Regression

~Abhishek Kumar

Scope

- What is regression?
- Types of regression
- Math in the Machine
  - Cost functions
  - Gradient Descent
  - Normal Equations
- Assumptions
- Pros and Cons

Regression

- Supervised
- Predicting real valued output/ continuous variables
- In statistics, a measure of the numerical relation between the mean value of one variable (e.g. output) and corresponding values of other variables

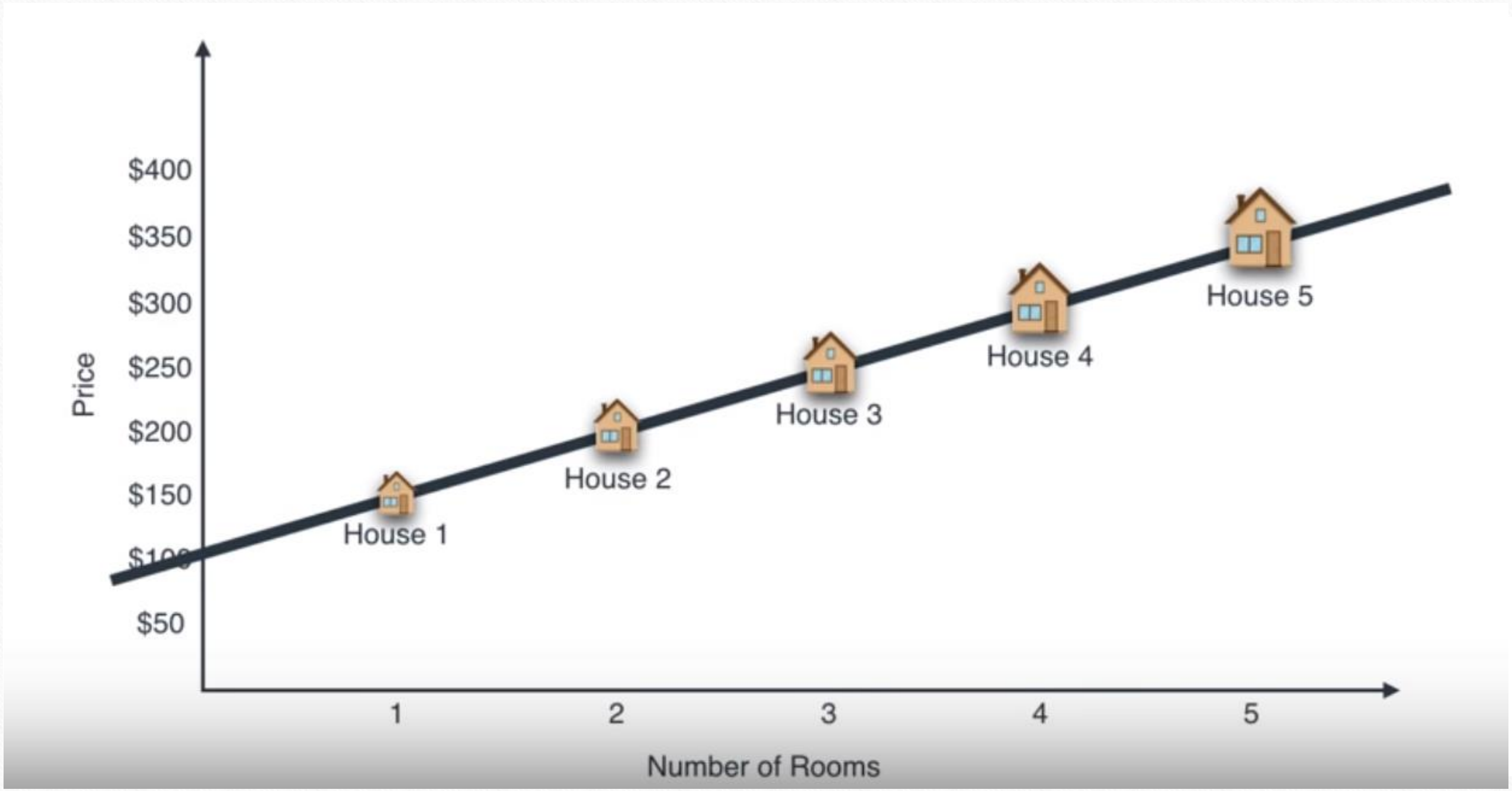
Housing Prices

House 1	House 2	House 3	House 4	House 5
1 room	2 rooms	3 rooms	4 rooms	5 rooms

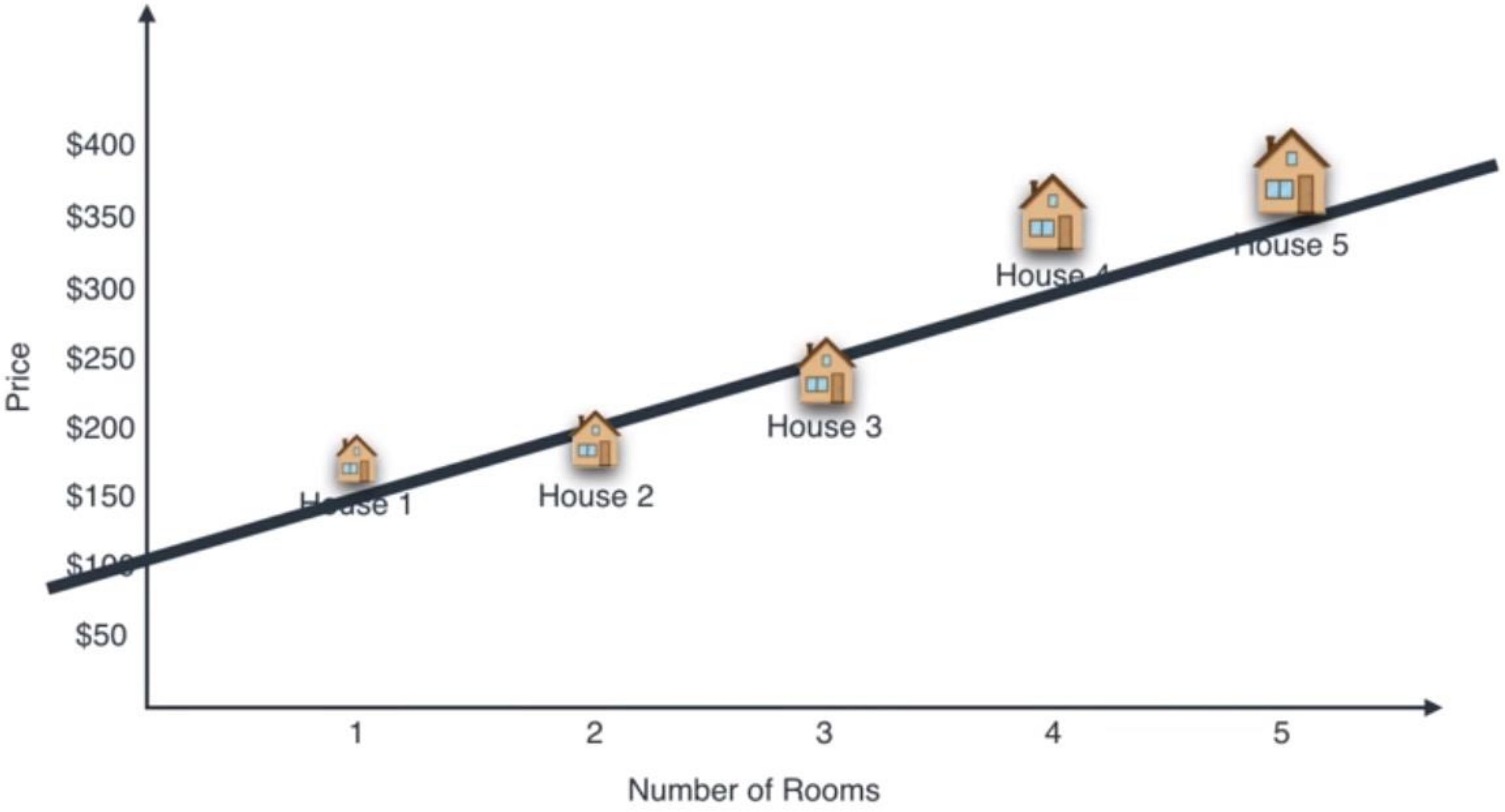
Housing Prices

House 1	House 2	House 3	House 4	House 5
1 room	2 rooms	3 rooms	4 rooms	5 rooms
$150K	$200K	???	$300K	$350K


Price
$400
$350
$300
$250
$200
$150
$100
$50
House 1
House 2
House 4
House 5
1
2
3
4
5

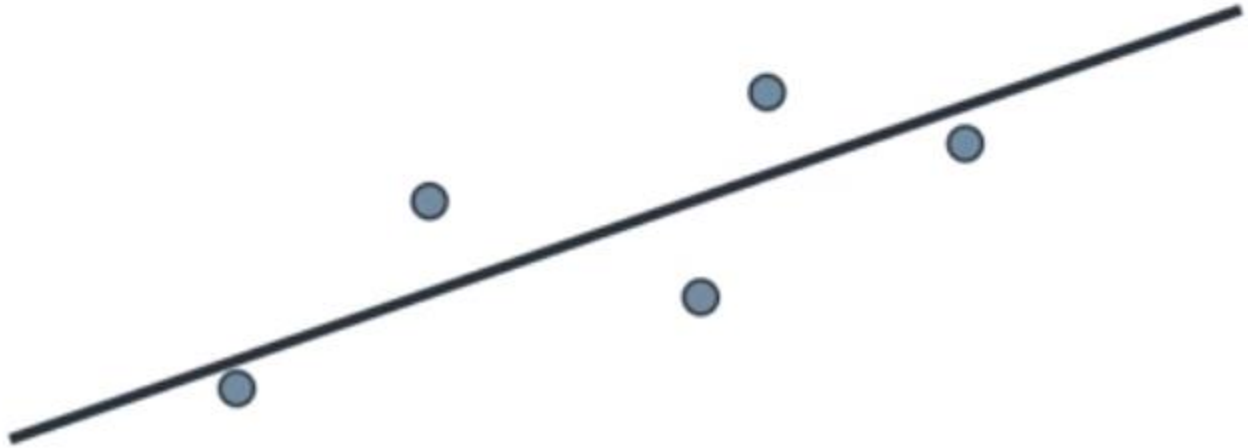


Price
$400
$350
$300
$250
$200
$150
$100
$50
House 1
House 2
House 3
House 4
House 5
1
2
3
4
5
Number of Rooms

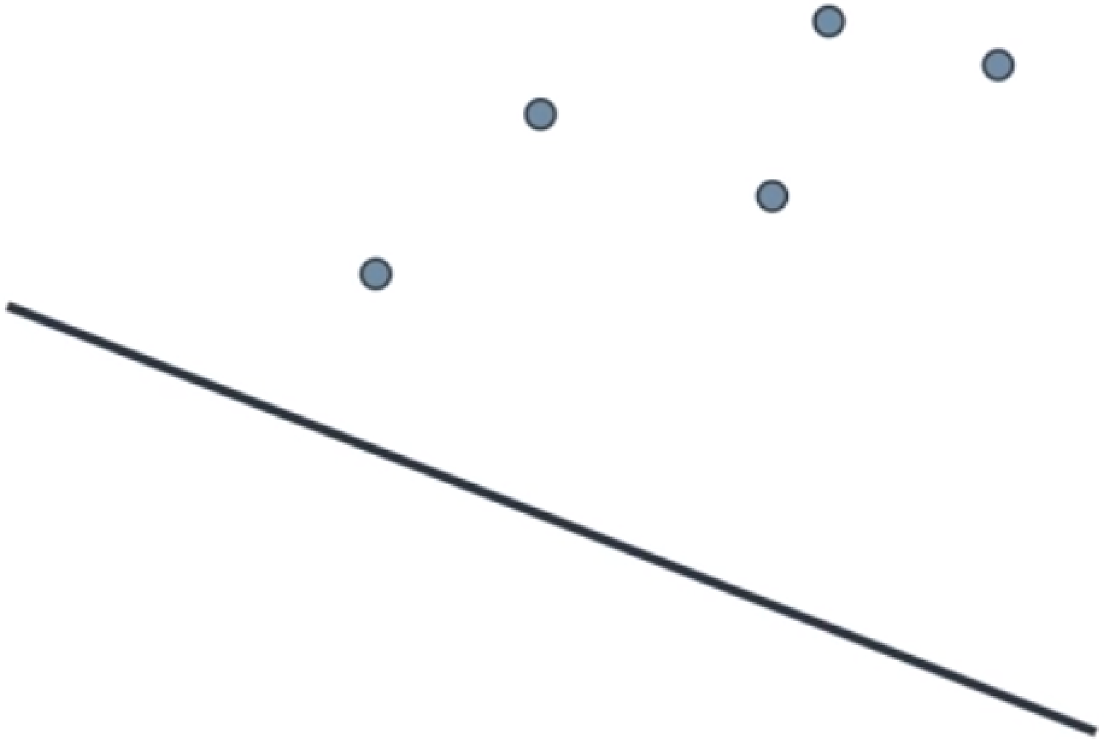


$400
$350
$300
$250
$200
$150
$50
Price
House 2
House 3
House 5
1
2
3
4
5
Number of Rooms

