Linear Regression

For model with one predictor [$y = b_0 + b_1 x$]

$$b_0 = \bar{y} - b_1 \bar{x}$$

$$b_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

= Covariance (x,y) / Variance (x)

= rx,y * (sy / sx)

- Covariance :
  - In probability theory and statistics, covariance is a measure of the joint variability of two random variables.
  - The property of a function of retaining its form when the variables are linearly transformed.
  - In statistics : it is the mean value of the product of the deviations of two variates from their respective means.

Here,

- $\bar{x}$ and $\bar{y}$ are the average of the x_i and y_i, respectively
- r_{xy} as the sample correlation coefficient between x and y
- s_x and s_y as the uncorrected sample standard deviations of x and y

Salary data.

x years experience	*y salary (in $1000s)*
3	30
8	57
9	64
13	72
3	36
6	43
11	59
21	90
1	20
16	83

$$y = w_0 + w_1 x.$$

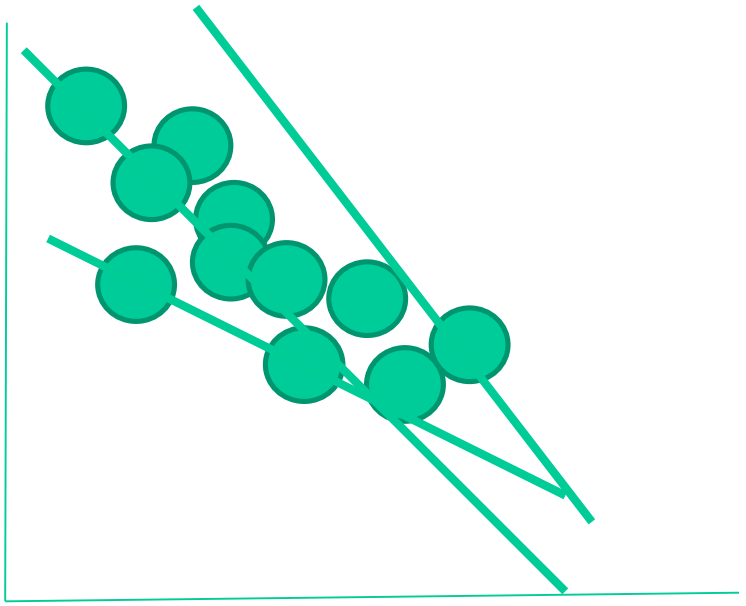
$$w_1 = \frac{\sum_{i=1}^{|D|} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{|D|} (x_i - \bar{x})^2} \qquad w_0 = \bar{y} - w_1 \bar{x}$$

Given the above data, we compute $\bar{x} = 9.1$ and $\bar{y} = 55.4$. Substituting these values into Equations (6.50) and (6.51), we get

$$w_1 = \frac{(3-9.1)(30-55.4)+(8-9.1)(57-55.4)+\cdots+(16-9.1)(83-55.4)}{(3-9.1)^2+(8-9.1)^2+\cdots+(16-9.1)^2} = 3.5$$

$$w_0 = 55.4 - (3.5)(9.1) = 23.6$$

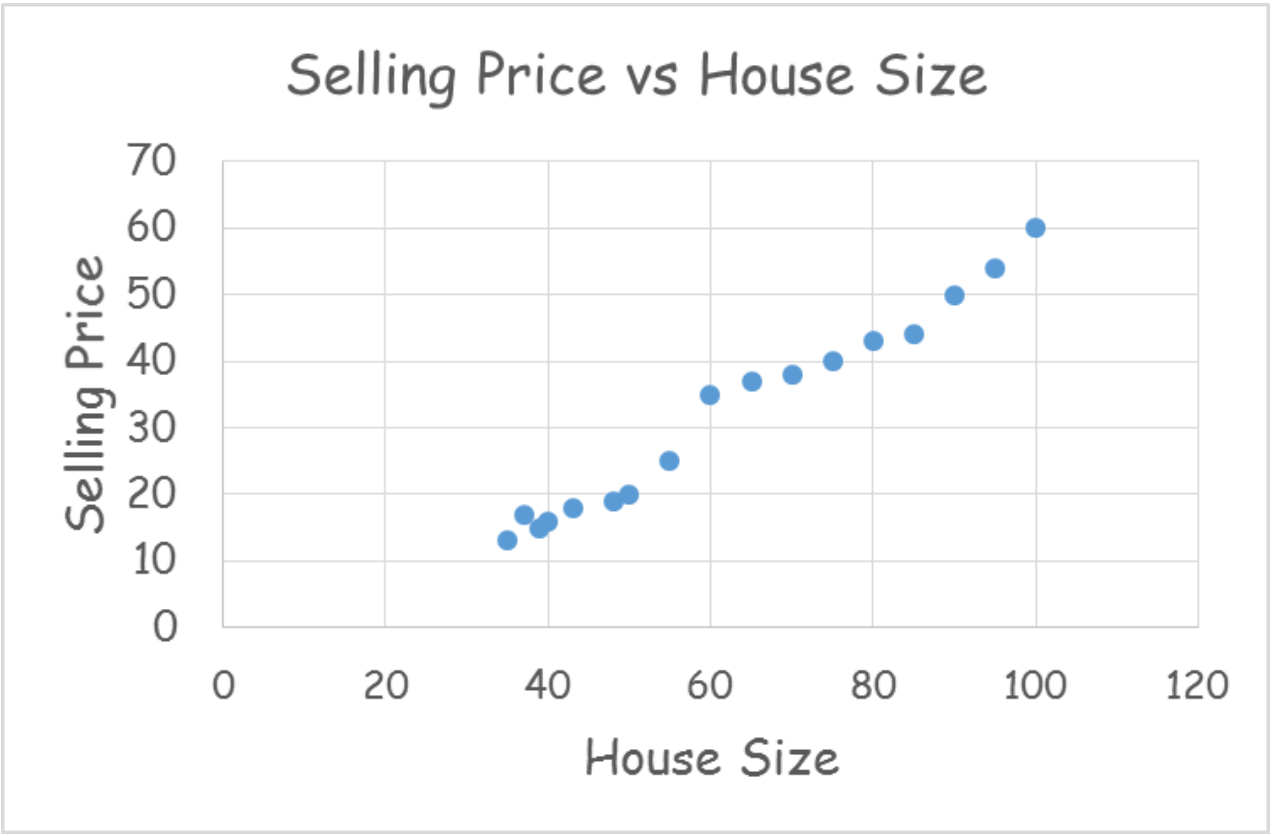
Thus, the equation of the least squares line is estimated by $y = 23.6 + 3.5x$. Using this equation, we can predict that the salary of a college graduate with, say, 10 years of experience is $58,600. ■




Rates of heart disease (% of population)
as a function of biking to work and smoking
Heart disease (% of population)
Biking to work (% of population)
0
5
10
15
20
0
20
40
60
Smoking
(% of population)
0.53
15.44
29.95

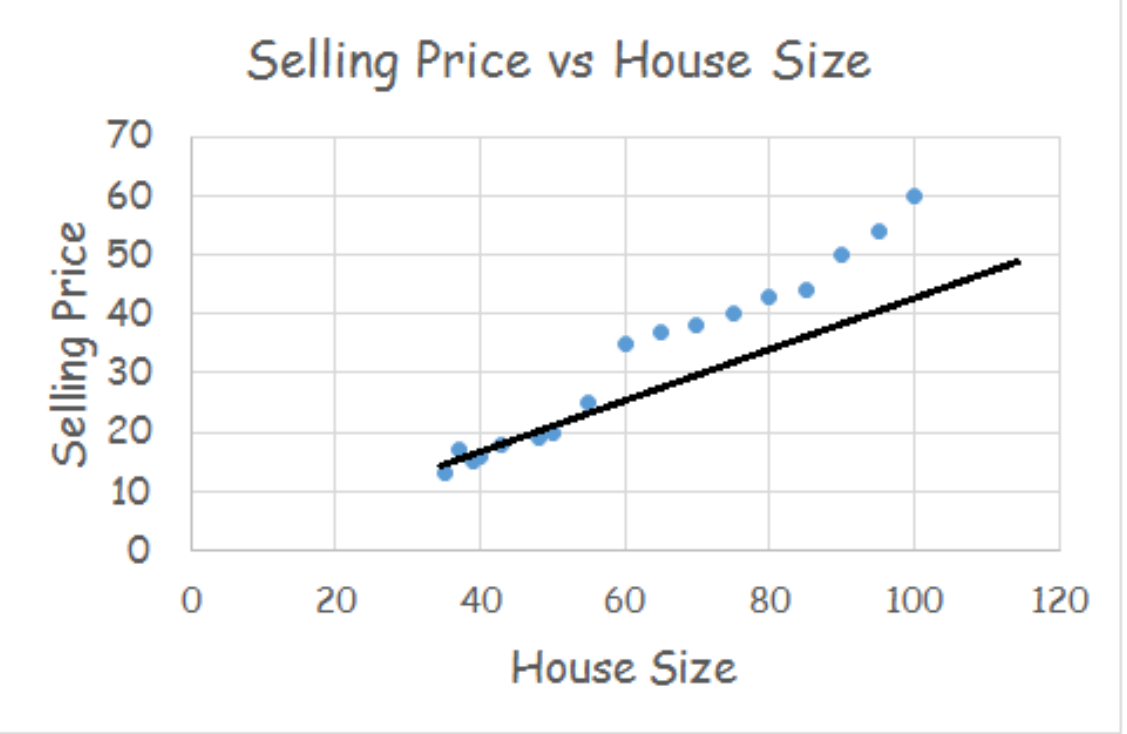
House Price Prediction

Area (SQM)	Selling Price (Lacs)
35	13
37	17
39	15
40	16
43	18
48	19
50	20
55	25
60	35
65	37
70	38
75	40
80	43
85	44
90	50
95	54
100	60



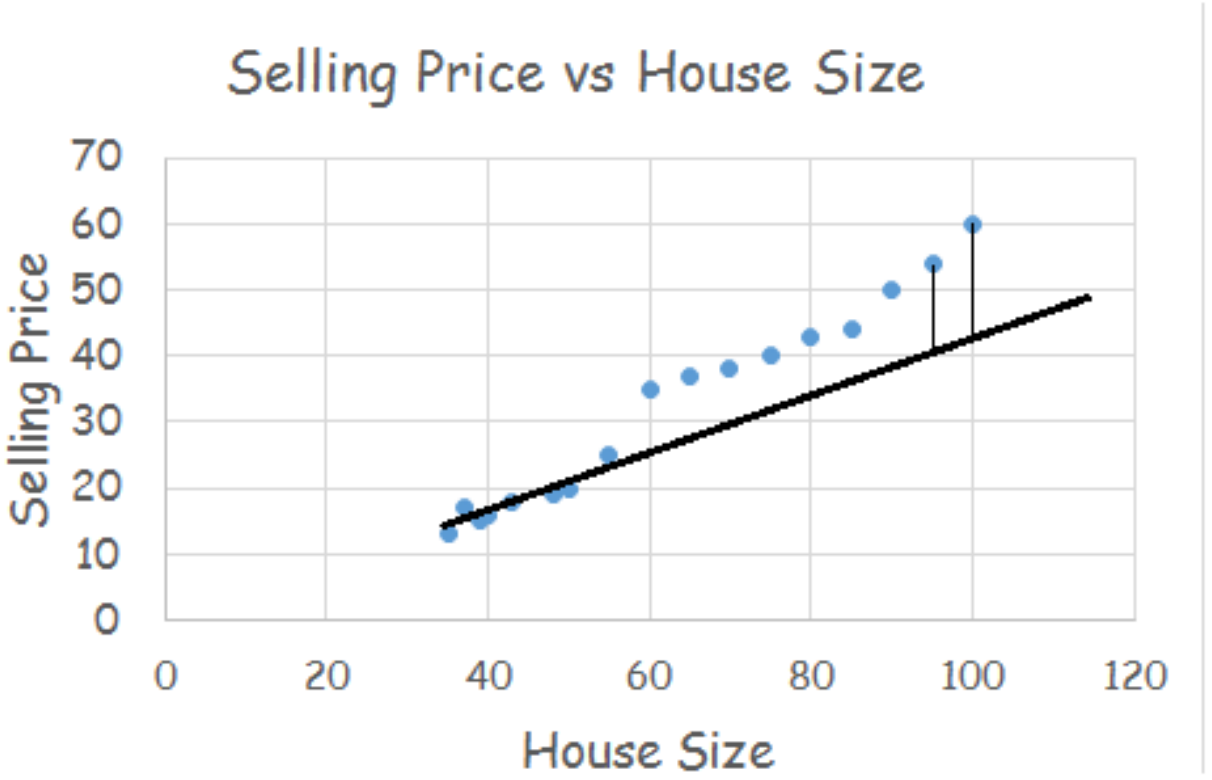
House Price Prediction

Area (SQM)	Selling Price (Lacs)
35	13
37	17
39	15
40	16
43	18
48	19
50	20
55	25
60	35
65	37
70	38
75	40
80	43
85	44
90	50
95	54
100	60



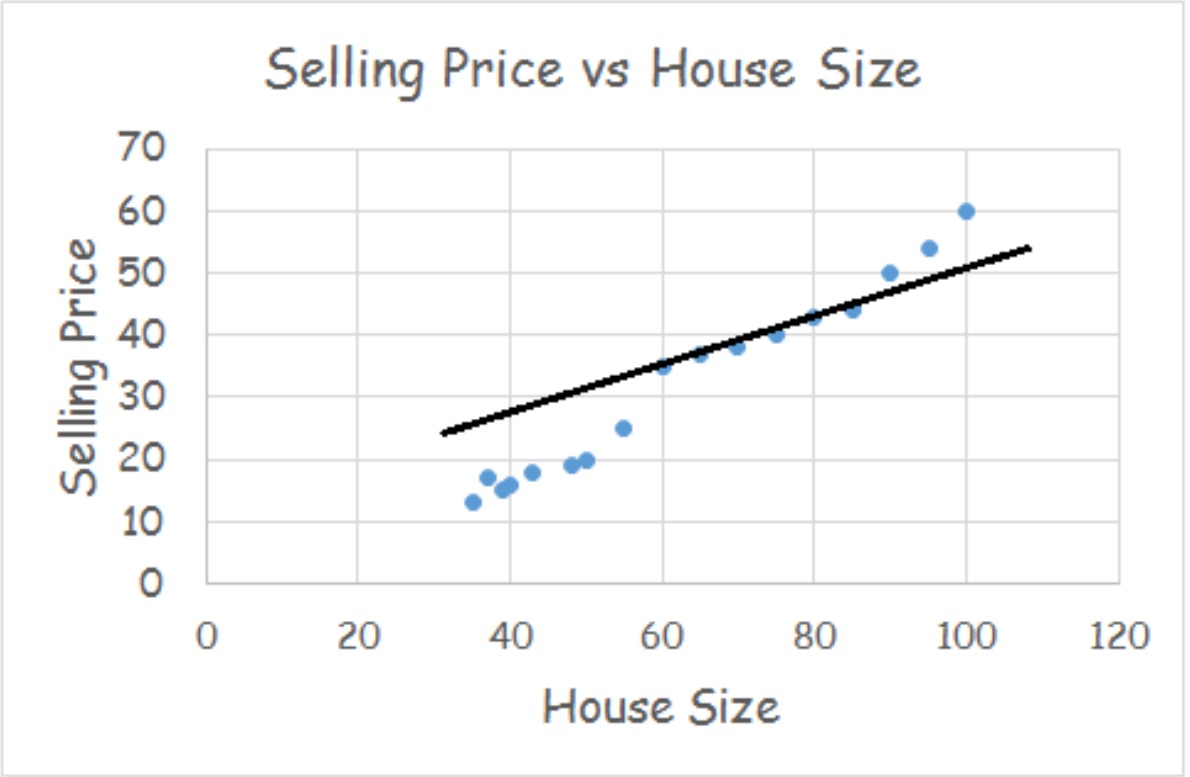
House Price Prediction

Area (SQM)	Selling Price (Lacs)
35	13
37	17
39	15
40	16
43	18
48	19
50	20
55	25
60	35
65	37
70	38
75	40
80	43
85	44
90	50
95	54
100	60



