Table of Contents

[Week 1 Review 2](#_Toc516235652)

[Welcome 2](#_Toc516235653)

[What is machine learning? 2](#_Toc516235654)

[Supervised learning 2](#_Toc516235655)

[Regression 2](#_Toc516235656)

[Classifications 2](#_Toc516235657)

[Unsupervised learning 3](#_Toc516235658)

[Model Representation 3](#_Toc516235659)

[Cost Function 4](#_Toc516235660)

[Cost Function - Intuition I 5](#_Toc516235661)

[Linear Algebra Review 6](#_Toc516235662)

[Matrix and Vectors 6](#_Toc516235663)

[Matrix 6](#_Toc516235664)

[Vector: An nx1 Matrix 6](#_Toc516235665)

[Matrix Math 6](#_Toc516235666)

[Addition 6](#_Toc516235667)

[Multiplication (Scalar (real number)) 7](#_Toc516235668)

[Matrix Matrix Multiplication 7](#_Toc516235669)

[Matrix Vector Multiplication 7](#_Toc516235670)

[Housing price trick with Matrix: 7](#_Toc516235671)

[Inverse 8](#_Toc516235672)

[Transpose 8](#_Toc516235673)

[Parameter Learning 8](#_Toc516235674)

[Gradient Descent 8](#_Toc516235675)

[Gradient Descent Intuition 8](#_Toc516235676)

[Week 2 – Multivariate Linear Regression 8](#_Toc516235677)

[Multiple Features 8](#_Toc516235678)

[Notation: 9](#_Toc516235679)

[Hypothesis 9](#_Toc516235680)

[Gradient Descent for Multiple Variables 9](#_Toc516235681)

[Gradient Descent in Practice 1 – Feature Scaling 10](#_Toc516235682)

[Gradient Descent in Practice II – Learning Rate 11](#_Toc516235683)

[Features and Polynomial Regression 11](#_Toc516235684)

[Housing Prices Prediction 11](#_Toc516235685)

[Computing Parameters Analytically 12](#_Toc516235686)

[Normal Equation 12](#_Toc516235687)

[Gradient DescentVrs Normal Equation 13](#_Toc516235688)

[Normal Equation Noninvertibility 13](#_Toc516235689)

[What if XTX is non-invertible? 13](#_Toc516235690)

# Week 1 Review

## Welcome

### What is machine learning?

1. Arthur Samuel = "The field of study that gives computers the ability to learn without being explicitly programmed"

2. Tom Mitchell = "A computer program is said to learn from experience E with respect to some class of tasks T and performance measured P,

If its performance as tasks in T, as measured by P, improves with experience E."

Example: 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

In general, any ML problem can be assigned to one of the two broad classifications

Supervised learning

Example: "right answers" given

Regression: predict continuous valued output (price) = housing price predict

Classifications: discrete valued output (0 or 1) = breast cancer (malignant, benign)

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 input and output.

Example 1:

Given data about the size of houses on a 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 instead of 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 basis of the given picture

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

Unsupervised learning

Example: news.google.com --> cluster articles together

Organize computing clusters, social network analysis, market segmentation, astronomical data analysis

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 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.

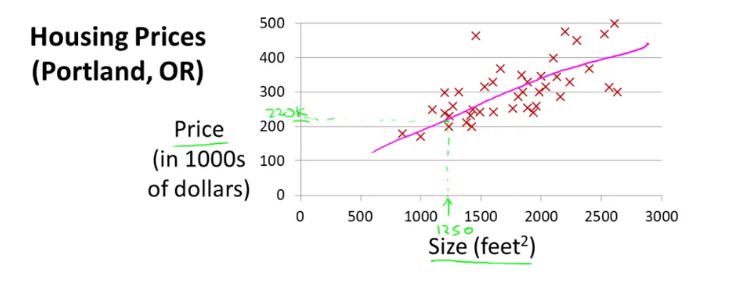
Example:

Clustering: Take a collection of 1m 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, and roles and so on.

Non-clustering: The "Cocktail Party Algorithm”, allows you to find structure in a chaotic environment.