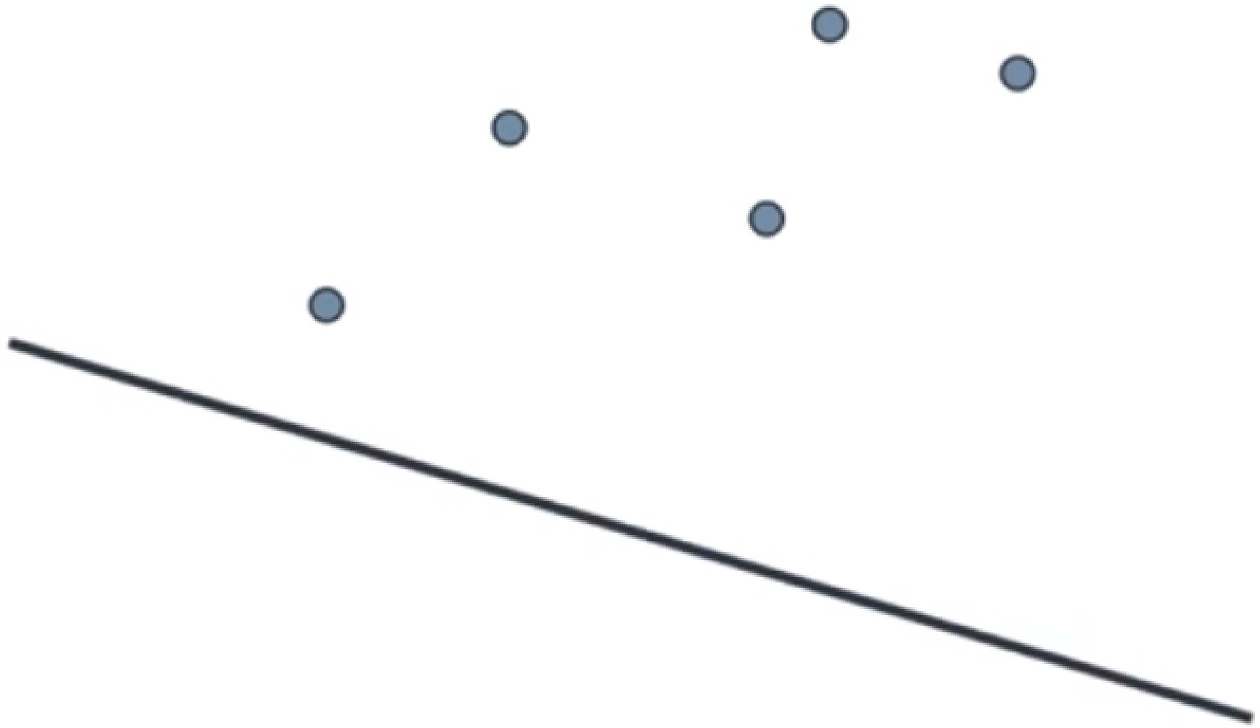
Linear Regression



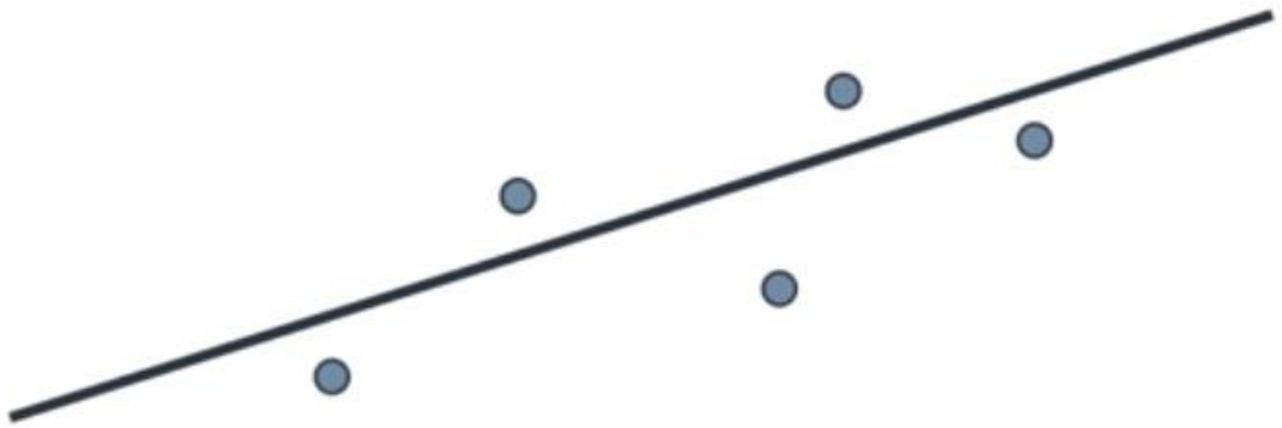
Linear Regression



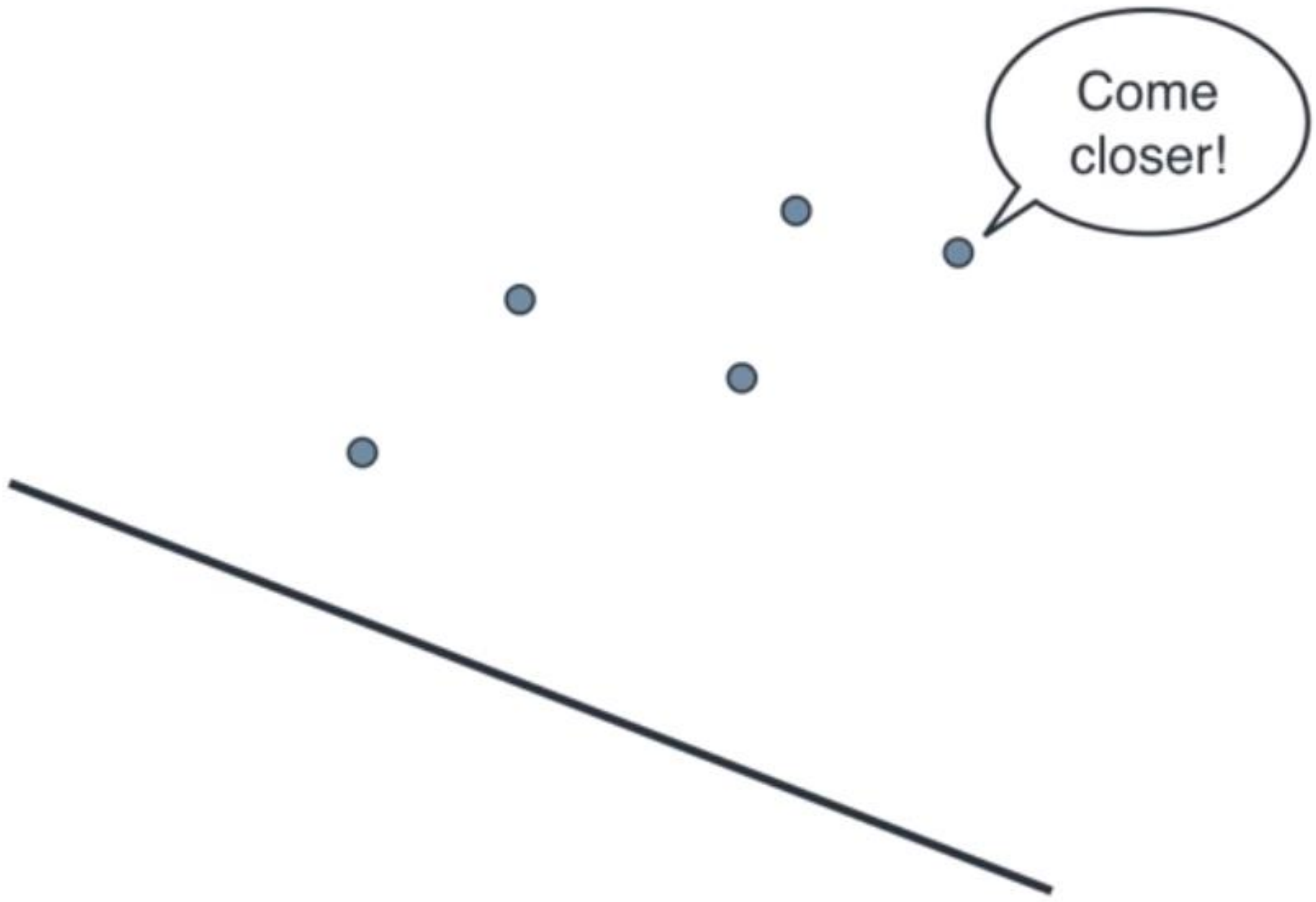
Linear Regression



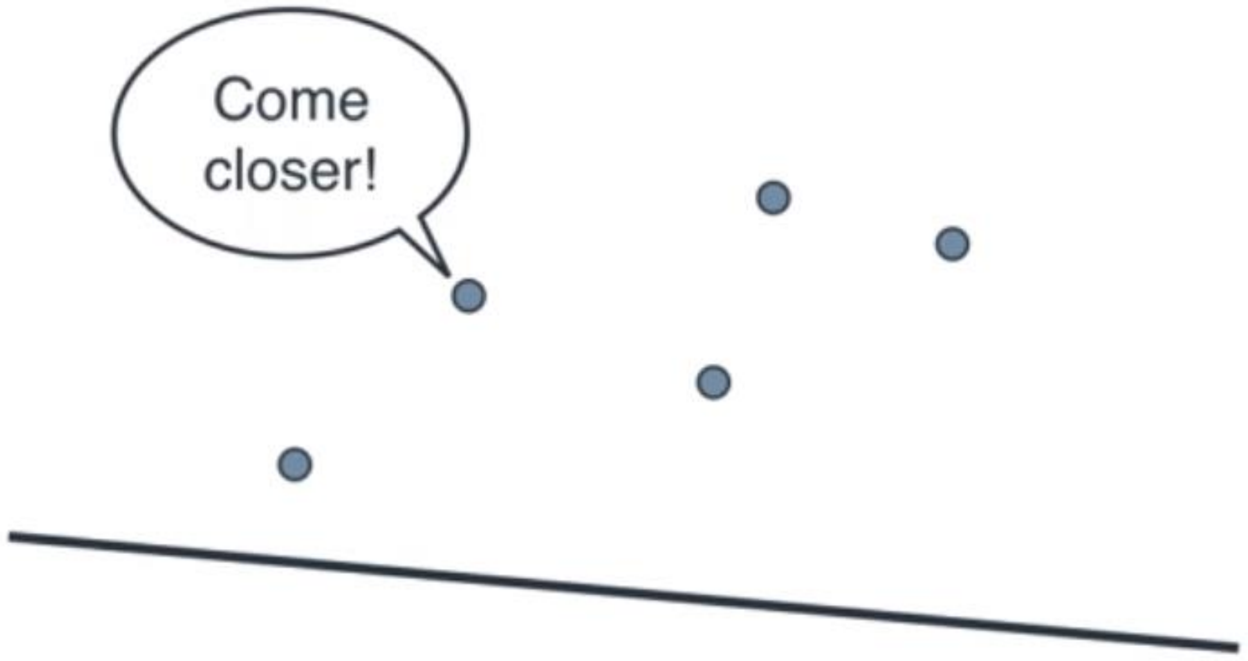
Linear Regression



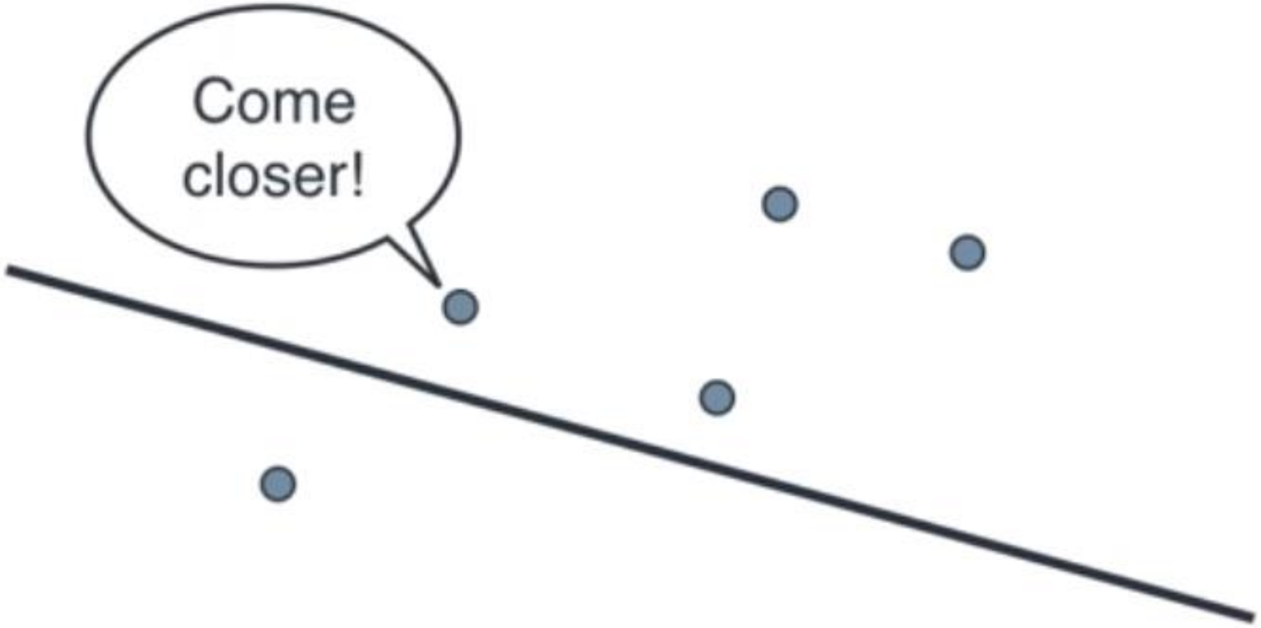
Linear Regression



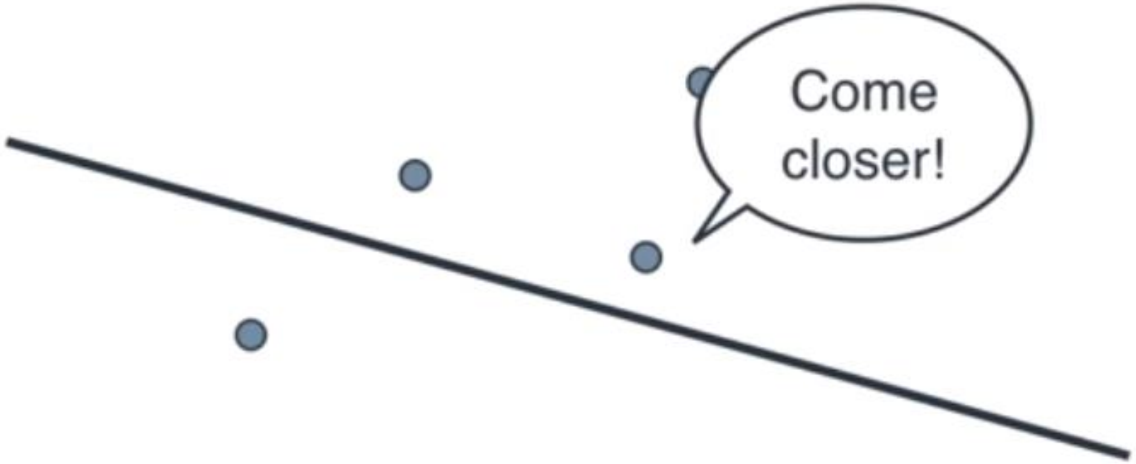
Linear Regression