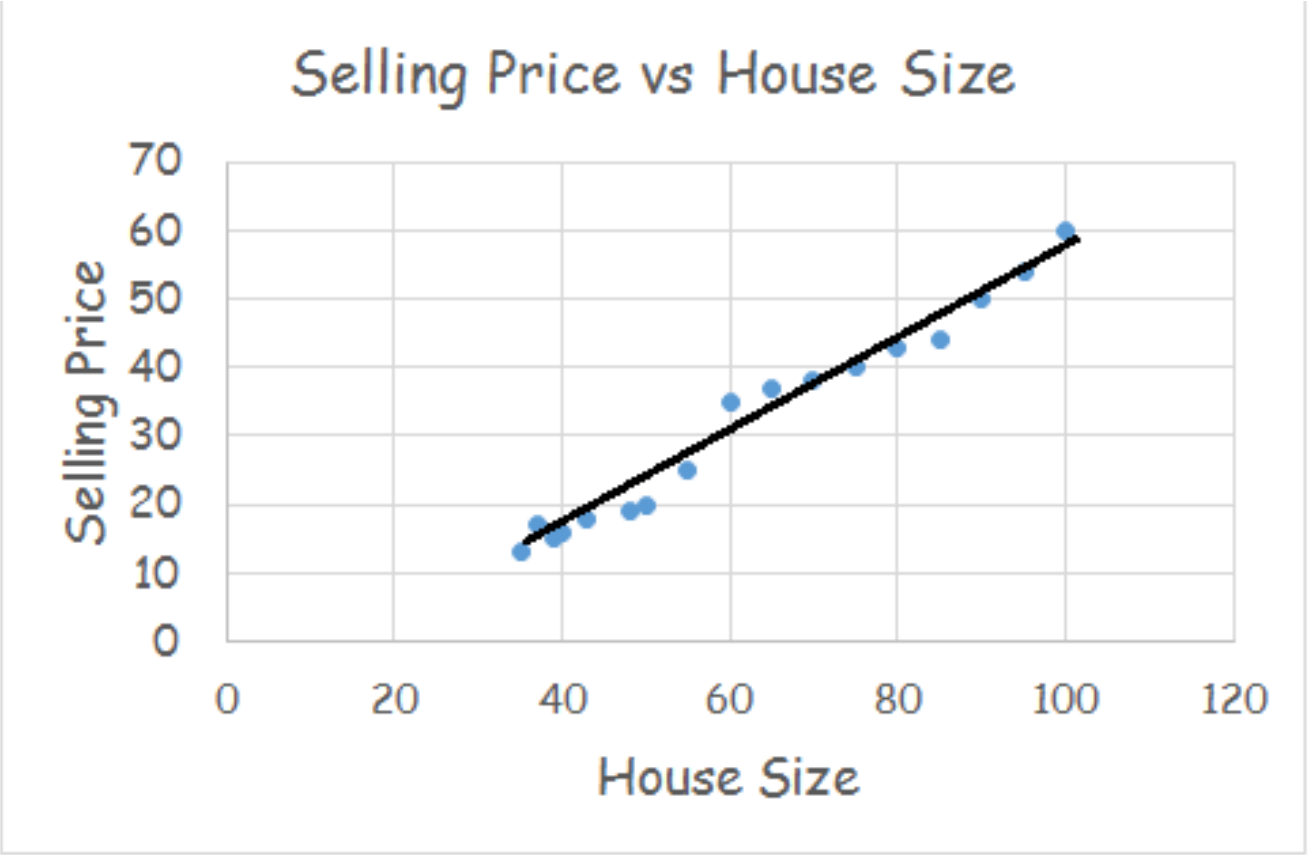
House Price Prediction

Area (SQM)	Selling Price (Lacs)
35	13
37	17
39	15
40	16
43	18
48	19
50	20
55	25
60	35
65	37
70	38
75	40
80	43
85	44
90	50
95	54
100	60



House Price Prediction

Area (SQM)	Selling Price (Lacs)
35	13
37	17
39	15
40	16
43	18
48	19
50	20
55	25
60	35
65	37
70	38
75	40
80	43
85	44
90	50
95	54
100	60



Hypothesis hΘ (x) = Θ0 + Θ1 x ; where Θi are termed as parameters of the model

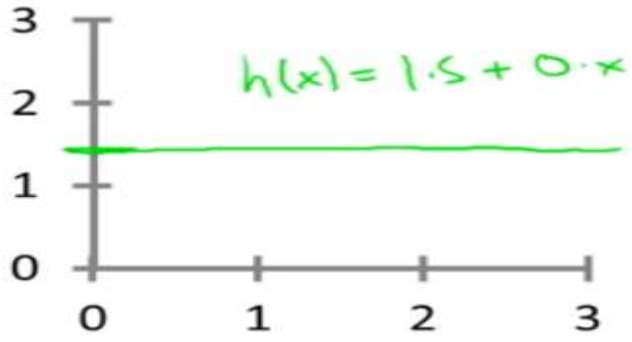
Challenge

how to choose optimal θ_i or different values of θ_1 and θ_0

This way we get different hypothesis functions

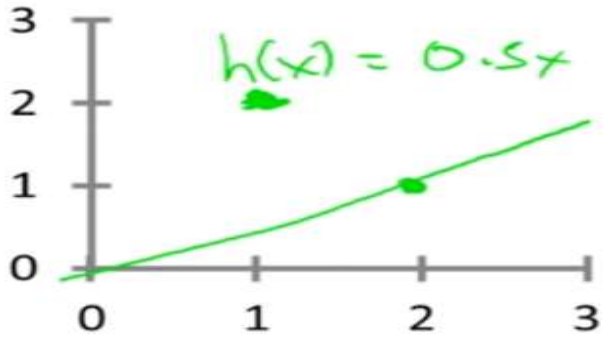
Idea is to choose Θ0 , Θ1, so that hypothesis hΘ (x) is close to y for our training examples (x,y)

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



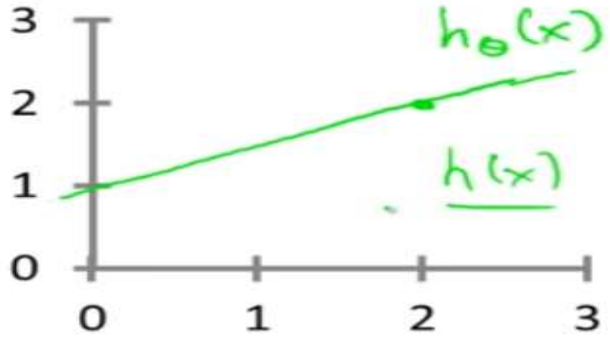
$\rightarrow \theta_0 = 1.5$

$\rightarrow \theta_1 = 0$



$\rightarrow \theta_0 = 0$

$\rightarrow \theta_1 = 0.5$



$\rightarrow \theta_0 = 1$

$\rightarrow \theta_1 = 0.5$

House Price Prediction

- Mathematically:
  - $Minimize \sum_{i=1}^{m}(y_i - \hat{y}_i)^2$ $\hat{y} = h_\theta(x) = \theta_0 + \theta_1 x$

OR

  - $Minimize(\frac{1}{2m})\sum_{i=1}^{m}(y_i - \hat{y}_i)^2$

OR

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

OR

  - $Minimize\ J(\theta_0, \theta_1)$
    θ_0, θ_1

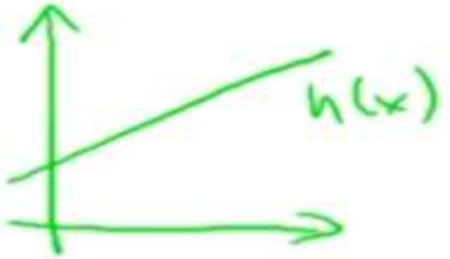
Linear regression : in summary

Hypothesis:

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

Parameters:

$$\theta_0, \theta_1$$



Cost Function:

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

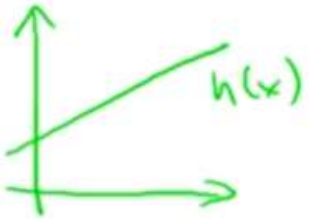
Goal: $\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$

Hypothesis:

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

Parameters:

$$\theta_0, \theta_1$$



Cost Function:

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

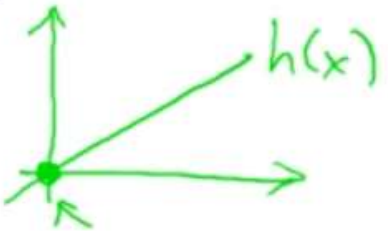
Goal: $\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$

Simplified

$$h_\theta(x) = \theta_1 x$$

$\theta_0 = 0$

θ_1



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

$\theta_1, x^{(i)}$

$\underset{\theta_1}{\text{minimize}}\ J(\theta_1)$

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

Simplified

$\theta_0 = 0$

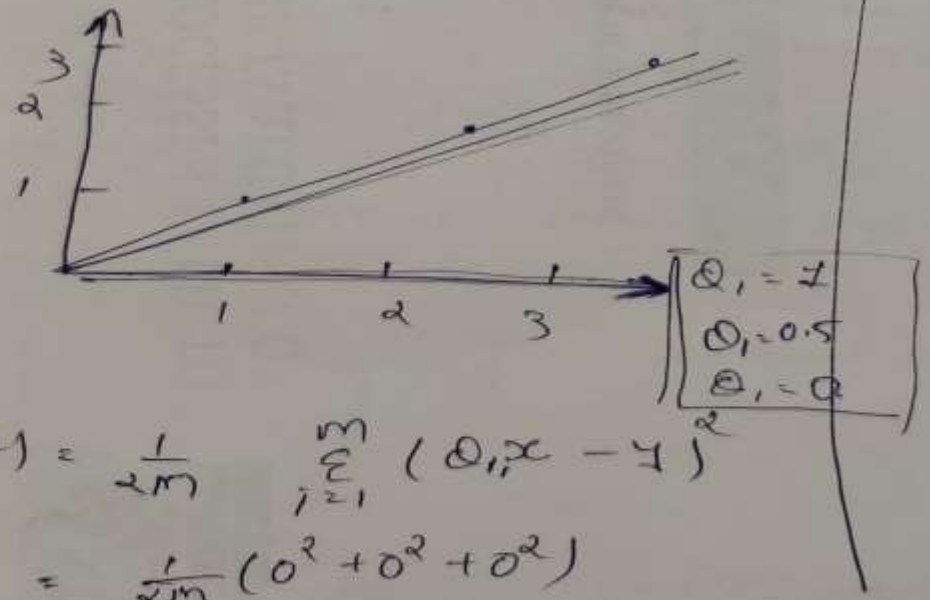
so, consider only θ_1

$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x) - y_i)^2$$

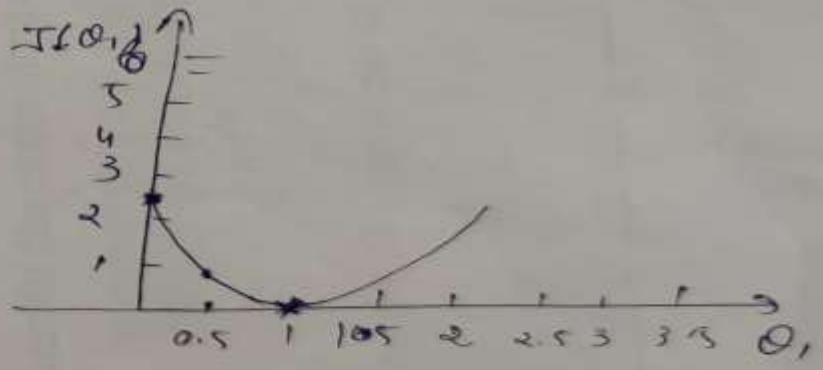
$\underset{\theta_1}{\text{minimize}} \; J(\theta_1)$

$h_\theta(x)$

(For fixed θ_1, this is a function of x)



