Machine Learning

Linear Regression

Mustafa Jarrar

Birzeit University

Watch this lecture and download the slides

Course Page: http://www.jarrar.info/courses/AI/

More Online Courses at: http://www.jarrar.info

Acknowledgement:
This lecture is based on (but not limited to) Andrew Ng's course about Machine Learning
https://www.youtube.com/channel/UCMoXOGX9mgrYNEwpcIQUcag

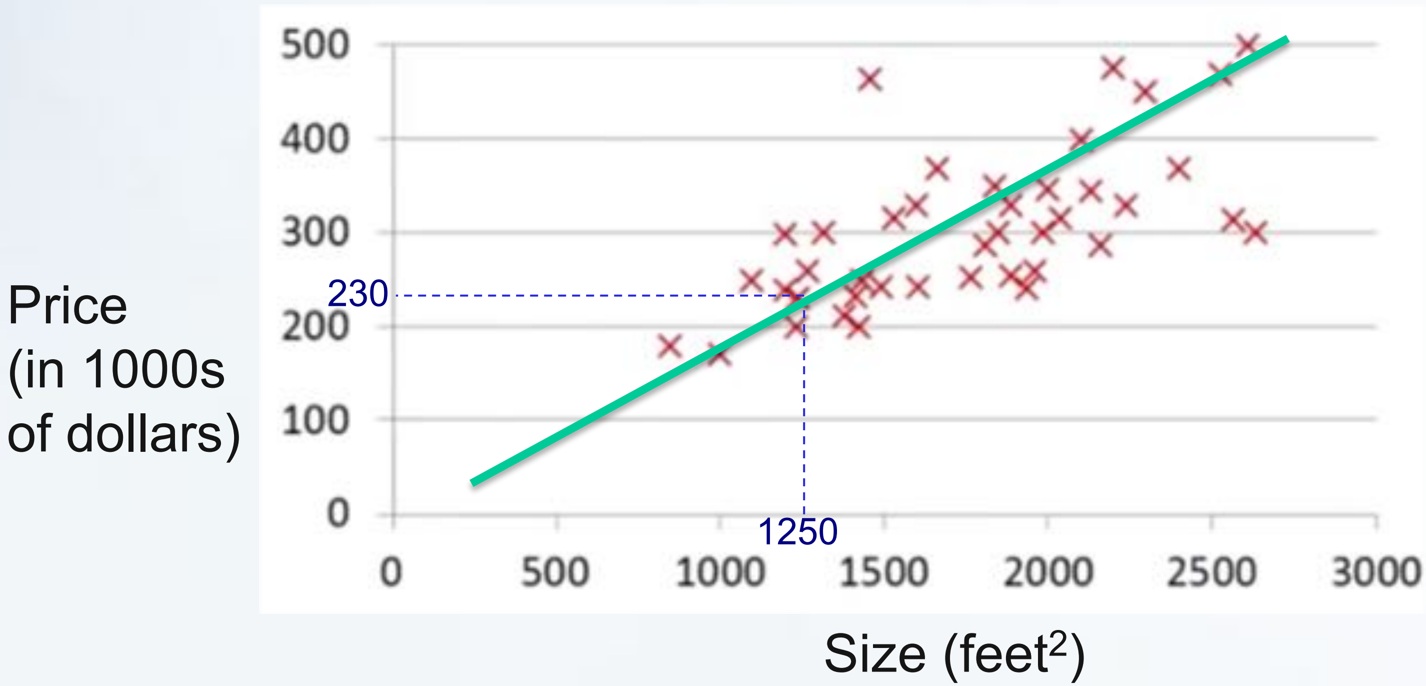
Machine Learning
Linear Regression

In this lecture:

- Part 1: **Motivation (Regression Problems)**
- Part 2: Linear Regression
- Part 3: The Cost Function
- Part 4: The Gradient Descent Algorithm
- Part 5: The Normal Equation
- Part 6: Linear Algebra overview
- Part 7: Using Octave
- Part 8: Using R

Motivation

Given the following **housing prices,**



How much a house of 1250ft^2 costs?

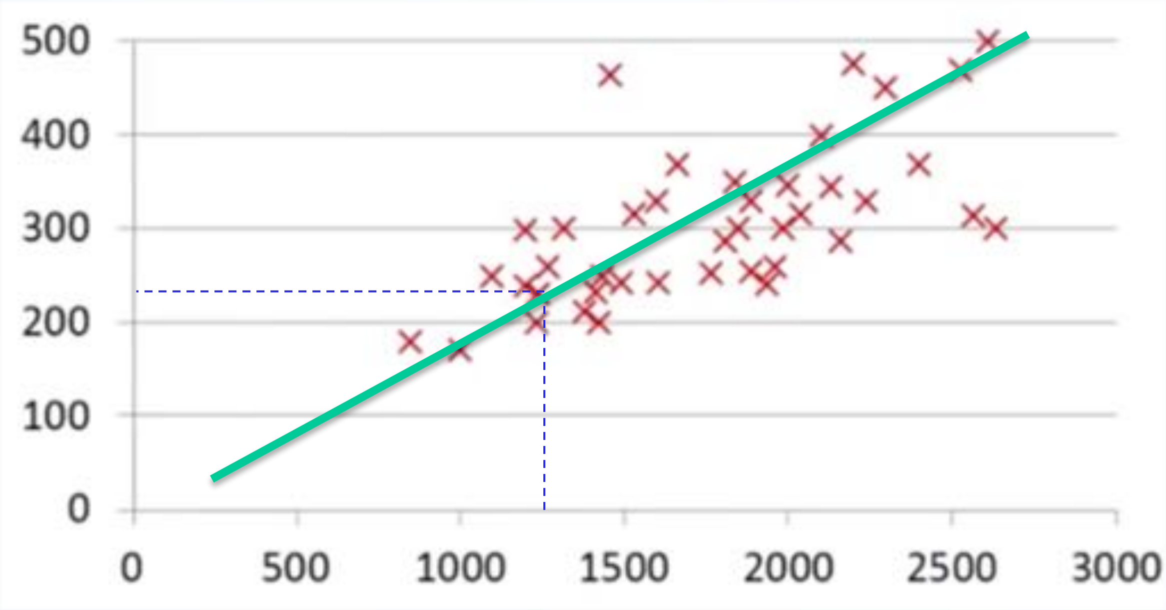
We may assume a linear curve,

So, conclude that a house with 1250ft^2 costs 230K$.