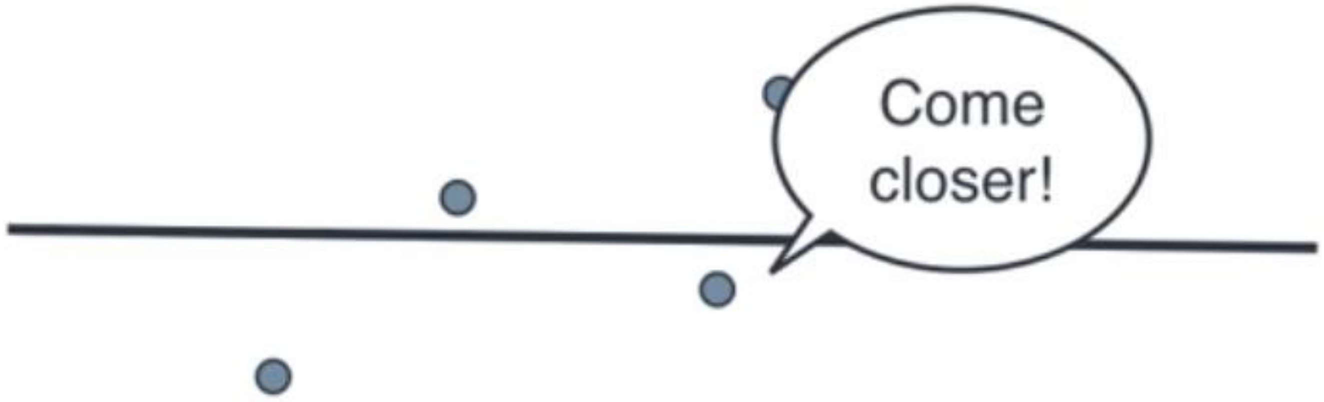
Linear Regression



Linear Regression

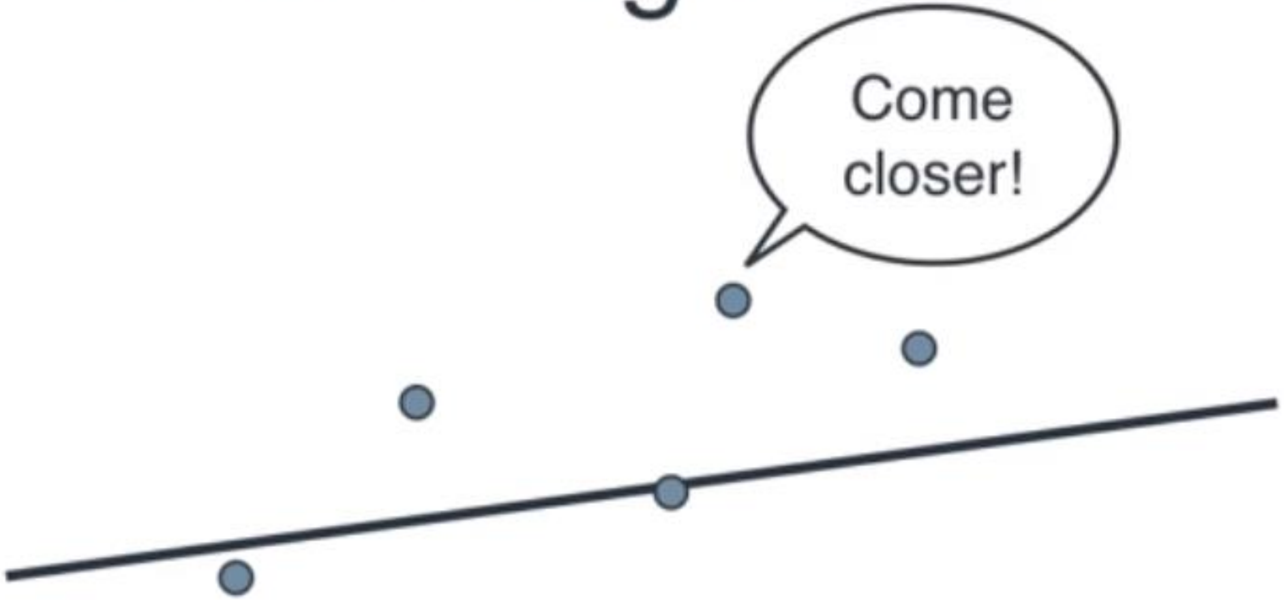


Linear Regression

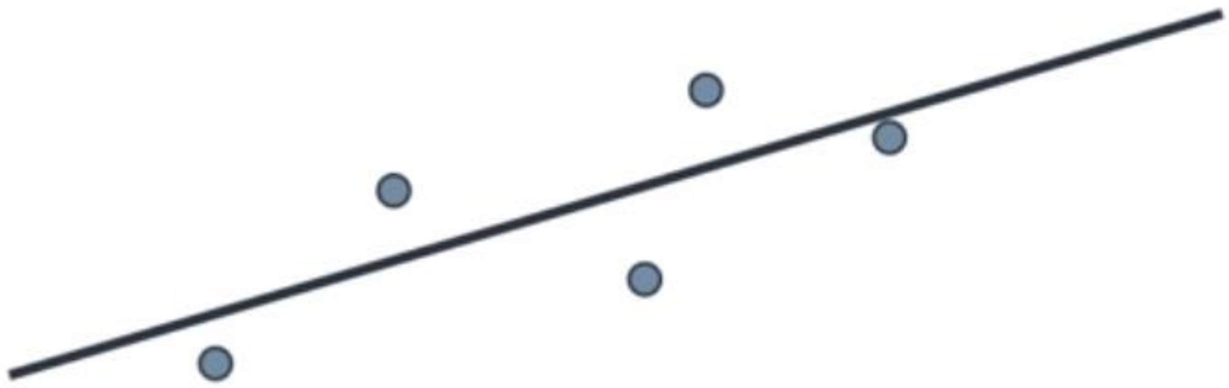


Linear Regression

Linear Regression

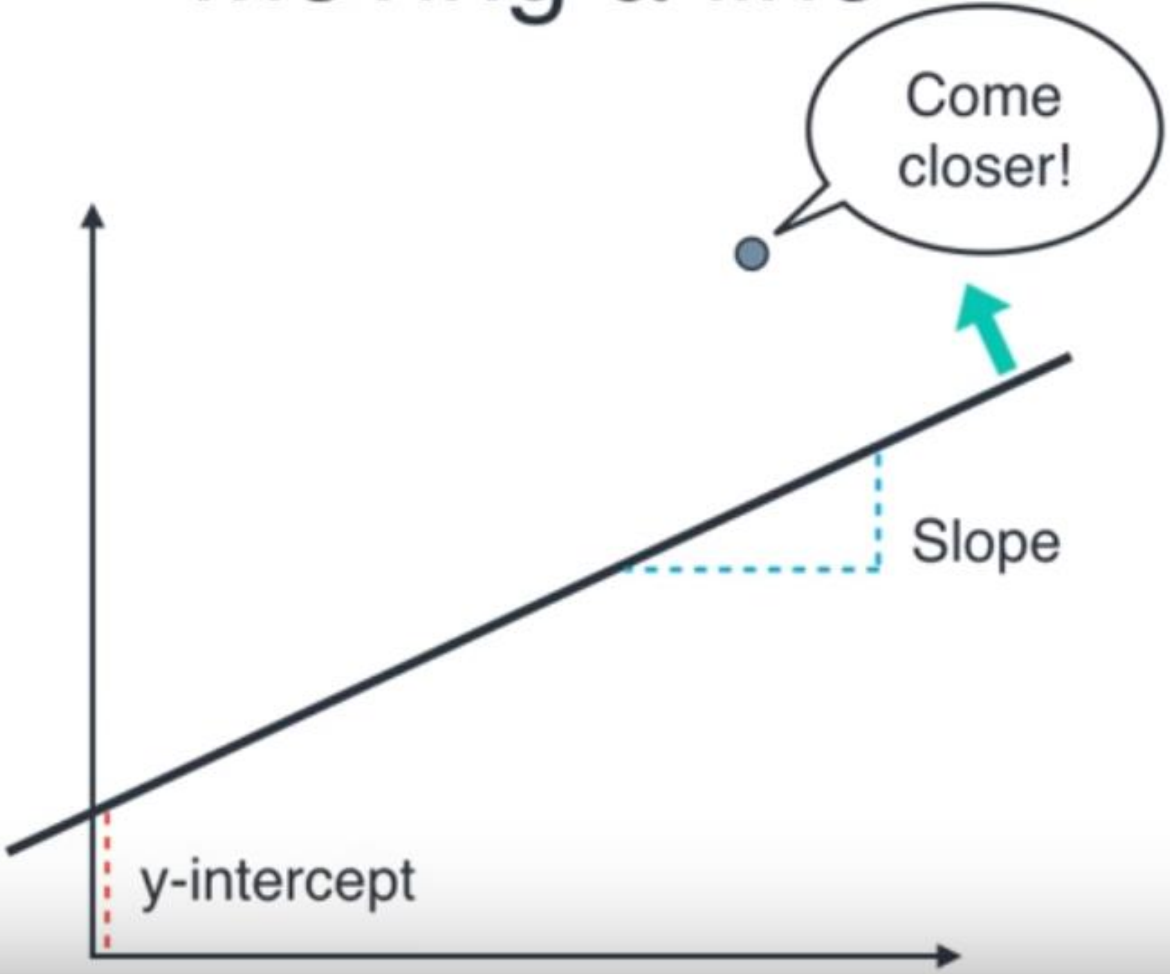


Linear Regression

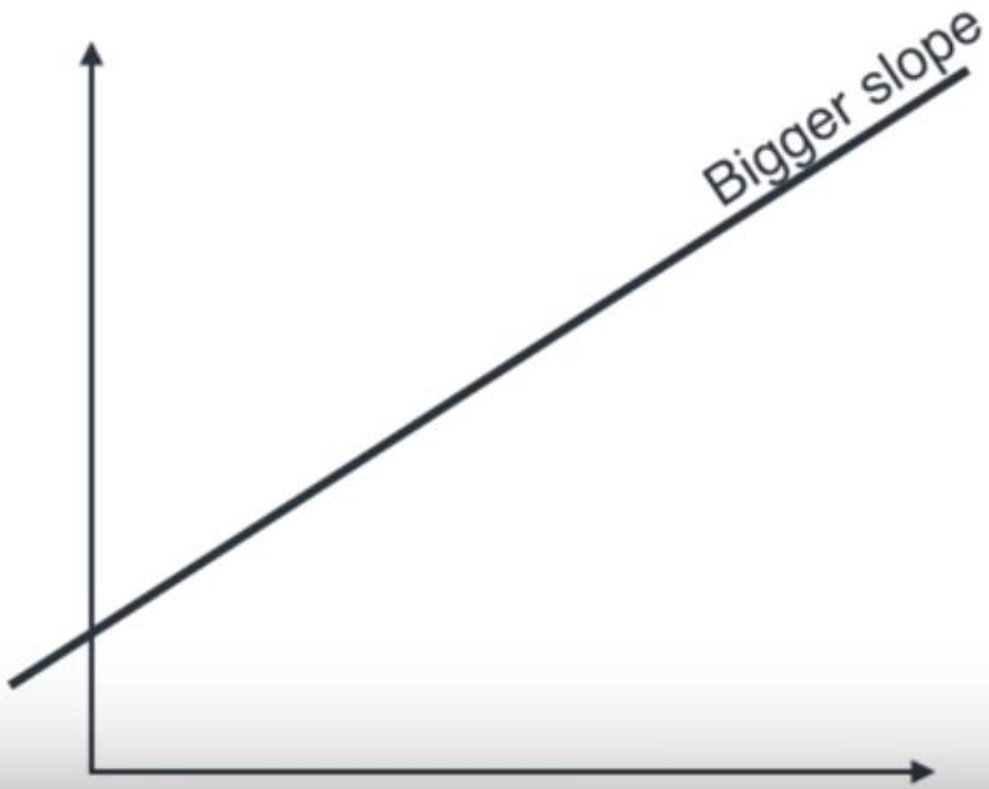


How do we move a line closer to a point?