$J(\theta_1)$



$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (\theta_1 x - y)^2$$

$$= \frac{1}{2m} (0^2 + 0^2 + 0^2)$$

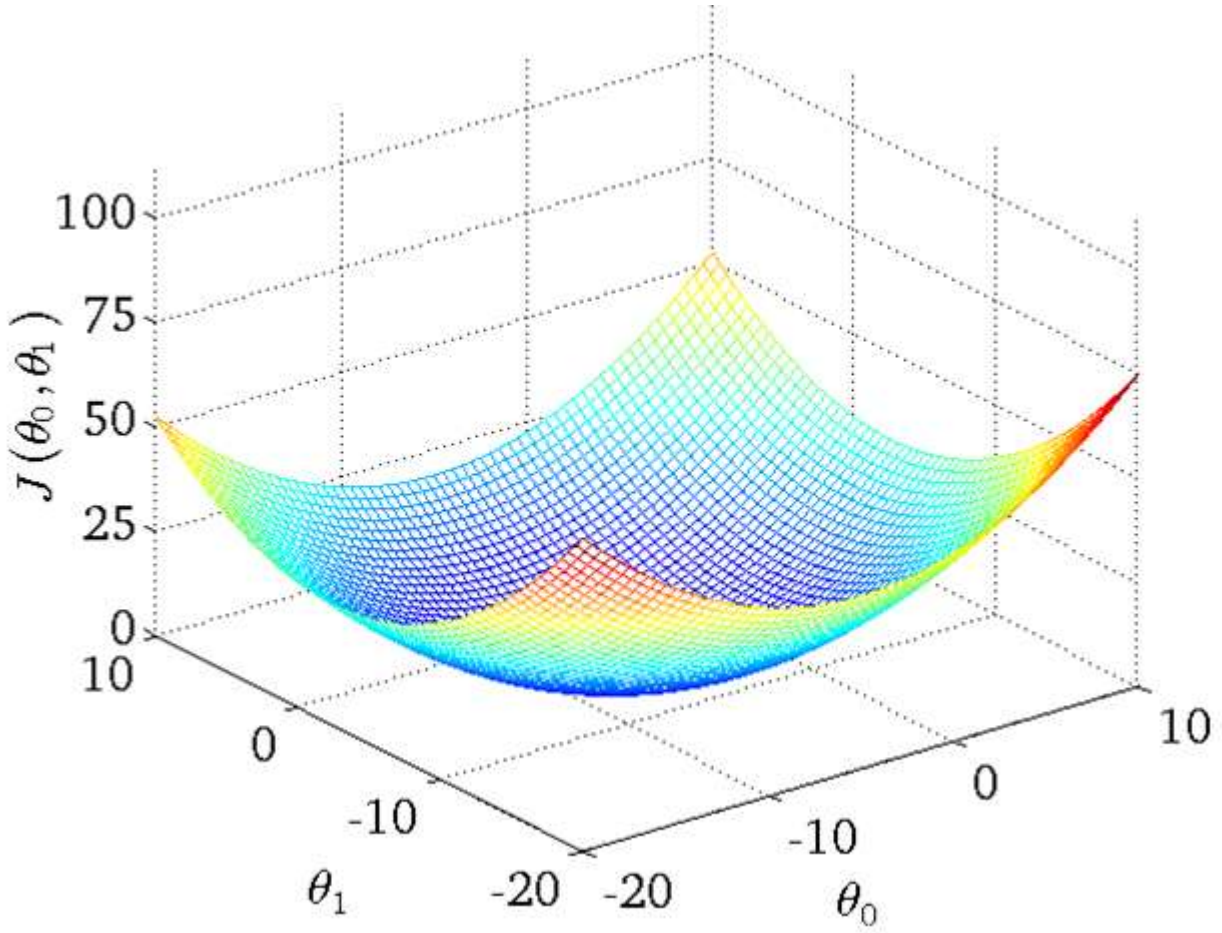
$J(0) = 0$

$J(0.5) = \frac{1}{2(3)} [(0.5-1)^2 + (1-2)^2 + (1.5-3)^2]$

$= 0.68$

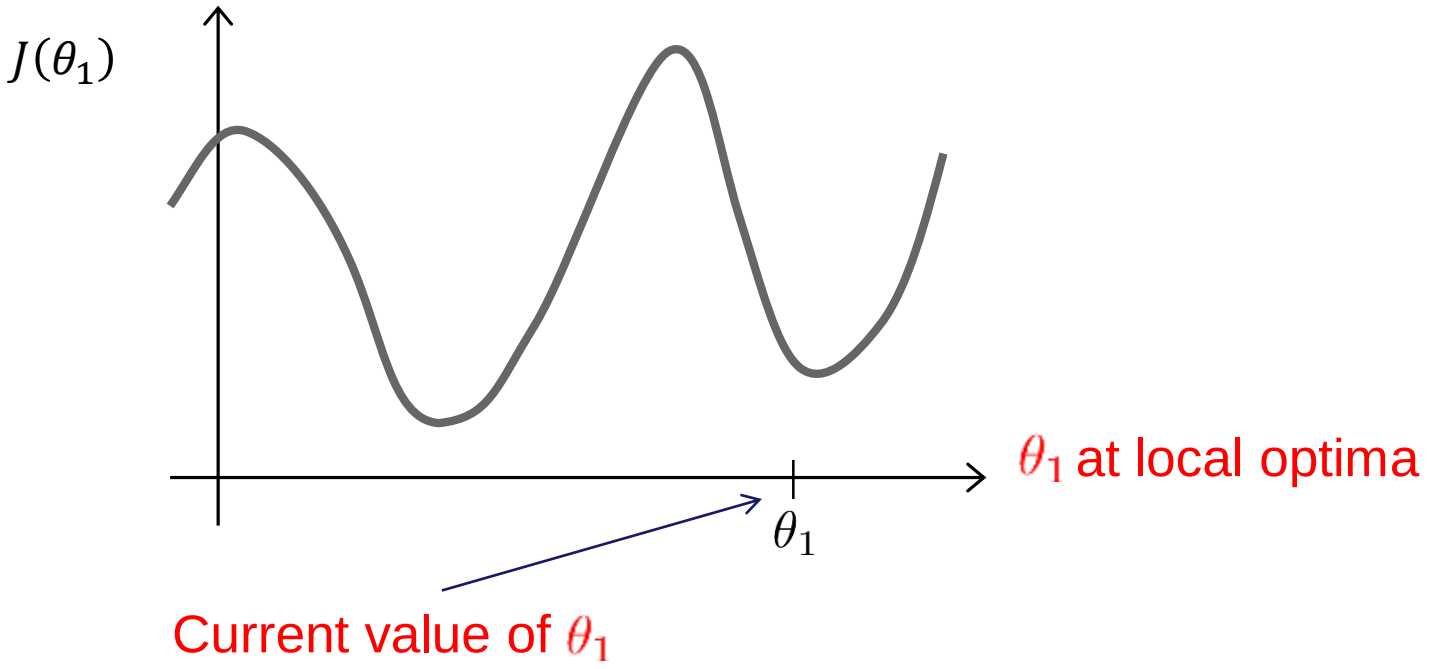
House Price Prediction

- Mathematically:



House Price Prediction

- Mathematically:



House Price Prediction

- Gradient Descent Algorithm

repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \quad (\text{for } j = 0 \text{ and } j = 1)$$

}

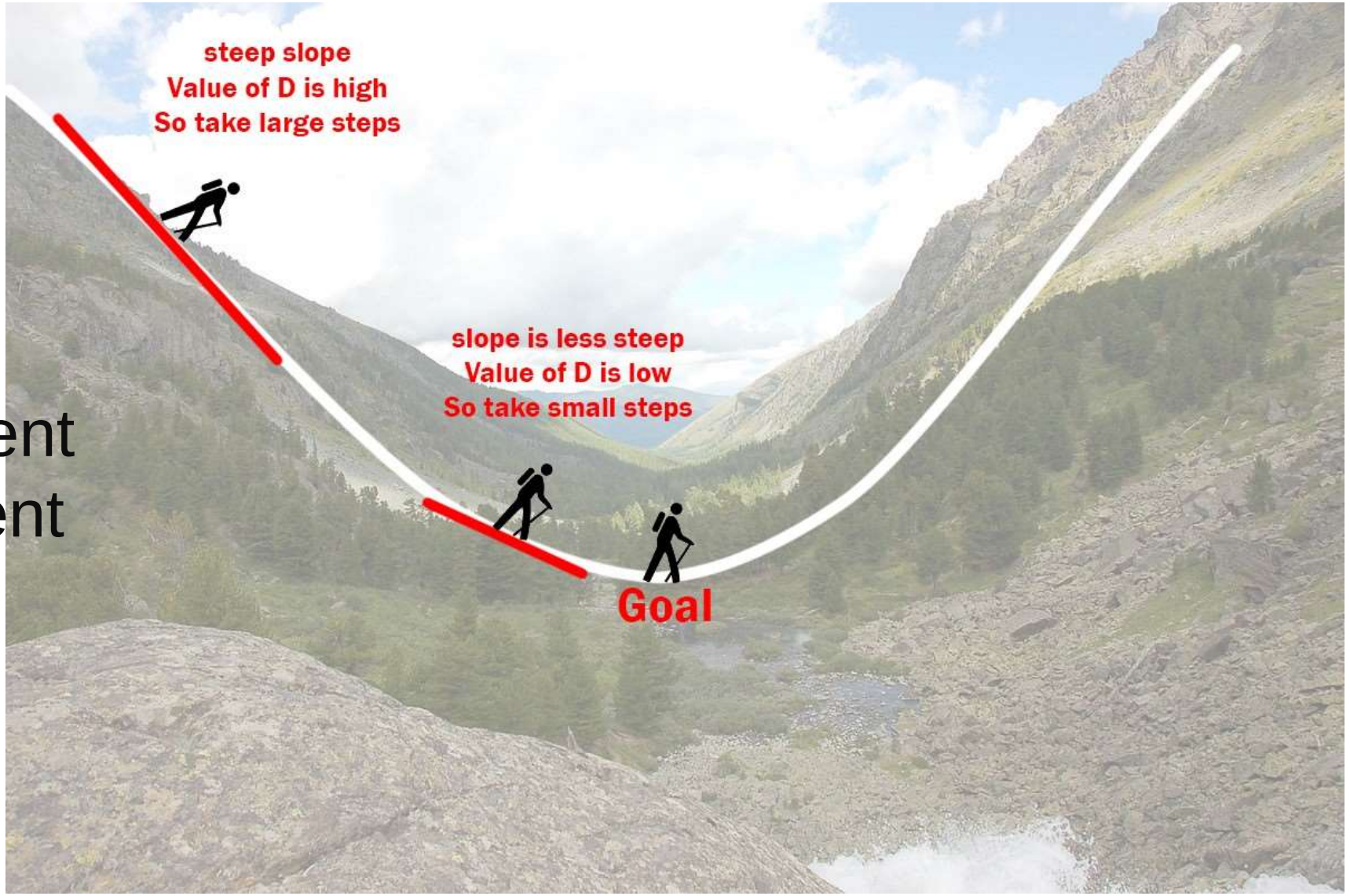
Correct: Simultaneous update

$$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$
$$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$
$$\theta_0 := \text{temp0}$$
$$\theta_1 := \text{temp1}$$

Incorrect:

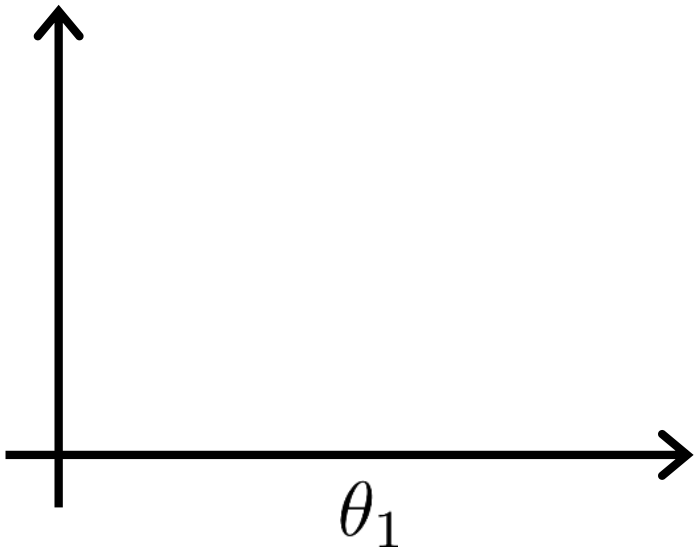
$$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$
$$\theta_0 := \text{temp0}$$
$$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$
$$\theta_1 := \text{temp1}$$

Gradient Descent

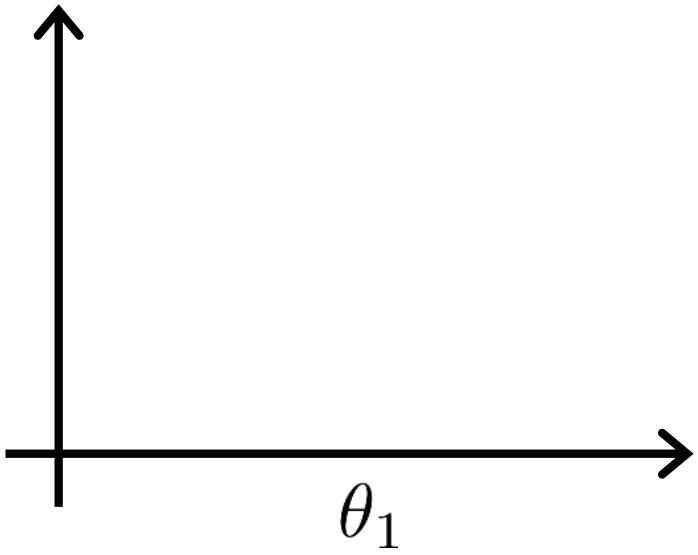


$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

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.



House Price Prediction

- Gradient Descent Algorithm

Gradient descent algorithm

repeat until convergence {

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

(for $j = 1$ and $j = 0$)

}

Linear Regression Model

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

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

House Price Prediction

- Gradient Descent Algorithm

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

$$j = 0 : \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) =$$

$$j = 1 : \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) =$$

House Price Prediction

- Gradient Descent Algorithm

repeat until convergence {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)$$

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

}

update θ_0 and θ_1 simultaneously

Area (SQM)	Selling Price (Lacs)
35	13
37	17
39	15
40	16
43	18
48	19
50	20
55	25
60	35
65	37
70	38
75	40
80	43
85	44
90	50
95	54
100	60

Area (sq. m)	Selling price (Lacs)
35	13
40	16
50	20
60	35
70	38
80	43
90	50
100	60

Initially $\theta_0 = \theta_1 = 0$, $\alpha = 0.1$

∴ Updates in θ_0 & θ_1 are

$$\text{temp}0 = \theta - 0.1 \left(\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)\right) = \theta - \frac{0.1}{m}\left[\sum_{i=1}^{m} (h(x_i) - y_i)\right]$$

$$\text{temp}1 = \theta - 0.1 \left(\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)\right) = \theta - \frac{0.1}{m}\left[\sum_{i=1}^{m} (h(x_i) - y_i) x_i\right]$$

Iteration I

$$\theta_0 = \theta - \frac{0.1}{8}\left[(-13) + (-16) + (-20) + (-35) + (-38) + (-43) + (-50) + (-60)\right]$$

$$= 0 - 0.0125\,[275]$$

Iteration 2: $\theta_0 = -1701.75$

$\theta_1 = -124289$

$$\theta_0 = 3.436$$

$$\theta_1 = 0 - \frac{0.1}{8}\left[-13\times35 - 16\times40 - 20(50) - 35(60) - 38(70) - 43(80) - 50(90) - 60(100)\right]$$

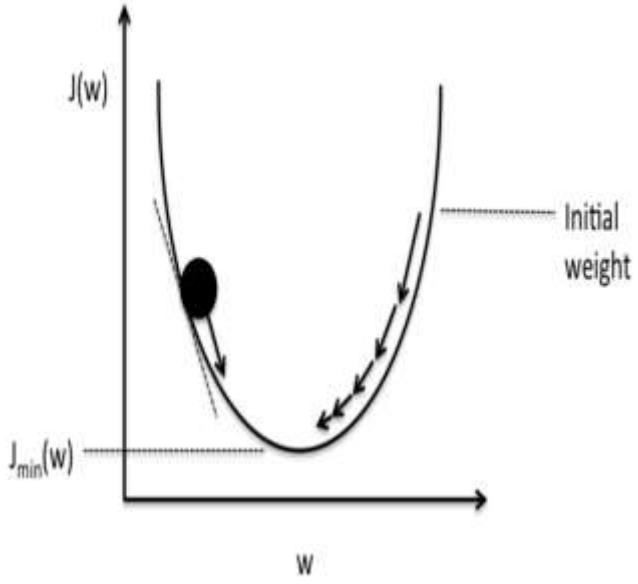
$$= 0 - 0.0125\left[+455 - 640 - 1000 - 2100 - 2660 - 3440 - 4500 - 6000\right]$$

$$= 0 + [0.0125 \times 20795] = \boxed{259.938}$$

In summary - Gradient Decent is

- **Its** a first-order iterative optimization algorithm for finding the minimum of a function OR
  - To find a local minimum of a function

- Finding minimum by taking steps proportional to the negative of the **gradient** (or of the approximate **gradient**) of the function at the current point.



