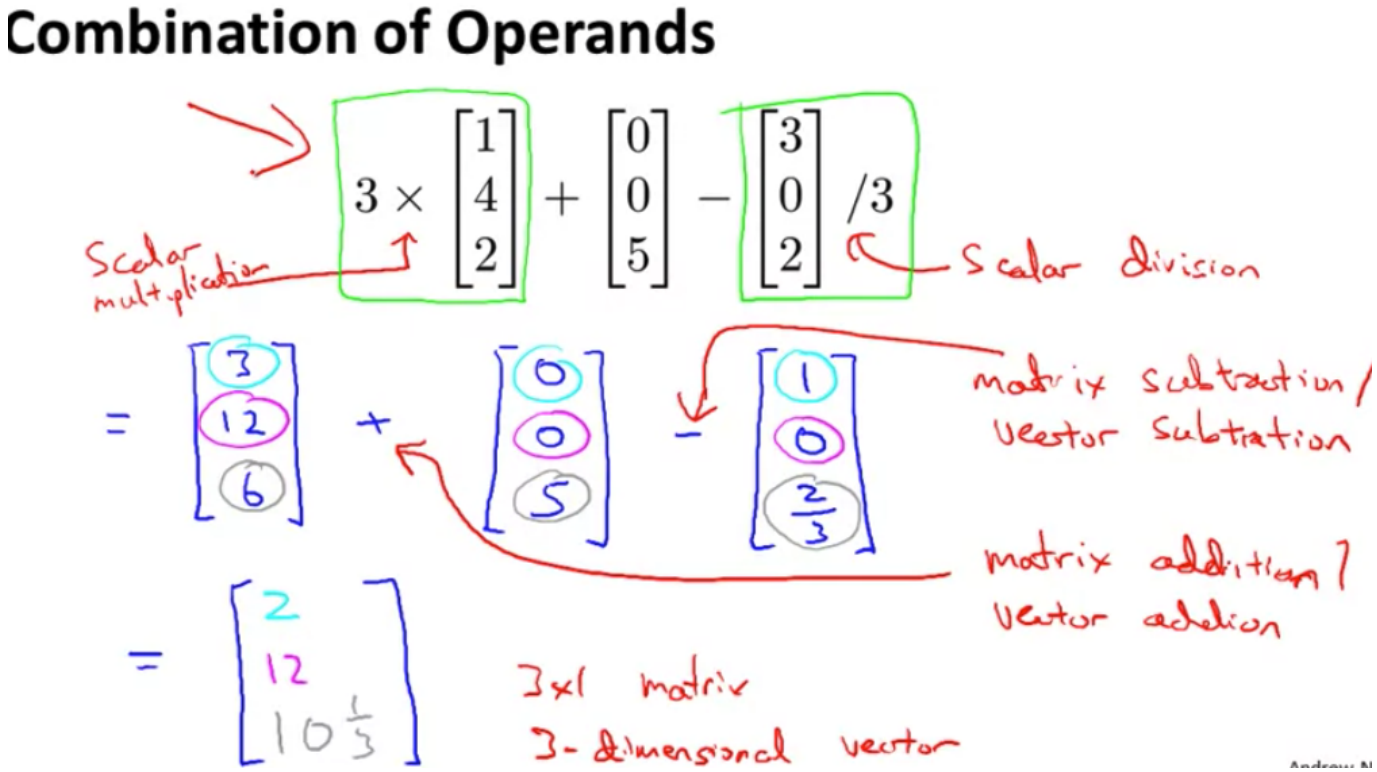
$$3 \times \begin{bmatrix} 1 & 0 \\ 2 & 5 \\ 3 & 1 \end{bmatrix} = \begin{bmatrix} 3 & 0 \\ 6 & 15 \\ 9 & 3 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 2 & 5 \\ 3 & 1 \end{bmatrix} \times 3$$