Moving a line

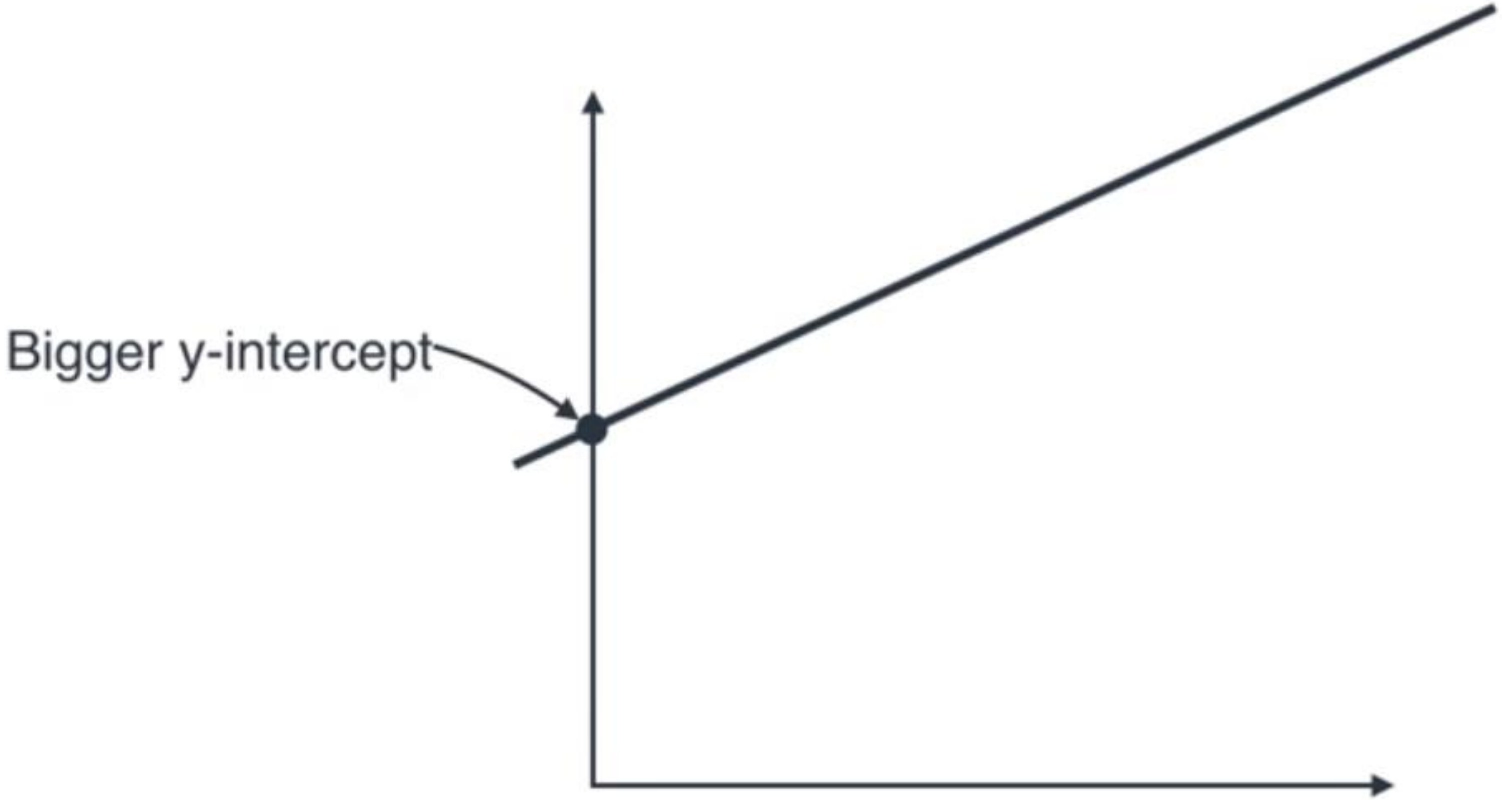


Changing the slope - Rotation

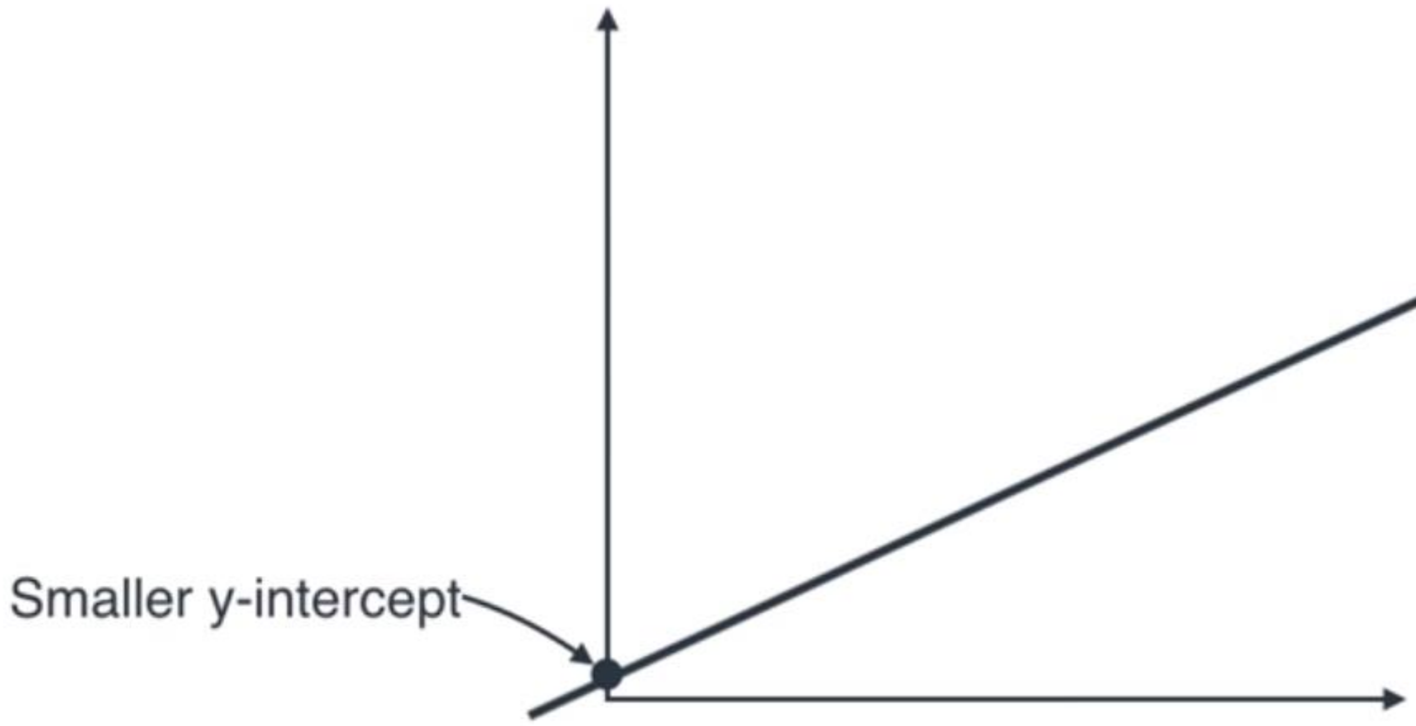


Changing the slope - Rotation

Changing the y-intercept - Translation

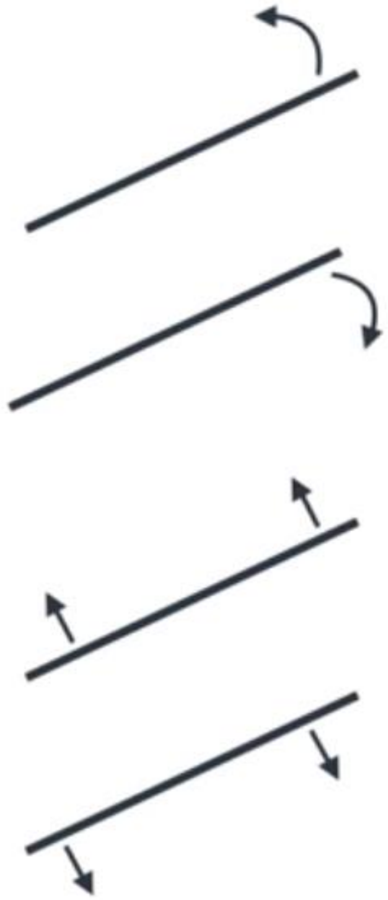


Changing the y-intercept - Translation

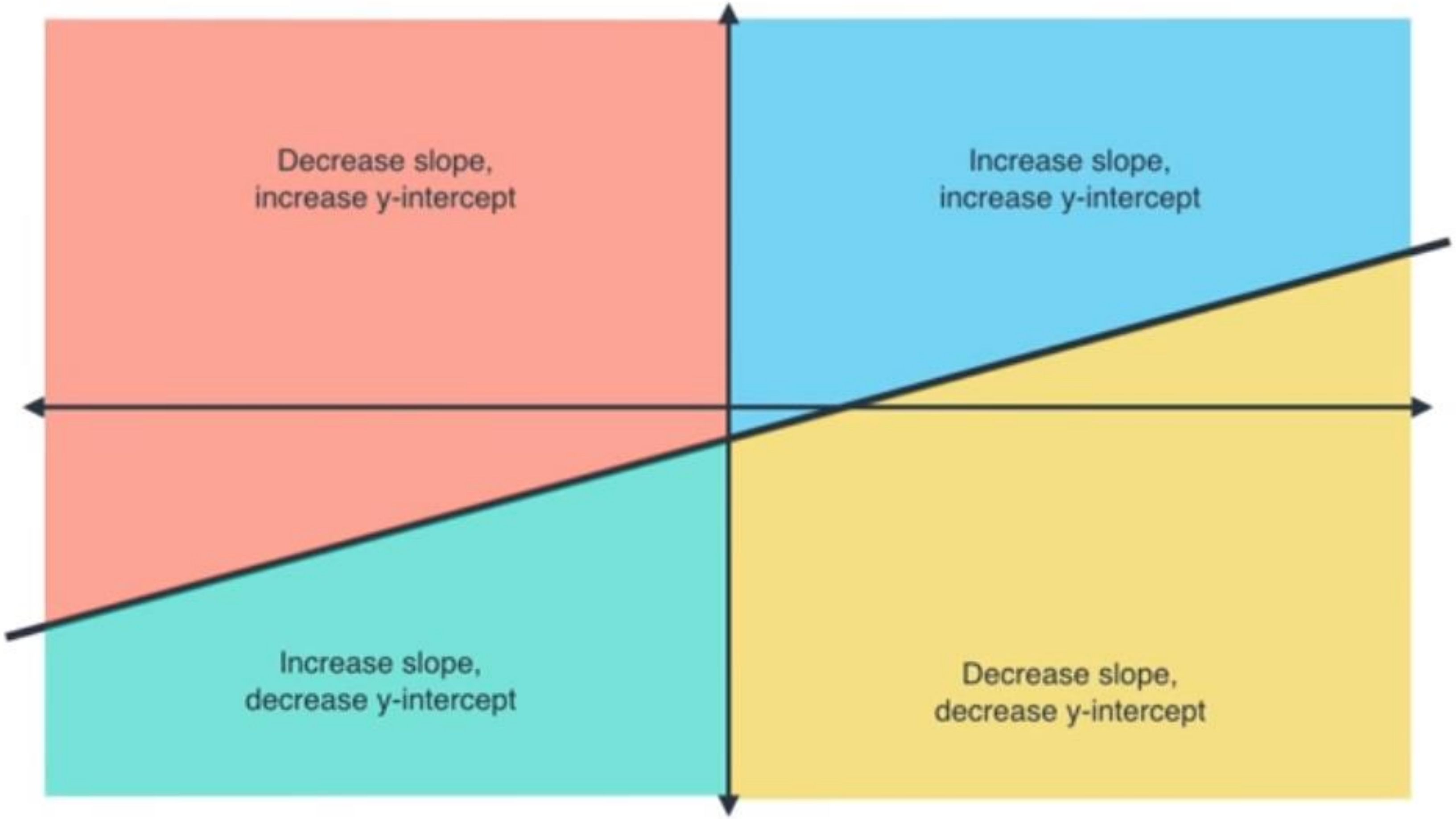


How to move a line

Rotate line counter-clockwise	Increase slope
Rotate line clockwise	Decrease slope
Translate line up	Increase y-intercept
Translate line down	Decrease y-intercept