## Model Representation

House Prices = Supervised learning, regression (predict real-valued output)



|  |  |
| --- | --- |
| Size in Feet2 (X) | Price ($) in 1000s (y) |
| 2104 | 460 |
| 1416 | 232 |
| 1534 | 315 |
| 852 | 178 |

Training set of size in feet squared and price in 1,000s

Notation:

m = Number of training examples

x's = "input" variable/features = size in feet

y's = "output" variable /"target" variable = price in 1,000s

(x,y) - one training example

(xi, yi) - ith training example = i is a index, not power of i

x1, y1 = 2104, 460

Training Set --> Learning Algorithm --> hypothesis

Input size of house and output the price of the house

h maps from x's and y's

Linear regression with one variable

Univariate linear regression

### Cost Function

Squared error Function - used a lot in regression for cost function.

Cost function = 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 results of the hypothesis with inputs from x's and the actual output y's.

Training Set

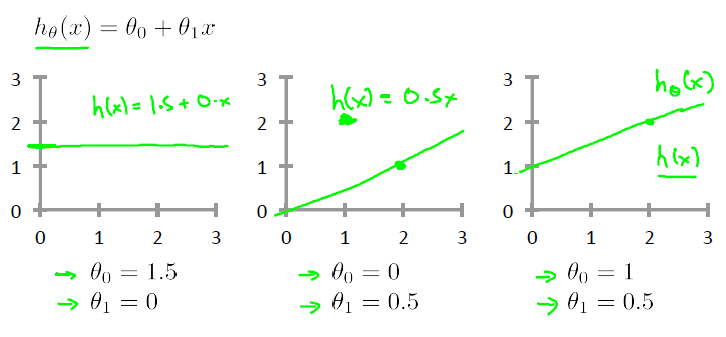
|  |  |
| --- | --- |
| Size in feet2 (x) | Price ($) in 1000’s (y) |
| 2104 | 460 |
| 1416 | 232 |
| 1534 | 315 |
| 852 | 178 |
| …… | ……. |

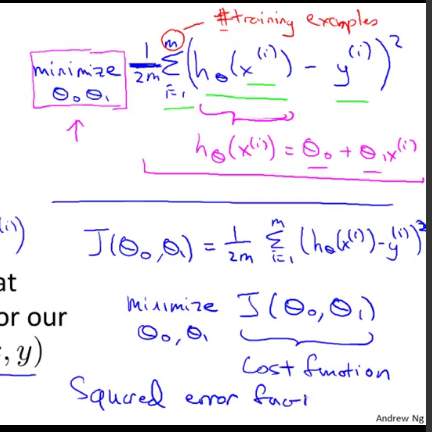
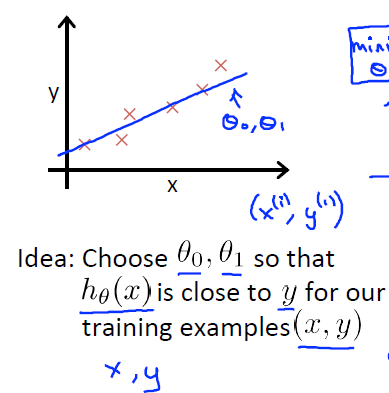
m = 47

hΘ(x) = Θ0 + Θ1x

Θi’s: Parameters

How to choose Θi’s?





### Cost Function - Intuition I

|  |  |
| --- | --- |
|  | **Simplified** |
| Hypothesis: |  |
| hΘ(x) = Θ0 + Θ1x | hΘ(x) = Θ1x |
| Parameters: |  |
| Θ0, Θ1 | Θ1 |
| Cost Function: |  |
|  |  |

y = mx + b

Points:

(1, 1), (2, 2), (3, 3)

J (Θ0.5) = (1/ (2m)) [(0.5-1)2 + (0.5-2)2 + (0.5-3) 2]

= (1 / (2\*3)) (3\*5)