Motivation

Given the following **housing prices**

Price
(in 1000s
of dollars)



Size (feet^2)

Supervised Learning:

Given the right answers for each example in the data (training data)

Regression Problem:

Predict real-valued output

Remember that classification (not regression) refers to predicting discrete-valued output

Motivation

Given the following **training set** of housing prices:

Size in feet^2 (x)	Price ($) in 1000s (y)
2104	460
1416	232
1534	315
852	178
…	…

Our job is to learn from this data how to predict prices

Notation:

m = Number of training examples

x's = "input" variable/features

y's = output variable/target variable

(x, y) : a training example

(x^i, y^i) : the i^{th} training example

For example:

$x^1 = 2104$

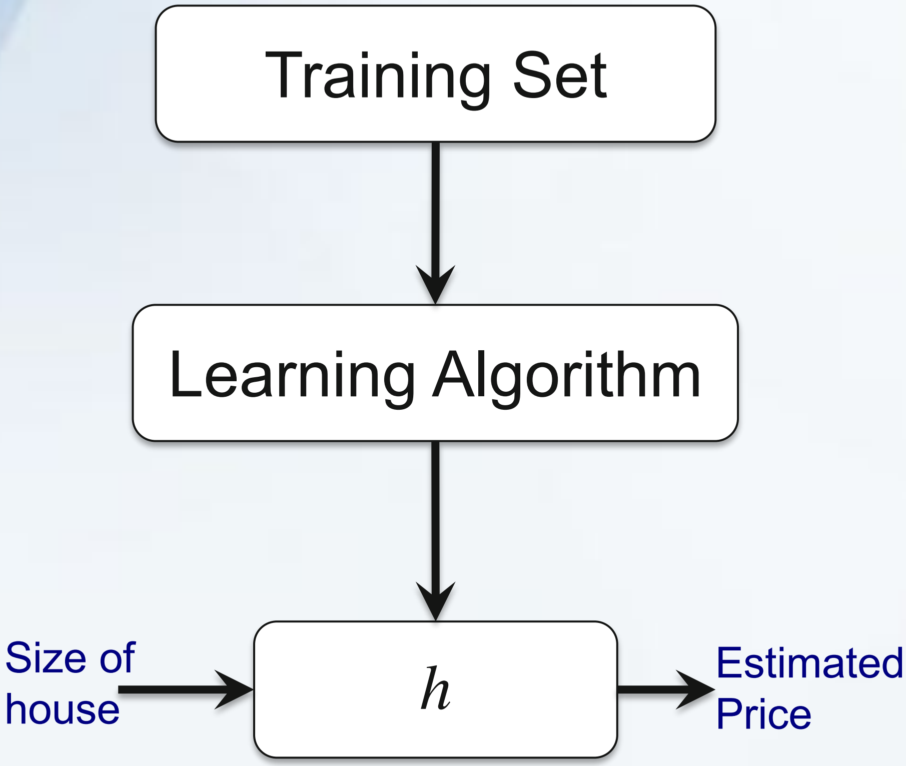
$y^1 = 460$

Machine Learning
Linear Regression

In this lecture:

- ❑ Part 1: Motivation (Regression Problems)
- ❑ Part 2: **Linear Regression**
- ❑ Part 3: The Cost Function
- ❑ Part 4: The Gradient Descent Algorithm
- ❑ Part 5: The Normal Equation
- ❑ Part 6: Linear Algebra overview
- ❑ Part 7: Using Octave
- ❑ Part 8: Using R

Linear Regression

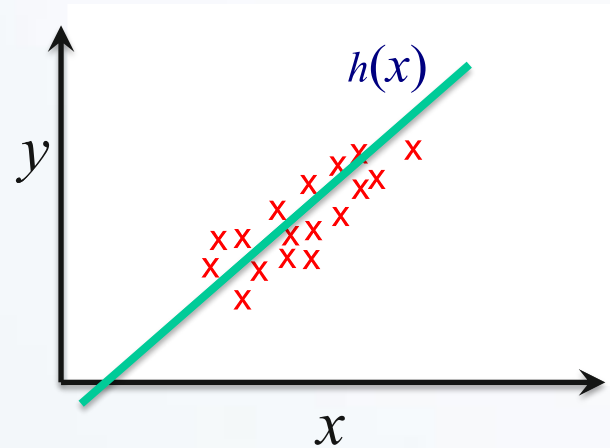


Hypothesis

h maps x's to y's

How to represent h?

$$h(x) = \theta_0 + \theta_1 x$$



- This is a linear function.
- Also called linear regression with one variable.

Linear Regression with Multiple Features

Linear regression with multiple features is also called *multiple linear regression*

Suppose we have the following features

x_1 Size ft^2	x_2 bedrooms	x_3 floors	x_4 Age	y Price
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
…	…	…	…	…

A hypothesis function $\boldsymbol{h(x)}$ might be:

$$h(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \ldots + \theta_n x_n$$

or $$h(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \ldots + \theta_n x_n \qquad (x_0 = 1)$$

e.g., $h(x) = 80 + 0.1 \cdot x_1 + 0.01 \cdot x_2 + 3 \cdot x_3 - 2 \cdot x_4$

Polynomial Regression

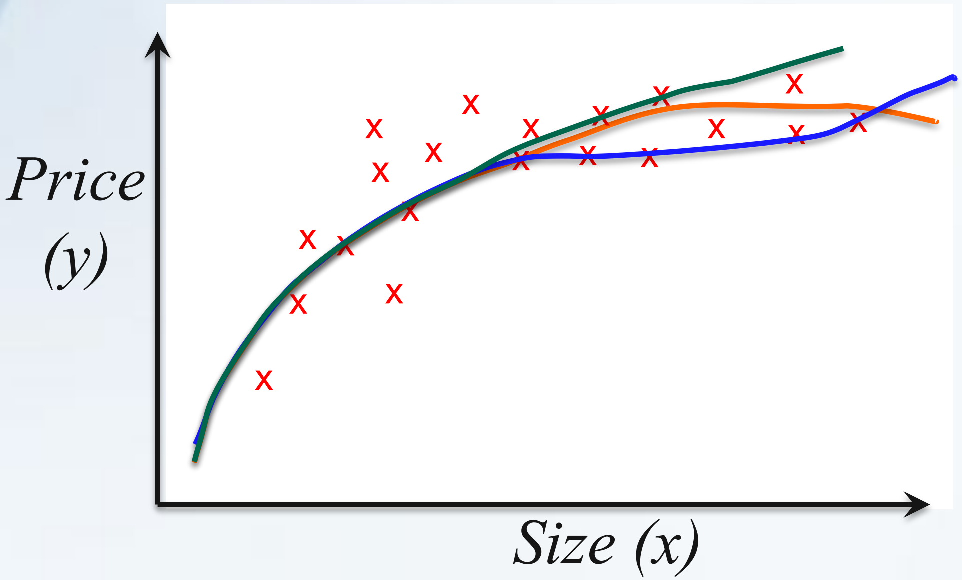
Suppose we have the following features

x_1 Size ft^2	x_2 bedrooms	x_3 floors	x_4 Age	y Price
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
…	…	…	…	…

Another hypothesis function $\boldsymbol{h(x)}$ might be (*polynomial*):

$$h(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2^{\,2} + \theta_2 x_3^{\,3} + \ldots + \theta_n x_n^{\,n} \qquad (x_0 = 1)$$

Polynomial Regression



$$h(x) = \theta_0 + \theta_1 x + \theta_2 x^2$$

$$h(x) = \theta_0 + \theta_1 x + \theta_2 \sqrt[2]{x}$$

$$h(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$$

- We may combine features (e.g., size = width * depth).
- We have the option of what features and what models (quadric, cubic,…) to use.
- Deciding which features and models that best fit our data and application, is beyond the scope of this course, but there are several algorithms for this.