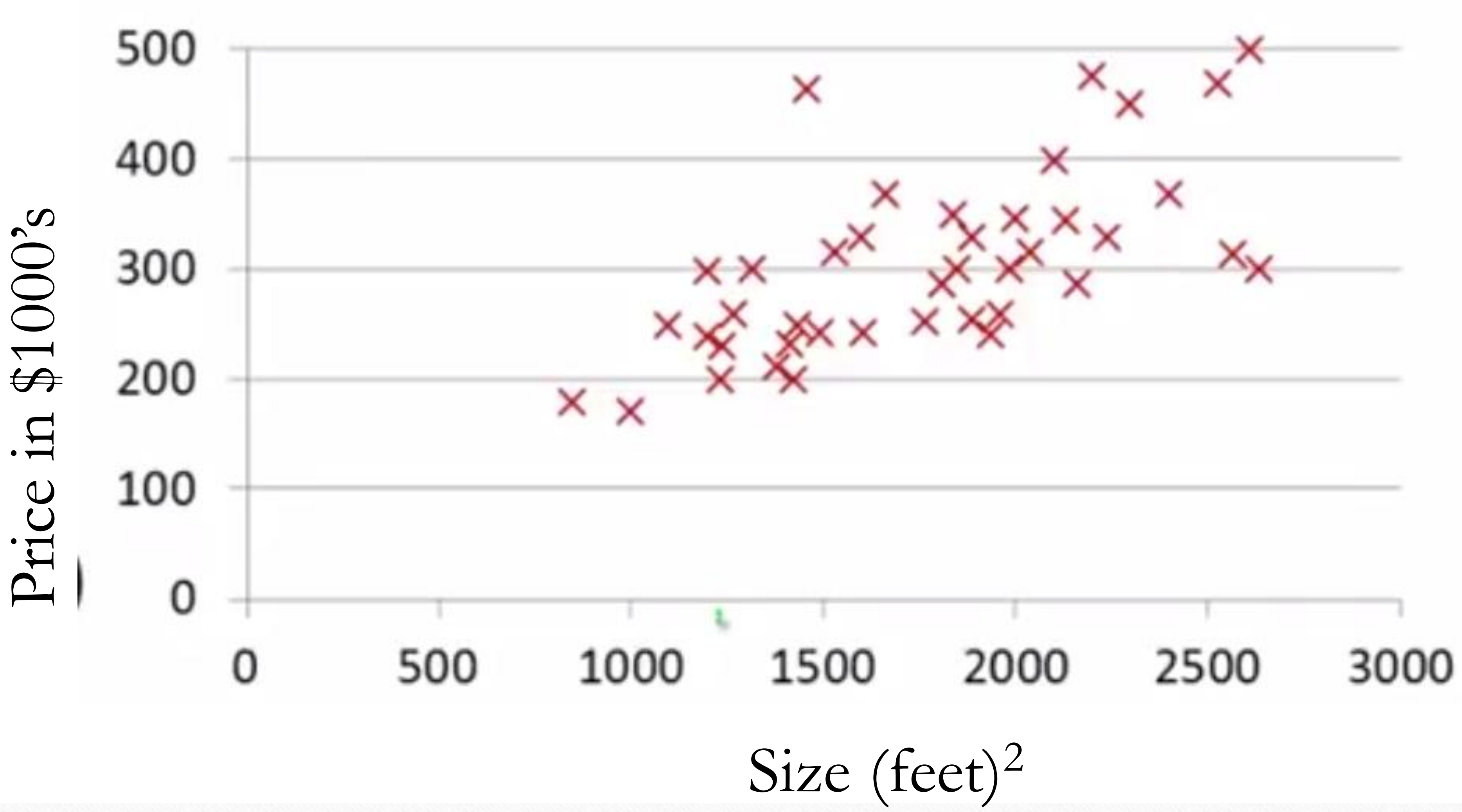
Four cases


Decrease slope,
increase y-intercept
Increase slope,
increase y-intercept
Increase slope,
decrease y-intercept
Decrease slope,
decrease y-intercept

What are these errors actually mean?

Carguru example

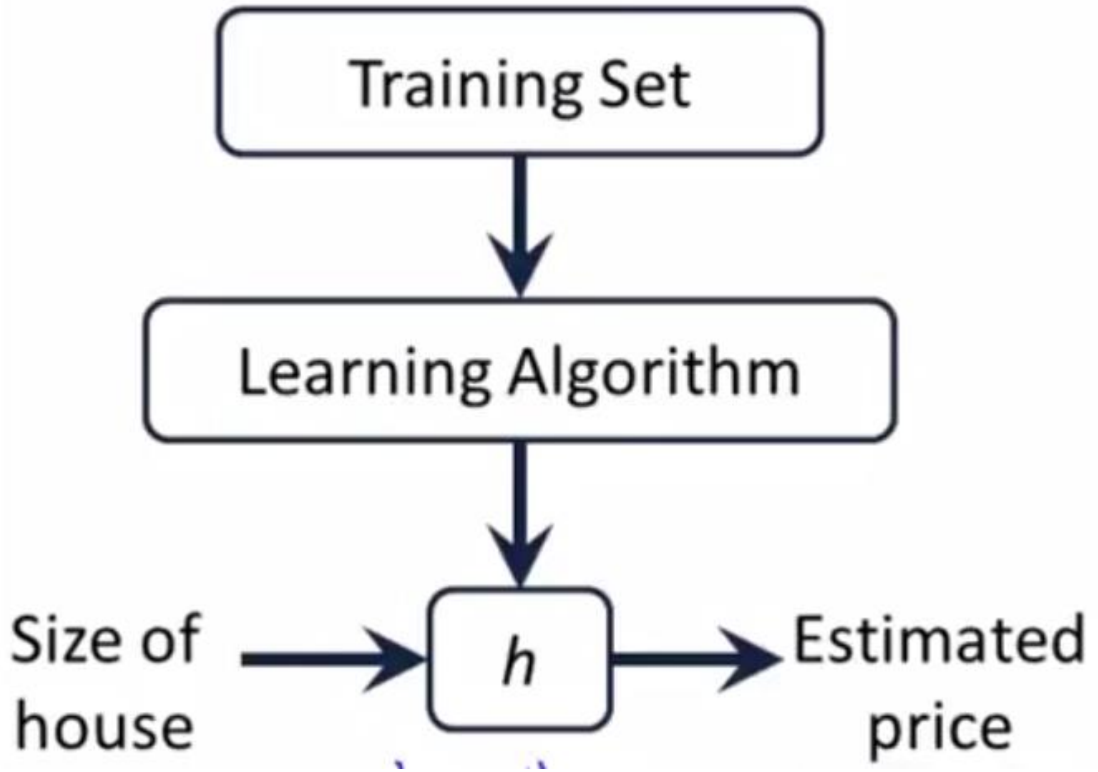
Math in Machine!!


500
400
300
200
100
0
0
500
1000
1500
2000
2500
3000
Price in $1000's
Size (feet)^2

Notations

- n = number of training examples
- x's = input variable/features
- y's = output variable/ "target" variable
- (x,y) = one training example
- $(x^{(i)}, y^{(i)})$ = i^{th} training example

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

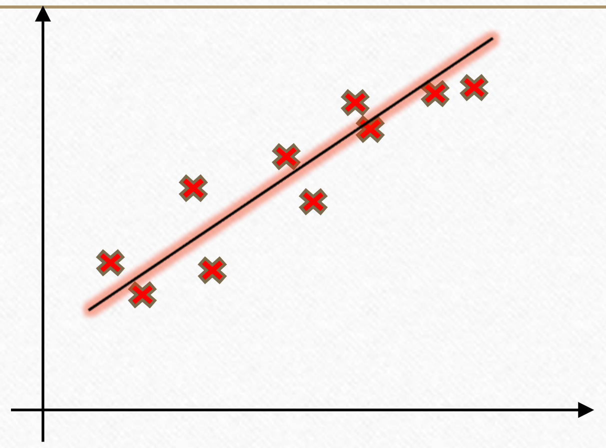


h maps from x's to y's

How do we represent h?

- h(x) = mx+c

- Linear Regression with one variable
  /Univariate Regression



How do we know that this line is the best fit?

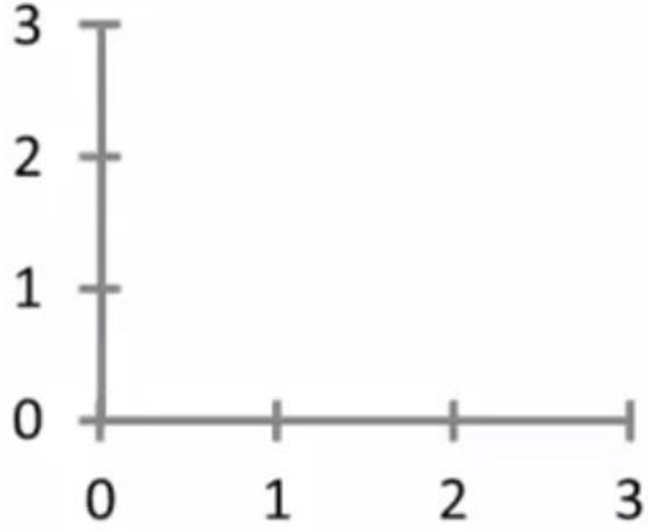
Univariate Regression COST FUNCTION!

- Help us to figure out how to best fit the line to our data

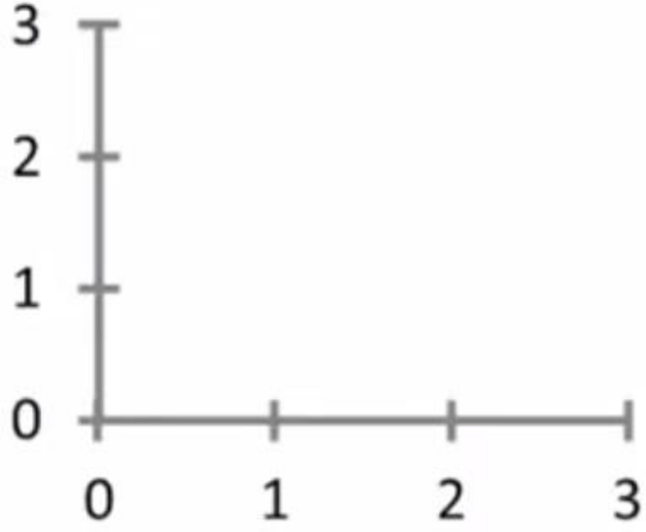
- **Hypothesis: $h(x) = mx+c$**

- m,c = parameters of the model

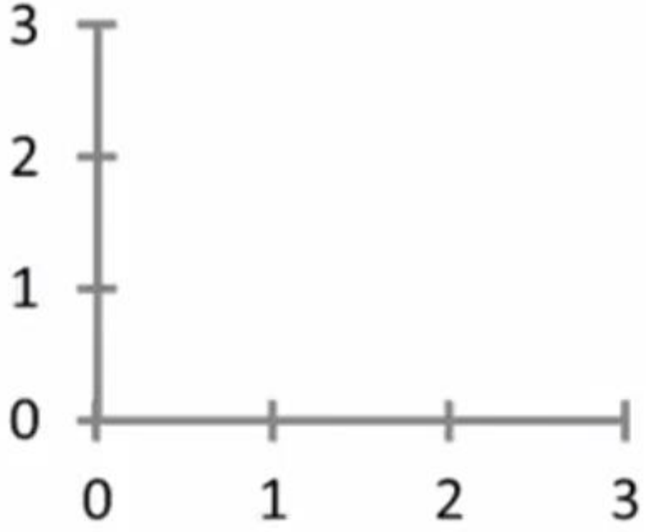
- How to choose m, c ?



$c = 1.5$
$m = 0$



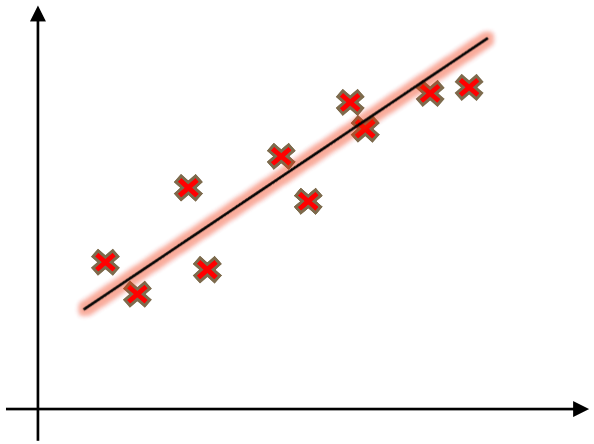
$c = 0$
$m = 0.5$



$c = 1$
$m = 0.5$

$h(x) = mx+c$

Idea; Choose m, c so that h(x) is close to y for our training examples (x,y)



Squared Error Cost Function

Recap!

