Linear regression

Tao Tan

Last time

- Supervised learning
  - Sensitivity
  - Specificity
  - ROC
- Unsupervised learning
  - Kmeans
- Other learning
  - Semi-supervised learning
  - Weakly-supervised learning

Simple Linear Regression Example

- A real estate agent wishes to examine the relationship between the selling price of a home and its size (measured in square meter)

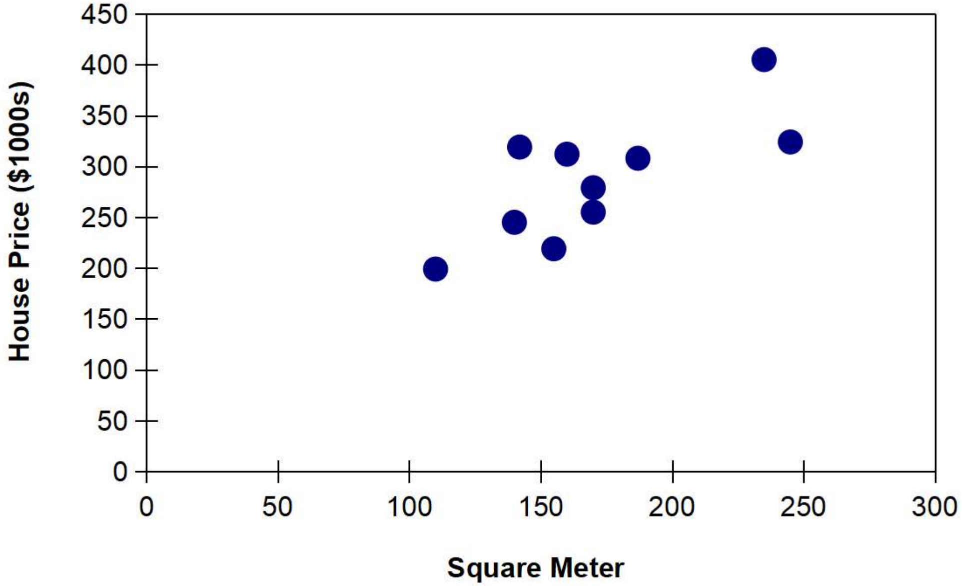
- A random sample of 10 houses is selected
  - Dependent variable (Y) = house price in $1000s
  - Independent variable (X) = square meter

Simple Linear Regression Example: Data

House Price in $1000s (Y)	Square Meter (X)
245	140
312	160
279	170
308	187
199	110
219	155
405	235
324	245
319	142
255	170

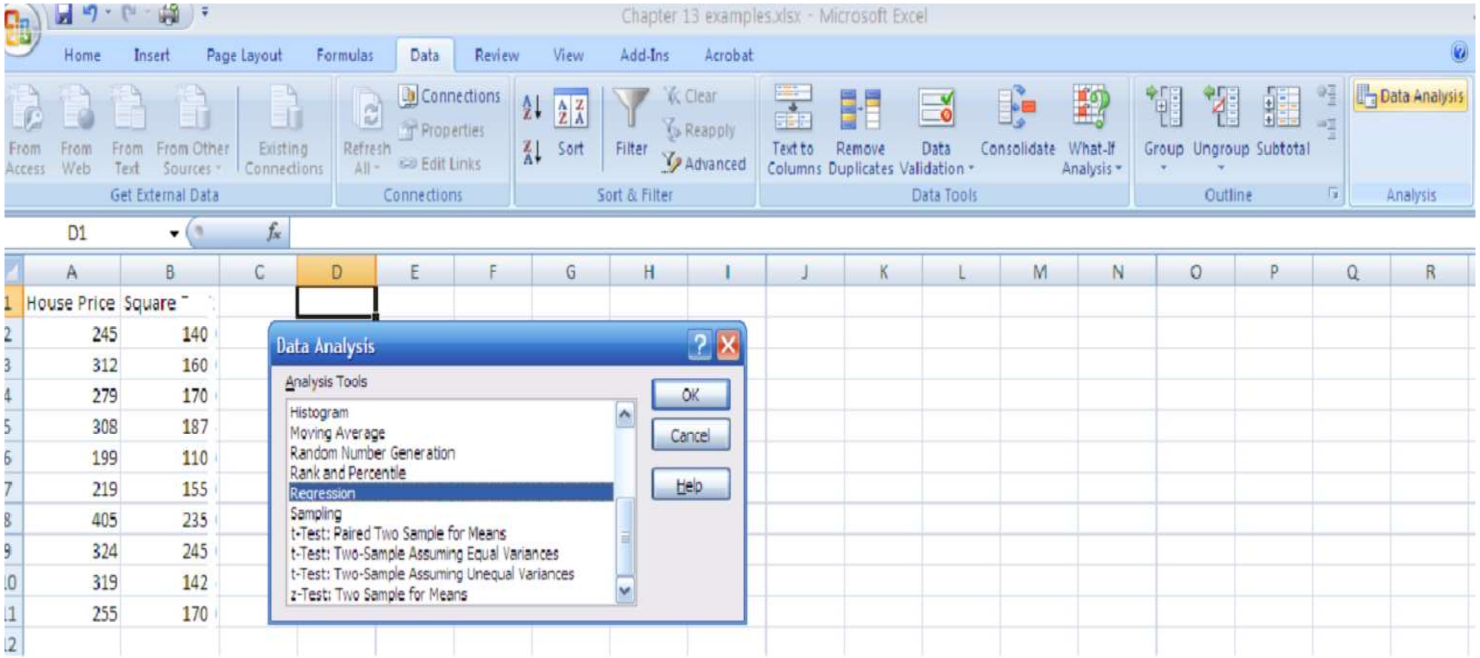
Simple Linear Regression Example: Scatter Plot