3 × 2

3 × 2

$$\begin{bmatrix} 4 & 0 \\ 6 & 3 \end{bmatrix} /4 = \frac{1}{4} \begin{bmatrix} 4 & 0 \\ 6 & 3 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \frac{3}{2} & \frac{3}{4} \end{bmatrix}$$

Combination of Operands:



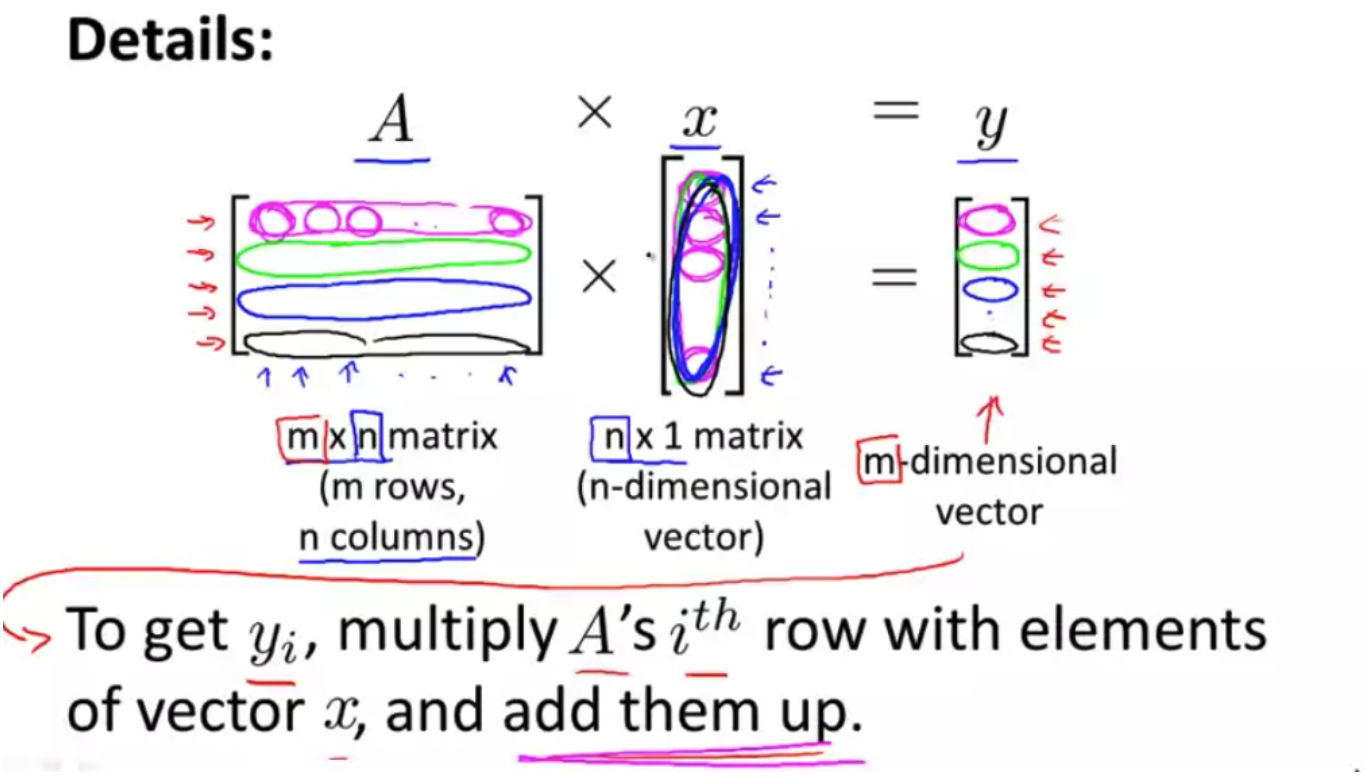
Matlab command:

```
% Initialize matrix A and B
A = [1, 2, 4; 5, 3, 2]
B = [1, 3, 4; 1, 1, 1]

% Initialize constant s
s = 2

% See how element-wise addition works
add_AB = A + B

% See how element-wise subtraction works
sub_AB = A - B

% See how scalar multiplication works
mult_As = A * s

% Divide A by s
div_As = A / s

% What happens if we have a Matrix + scalar?
add_As = A + s

A =
   1   2   4
   5   3   2

B =
   1   3   4
   1   1   1
s =  2

add_AB =
   2   5   8
   6   4   3

sub_AB =
   0  -1   0
   4   2   1

mult_As =
    2    4    8
   10    6    4

div_As =
   0.50000   1.00000   2.00000
   2.50000   1.50000   1.00000
```

```
add_As =
   3   4   6
   7   5   4
```