- Hypothesis: h(x) = mx +c
- Parameters = m , c
- Cost function: J (c,m) = $\frac{1}{2n}\sum_{i=1}^{n}\left(h\ (x^{(i)}) - y^{(i)}\right)^2$
- Goal = minimize J (c,m)

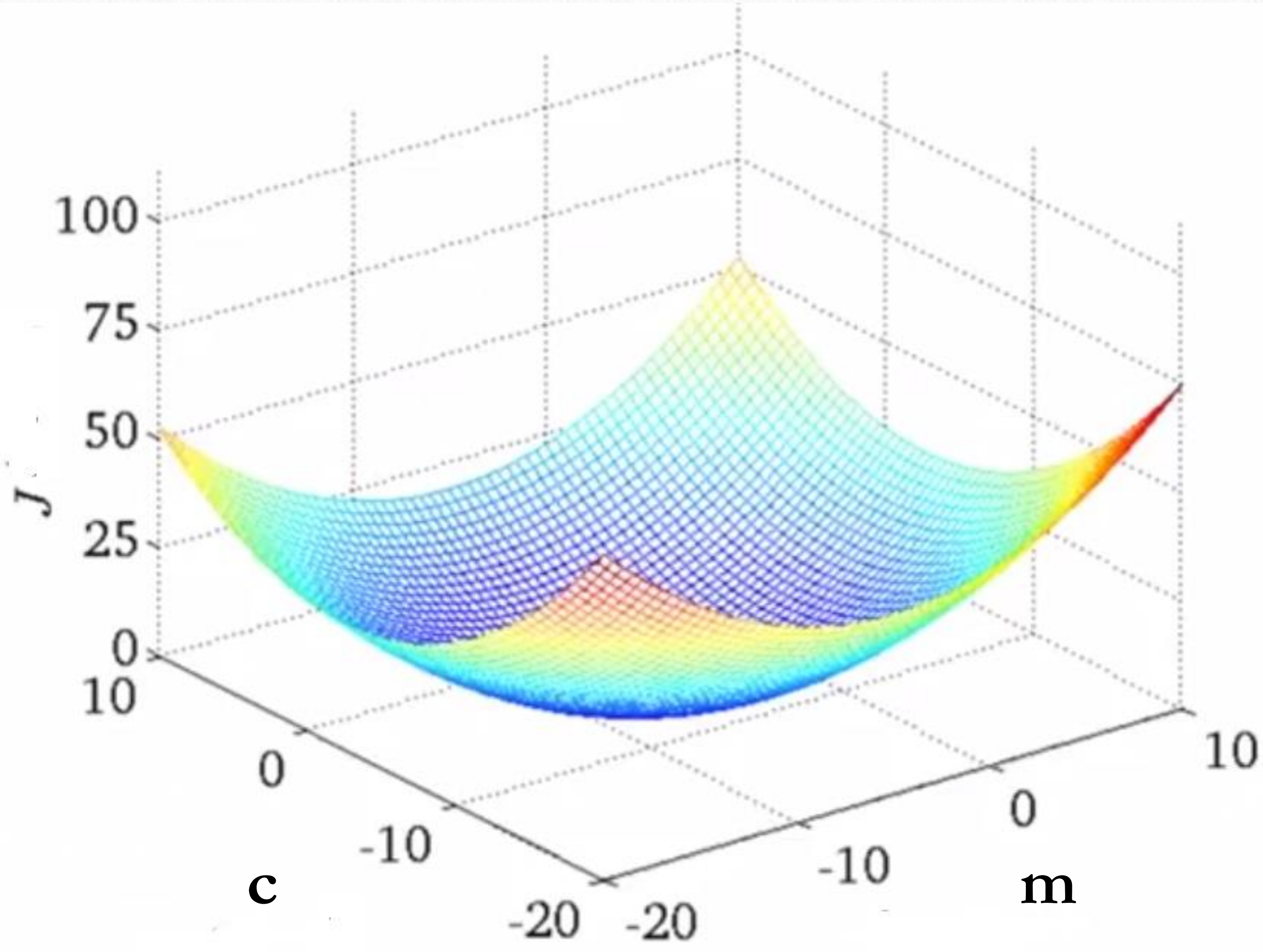
$C = 0$ (simplied version)

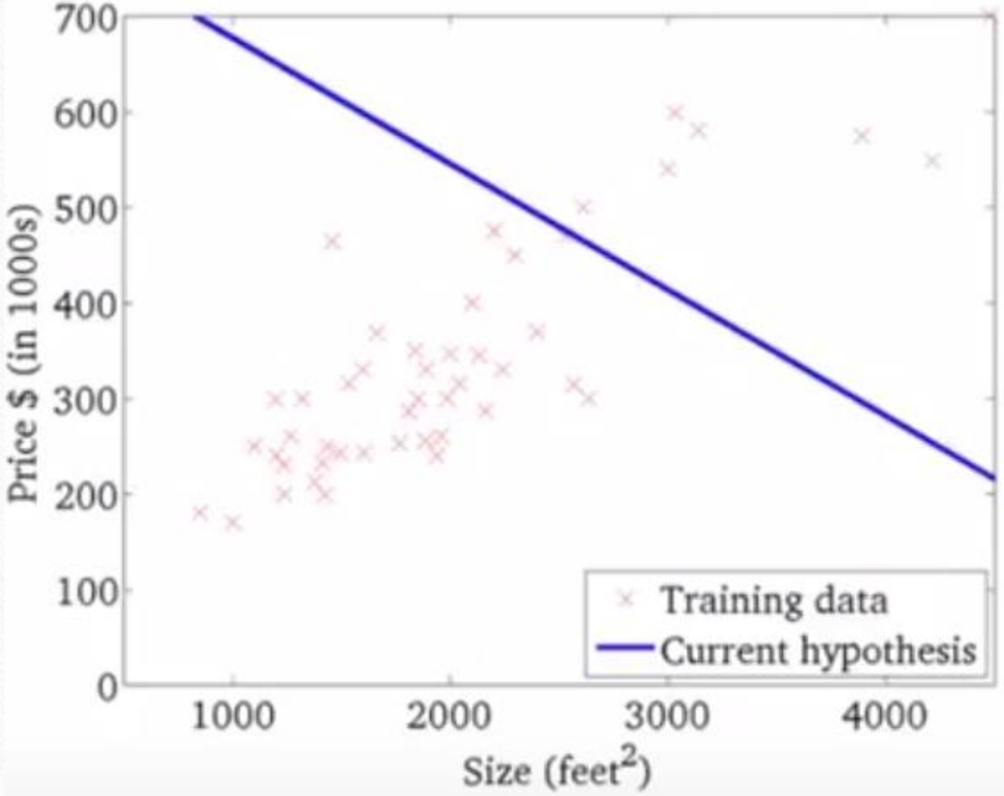
- $h(x)$ for fixed m is the function of x
- $J(m)$ is a function of m

- $m = 1, J = ?$
- $m = 0.5, J = ?$
- $m = 0, J = ?$

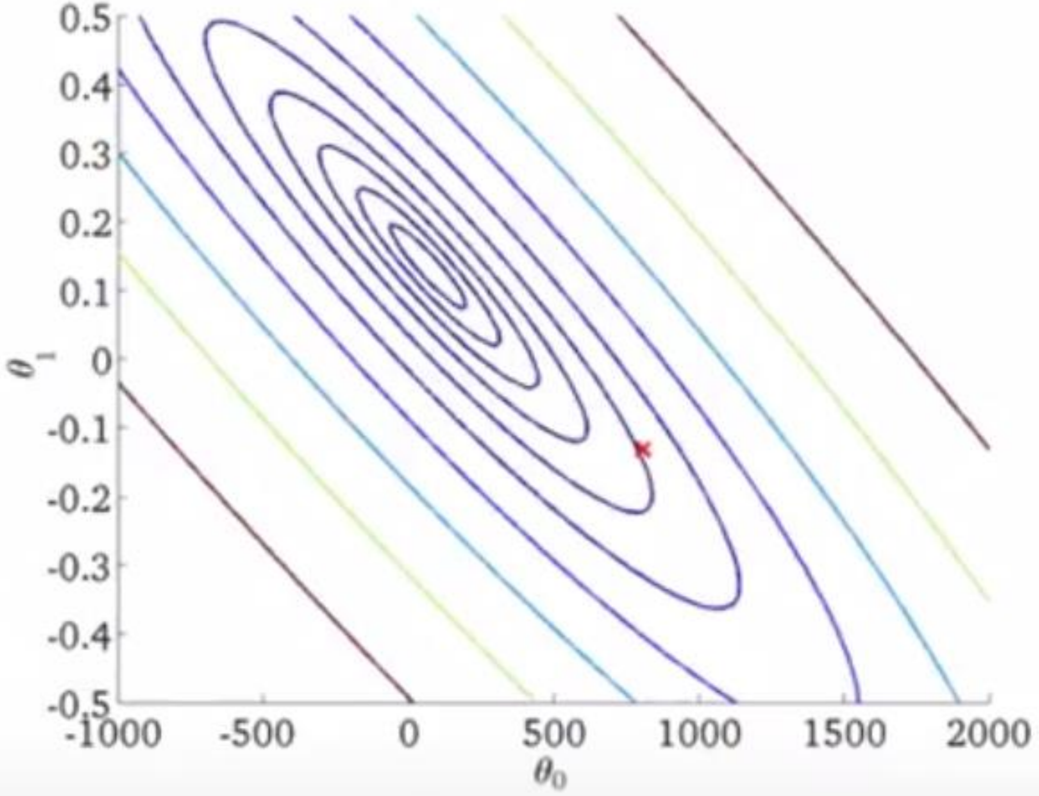
What is optimal value of m?

For c,m

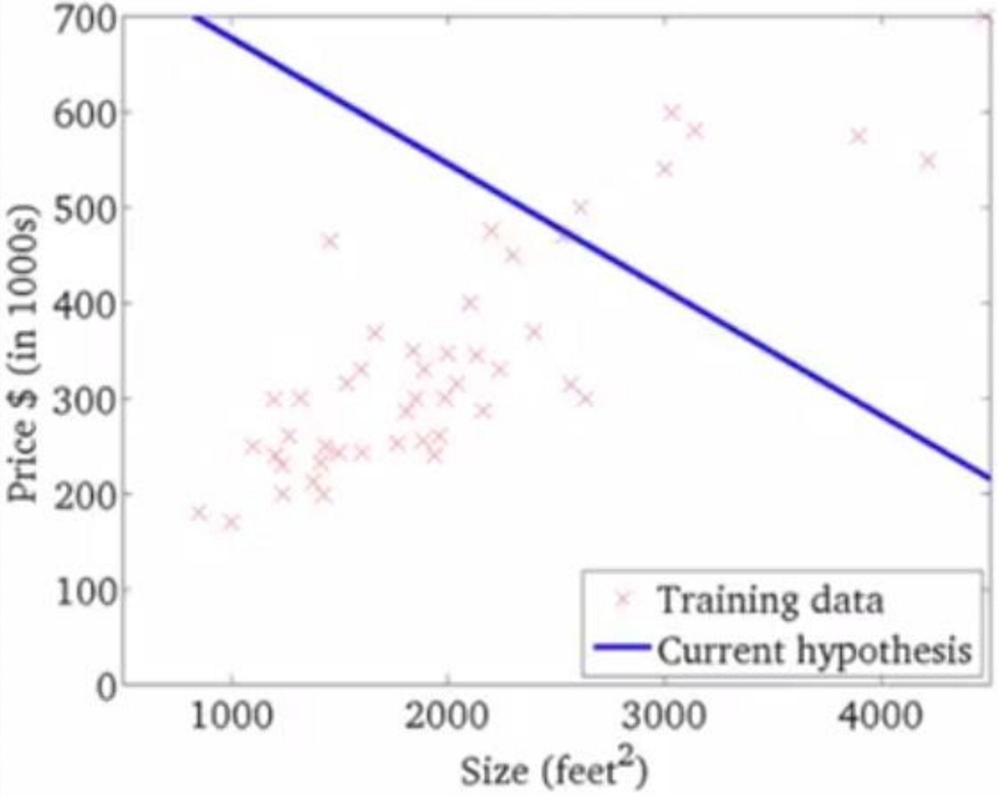




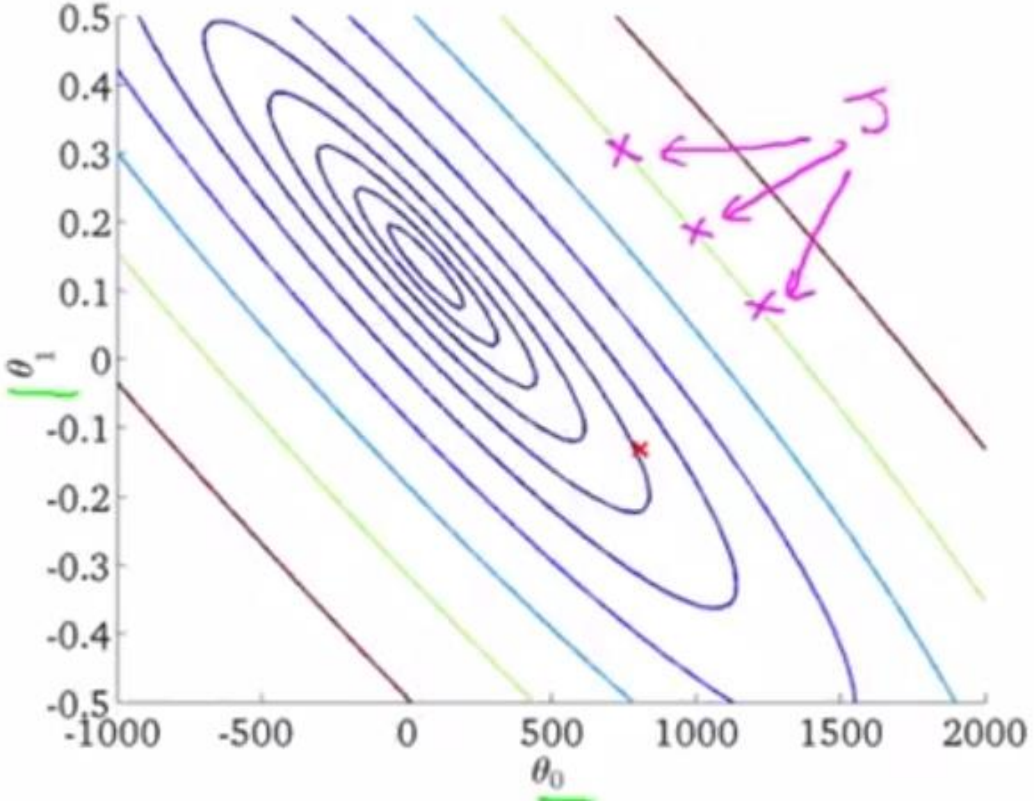
700
600
500
400
300
200
100
0
Price $ (in 1000s)
1000
2000
3000
4000
Size (feet^2)
Training data
Current hypothesis