Machine Learning
Linear Regression

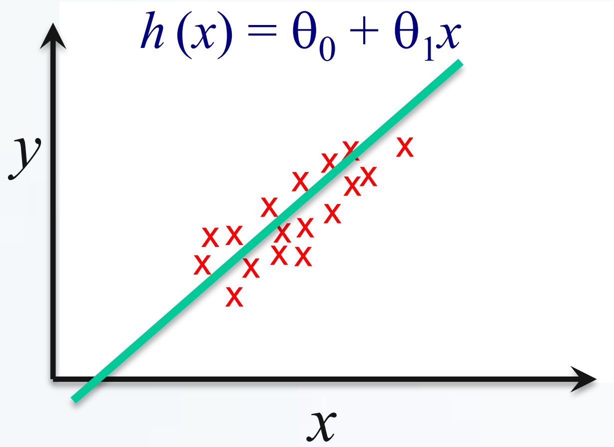
In this lecture:

- Part 1: Motivation (Regression Problems)
- Part 2: Linear Regression
- Part 3: **The Cost Function**
- Part 4: The Gradient Descent Algorithm
- Part 5: The Normal Equation
- Part 6: Linear Algebra overview
- Part 7: Using Octave
- Part 8: Using R

Understanding θs Values

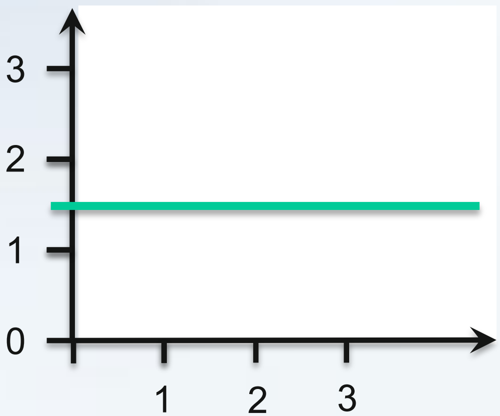
θ_i's: Parameters

How to choose θ_i's?



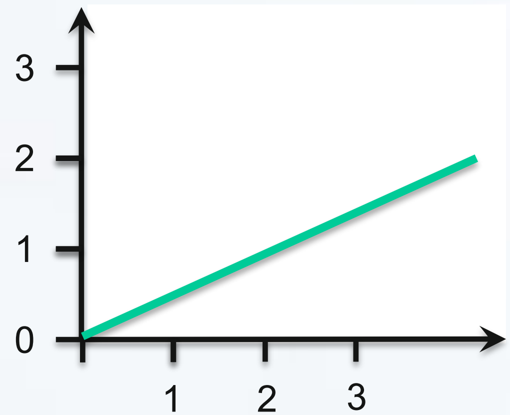
Understanding θs Values

$h(x) = 1.5 + 0 \cdot x$



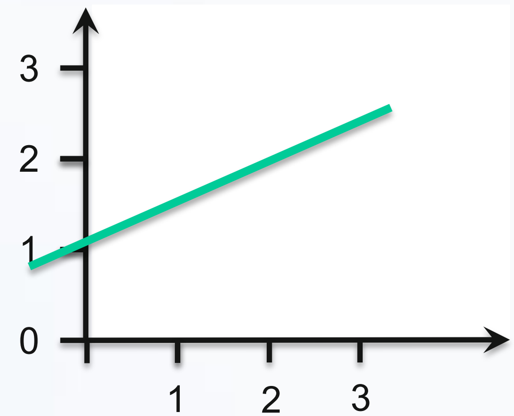
$\theta_0 = 1.5$
$\theta_1 = 0$

$h(x) = 0 + 0.5x$



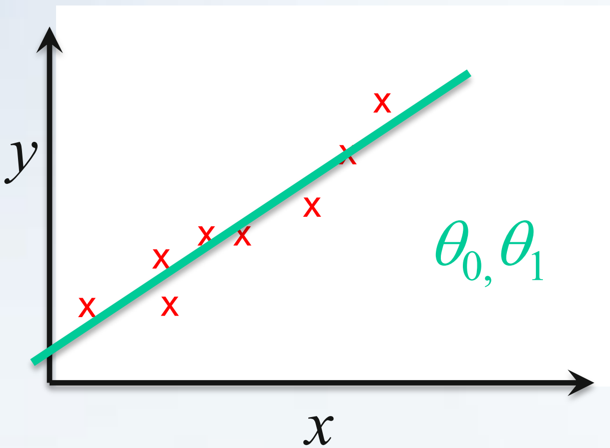
$\theta_0 = 0$
$\theta_1 = 0.5$

$h(x) = 1 + 0.5x$



$\theta_0 = 1$
$\theta_1 = 0.5$

The Cost Function



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

***y**: is the actual value.*
***h(x)**: is the estimated value.*

The **Cost Function** $\boldsymbol{J}$ aims to find θ_0, θ_1 that minimizes the error.

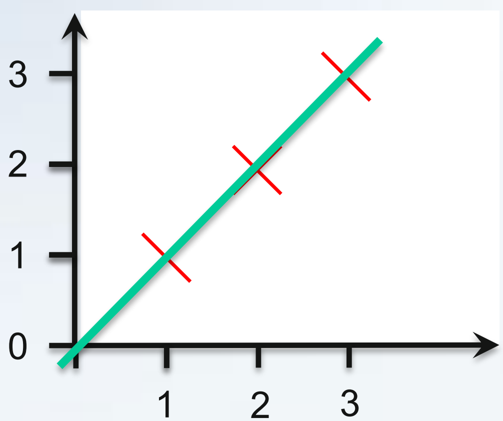
$$\boldsymbol{J}(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (\, h(x^i) - y^i\,)^2$$

This cost function is also called a **Squared Error function**

Undersetting the Cost Function

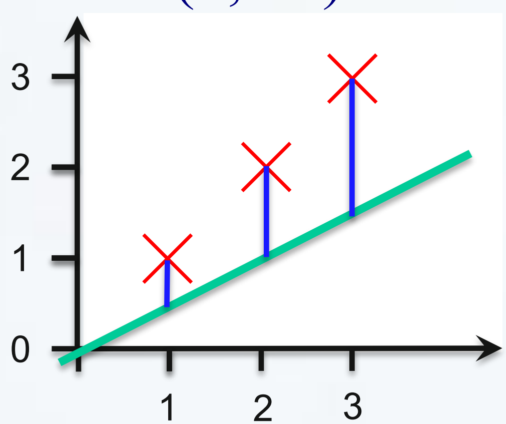
Lets try different values of θs and see how the cost function $J(\theta_{0,}\theta_1)$ behave!