Batch gradient descent refers to calculating the derivative from all training data before calculating an update

- Calculate the error for each training example.
- But only updates the model after all training examples have been evaluated

Mini batch gradient descent

- Divide the training examples into small batches that are used to calculate the model error and accordingly update the model coefficients
- Sum of the gradients is calculate over the mini batch.
- It reduces the variance in the gradients, ie gradual updation

Linear Regression with multiple variables

Multiple features

- Different names
- Number of dependent & independent variables
- Regression and nature of dependent variable
- Nature of the model assuming $y=f(x_1, x_2, \ldots, x_n)$
- House Price Prediction Example

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

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

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:

n = number of features

$x^{(i)}$ = input (features) of i^{th} training example.

$x_j^{(i)}$ = value of feature j in i^{th} training example.

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

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

For convenience of notation, define $x_0 = 1$.

Multivariate linear regression.

Linear Regression with multiple variables

Gradient descent for multiple variables

Hypothesis: $h_\theta(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$

Parameters: $\theta_0, \theta_1, \ldots, \theta_n$

Cost function:

$$J(\theta_0, \theta_1, \ldots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

Gradient descent:

Repeat $\{$

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

$\}$ (simultaneously update for every $j = 0, \ldots, n$)

Gradient Descent

Previously (n=1):

Repeat $\{$

$$\theta_0 := \theta_0 - \alpha \underbrace{\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})}_{\frac{\partial}{\partial\theta_0}J(\theta)}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x^{(i)}$$

(simultaneously update θ_0, θ_1)

$\}$

New algorithm $(n \geq 1)$:

Repeat $\{$

$$\theta_j := \theta_j - \alpha \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$

simultaneously update θ_j for $j = 0, \ldots, n$

$\}$

$$\theta_0 := \theta_0 - \alpha \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_0^{(i)}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_1^{(i)}$$

$$\theta_2 := \theta_2 - \alpha \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_2^{(i)}$$

$\ldots$

Linear Regression with multiple variables

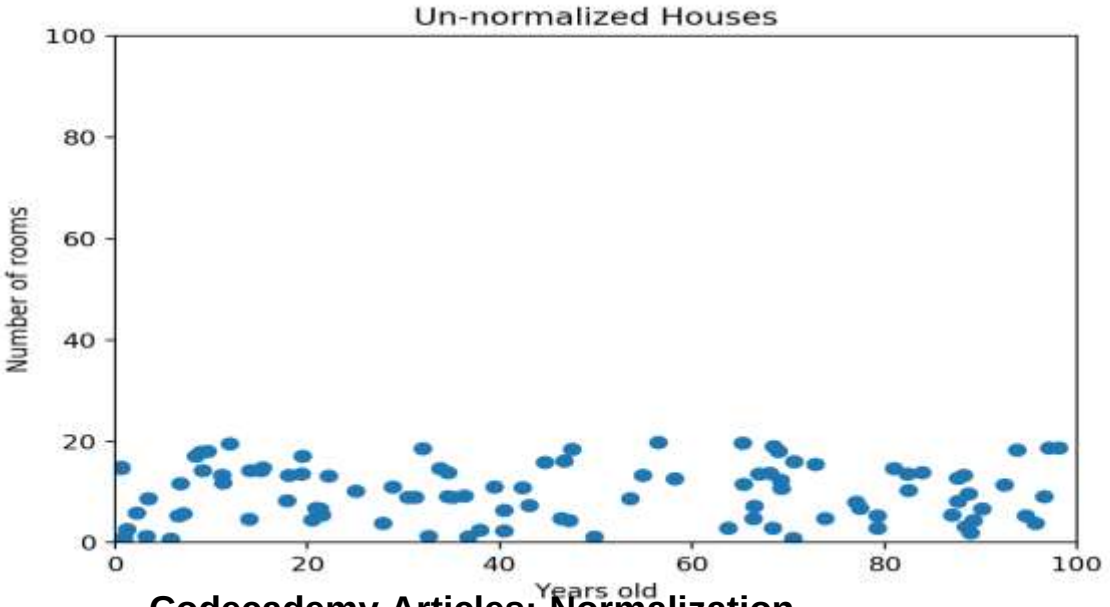
Gradient descent in practice I: Feature Scaling

Need of Feature Scaling?

- Consider a dataset of houses. Two potential features might be (1) the number of rooms in the house, and (2) the total age of the house in years.

- A machine learning algorithm could try to predict which house would be best for you.

- However, when the algorithm compares data points, the feature with the larger scale will completely dominate the other.



Codecademy Articles: Normalization

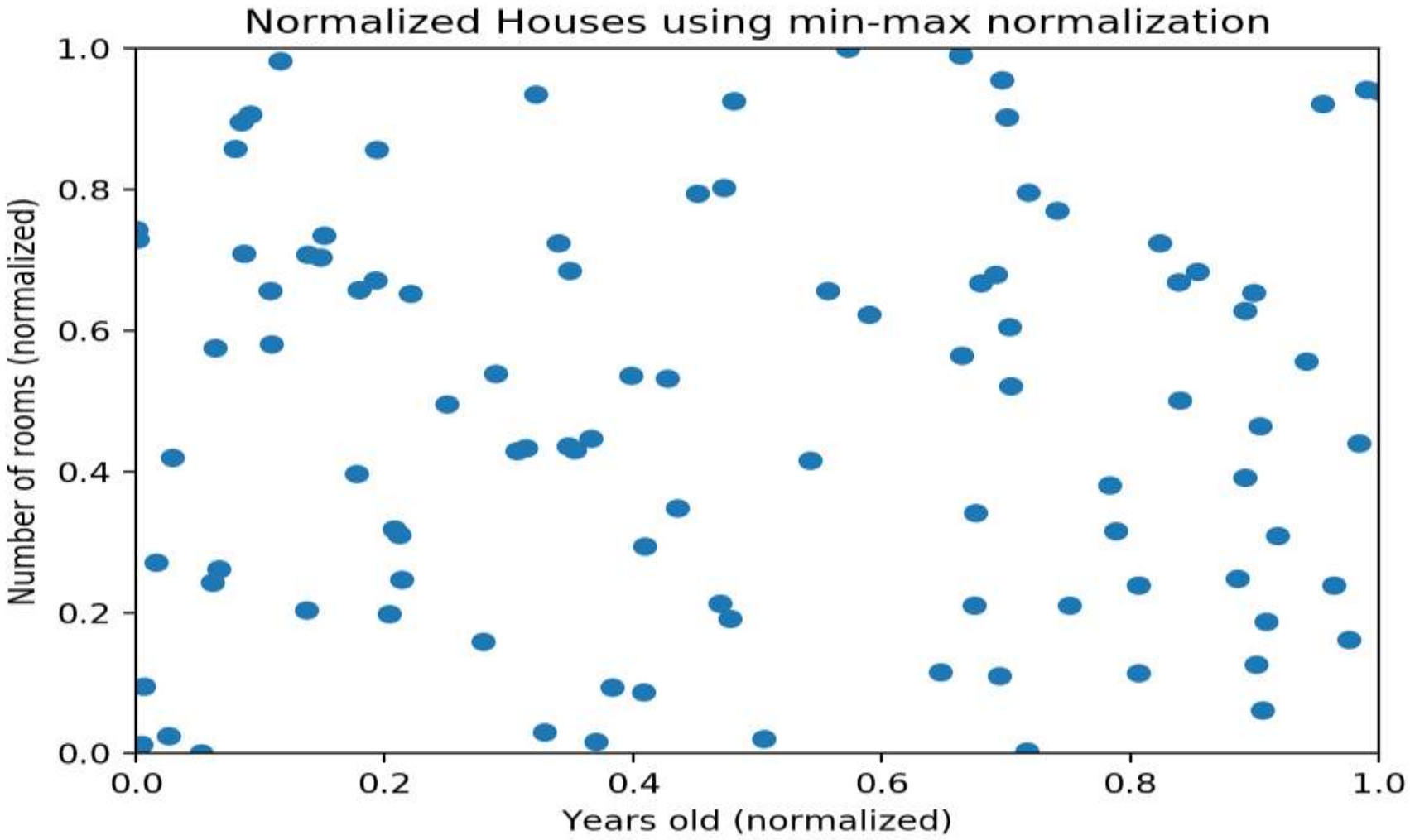
- When the data looks squished like that, we know we have a problem.
- The machine learning algorithm should realize that there is a huge difference between a house with 2 rooms and a house with 20 rooms. But right now, because two houses can be 100 years apart, the difference in the number of rooms contributes less to the overall difference.

Min-max normalization performs a linear transformation on the original data. Suppose that min_A and max_A are the minimum and maximum values of an attribute, A. Min-max normalization maps a value, v, of A to v' in the range $[new\_min_A, new\_max_A]$ by computing

$$v' = \frac{v - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A. \tag{2.11}$$

Min-max normalization preserves the relationships among the original data values. It will encounter an "out-of-bounds" error if a future input case for normalization falls outside of the original data range for A.

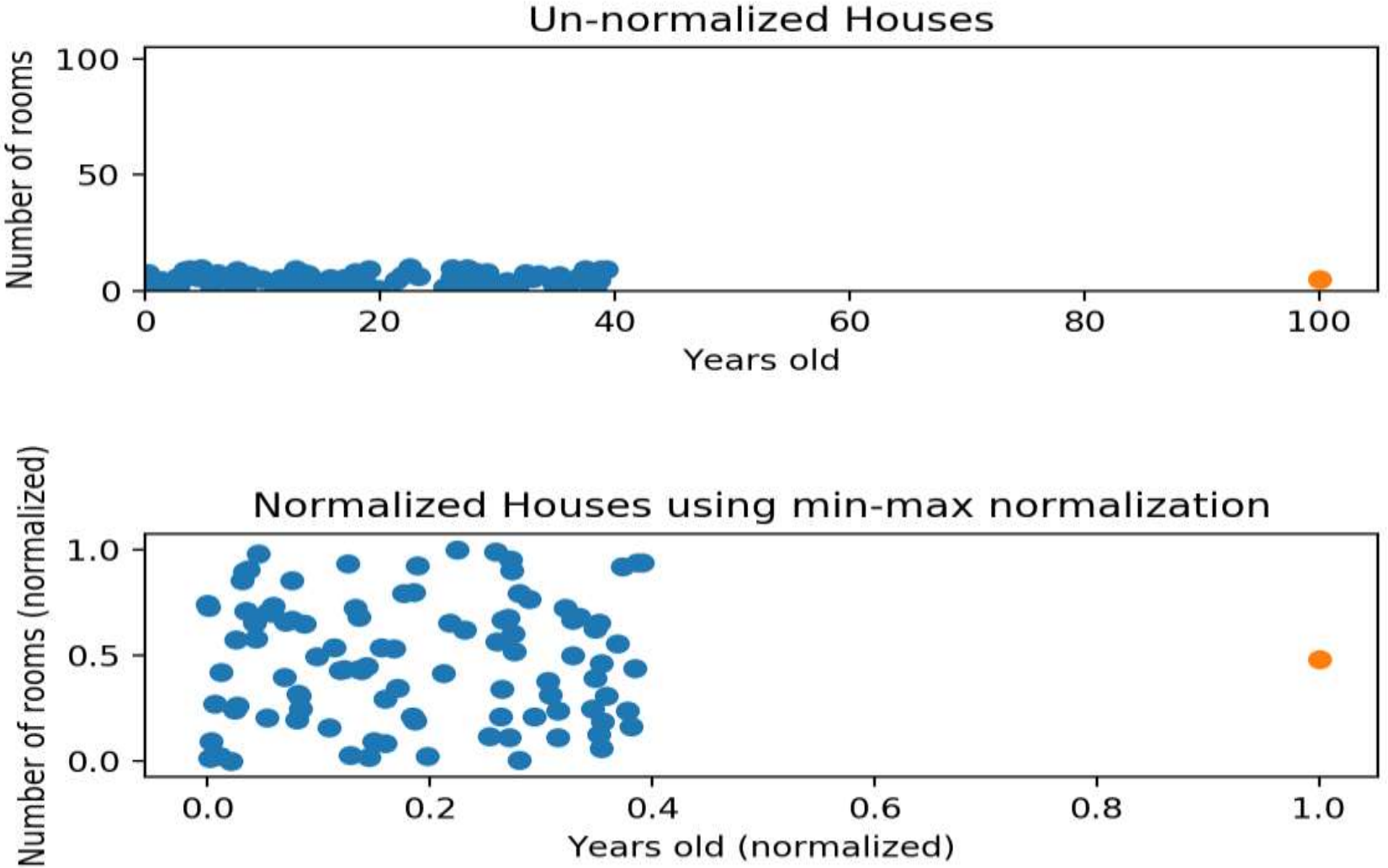
Min-max normalization. Suppose that the minimum and maximum values for the attribute *income* are \$12,000 and \$98,000, respectively. We would like to map *income* to the range [0.0, 1.0]. By min-max normalization, a value of \$73,600 for *income* is transformed to $\frac{73{,}600-12{,}000}{98{,}000-12{,}000}(1.0-0)+0 = 0.716$. ■



Codecademy Articles: Normalization

Guarantees all features will have the exact same scale but does not handle outliers well.

Outlier: an **outlier** is a data point that differs significantly from other observations.