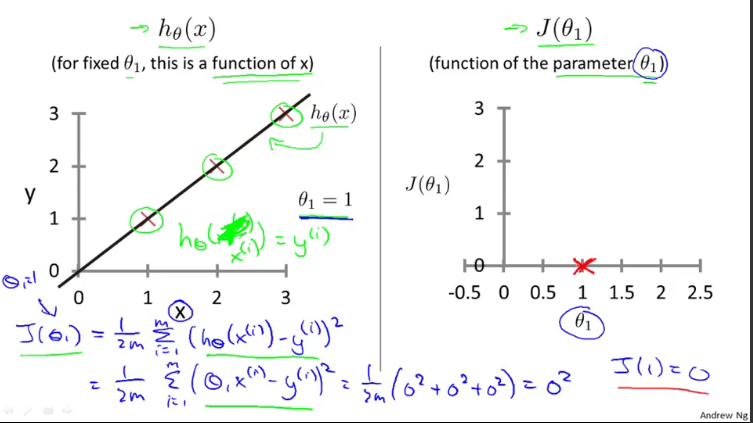
= (3\*5)/6

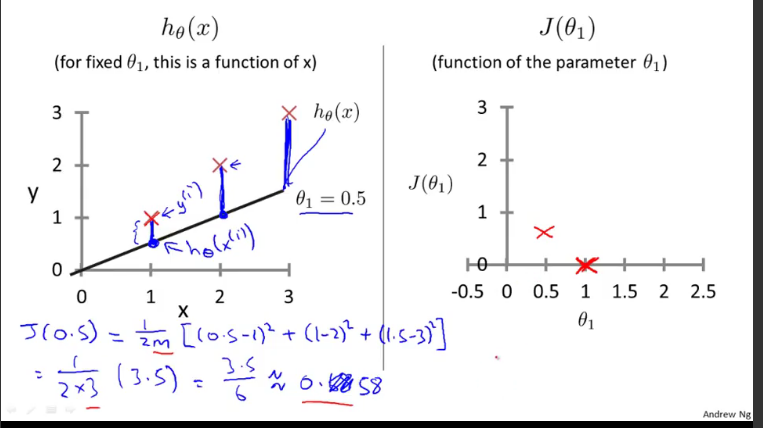
= 0.58

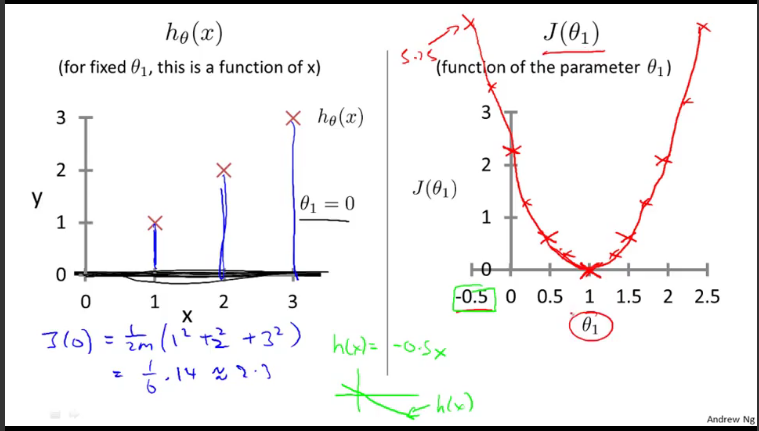
J (theta 0) = (1/ (2m)) (12 + 22 +32)

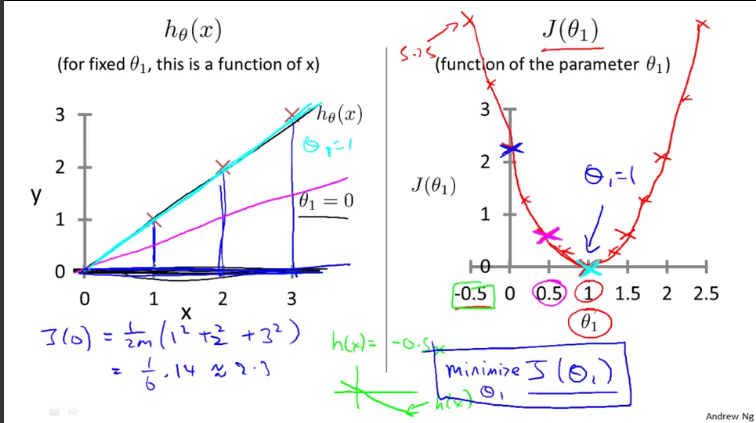
= (1/6) \* 14

= 2.3

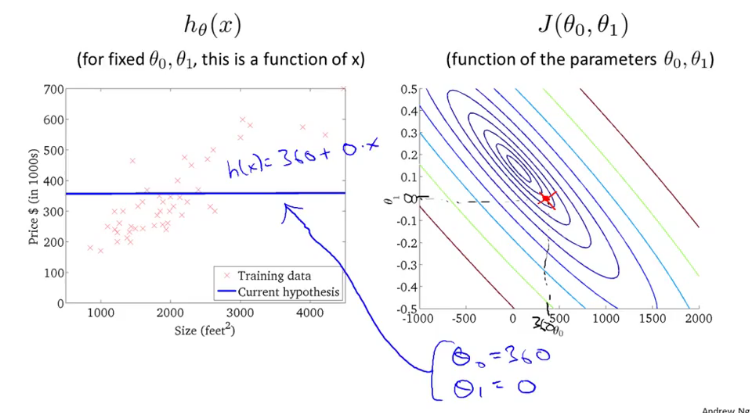








### Cost Function - Intuition II



Gradient Descent

Assignment

a := b --> a := a+1

Truth Assertion

a = b --> a = a+1 (wrong)

Gradient Descent Intuition

As we approach a local minimum, gradient descent will automatically take smaller steps.