$J(0,1)$



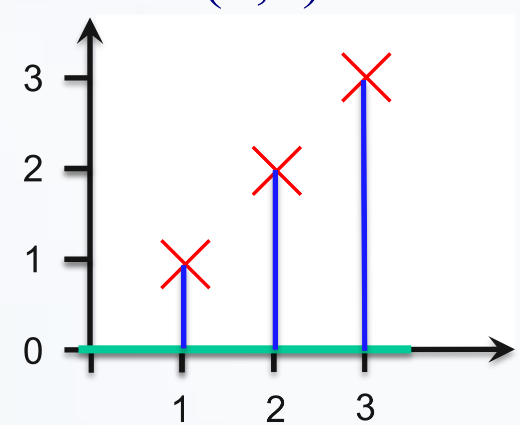
$J(1)$= 1/2·3 $[(0)^2+(0)^2(0)^2]$
$= 0$

$J(0,0.5)$



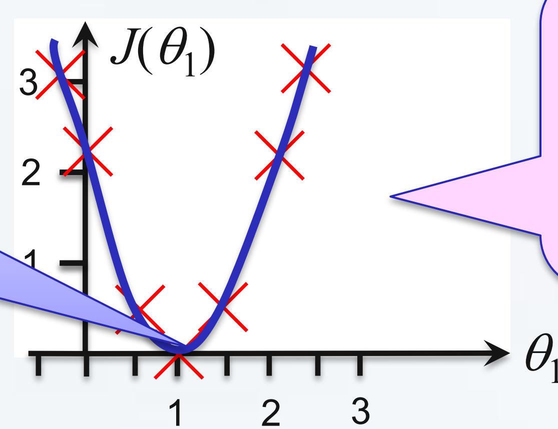
$J(0.5)$= 1/2·3 $[(0.5\text{-}1)^2+(1\text{-}2)^2(1.5\text{-}3)^2]$
$\simeq 0.68$

$J(0,0)$



$J(0)$= 1/2·3 $[(1)^2+(2)^2(3)^2]$
$\simeq 2.3$

Given our training data, this is the best value of θ_1.



But this figure plots only θ_1, the problem becomes complicated if we have also $(\theta_{0.}\ \theta_{1})$

Cost Function Intuition II

Remember that our goal is find the minimum values of θ_0 and θ_1

Hypothesis: $h(x) = \theta_0 + \theta_1$

Parameters: $\theta_0,\ \theta_1$

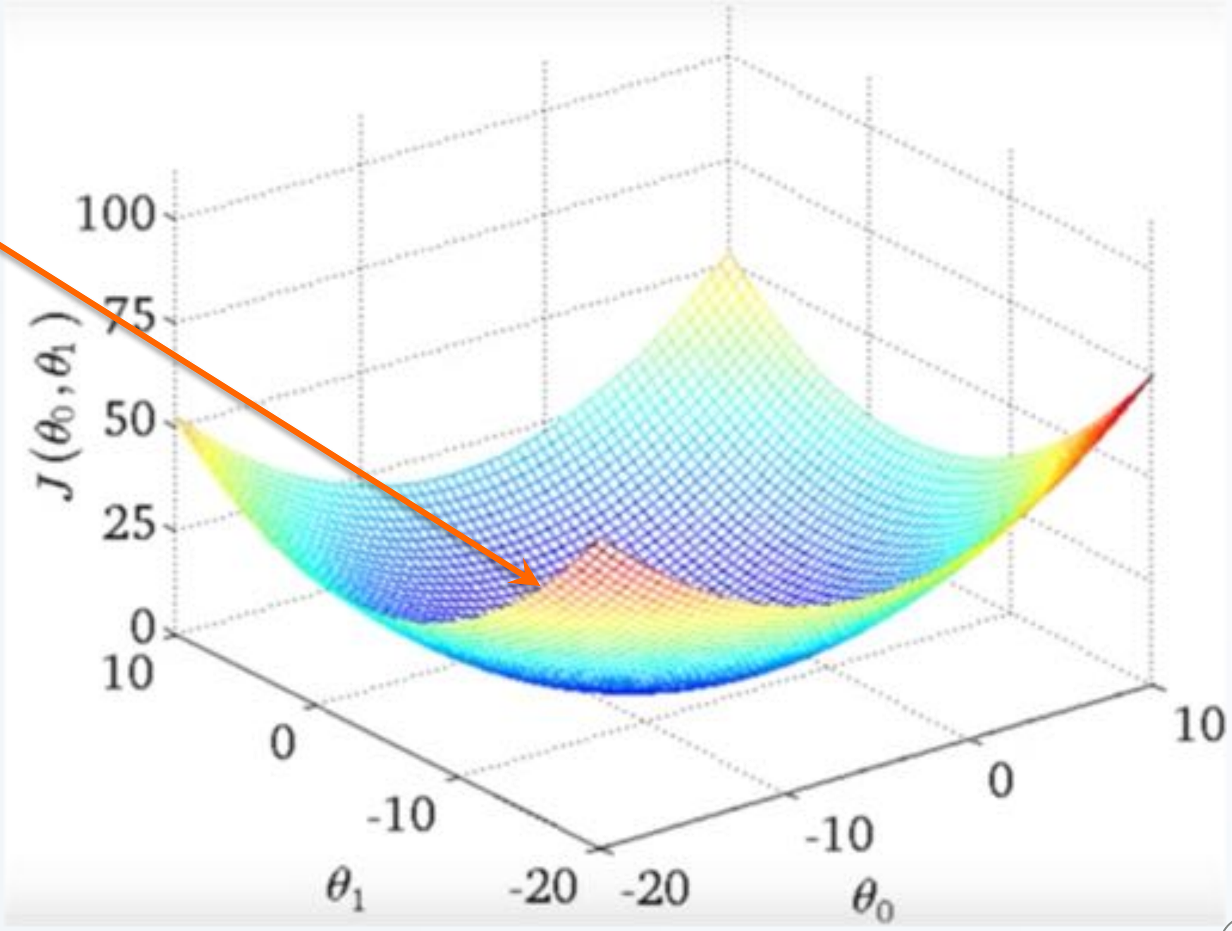
Cost Function: $J(\theta_0, \theta_1) = \dfrac{1}{2m} \sum_{i=1}^{m} (\ h(x^i) - y^i\ )^2$

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

Cost Function Intuition II

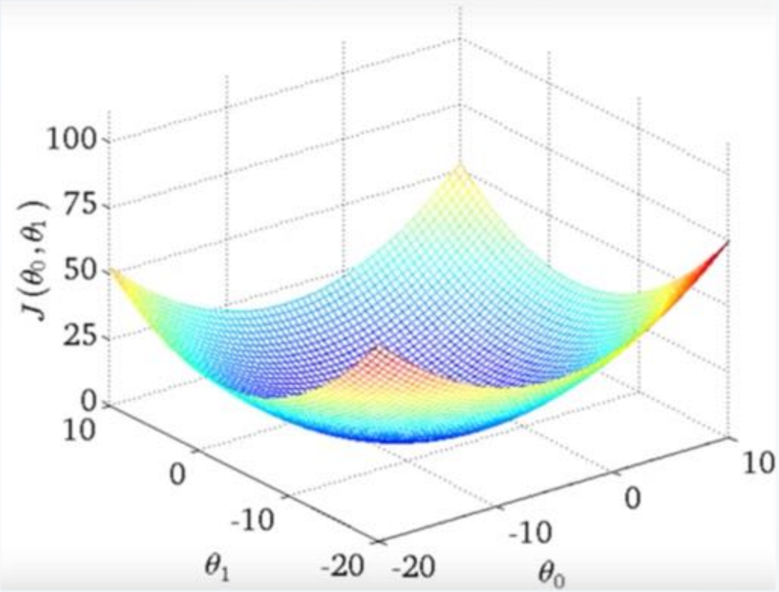
Given any dataset, when we try to draw the cost function $J(\theta_0,\theta_1)$, we may get this 3D shape:

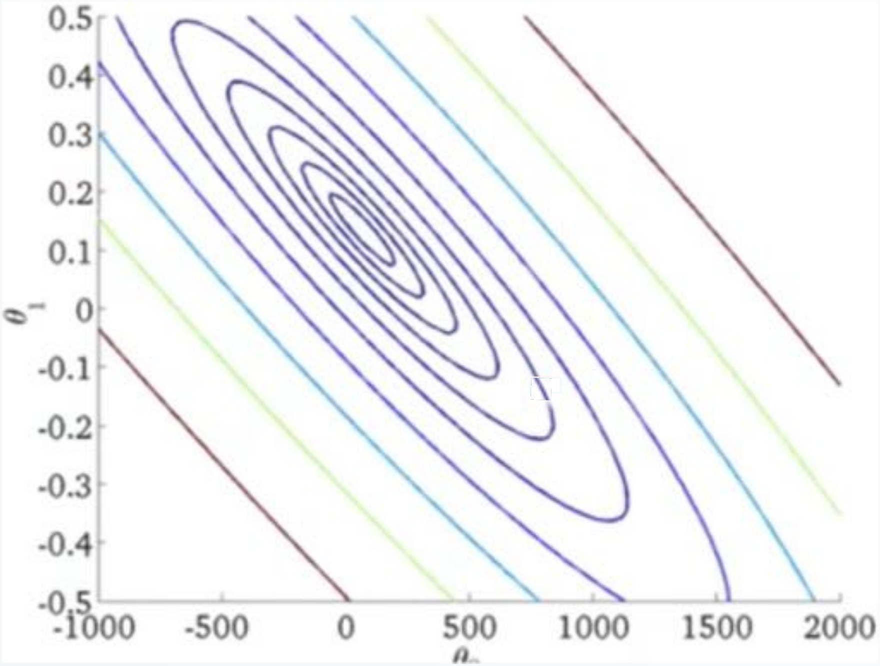
Given a dataset, our goal is find the minimum values of $J(\theta_0,\theta_1)$



Cost Function Intuition II

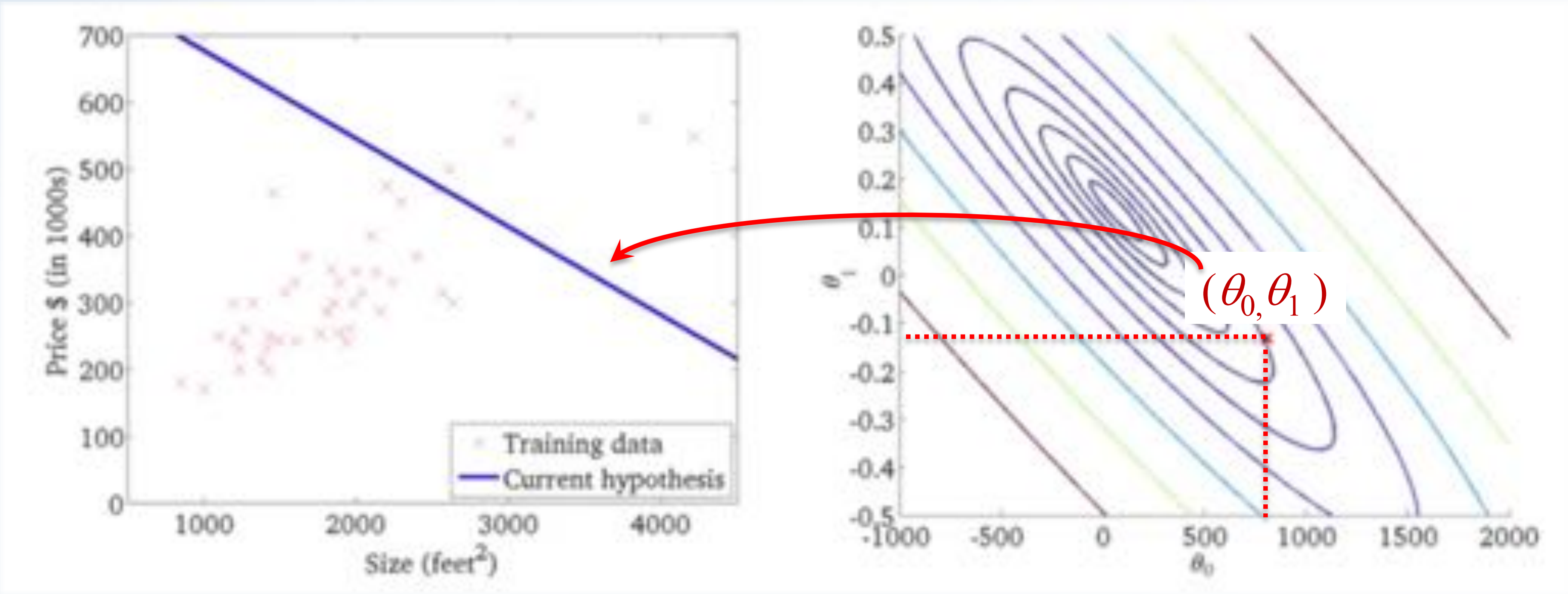
We may draw the cost function also using contour figures:





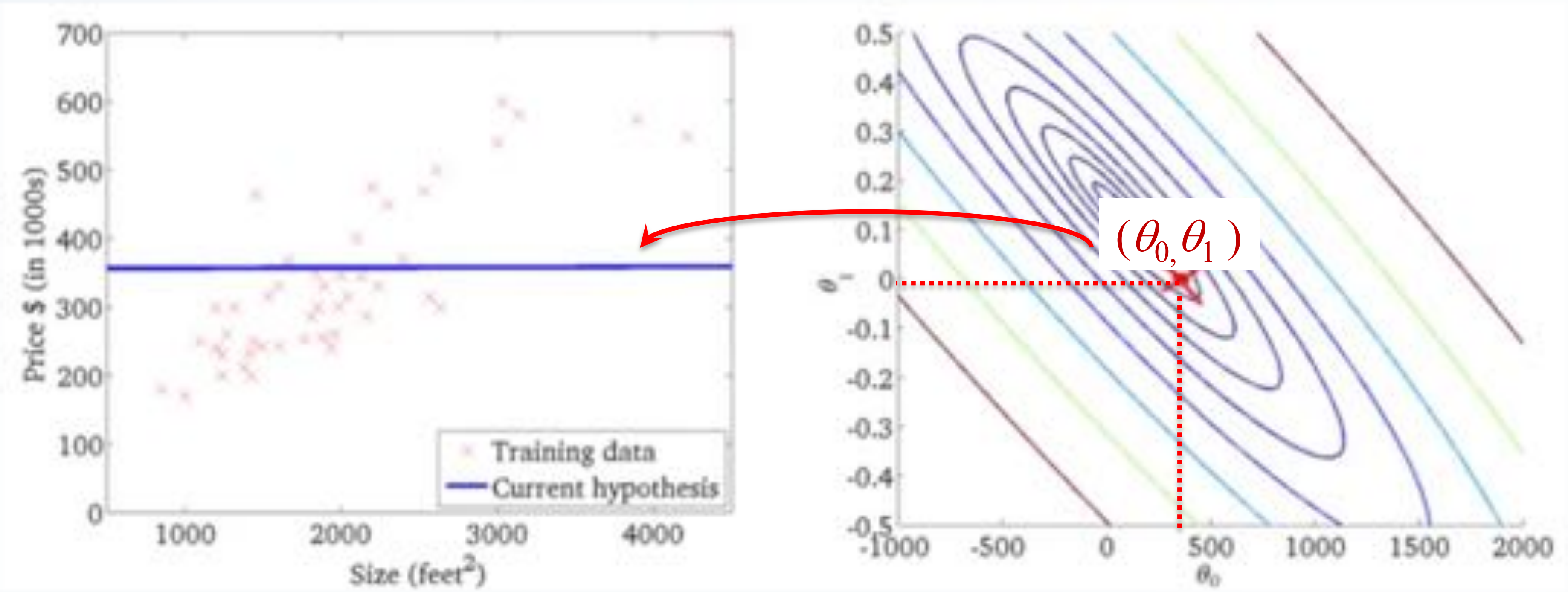
Cost Function Intuition II

$$h(x) = 800 + 0.15 \cdot x$$

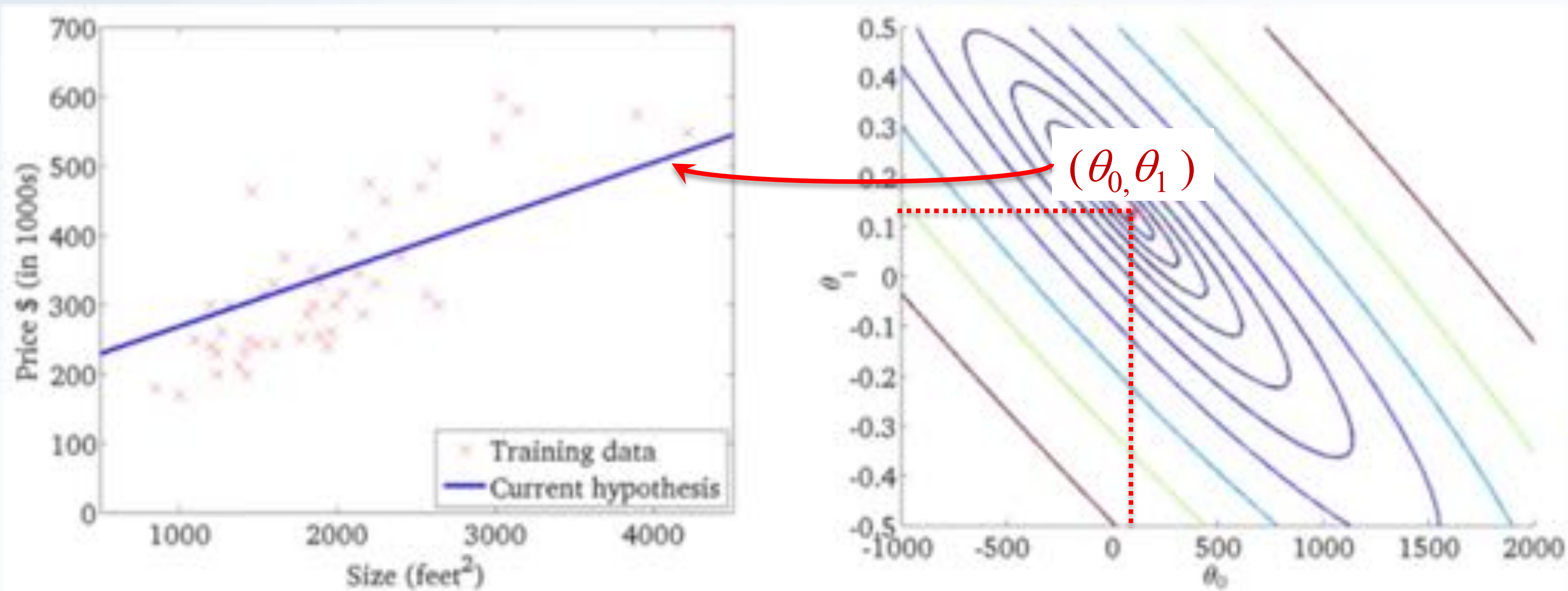


Cost Function Intuition II

$$h(x) = 360 + 0 \cdot x$$



Cost Function Intuition II



Is there any way/algorithm to find θs *automatically?*
Yes, e.g., the **Gradient Descent** Algorithm

Machine Learning
Linear Regression

In this lecture:

- Part 1: Motivation (Regression Problems)
- Part 2: Linear Regression
- Part 3: The Cost Function
- Part 4: **The Gradient Descent Algorithm**
- Part 5: The Normal Equation
- Part 6: Linear Algebra overview
- Part 7: Using Octave
- Part 8: Using R

The **Gradient Descent** Algorithm

Starts with some initial values of θ_0 and θ_1

Keep changing θ_0 and θ_1 to reduce $J(\theta_0, \theta_1)$

Until hopefully we end up at a minimum

J is 0 or 1

Repeat until convergence {

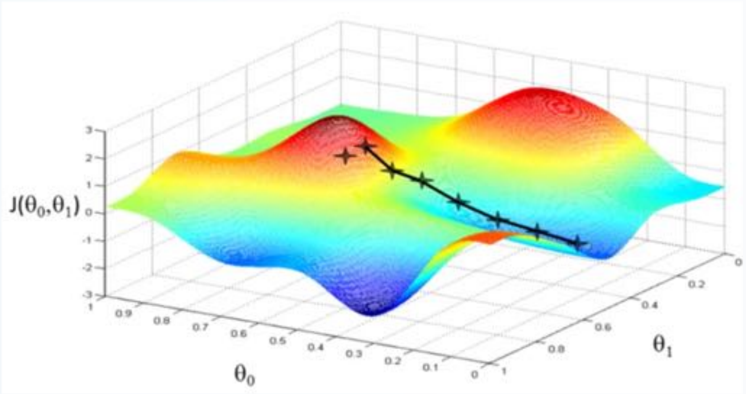
$$Temp0 := \theta_0 - \alpha \, . \frac{\partial}{\partial \theta_0} \, . \, J(\theta_0, \theta_1)$$

$$Temp1 := \theta 1 - \underbrace{\alpha}_{\text{Learning Rate}} . \underbrace{\frac{\partial}{\partial \theta_1} \, . \, J(\theta_0, \theta_1)}_{\text{Partial Derivation}}$$