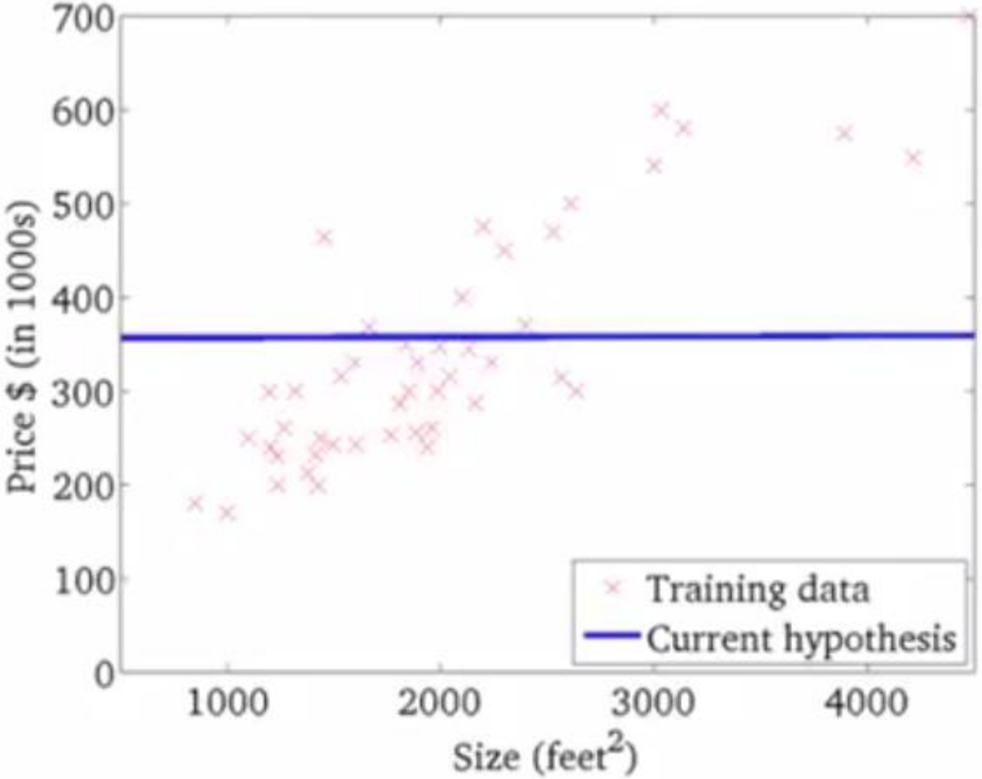
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
θ_1
-1000
-500
0
500
1000
1500
2000
θ_0



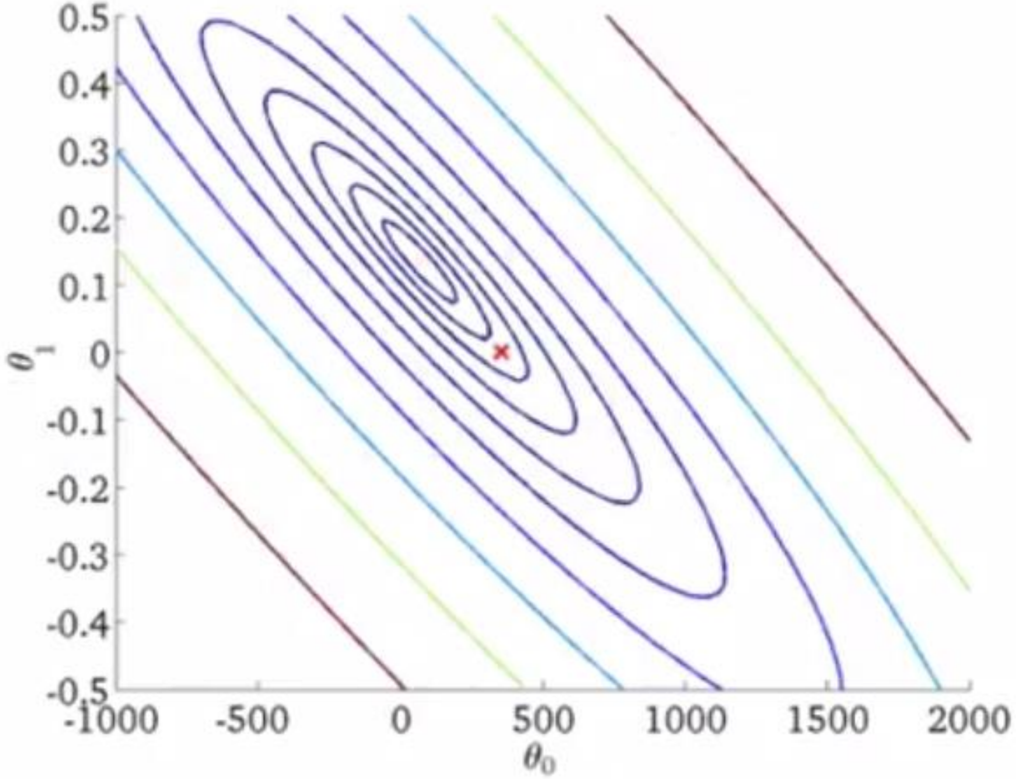
700
600
500
400
300
200
100
0
1000
2000
3000
4000
Price $ (in 1000s)
Size (feet^2)
Training data
Current hypothesis



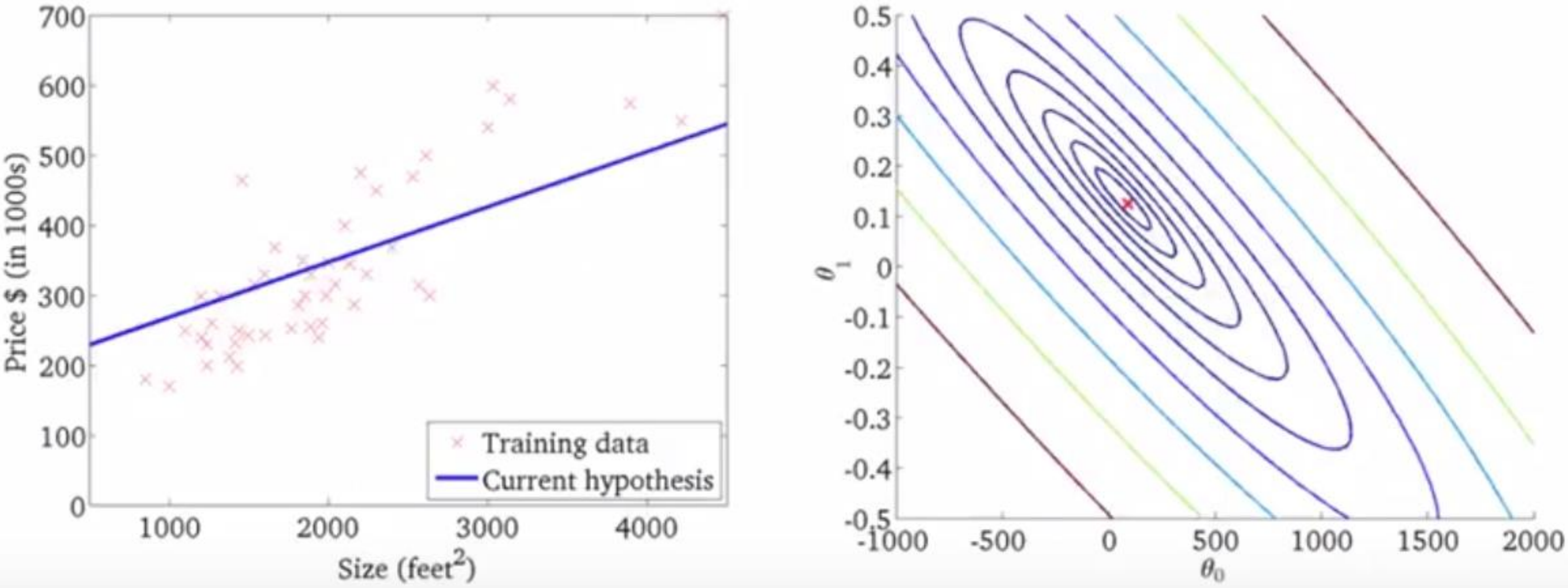
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
-1000
-500
0
500
1000
1500
2000
θ_1
θ_0
J



700
600
500
400
300
200
100
0
1000
2000
3000
4000
Size (feet^2)
Price $ (in 1000s)
Training data
Current hypothesis



0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
-1000
-500
0
500
1000
1500
2000
θ_0
θ_1



SSE is minimum in this case

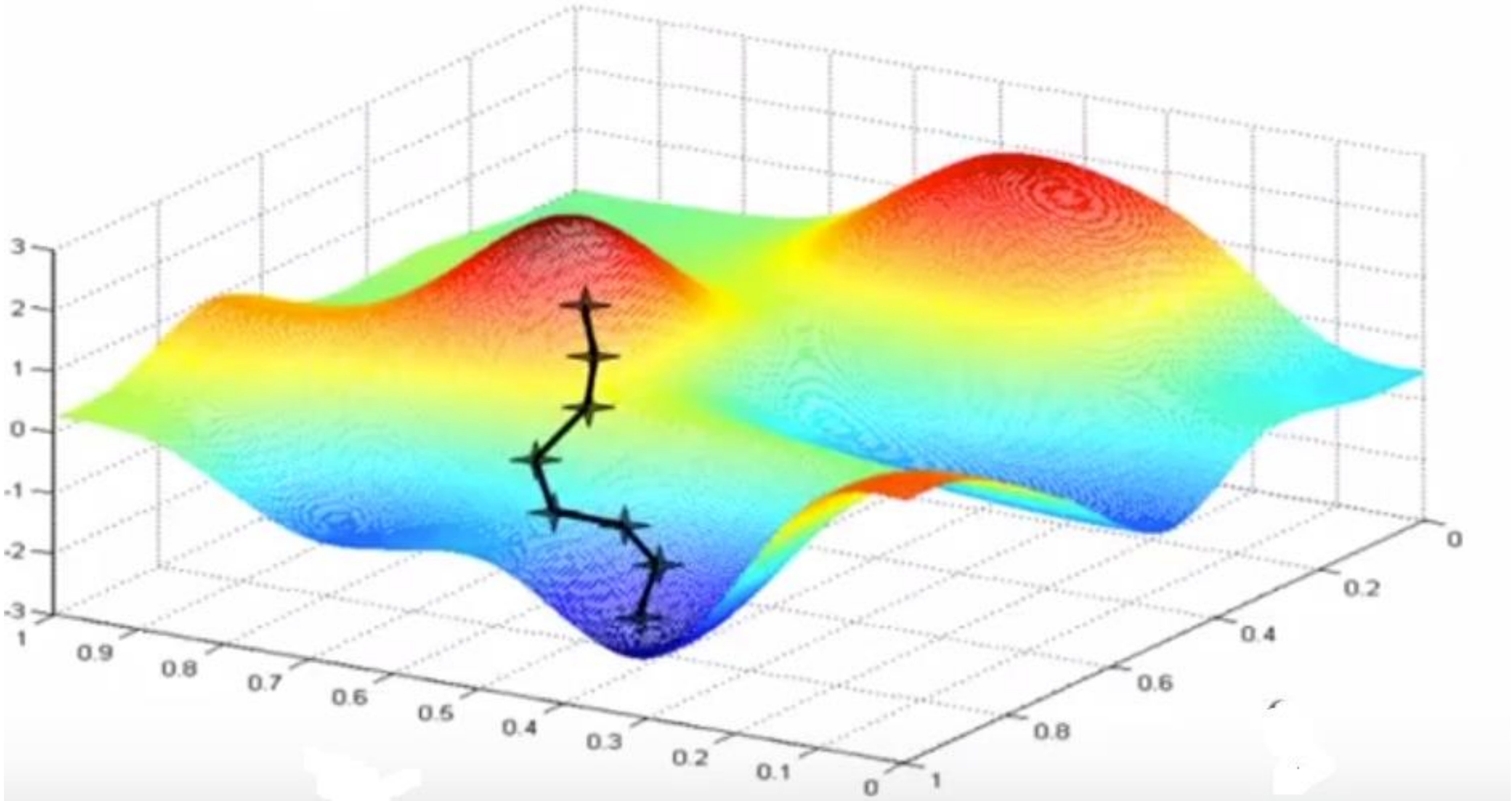
What do we need now?