Codecademy Articles: Normalization

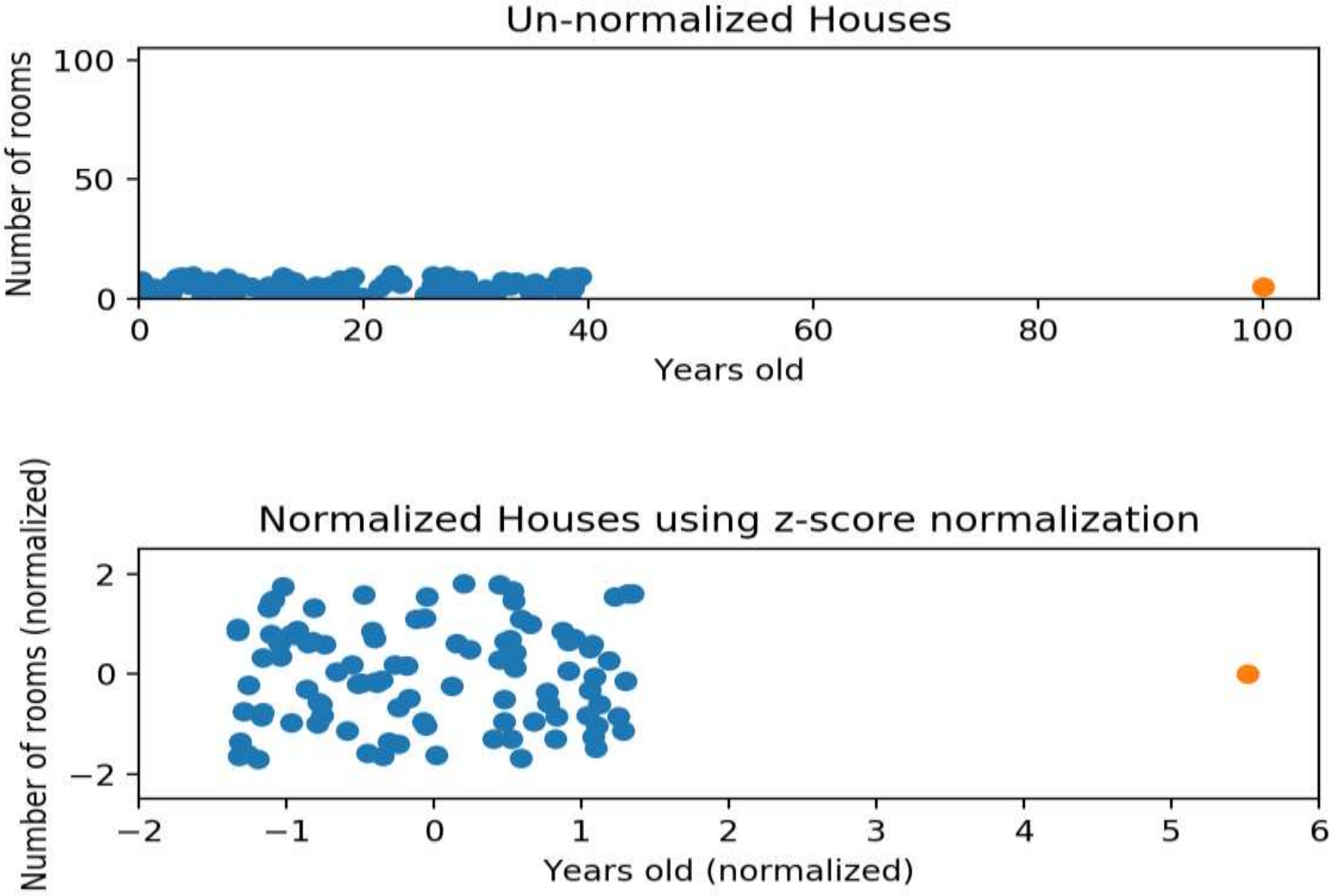
In **z-score normalization** (or *zero-mean normalization*), the values for an attribute, A, are normalized based on the mean and standard deviation of A. A value, v, of A is normalized to v' by computing

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2 \qquad v' = \frac{v - \bar{A}}{\sigma_A}, \tag{2.12}$$

where $\bar{A}$ and σ_A are the mean and standard deviation, respectively, of attribute A. This method of normalization is useful when the actual minimum and maximum of attribute A are unknown, or when there are outliers that dominate the min-max normalization.

z-score normalization Suppose that the mean and standard deviation of the values for the attribute *income* are \$54,000 and \$16,000, respectively. With z-score normalization, a value of \$73,600 for *income* is transformed to $\frac{73{,}600-54{,}000}{16{,}000} = 1.225$. ■

Handles outliers, but does not produce normalized data with the *exact* same scale.



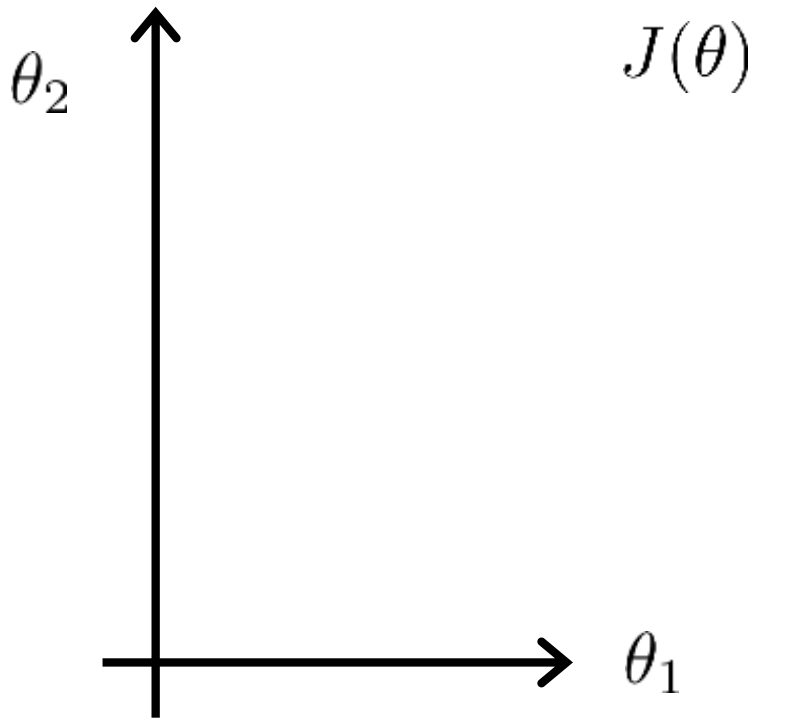
Codecademy Articles: Normalization

Feature Scaling

Idea: Make sure features are on a similar scale.

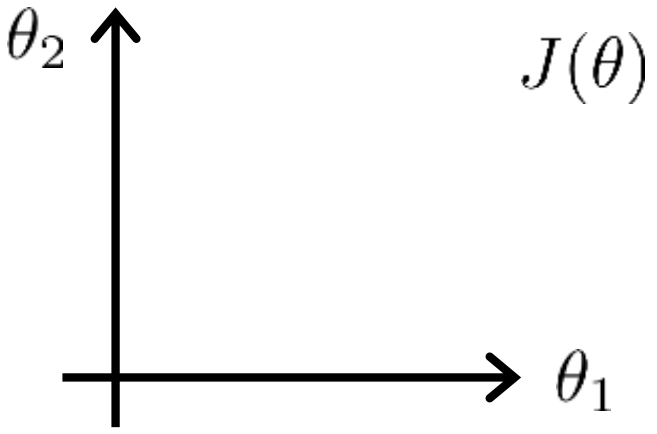
E.g. x_1 = size (0-2000 feet^2)

x_2 = number of bedrooms (1-5)



$$x_1 = \frac{\text{size (feet}^2\text{)}}{2000}$$

$$x_2 = \frac{\text{number of bedrooms}}{5}$$



Feature Scaling

Get every feature into approximately a$-1 \leq x_i \leq 1$ range.

Linear Regression with multiple variables

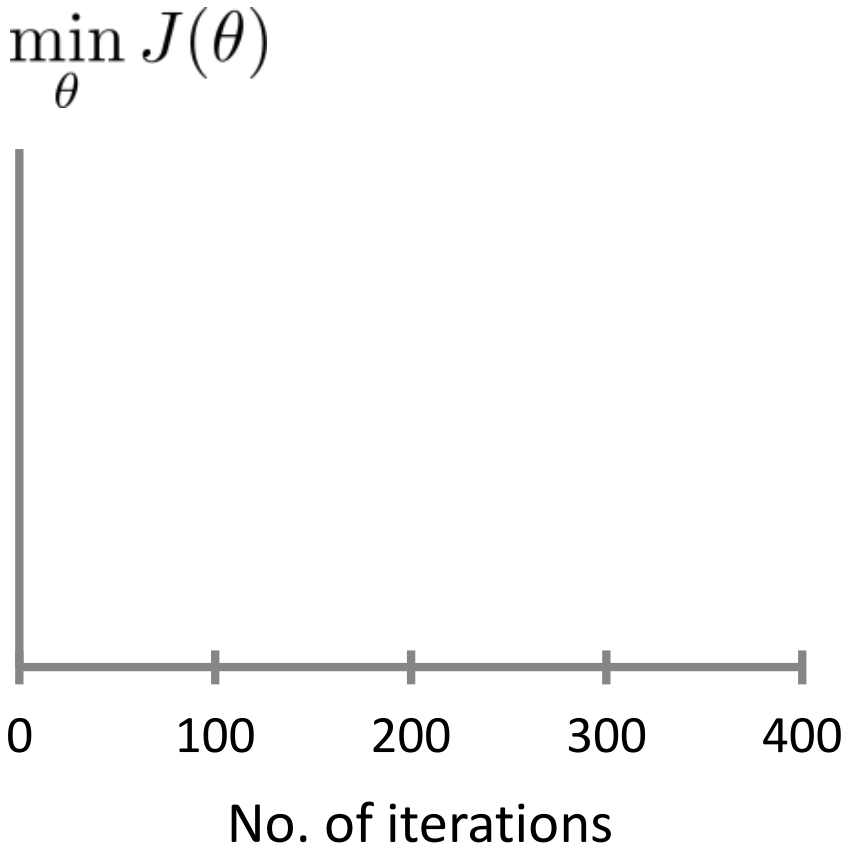
Gradient descent in practice II: Learning rate

Gradient descent

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

- "Debugging": How to make sure gradient descent is working correctly.
- How to choose learning rate α.

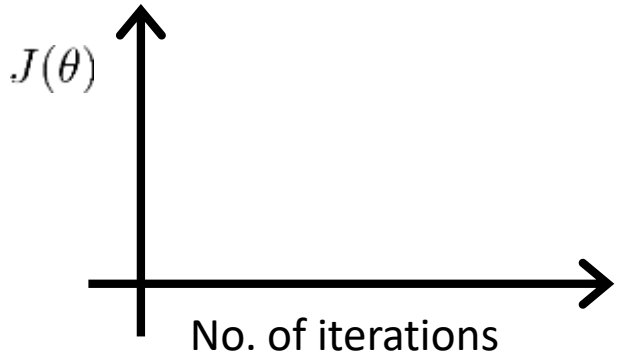
Making sure gradient descent is working correctly.



Example automatic convergence test:

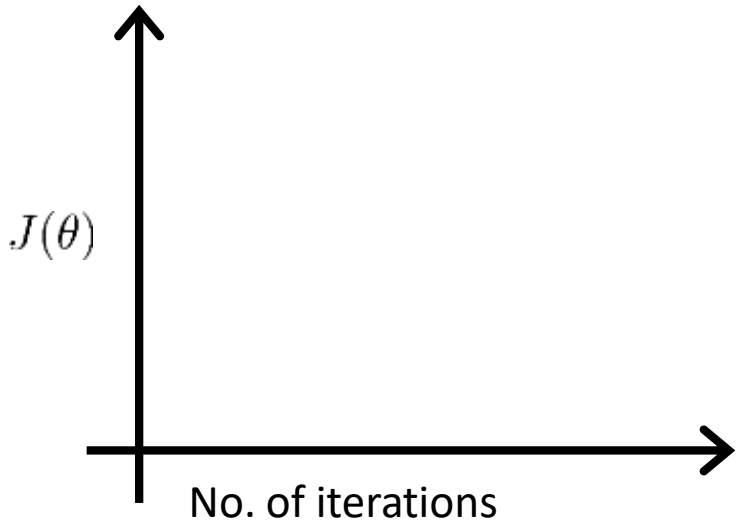
Declare convergence if $J(\theta)$ decreases by less than 10^{-3} in one iteration.

Making sure gradient descent is working correctly.



Gradient descent not working.

Use smaller α.



- For sufficiently small α, $J(\theta)$ should decrease on every iteration.
- But if α is too small, gradient descent can be slow to converge.

Summary:

- If α is too small: slow convergence.
- If α is too large: $J(\theta)$ may not decrease on every iteration; may not converge.

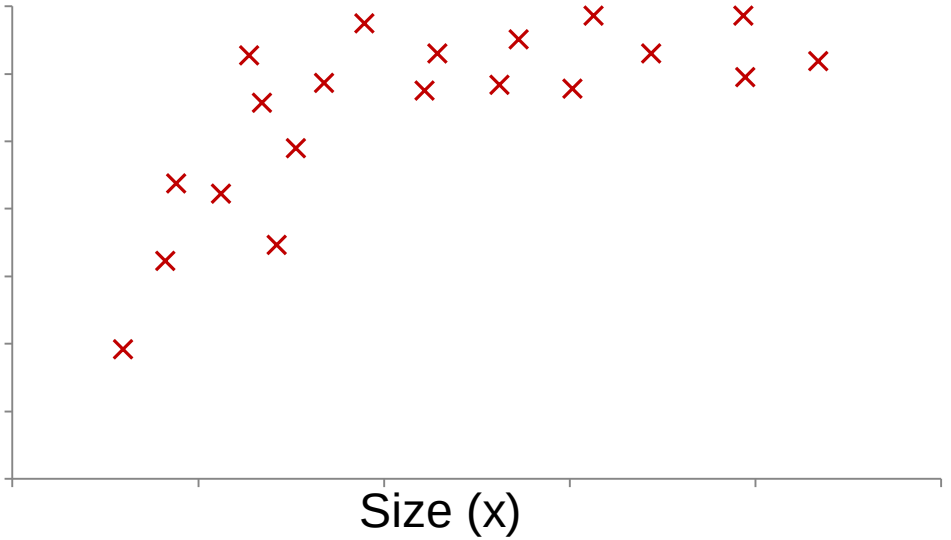
To choose α, try

$$\ldots, 0.001, \qquad , 0.01, \qquad , 0.1, \qquad , 1, \ldots$$

Linear Regression with multiple variables

Features and polynomial regression

Polynomial regression



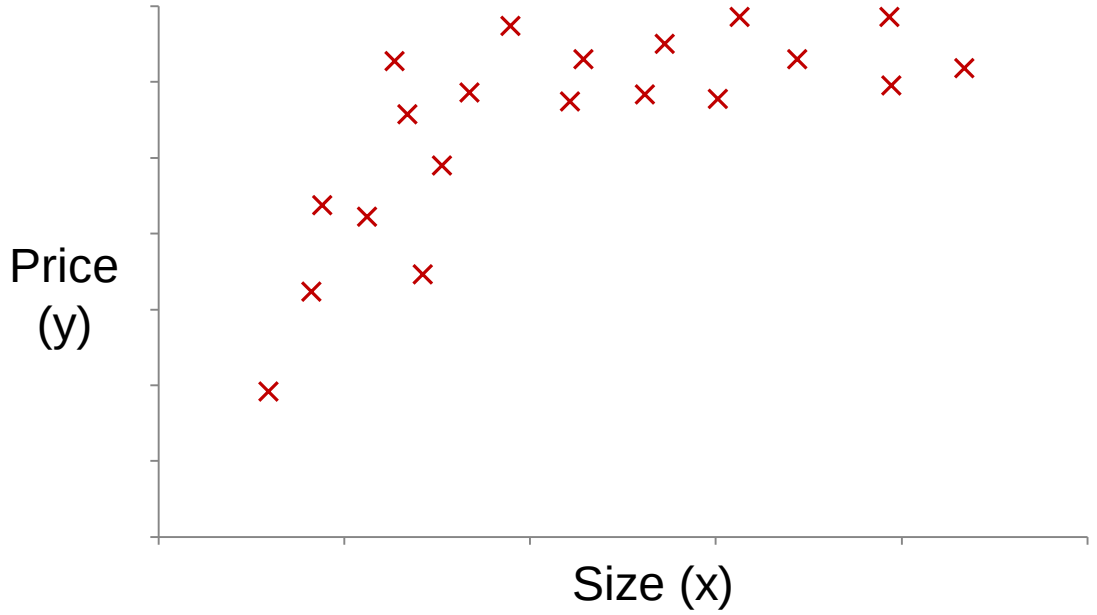
$$\theta_0 + \theta_1 x + \theta_2 x^2$$