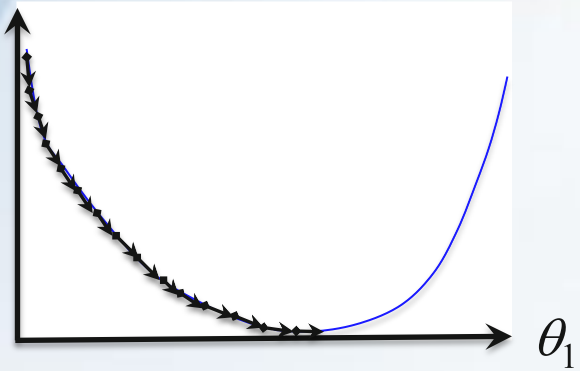
$$\theta_0 := Temp0$$

$$\theta_1 := Temp1$$

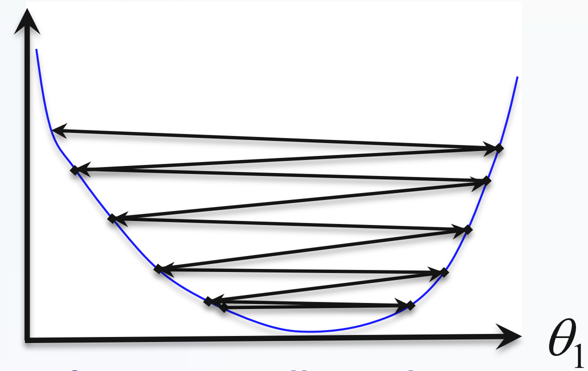
}



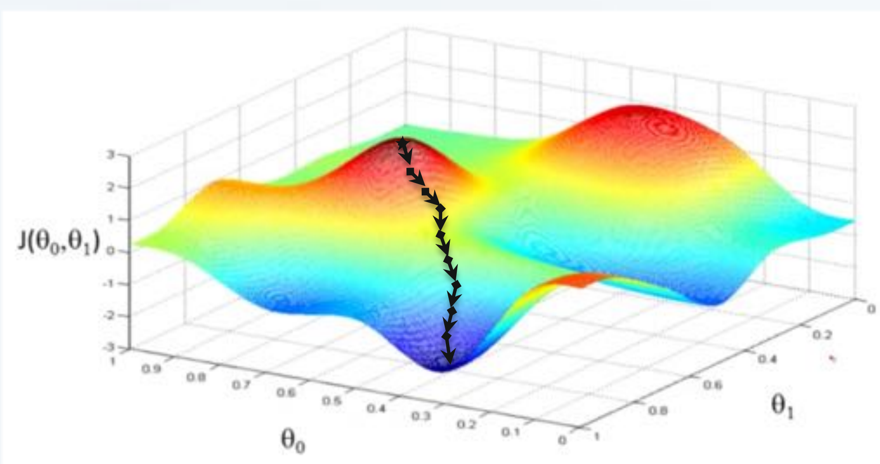
The Problem of the Gradient Descent algo.



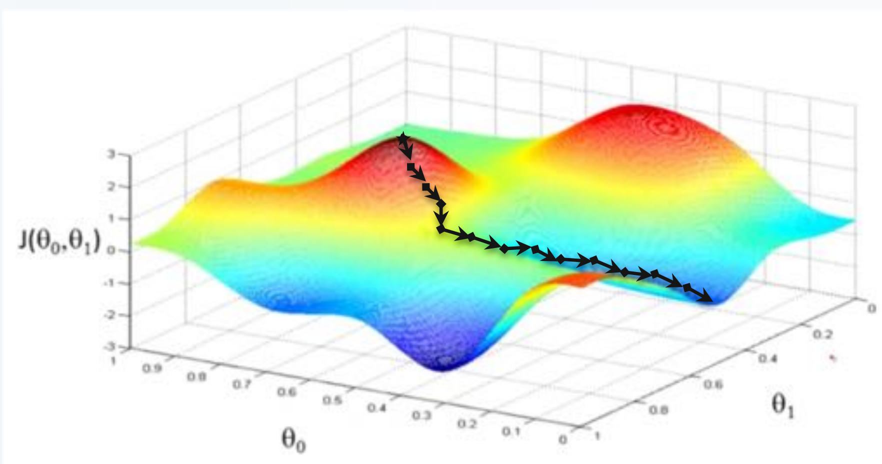
If α is too small, gradient descent can be slow



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



May converge to *global* minimum



May converge to a *local* minimum

The Gradient Descent for Linear Regression

Simplified version for linear regression

J is 0 and 1

Repeat until convergence {

$$\theta_0 := \theta_0 - \alpha \, . \frac{1}{m} . \sum_{i=1}^{m} (h(x^i) - y^i)$$

$$\theta_1 := \theta_1 - \alpha \, . \frac{1}{m} . \sum_{i=1}^{m} (h(x^i) - y^i) \, . \, x^i$$

}

Next, we will try to use Linear Algebra to *numerically* minimize θs (called **Normal Equation**) without needing to use iterating algorithms like Gradient Descent. However Gradient Descent scales better for bigger datasets.

Machine Learning
Linear Regression

In this lecture:

- Part 1: Motivation (Regression Problems)
- Part 2: Linear Regression
- Part 3: The Cost Function
- Part 4: The Gradient Descent Algorithm
- Part 5: **The Normal Equation**
- Part 6: Linear Algebra overview
- Part 7: Using Octave
- Part 8: Using R

Minimizing θs numerically

Another way to numerically (using linear algebra) estimate the optimal values of θs.

Given the following features:

x_1 Size ft^2	x_2 bedrooms	x_3 floors	x_4 Age	y Price
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
…	…	…	…	…

Convert the features and the target value into matrixes:

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

Minimizing θs numerically

x_1	x_2	x_3	x_4	y
Size ft^2	**bedrooms**	**floors**	**Age**	**Price**
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
...	...	...	...	...

add x_0, so to represent θ_0

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

The Normal Equation

To obtain the optimal values (minimized) of θs,
Use the following **Normal Equation**:

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

Where:

θ: the set of θs we want to minimize
X: the set of features in the training set
y: the output values we try to predict
X^T: the transpose of X
X^{-1}: the inverse of X

In Octave: `pinv(X'*X)*X' *y`

Gradient Descent Vs Normal Equation

***m* training examples, *n* features**

Gradient Descent

- Needs to choose the learning rate (α)
- Needs many iterations
- Works well even when n is large

Normal Equation

- No need to choose (α)
- Don't need to iterate
- Need to compute (X^TX), which takes about $O(n^3)$
- Slow if n is very large
- Some matrices (e.g., singular) are non-invertible

Recommendation: use the Normal Equation If the number of features in less than 1000, otherwise the Gradient Descent.