So, no need to decrease theta over time.

## Linear Algebra Review

### Matrix and Vectors

#### Matrix

4x2 Matrix 2x3 Matrix

Dimension of Matrix: number of rows X number of columns

Matrix Elements (entries of matrix)

A = Aij = “I, j entry” in the ith row, jth column  
 A11 = 1402; A12 = 191

#### Vector: An nx1 Matrix

y = n = 4

4 Dimensional Vector

y1 = 460 y2 = 232

### Matrix Math

#### Addition

+ = =

*Can only add matrix of same size*

#### Multiplication (Scalar (real number))

X = =

*3x2 matrix multi 2x1 matrix = 3x1 matrix*

Subtract

Divide

#### Matrix Matrix Multiplication

= (see below with logic)

= =

= =

#### Matrix Vector Multiplication

= =

### Housing price trick with Matrix:

House sizes: 2104, 1416, 1534, 852

h Θ (x) = -40 + 0.25x

X = =

Prediction = Data Matrix \* parameters

*This is a lot faster and simpler code than for loop.*

More complex example of this:

House sizes = 2104, 1416, 1534, 852

3 competing hypotheses:

1. h Θ(x) = -40 + 0.25x
2. h Θ(x) = 200 + 0.1x
3. h Θ(x) = -150 + 0.4x

X =

### Inverse

1 = “identity”

Examples:

3 (3-1) = 1 3-1 = 1/3

If A is an m x m matrix (square matrix), and if it has an inverse,

AA-1 = A-1 A = 1

= = = I2x2

### Transpose

A = AT =

Let A be an m x n matrix, and let B = AT.

Then B is an n x m matrix, and Bij = Aij : B12 = A21

### Parameter Learning

#### Gradient Descent

Need to fill out

#### Gradient Descent Intuition

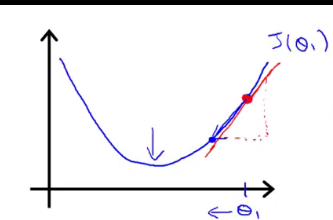
Repeat until convergence:{

*θj*:=*θj*−*α*1*m*∑*i*=1*m*(*hθ*(*x*(*i*))−*y*(*i*))⋅*x*(*i*)*j*for j := 0...n

}

*α = alpha or learning rate*

∑*i*=1*m*(*hθ*(*x*(*i*))−*y*(*i*))⋅ = derivative

Θ1 = Θ1 – *α (d/d*Θ1) J (Θ1)

=Θ1 \* *α (positive number)*

# Week 2 – Multivariate Linear Regression

## Multiple Features

|  |  |
| --- | --- |
| Size (feet2) | Price ($1000) |
| **X** | **y** |
| 2104 | 460 |
| 1416 | 232 |
| 1534 | 315 |
| 852 | 178 |

H Θ (x) = Θ0 + Θ1x

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Size (feet2) | Number of Bedrooms | Number of Floors | Are of home (years) | Price($1000) |
| **X1** | **X2** | **X3** | **X4** | **y** |
| 2104 | 5 | 1 | 45 | 460 |
| 1416 | 3 | 2 | 40 | 232 |
| 1534 | 3 | 2 | 30 | 315 |
| 852 | 2 | 1 | 36 | 178 |
| …. | … | ….. | …. | … |

n = 4 m = 47 (47 rows) x(2) = x3(2) = 2

### Notation:

n = number of features

x (i) = input (features) of i*th* training example.

x*j*(*i*) = value of feature j in the i*th* training example

### Hypothesis

Previously: hΘ(x) = Θ0 + Θ1x

New: hΘ(x) = Θ0 + Θ1x1 + Θ2x2 +Θ2x2 +Θ3x3 +Θ4x4

E.g. hΘ(x) = 80 +0.1x1 + 0.01x2 + 3x3 – 2x4

Base = 80, price goes up for bedrooms, price goes up for floor, bed rooms, price down for age