$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$$

$$\begin{aligned} h_\theta(x) &= \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 \\ &= \theta_0 + \theta_1 (size) + \theta_2 (size)^2 + \theta_3 (size)^3 \end{aligned}$$

$$\begin{aligned} x_1 &= (size) \\ x_2 &= (size)^2 \\ x_3 &= (size)^3 \end{aligned}$$

Choice of features



$$h_\theta(x) = \theta_0 + \theta_1(size) + \theta_2(size)^2$$

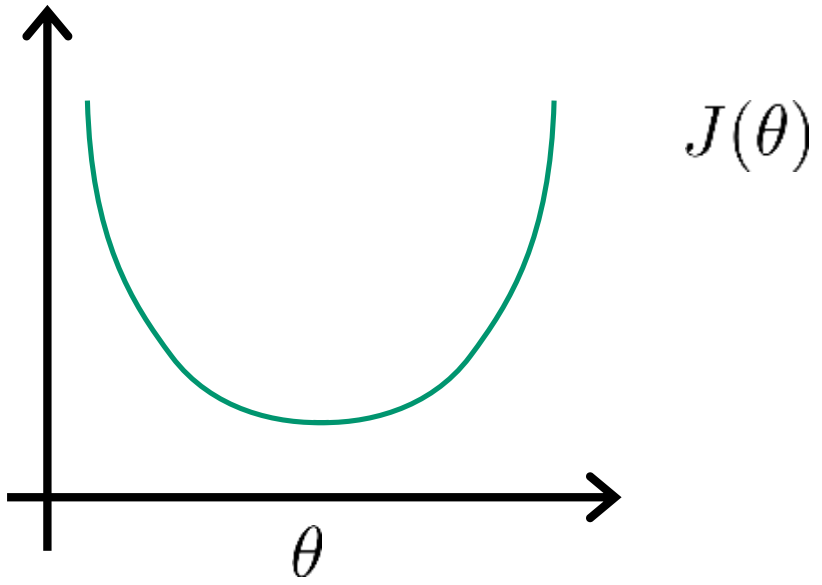
$$h_\theta(x) = \theta_0 + \theta_1(size) + \theta_2\sqrt{(size)}$$

Linear Regression with multiple variables

Normal equation

1) Alpha
2) Derivatives
3) Simultaneous Update

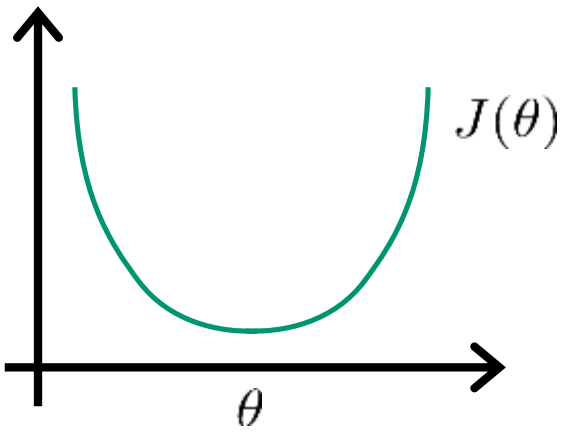
Gradient Descent



Normal equation: Method to solve for θ analytically. Normal equations help to find the optimum solution in one go with help of matrix multiplication

Intuition: If 1D($\theta \in \mathbb{R}$)

$$J(\theta) = a\theta^2 + b\theta + c$$



$\theta \in \mathbb{R}^{n+1}$ $\quad J(\theta_0, \theta_1, \ldots, \theta_m) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$

$\frac{\partial}{\partial \theta_j} J(\theta) = \cdots = 0$ (for every j)

Solve for $\theta_0, \theta_1, \ldots, \theta_n$

Examples: $m = 4$.

x_0	Size (feet²) x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price ($1000) 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

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

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

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

Show the derivation for this equation on Board.

$(X^T X)^{-1}$ is inverse of matrix $X^T X$

.

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

m training examples, n features.

Gradient Descent

- Need to choose 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 $(X^T X)^{-1}$
- Slow if n is very large.

Linear Regression with multiple variables

Normal equation and non-invertibility (optional)

Normal equation

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

- What if $X^T X$ is non-invertible? (singular/ degenerate)
- Octave: `pinv(X'*X)*X'*y`

What if $X^T X$ is non-invertible?

- Redundant features (linearly dependent).
  E.g. x_1 = size in feet^2
  x_2 = size in m^2

- Too many features (e.g. $m \leq n$).
  - Delete some features, or use regularization.

Error Measures

Absolute error or squared error?

$$\text{Absolute error}: \quad |y_i - y_i'|$$

$$\text{Squared error}: \quad (y_i - y_i')^2$$

$$\text{Mean absolute error}: \quad \frac{\sum_{i=1}^{d} |y_i - y_i'|}{d}$$

$$Mean\ Absolute\ Percentage\ Error: \frac{1}{d}\sum_{i=1}^{d}\left|\frac{y_i - y_i'}{y_i}\right|$$

$$Root\ Mean\ Square\ Error: \sqrt[2]{\sum_{i=1}^{d}\frac{(y_i - y_i')^2}{d}}$$

$$NRMSE = \frac{RMSE}{y_{max} - y_{min}}$$

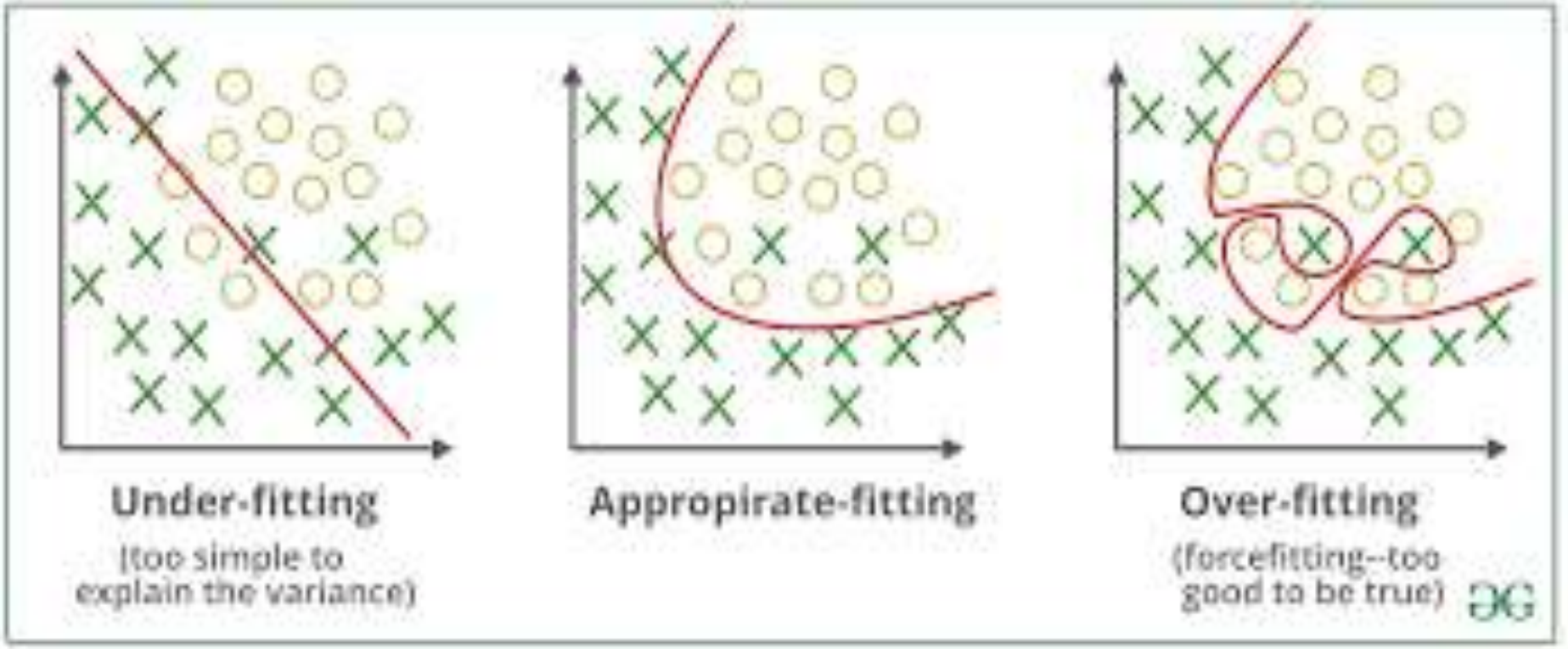
https://julienbeaulieu.gitbook.io/wiki/sciences/machine-learning/linear-regression/error-functions-1

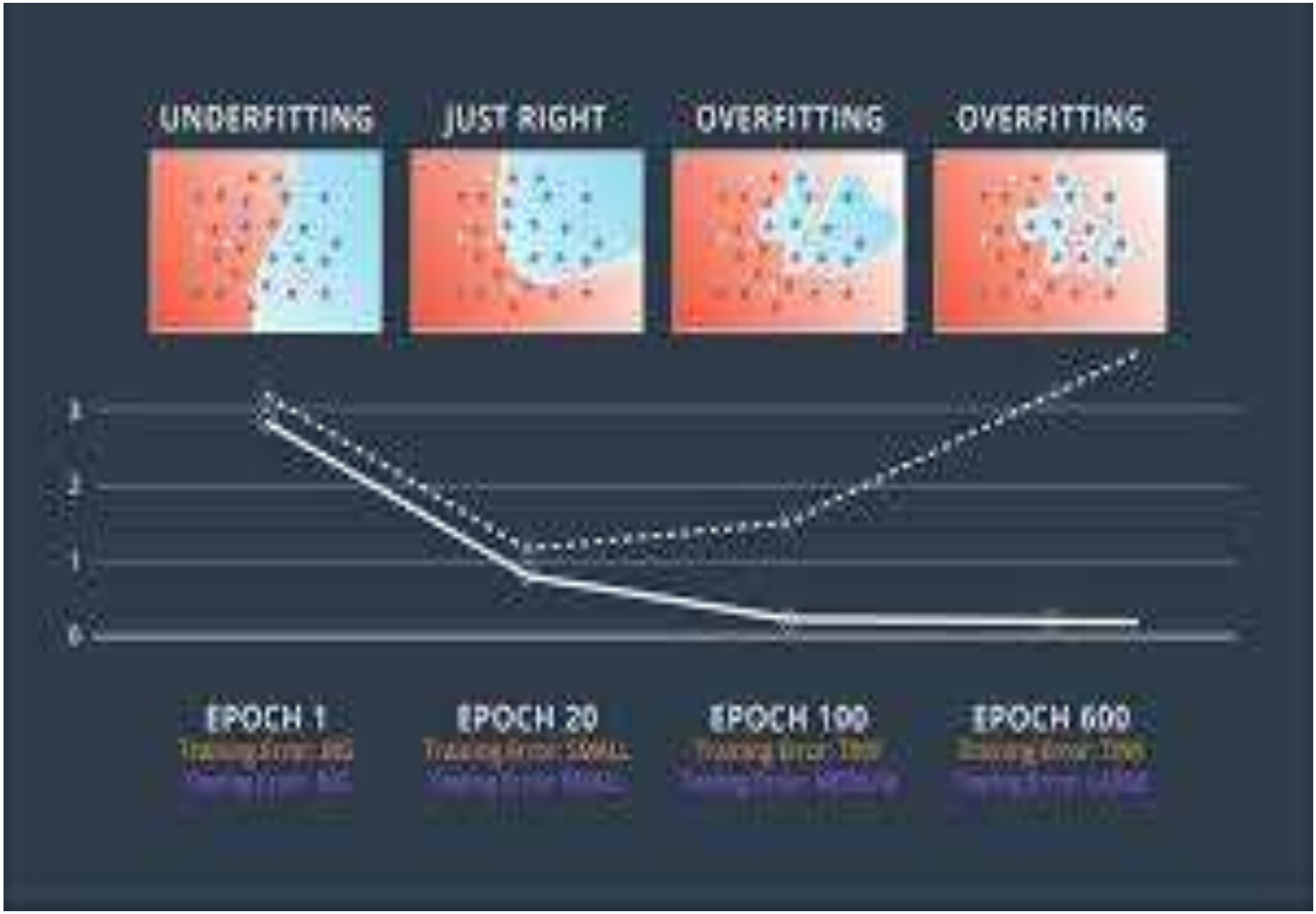
Evaluating Predictor

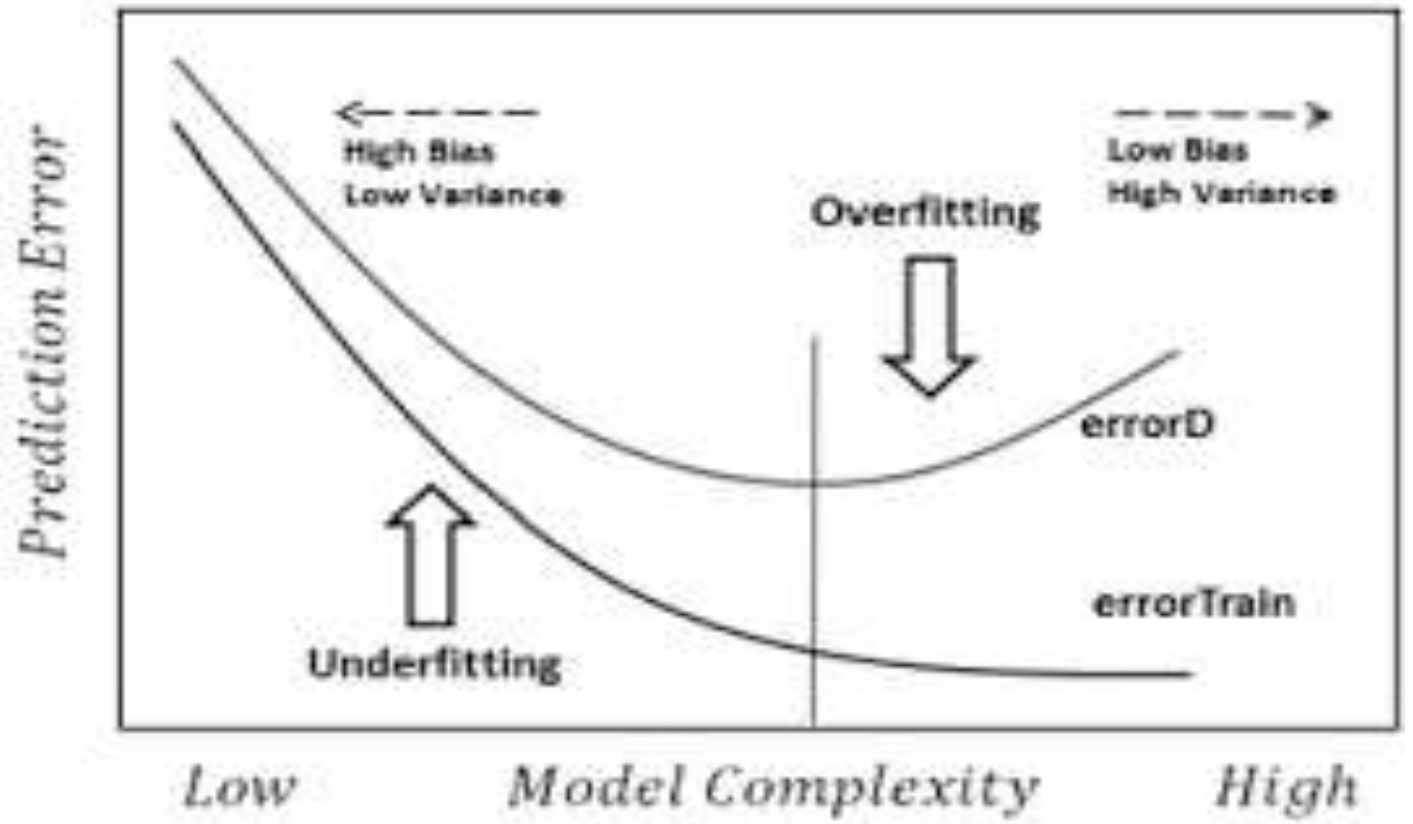
Training + Test Set (Holdout Set)
Cross Validation
Training + Validation + Test Set
Training + Test Set and Cross Validation of Training Set for Model Selection