Matrix Vector Multiplication

The number of columns of matrix must match the number of rows of vector.

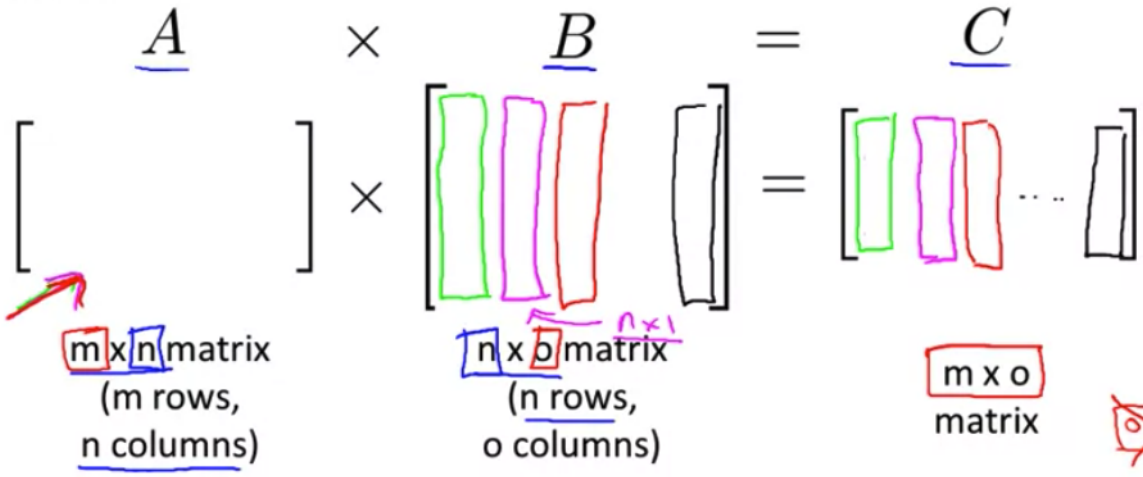


m x n matrix multiply n x 1 matrix [n-dimensional vector] = m-dimensional vector

Matrix Matrix Multiplication

Details:

Details:



The i^{th} column of the matrix C is obtained by multiplying A with the i^{th} column of B. (for i = 1,2,...,o)

Andrew N

Apply:

House sizes:

$$\begin{Bmatrix} 2104 \\ 1416 \\ 1534 \\ 852 \end{Bmatrix}$$

Have 3 competing hypotheses:

1. $h_\theta(x) = -40 + 0.25x$
2. $h_\theta(x) = 200 + 0.1x$
3. $h_\theta(x) = -150 + 0.4x$

Matrix

$$\begin{bmatrix} 1 & 2104 \\ 1 & 1416 \\ 1 & 1534 \\ 1 & 852 \end{bmatrix}$$

Matrix

$$\times \begin{bmatrix} -40 & 200 & -150 \\ 0.25 & 0.1 & 0.4 \end{bmatrix} = \begin{bmatrix} 486 & 410 & 692 \\ 314 & 342 & 416 \\ 344 & 353 & 464 \\ 173 & 285 & 191 \end{bmatrix}$$