For convenience of notation, define x0 = 1

hΘ(x) = Θ0Xo + Θ1x1 + Θ2x2 +Θ2x2 +Θ3x3 +Θnxn

= ΘTx

## Gradient Descent for Multiple Variables

repeat until convergence:{

*θ*0:=*θ*0−*α*1*m*∑*i*=1*m*(*hθ*(*x*(*i*))−*y*(*i*))⋅*x*(*i*)0

*θ*1:=*θ*1−*α*1*m*∑*i*=1*m*(*hθ*(*x*(*i*))−*y*(*i*))⋅*x*(*i*)1

*θ*2:=*θ*2−*α*1*m*∑*i*=1*m*(*hθ*(*x*(*i*))−*y*(*i*))⋅*x*(*i*)2

……

}

In other words

repeat until convergence:{

*θj*:=*θj*−*α*1*m*∑*i*=1*m*(*hθ*(*x*(*i*))−*y*(*i*))⋅*x*(*i*)*j*for j := 0...n

}

## Gradient Descent in Practice 1 – Feature Scaling

We can speed up gradient descent by having each of our input values in roughly the same range. This is because Θ will descend quickly on small ranges and slowly on larger rangers, and so we will oscillate inefficiently down to the optimum when the variables are very uneven.

The way to prevent this is to modify the ranges of our input variables so that they are all roughly the same, ideally:

-1 <= x(i) <= 1 or -0.5 <= x(i) <= 0.5

Two techniques to help with this are **feature scaling** and **mean normalization**. Feature scaling involves dividing the input values by the range (i.e. the maximum value minus the minimum value) of the input variable, resulting in a new range of just 1. Mean Normalization involves subtracting the average value of an input variable from the values for that input variable resulting in a new average value for the input variable to just zero. To implement both of these techniques, adjust your input values as shown in this formula:

Xi := | Xi :=

Where is the average of all the values for feature (i) and si is the range of values (max – mine), or si is the standard deviation.

Note that dividing by the range, or dividing by standard deviation, give different results. The quizzes in this course use range – the programming exercises use standard deviation.

For example, if xi represents housing prices with a range of 100 to 2000 and a mean value of 1000,

Then xi: =

## Gradient Descent in Practice II – Learning Rate

* If Θ is too small; slow convergence
* If Θ is too large, J(Θ) may not decrease on every iteration; may not converge.

To choose Θ try

…., 0.0001, , 0.01, , 0.1, , 1,…..

## Features and Polynomial Regression

### Housing Prices Prediction

hΘ(x) = Θ0+ Θ1 \* frontage + Θ2 \* depth

x1 x2

Area:

X = frontage \* depth

hΘ(x) = Θ0+ Θ1x

## Computing Parameters Analytically

### Normal Equation

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Size (feet2) | Number of Bedrooms | Number of Floors | Are of home (years) | Price($1000) |
| **X**0 | **X1** | **X2** | **X3** | **X4** | **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 = y =

Θ = (XTX)-1XTy = minJ(Θ)

(XTX)-1 is inverse of matrix XTX

Set A = XTX = (XTX)-1 = A-1

Octave: pinv(X’ \*X)X’ \*y

X’ = XT

Please note there is no need to do feature scaling if the following is true.

0 <= X1 <= 1 | 0 <= X2 <= 1000 | 0 <= X3 <= 10-5

### Gradient DescentVrs Normal Equation

*m* training examples, *n* features

|  |  |
| --- | --- |
| Gradient Descent | Normal Equation |
| * Need to choose Θ * Needs many iterations * Works well even when n is large | * No need to choose Θ * Don’t need to iterate * Need to compute (XTX)-1 * Slow if n is very large   + N >= 10,000 |

## Normal Equation Noninvertibility

### What if XTX is non-invertible?

* Redundant features (linearly dependent)
  + X1 = size in feet2
  + X2 = size in m2

X1 = (3.28)2X2

* Too many features (e.g. m <= n)
  + Delete some features, or use regularization

# Week 3

## Classification and Representation

Classification examples

Email: Spam / No Spam?

Online Transactions: Fraudulent (Yes/No)?

Tumor: Malignant / Benign?

y ε {0,1} 0: “Negative Class” (e.g. Benign tumor)

1: “Positive Class” (e.g. malignant tumor)