Underfitting and Overfitting

Overfitting and underfitting




UNDERFITTING
JUST RIGHT
OVERFITTING
OVERFITTING
EPOCH 1
EPOCH 20
EPOCH 100
EPOCH 600


Prediction Error
High Bias
Low Variance
Low Bias
High Variance
Overfitting
errorD
errorTrain
Underfitting
Low
Model Complexity
High

Model - Bias and Variance

- Bias: expected difference between model's prediction and truth
  - High Bias: Model makes inaccurate predictions on training data and has strong assumptions on the data
  - Low Bias: Model makes accurate predictions on training data

- Variance: how much the model differs among sample in the training set OR how much the model is dependent on the training data
  - High Variance: Model does not generalize to new datasets
  - Low Variance: Model generalizes to new datasets. ie Prediction results do not depend on the training data used

- An example of a high bias and low variance model is a model that always predicts cat
  - It's bias is high because it frequently miss classifies the image, but its variance is low because its prediction results do not depend on the training data used

- An example of a low bias and high variance model is a model that predicts cat only if the image matches pixel-to-pixel with a cat in the training data
  - It's bias is low in the training data because it has 100% correct labels, but its variance is high because it's predictions will differ depending on what images are in the training data

Underfitting

- Model passes straight through the training set with no regard for the data!
- This is because an **underfit model has low variance and high bias.**

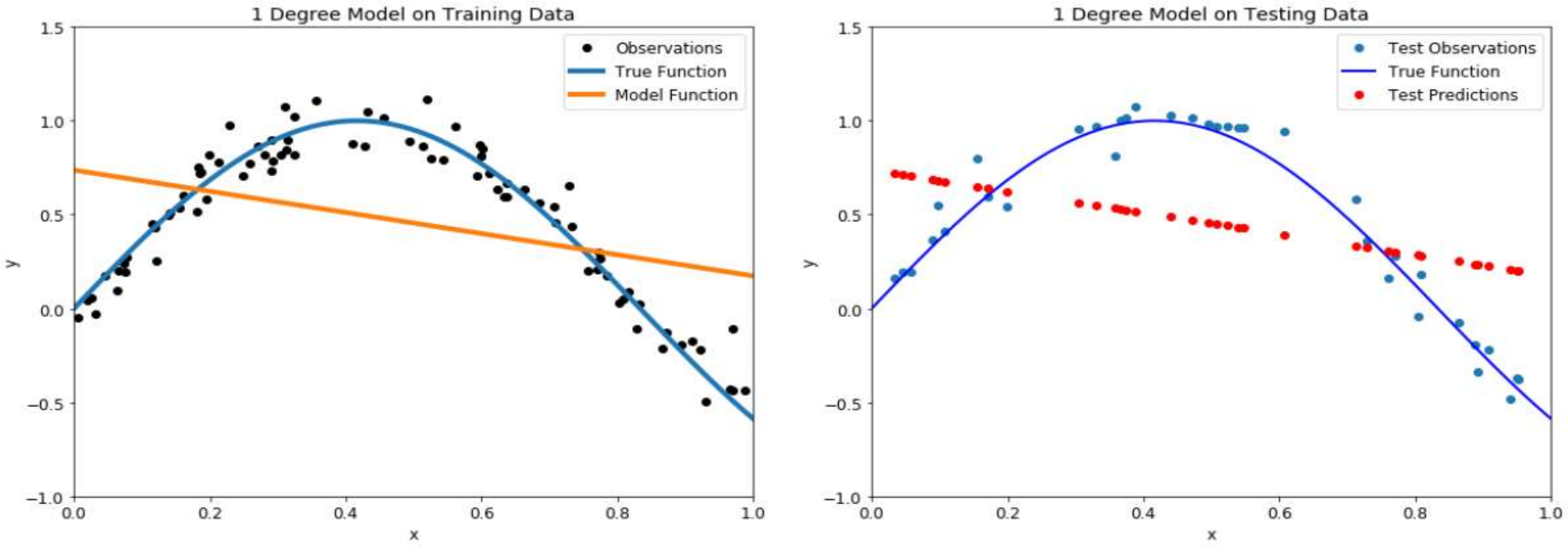
- For the case of a 1 degree polynomial, the model depends very little on the training data because it barely pays any attention to the points!

- Instead, the model has high bias, which means it makes a strong assumption about the data.

- For this example, the assumption is that the data is linear, which is evidently quite wrong.

- When the model makes test predictions, the bias leads it to make inaccurate estimates. The model failed to learn the relationship between x and y because of this bias, a clear example of underfitting.

Underfitting : both training and validation loss is high



Underfit 1 degree polynomial model on training (left) and testing (right) datasets

Techniques to reduce underfitting :

1. Increase model complexity
2. Increase number of features, performing feature engineering
3. Remove noise from the data.
4. Increase the number of epochs or increase the duration of training to get better results.

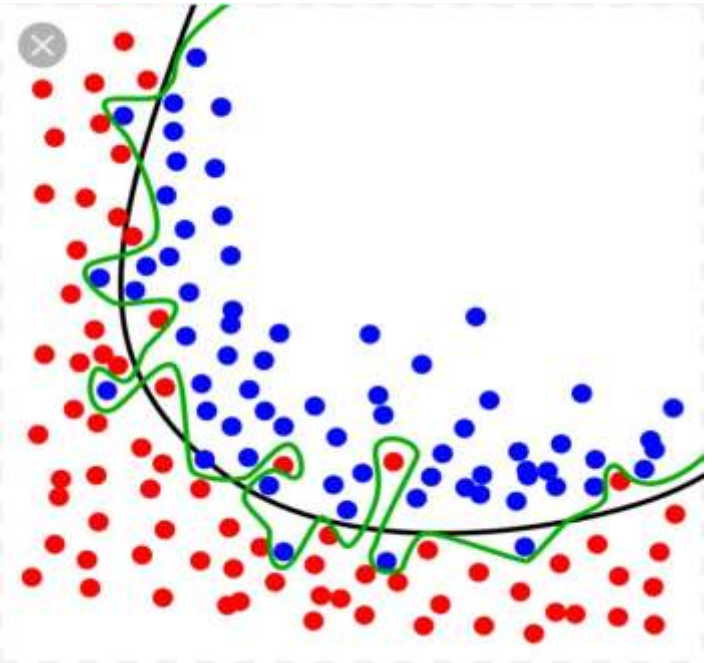
Overfitting

–Overfitting happens when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new data

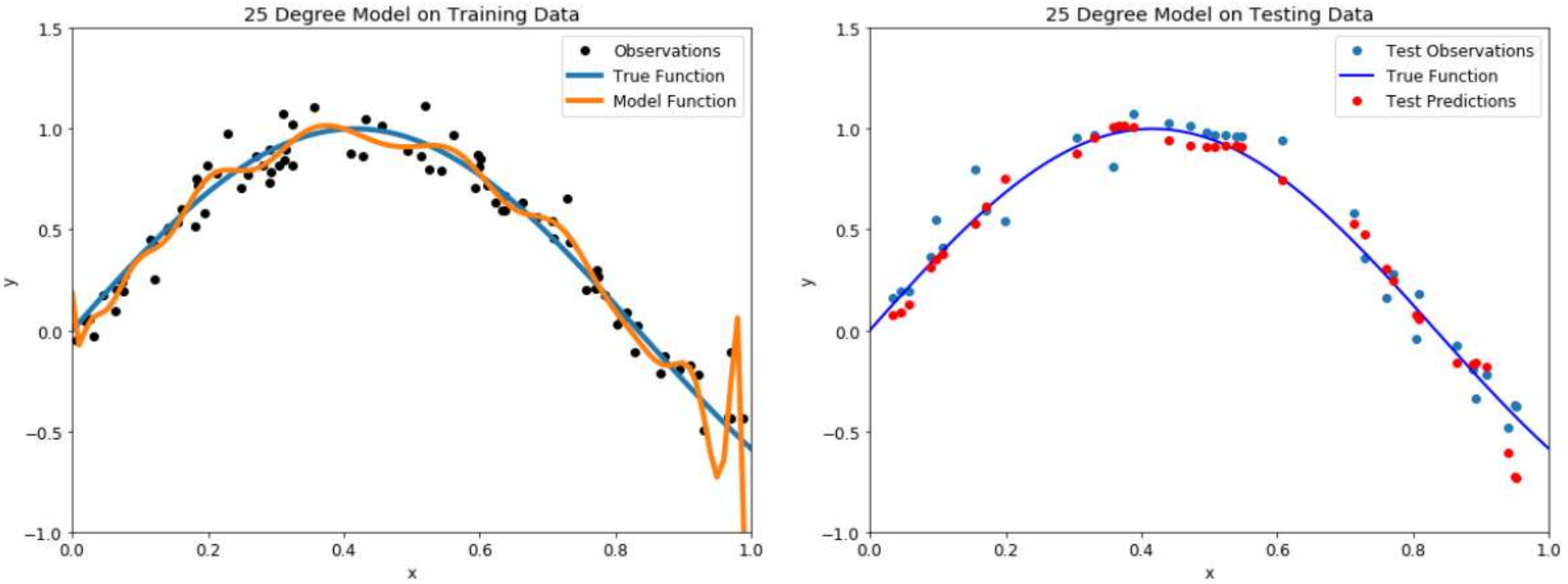
–Model (network) memorizes rather than generalizes

–Training model with higher variance and less bias

–Overly complex model with too many parameters



Overfitting



Overfit 25 degree polynomial model on training (left) and testing (right) datasets

Avoiding overfitting

–Increase the training data : Data augmentation (adding more data) through horizontal flipping, zoom, color filter, rotate.

–Use a resampling technique to estimate model accuracy.

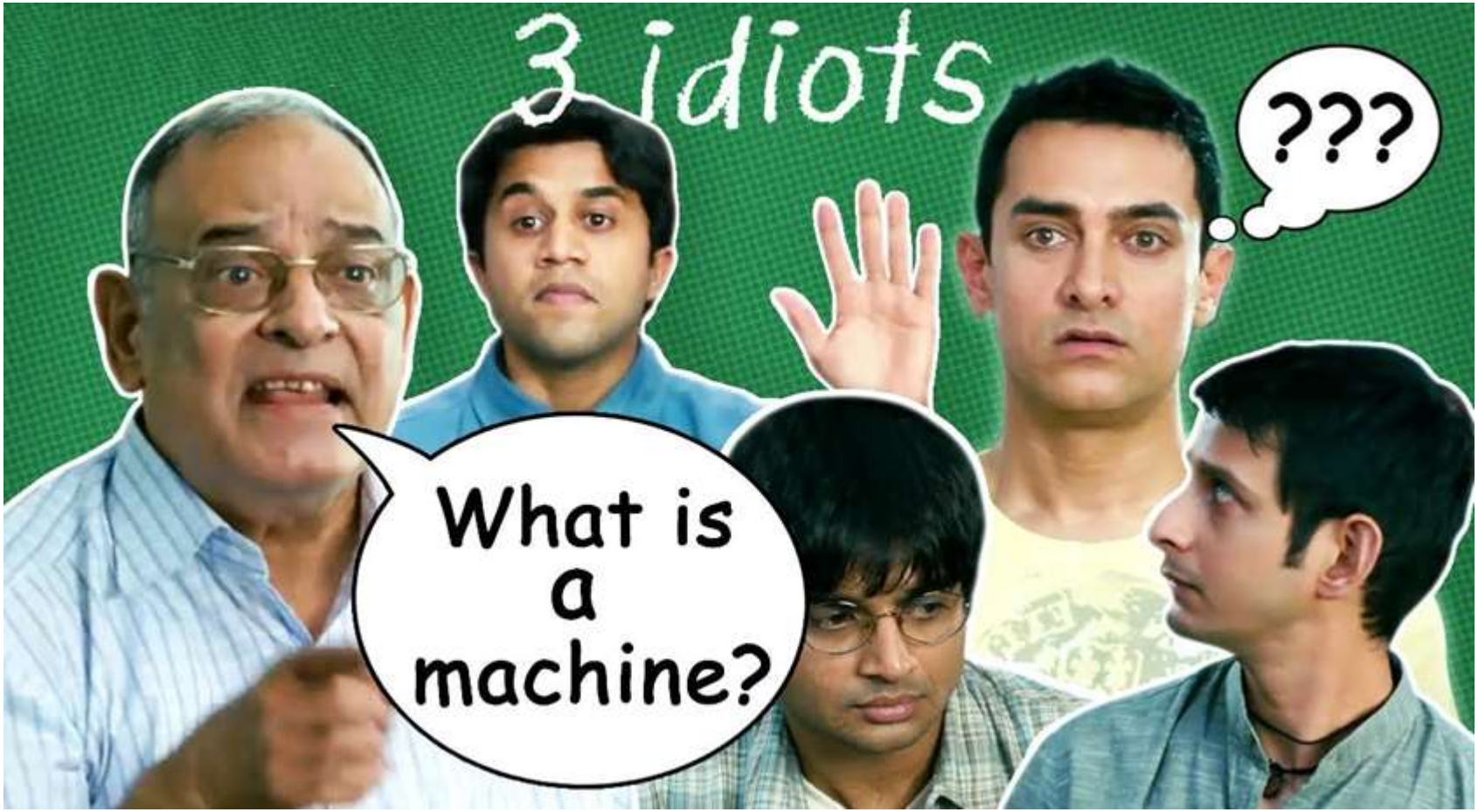
- **K fold cross validation:**
  - Validation is a part of the model training phase.
  - It allows you to train and test your model k-times on different subsets of training data and build up an estimate of the performance of a machine learning model on unseen data.