Prediction of first ho

Predictions of 2nd ho

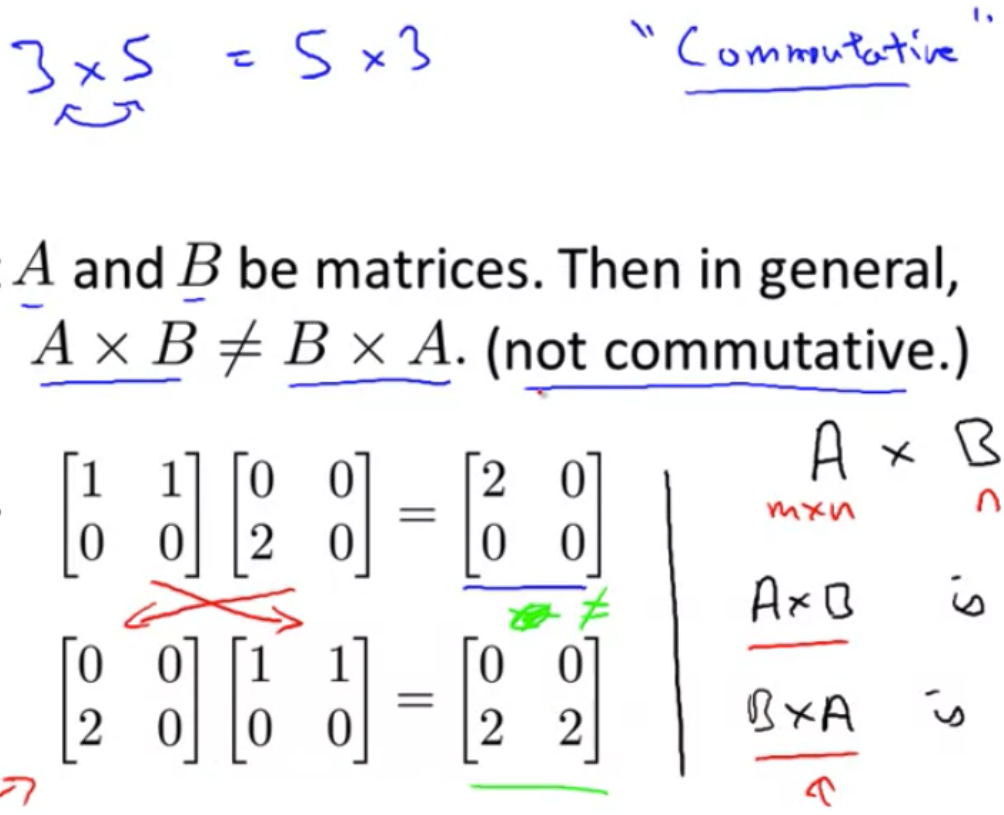
Andrew

Matlab command:

```
% Initialize a 3 by 2 matrix
A = [1, 2; 3, 4;5, 6]

% Initialize a 2 by 1 matrix
B = [1; 2]

% We expect a resulting matrix of (3 by 2)*(2 by 1) = (3 by 1)
mult_AB = A*B

% Make sure you understand why we got that result
A =
     1     2
     3     4
     5     6

B =
     1
     2

mult_AB =
     5
    11
    17
```