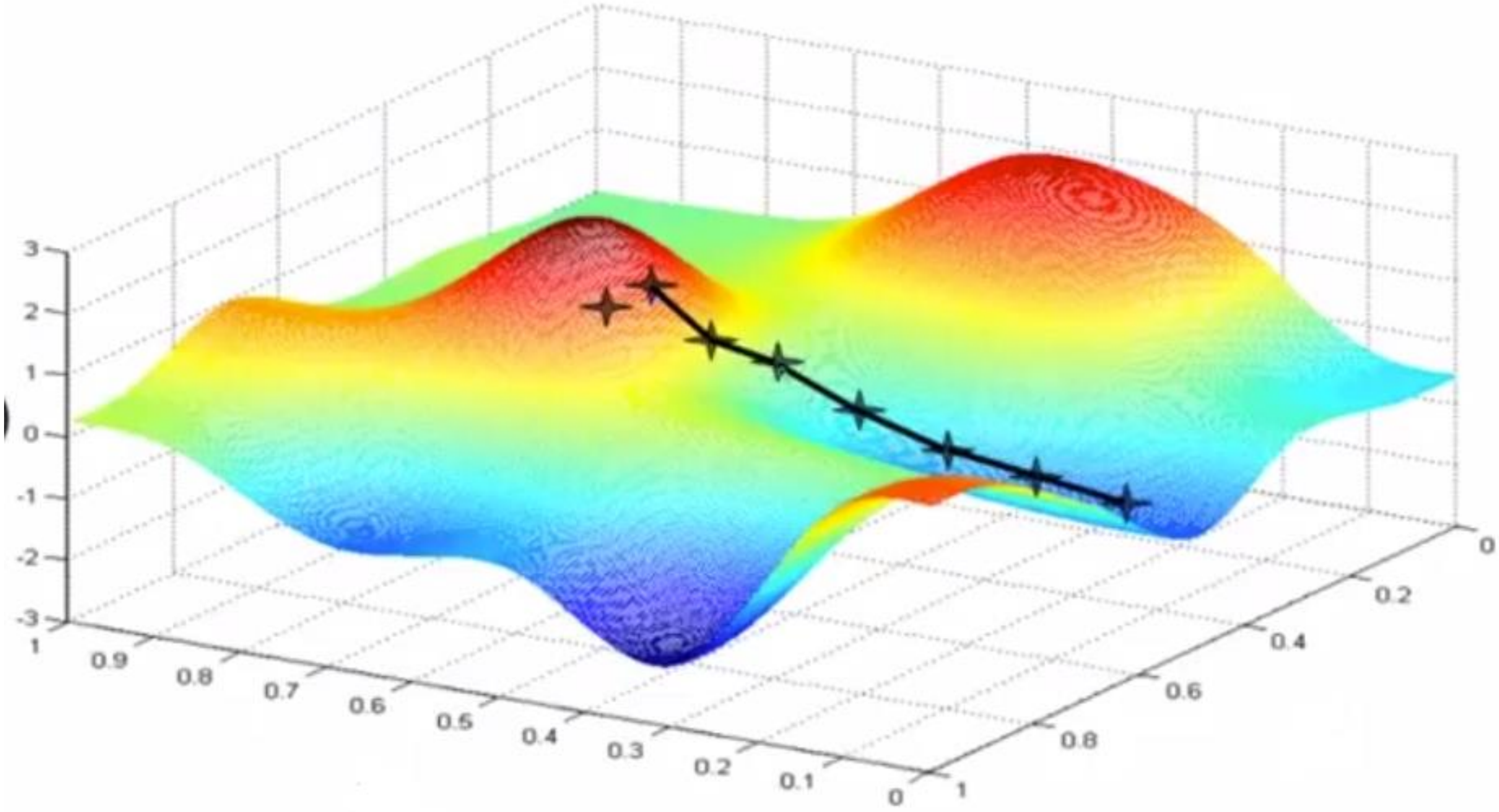
- We need some efficient algorithm which automatically find out best value of m,c which minimize the cost finction J

Gradient Descent

- Used for minimizing cost function.
- Have some function J(c,m)
- Want min J(c,m)
- Outline
  - Start with some c,m
  - Keep changing c, m to reduce J(c,m) until we hopefully end at a minimum.

https://www.youtube.com/watch?v=vWFjqgb-ylQ





3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.2
0.4
0.6
0.8
1

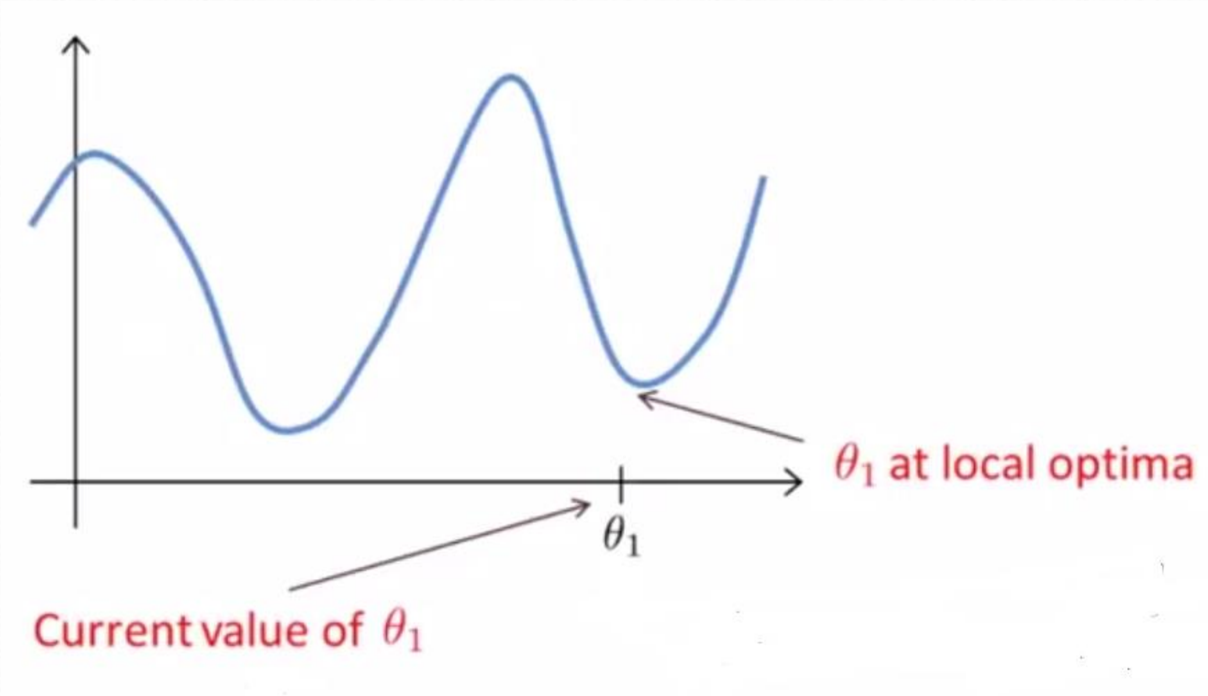
Learning rate (alpha)

- If alpha is too small, gradient descent can be slow.

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

(show example)

Question?



Hint: Slope

Time independent

- Gradient Descent can converge to a local minimum, even with a fixed learning rate.

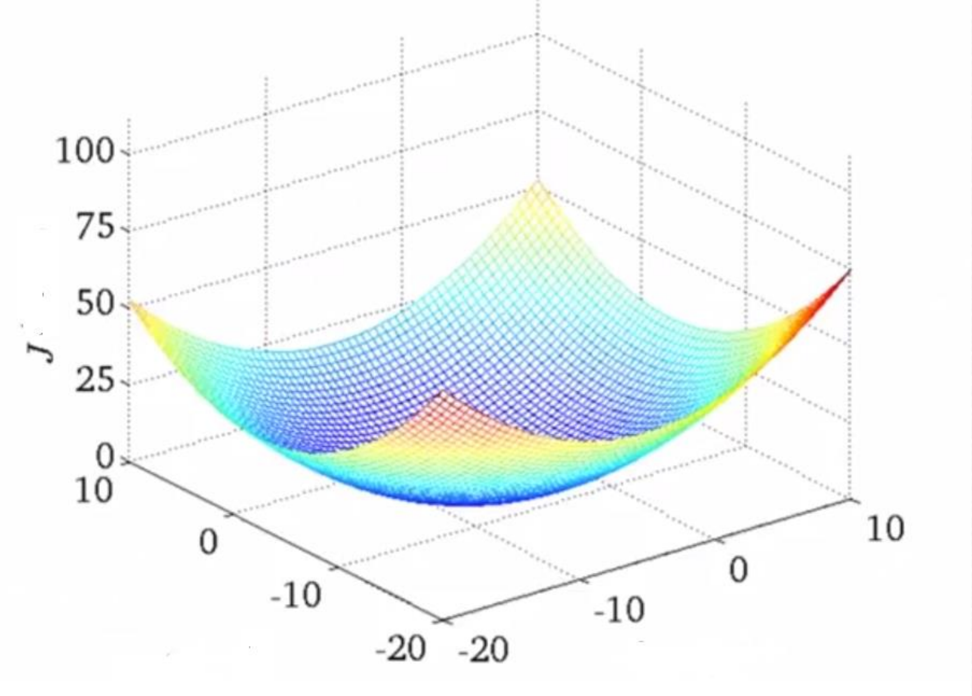
- As we approach a local minimum, gradient descent will automatically take smaller steps. (as slope is also decreasing) So, no need to decrease alpha over time

(Show example)

Gradient Descent for Linear regression

- Formula

Cost function for LR is always a bell function



Convex function

One global/local optima

Batch Gradient Descent

- Each step of gradient descent uses all training examples

Multiple Linear Regression

Size ($feet^2$)	Number of bedrooms	Number of floors	Age of home (years)	Price ($1000)
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
...	...	...	...	...

Notation

- k = 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 of MLR

$h(x) = cx_0 + m_1x_1 + m_2x_2 + m_3x_3 + m_4x_4$

For convienence of notation lets say $x_0 = 1$

$h(x) = m^Tx$

Gradient Descent for Multiple LR

- Formula

Feature Scaling

- Idea: Make sure features are on same scale

- E.g. : x_1 =size (0-4000 $feet^2$)
- x_2 = number of bedrooms (1-5)
- x_1 = size/4000
- x_2 = number of bedrooms/5

Feature Scaling

- Get every features approx. in range from $-1<x_i<+1$
- Other accepted range of values.

Mean Normalisation

Replace x_i with x_i-u_i to make feature has approx. zero mean.

x_1 = size – 2000/4000

x_2 = #bedrooms – 2/5 $-0.5<x_i<0.5$

Standardization

We can also use standard deviation in denominator instead of range/max value

How to know if gradient descent is working correctly?

(show example)

- Automatic convergence test:

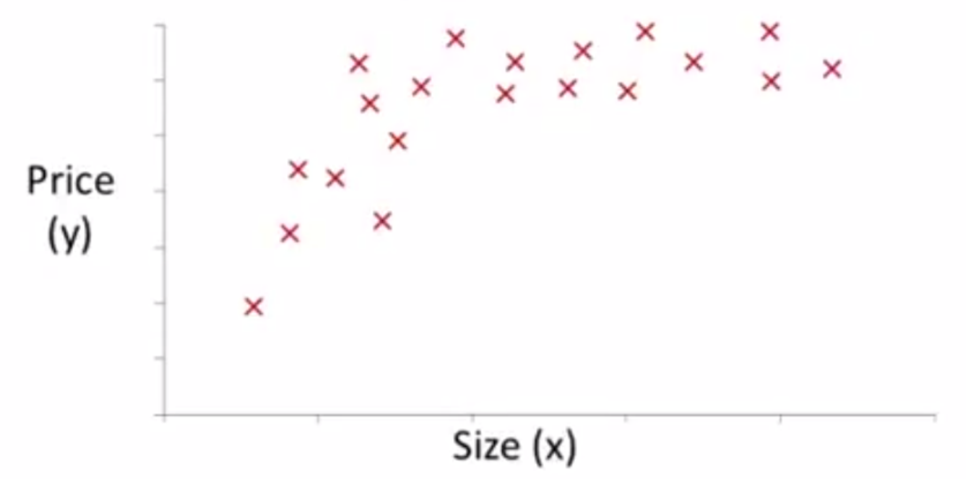
  declare convergence if J decreasesby less than 10^{-3} in one iteration.

If not working!

- Use smaller alpha (show example)
- For sufficiently small alpha, J should decrease on every iteration
- But if alpha is too small, gradient can be slow to converge
- If alpha is too large may not decrease on every iteration

What value of alpha should I choose?

Polynomial Regression



$m_1x + m_2x^2 + c$

or

$m_1x + m_2x^2 + m_3\, x^3 + c$

In this case feature scaling is very important

Normal Equation

- Method to solve for ‘m’ analytically
- Instead of solving value of m iteratively we can solve the optimum value of m in one step.

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

Gradient Descent	Normal Equation
• Need to choose alpha	• No need to choose alpha
• Needs many iteration	• Don’t needs many iterate
• Needs feature scaling	• No needs feature scaling
• Works well even when k(#features) is large	• Need to compute $(X^TX)^{-1}$ (Slow if k is large) $O(n^3)$ $n < 10000$

Assumptions

- Linear relationship
- Multivariate normality
- No or little multicollinearity
- Homoscedasticity

Problems With Linear Regression

- Limited to Linear Relationships

  Which can be overcome by using higher degree polynomial

- Only Looks at the Mean of the Dependent Variable

  e.g., babies are at risk when their weights are low, so you would want to look at the extremes in this example.

- Sensitive to Outliers

- Data Must Be Independent

Advantages

1. Very simple and easy to implement
2. Need less training data
3. No parameter tuning
4. As it is fast, it can be used in real time prediction
5. Performs well on new data which is not present in the training data

Applications in real world

Extensively used in all the businesses

Banking/Financial domain

Retail Industry

Python code implementation

- Jupyter Notebook

Discussion

Thank you!!!!!!