–Pruning (in case of decision trees)

Avoid Overfitting

- Reducing number of features
  - Manually select which features to keep.
  - Model selection algorithm (later in course).
- Add regularization
  - Keep all the features, but reduce magnitude/values of parameters θ_j.
  - Works well when we have a lot of features, each of which contributes a bit to predicting y.
- So, use a model that generalizes well.

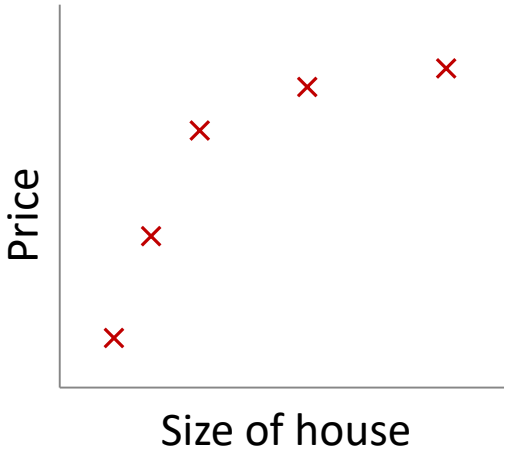
Regularization



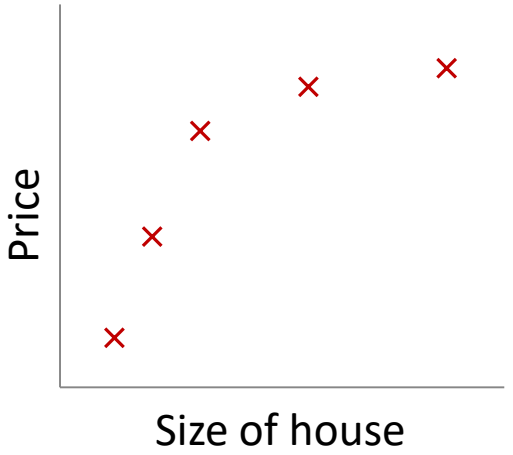
A theoretical justification for regularization is that it attempts to impose Occam's razor on the solution.

Occam's razor paraphrased by the statement, "the simplest explanation is most likely the right one."

Intuition



$\theta_0 + \theta_1 x + \theta_2 x^2$



$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

Suppose we penalize and make θ_3, θ_4 really small.

$$\min_{\theta} \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

Regularization

Small values for parameters $\theta_0, \theta_1, \ldots, \theta_n$

- "Simpler" hypothesis
- Less prone to overfitting

Housing:

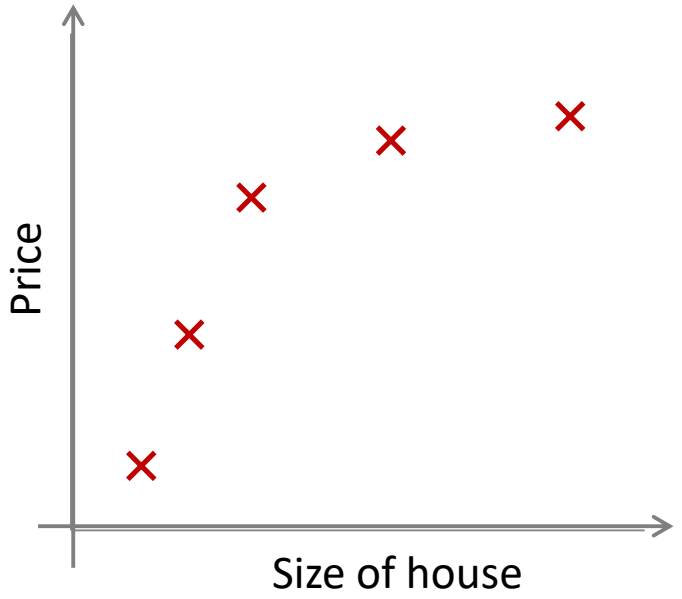
- Features: $x_1, x_2, \ldots, x_{100}$
- Parameters: $\theta_0, \theta_1, \theta_2, \ldots, \theta_{100}$

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

Regularization

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} \theta_j^2\right]$$

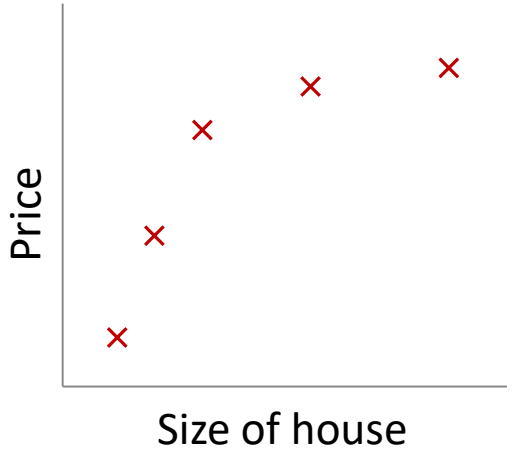
$$\min_\theta J(\theta)$$



In regularized linear regression, we choose θ to minimize

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right]$$

What if λ is set to an extremely large value (perhaps for too large for our problem, say $\lambda = 10^{10}$)?



$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

Regularized linear regression

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right]$$

$$\min_\theta J(\theta)$$

λ, the regularization parameter controls the trade off between two goals of the cost function. (goal 1: reduce the error (fit to the training set well, goal 2: keep the Θ small to make the model simple. The hyper parameter λ controls this tradeoff by adjusting the weight of the penalty term. If λ is increased, model complexity will have a greater contribution to the cost. Because the minimum cost hypothesis is selected, this means that higher λ will bias the selection toward models with lower complexity.

Gradient descent

Repeat $\{$

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

$$(j = 0, 1, 2, 3, \ldots, n)$$

$\}$

$$\theta_j := \theta_j (1 - \alpha \frac{\lambda}{m}) - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

Normal equation

$$X = \begin{bmatrix} (x^{(1)})^T \\ \vdots \\ (x^{(m)})^T \end{bmatrix} \qquad y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

$$\min_{\theta} J(\theta)$$

Non-invertibility (optional/advanced).

Suppose $m \leq n$,

(#examples) (#features)

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

If $\lambda > 0$,

$$\theta = \left(X^T X + \lambda \begin{bmatrix} 0 & & & \\ & 1 & & \\ & & 1 & \\ & & & \ddots \\ & & & & 1 \end{bmatrix} \right)^{-1} X^T y$$

Summary:Regularization

Regularization basically adds the penalty as model complexity increases.

Regularization parameter (lambda) penalizes all the parameters or coefficients except intercept so that model generalizes the data and won't overfit.

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right]$$

$$\min_\theta J(\theta)$$

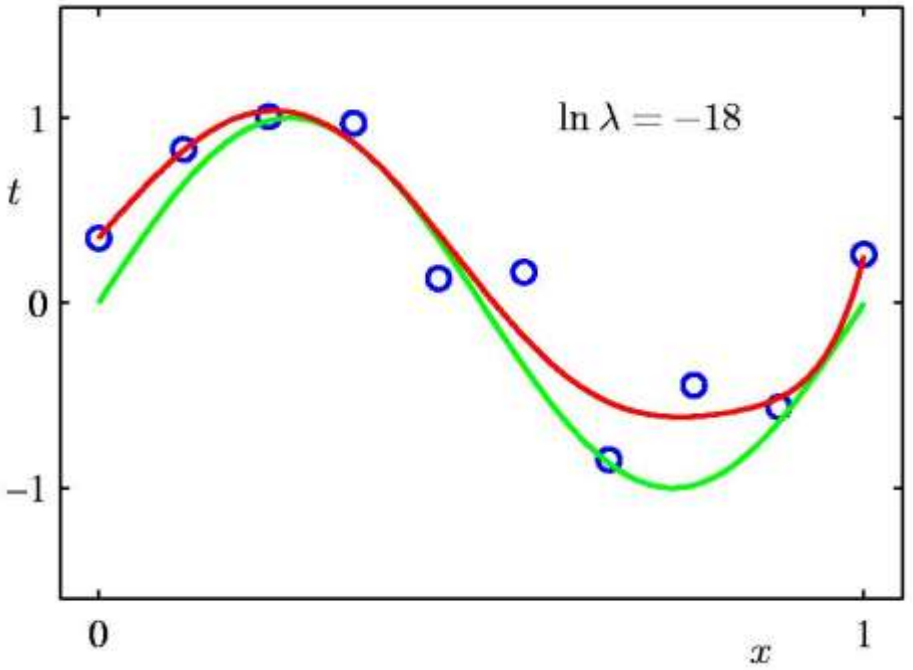
λ governs the relative importance of the regularization term compared with the sum of square error term.

Known as shrinkage methods, since they reduce the value of coefficients.

The particular case of quadratic regularizer is called ridge regression.

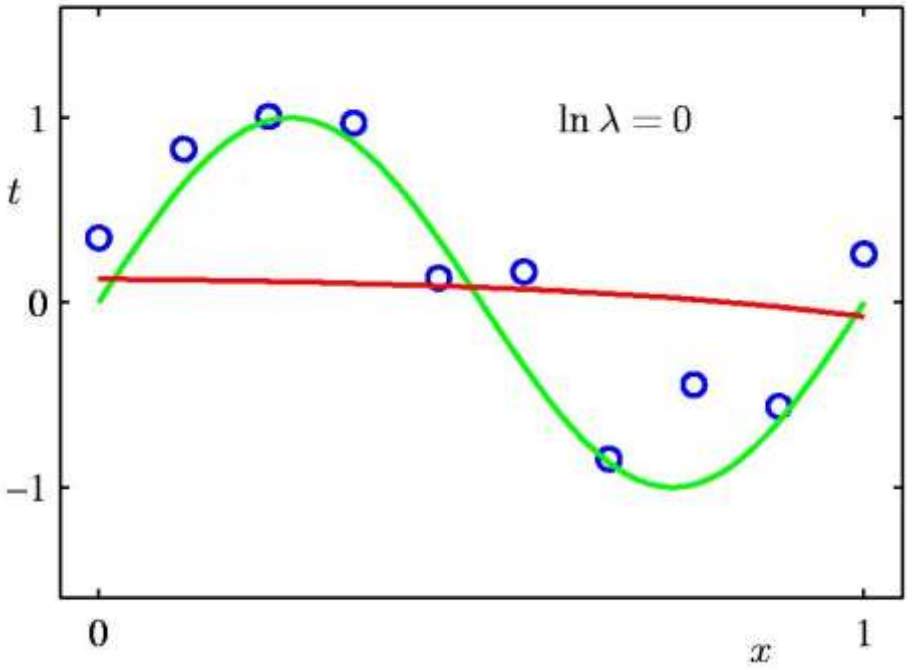
Regularization:

$$\ln \lambda = -18$$



Regularization:

$$\ln \lambda = 0$$



Regularization Methods (L1 and L2)

A regression model that uses L1 regularization technique is called ***Lasso Regression*** and model which uses L2 is called ***Ridge Regression***.

The key difference between these two is the penalty term.

$$\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

Ridge or L2 regression adds ¨*squared magnitude*¨ of coefficient as penalty term to the loss function. Here the *highlighted* part represents L2 regularization element.

Here, if *lambda* is zero or too low then – we will get back OLS (ordinary least square) – that means does not do anything

If *lambda* is very large - then it will add too much weight and it will lead to under-fitting.

Its important how *lambda* is chosen. This technique works very well to avoid over-fitting issue.

Ridge or L2 regularization tends to yield a “dense” solution, where the magnitude of the coefficients are evenly reduced.

For example, for a model with 3 parameters, B1, B2, and B3 will reduce by a similar factor

Lasso Regression - (Least Absolute Shrinkage and Selection Operator) adds "*absolute value of magnitude*" of coefficient as penalty term to the loss function.

$$\sum_{i=1}^{n}(Y_i - \sum_{j=1}^{p} X_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

Again, if *lambda* is zero then we will get back OLS whereas very large value will make coefficients zero hence it will under-fit.

The **key difference** between these techniques is that Lasso shrinks the less important feature's coefficient to zero thus, removing some feature altogether.

So, this works well for **feature selection** in case we have a huge number of features.

Comparison between L2 and L1

Often neither one is overall better.

Lasso can set some coefficients to zero, thus performing variable selection, while ridge regression cannot.

Both methods allow to use correlated features, but they solve multicollinearity issue differently:

- In ridge regression, the coefficients of correlated features are similar;
- In lasso, one of the correlated feature has a larger coefficient, while the rest are (nearly) zeroed.

Lasso tends to do well if there are a small number of significant parameters and the others are close to zero (when only a few features actually influence the response).

Ridge works well if there are many large parameters of about the same value (when most predictors impact the response).

In practice, the true parameter values are not available, so the previous two points are somewhat theoretical. Just run cross-validation to select the more suited model for a specific case or combine the two.

Model Selection

- Model Parameters
- Hyper Parameters

Normally we randomly set the value for these hyper parameters and see what parameters result in best performance. However randomly selecting the parameters for the algorithm can be exhaustive.

Also, it is not easy to compare performance of different algorithms by randomly setting the hyper parameters because one algorithm may perform better than the other with different set of parameters. And if the parameters are changed, the algorithm may perform worse than the other algorithms.

Therefore, instead of randomly selecting the values of the parameters, a better approach would be to develop an algorithm which automatically finds the best parameters for a particular model. Grid Search is one such algorithm.

Grid Search

Grid-search is used to find the optimal hyperparameters of a model which results in the most ‘accurate’ predictions.

Grid search builds a model for every combination of hyperparameters specified and evaluates each model. A more efficient technique for hyperparameter tuning is the Randomized search — where random combinations of the hyperparameters are used to find the best solution.

Evaluating Predictor

Training + Test Set (Holdout Set)
Cross Validation
Training + Validation + Test Set
Training + Test Set and Cross Validation of Training Set for Model Selection

Model Validation

A process of deciding whether the numerical results quantifying hypothesized relationships between variables, are acceptable as descriptions of the data, is known as **validation**

Training error –

only gives us an idea about how well our model does on data used to train it. Its possible that the model is under-fitting or overfitting the data.

So, the **problem with this evaluation technique is that it does not give an indication of how well the learner will generalize to an independent / unseen data set**.

Getting this idea about our model is known as Cross Validation.

Two types:

Non-exhaustive – do not compute all ways of splitting the data. Just decides on the number of subsets

Exhaustive - that computes all possible ways the data can be split into training and test sets.

References

- Andrew Ng's slides on Multiple Linear Regression from his Machine Learning Course on Coursera.
- Data Mining Book from Han and Kamber

Disclaimer

- Content of this presentation is not original and it has been prepared from various sources for teaching purpose.