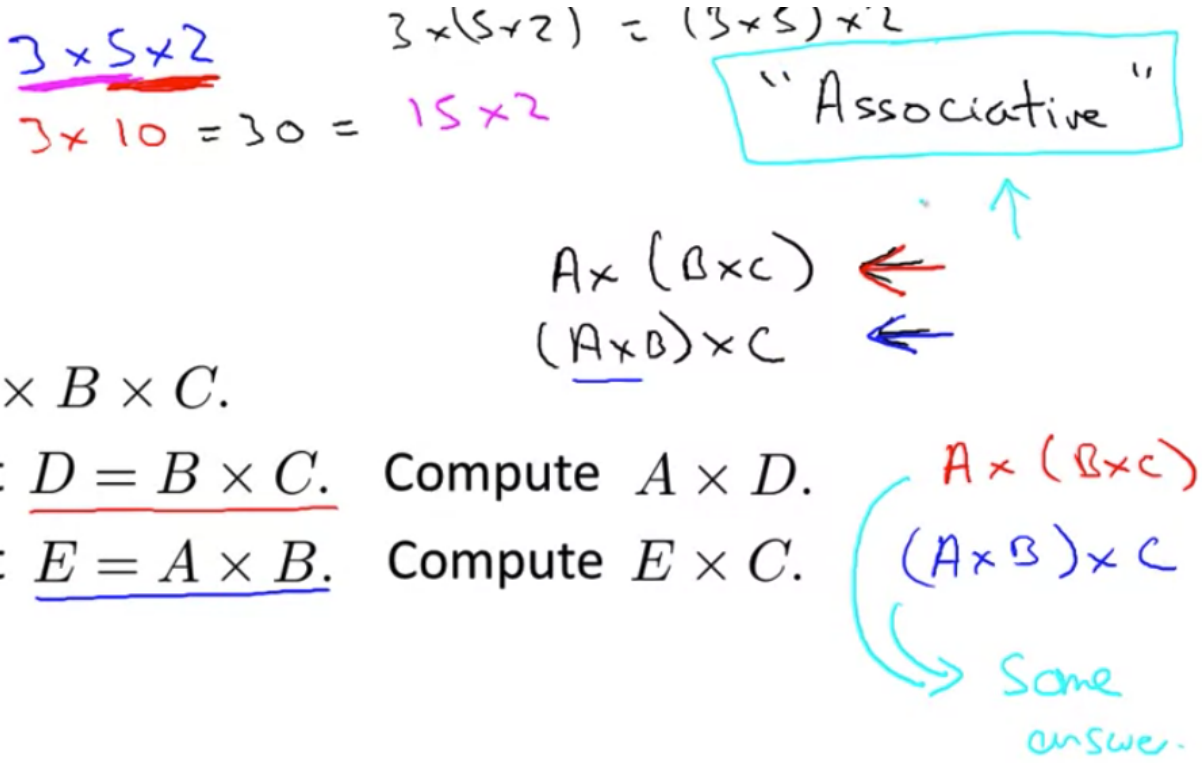
Matrix Multiplication Properties

Not commutative



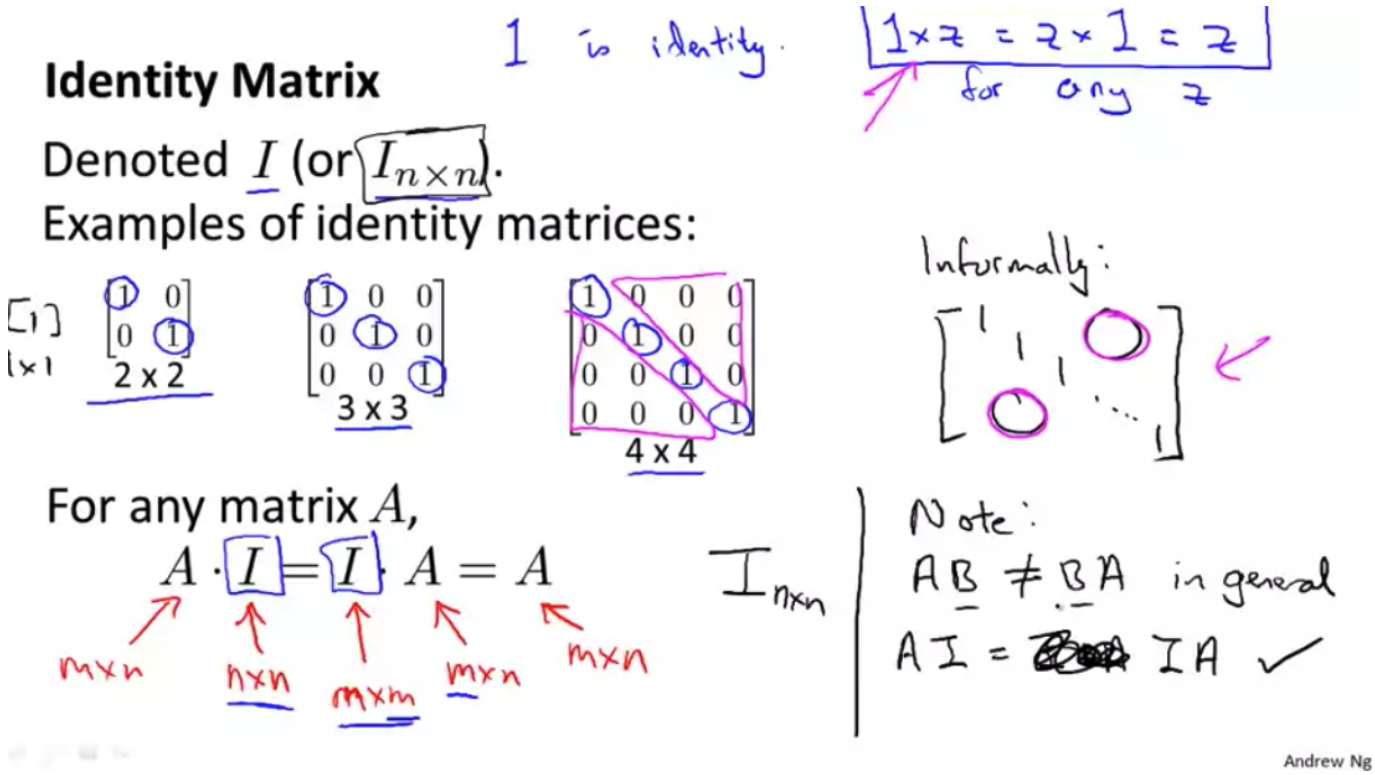
Associative

(A * B) * C = A * (B * C)



Identity Matrix:

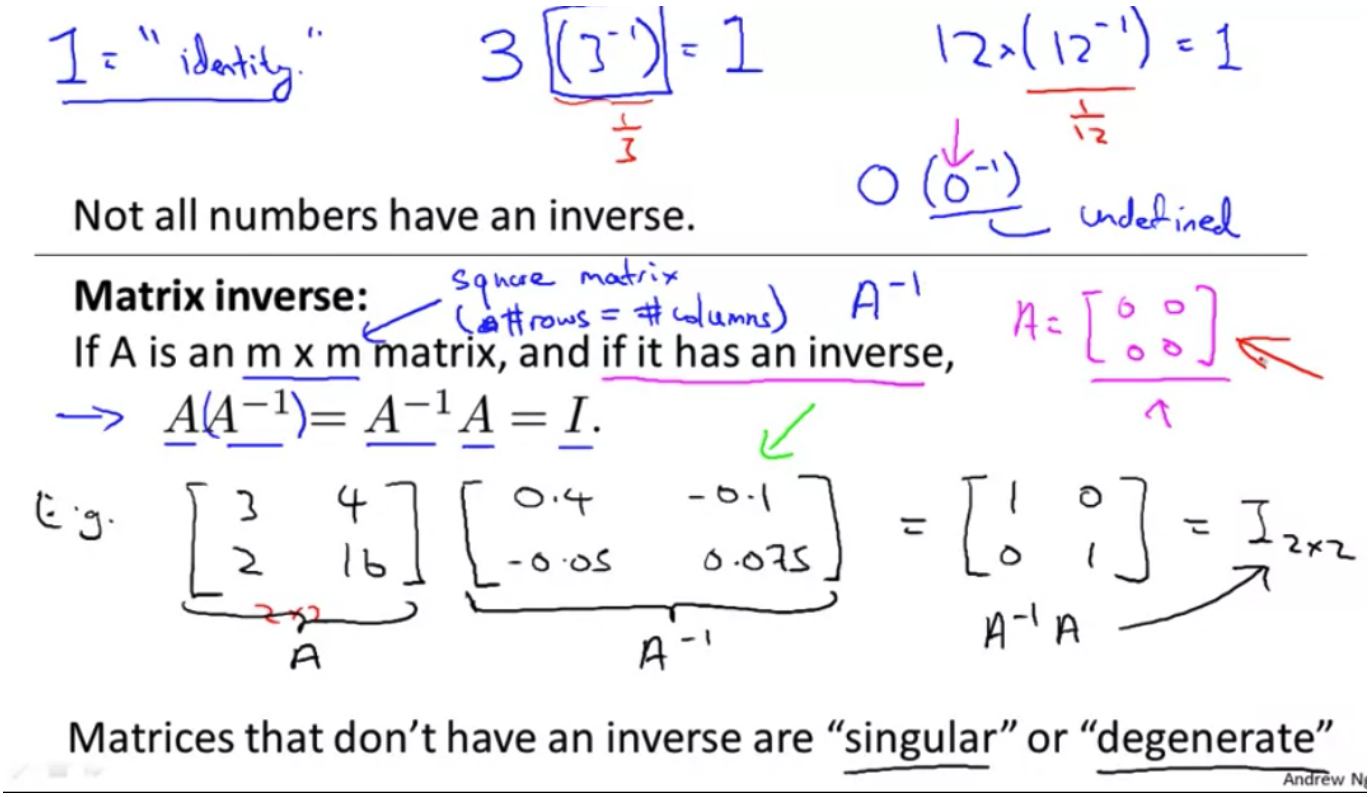
The identity matrix of size n is the n x n square matrix **with ones on the main diagonal and zeros elsewhere. Details of Identity Matrix:**