Machine Learning
Linear Regression

In this lecture:

Matrix Vector Multiplication

$$\begin{bmatrix} 1 & 3 \\ 4 & 0 \\ 2 & 1 \end{bmatrix} \times \begin{bmatrix} 1 \\ 5 \end{bmatrix} = \begin{bmatrix} 16 \\ 4 \\ 7 \end{bmatrix}$$

$$1\times1 + 3\times5 = 16$$

$$4\times1 + 0\times5 = 4$$

$$2\times1 + 1\times5 = 7$$

Matrix Matrix Multiplication

$$\begin{bmatrix} 1 & 3 & 2 \\ 4 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 3 \\ 0 & 1 \\ 5 & 2 \end{bmatrix} = \begin{bmatrix} 11 & 10 \\ 9 & 14 \end{bmatrix}$$

Multiple with the first column

$$\begin{bmatrix} 1 & 3 & 2 \\ 4 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 \\ 0 \\ 5 \end{bmatrix} = \begin{bmatrix} 11 \\ 9 \end{bmatrix}$$

Multiple with the second column

$$\begin{bmatrix} 1 & 3 & 2 \\ 4 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 3 \\ 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 10 \\ 14 \end{bmatrix}$$

Matrix Inverse

If A is an $m \times m$ matrix, and if it has an inverse, then

$$A \times A^{-1} = A^{-1} \times A = I$$

The multiplication of a matrix with its inverse produce an identity matrix:

$$\begin{bmatrix} 3 & 4 \\ 2 & 16 \end{bmatrix} \times \begin{bmatrix} 0.4 & -0.1 \\ -0.05 & 0.075 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Some matrices do not have inverses e.g., if all cells are zeros
For more, please review Linear Algebra

Matrix Transpose

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

Let A be an $m \times n$ matrix, and let $B = A^T$

Then B is an $n \times m$ matix, and $B_{ij} = A_{ji}$

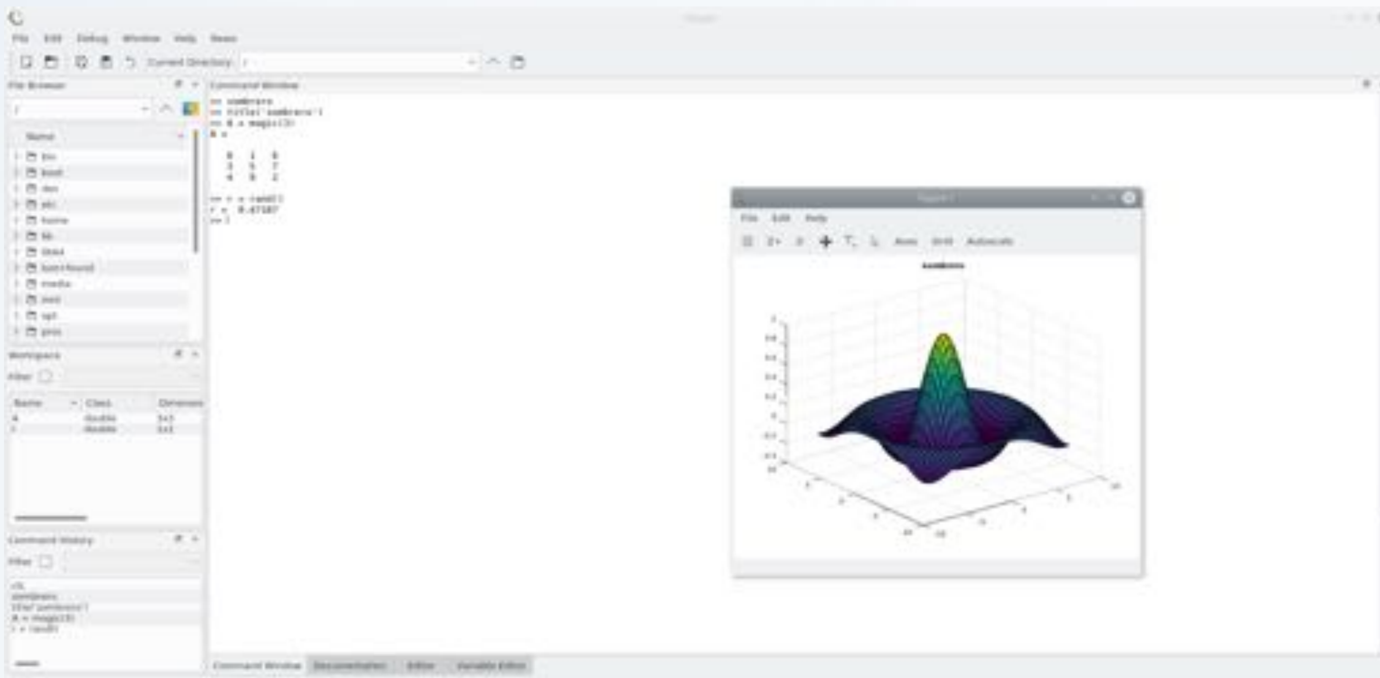
Machine Learning
Linear Regression

In this lecture:

- Part 1: Motivation (Regression Problems)
- Part 2: Linear Regression
- Part 3: The Cost Function
- Part 4: The Gradient Descent Algorithm
- Part 5: The Normal Equation
- Part 6: Linear Algebra overview
- Part 7: **Using Octave**
- Part 8: Using R

Octave

- Download from **https://www.gnu.org/software/octave/**
- High-level Scientific Programming Language
- Free alternatives to (and compatible with) Matlab
- helps in solving linear and nonlinear problems numerically



Online Tutorial https://www.youtube.com/playlist?list=PLnnr1O8OWc6aAjSc50lzzPVWgjzucZsSD

Compute the normal equation using Octave

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

In Octave:

```
load X.txt
load y.txt
C= pinv(X'*X)*X' *y
Save θ.txt
```

$\theta = \begin{bmatrix} 188.4 \\ 0.4 \\ -56 \\ -93 \\ -3.7 \end{bmatrix}$

These are the values of θs we need to use in our hypothesis function h(x)

$$h(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

$$h(x) = 188.4 + 0.4x - 56x^2 - 93x^3 - 3.7x^4$$

Machine Learning
Linear Regression

In this lecture:

- Part 1: Motivation (Regression Problems)
- Part 2: Linear Regression
- Part 3: The Cost Function
- Part 4: The Gradient Descent Algorithm
- Part 5: The Normal Equation
- Part 6: Linear Algebra overview
- Part 7: Using Octave
- Part 8: **Using R**

R (programming language)

Free software environment for statistical computing and graphics that is supported by the R Foundation for Statistical Computing.

R comes with many functions that can do sophisticated stuff, and the ability to install additional packages to do much more.

Download R: https://cran.r-project.org/

A very good IDE for R is the RStudio:
https://www.rstudio.com/products/rstudio/download/

R basics tutorial:
https://www.youtube.com/playlist?list=PLjgj6kdf_snYBkIsWQYcYtUZiDpam7ygg

Compute the normal equation using R

Given

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

In R:

```
X = as.matrix(read.table("~/x.txt", header=F, sep=","))
Y = as.matrix(read.table("~/y.txt", header=F, sep=","))

thetas = solve( t(X) %*% X ) %*% t(X) %*% Y

write.table(thetas, file="~/thetas.txt",
            row.names=F, col.names=F)
```

t(X): is the transpose of matrix X.
solve(X): is the inverse of matrix X.

References

[1] Andrew Ng's course about Machine Learning
https://www.youtube.com/channel/UCMoXOGX9mgrYNEwpcIQUcag

[2] Sami Ghawi, Mustafa Jarrar: Lecture Notes on Introduction to Machine Learning, Birzeit University, 2018

[3] Mustafa Jarrar: Lecture Notes on Decision Tree Machine Learning, Birzeit University, 2018

[4] Mustafa Jarrar: Lecture Notes on Linear Regression Machine Learning, Birzeit University, 2018