Inverse and Transpose

Start by how it relates to real numbers.

Inverse Matrix:



- Matrices that don't have an inverse are **"singular" or "degenerate"**

So how can we get the inverse matrix?

Sometimes you can compute inverses by hand but almost no one does that those day. It turns out there is very good **numerical software** for taking a matrix and computing.

Matrix Transpose:

The transpose of a matrix is an operator which flips a matrix over its diagonal.

Matrix Transpose

Example: $A = \begin{bmatrix} 1 & 2 & 0 \\ 3 & 5 & 9 \end{bmatrix}$ $B = A^T = \begin{bmatrix} 1 & 3 \\ 2 & 5 \\ 0 & 9 \end{bmatrix}$

2×3 3×2

Let A be an m x n matrix, and let $B = A^T$.
Then B is an n x m matrix, and

$$B_{ij} = A_{ji}.$$

$B_{12} = A_{21} = 2$

$B_{32} = 9$ $A_{23} = 9$.

Matlab Command

```
% Initialize matrix A
A = [1,2,0;0,5,6;7,0,9]

% Transpose A
A_trans = A'

% Take the inverse of A
A_inv = inv(A)

% What is A^(-1)*A?
A_invA = inv(A)*A

A =
     1     2     0
     0     5     6
     7     0     9
A_trans =

     1     0     7
     2     5     0
     0     6     9
A_inv =

    0.3488   -0.1395    0.0930
    0.3256    0.0698   -0.0465
   -0.2713    0.1085    0.0388
A_invA =

    1.0000   -0.0000    0.0000
    0.0000    1.0000   -0.0000
   -0.0000         0    1.0000
```

FAQ for Week1:

Q1: In the cost function, why don't we use absolute value(or mod(), or some other function) instead of the squared error?

The absolute value has some bad characteristics for minimization.

- The gradient is not continuous because the absolute value function is not differentiable at its minimum point.
- It does not emphasize the correction of large errors.

- The abs() function is also not very mathematically efficient.

However, the squared error cost function has some very desirable characteristics:

- The cost can be computed very efficiently.
- Its partial derivative is easily computed.
- Its partial derivative is continuous.

Q2: How does the Cocktail Party problem work?

It isn't clustering.

Clustering would not preserve the time-sequence of the sound samples, so the results would not sound like speech.

The method used is Principal Component Analysis. This method is a mathematical trick that takes two sets of correlated data, and returns two new sets of data that are not correlated. The sequence of the data is retained.

If the data is audio recordings, the result has the effect to the human ear of sounding like two separate audio tracks.