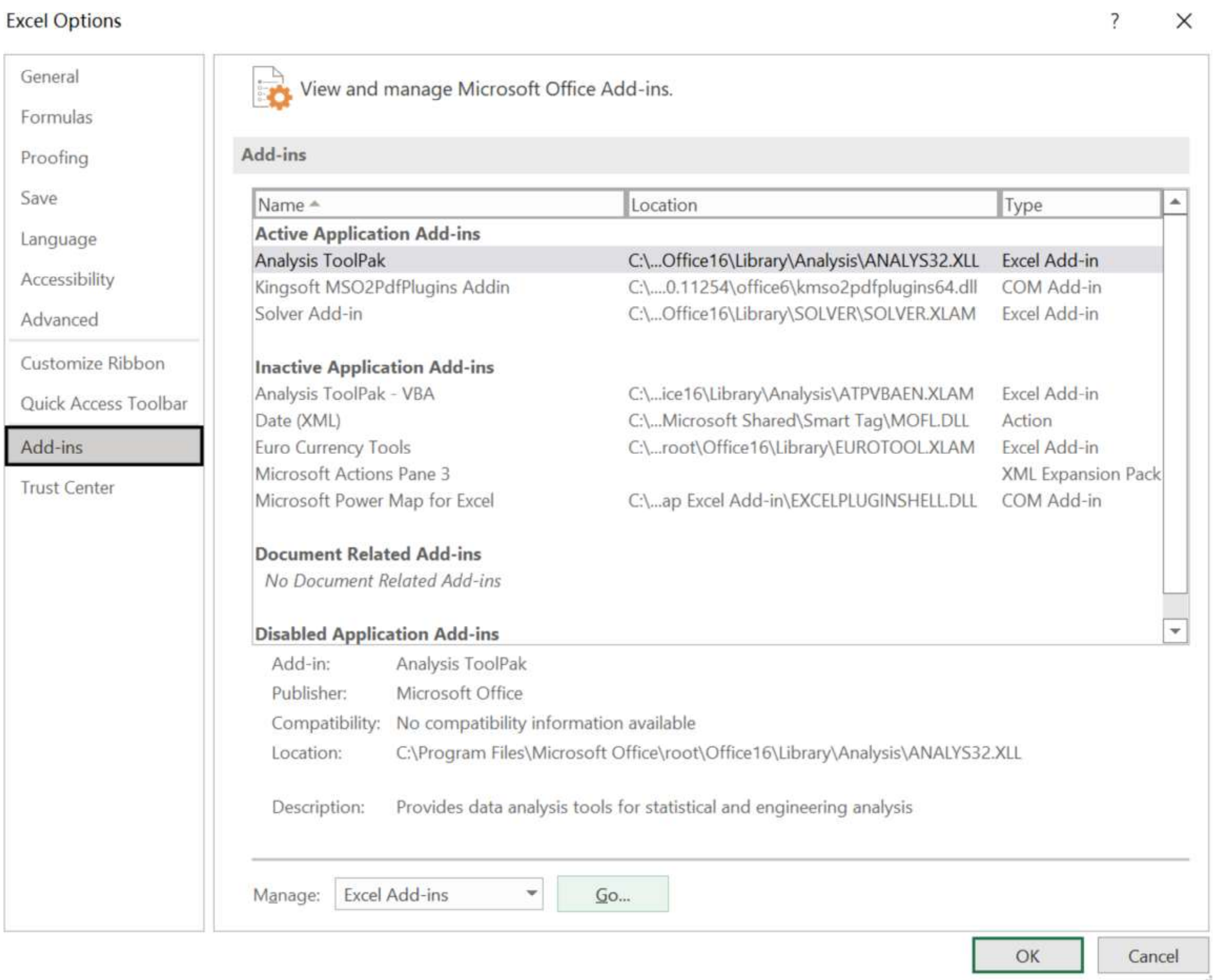
House price model: Scatter Plot



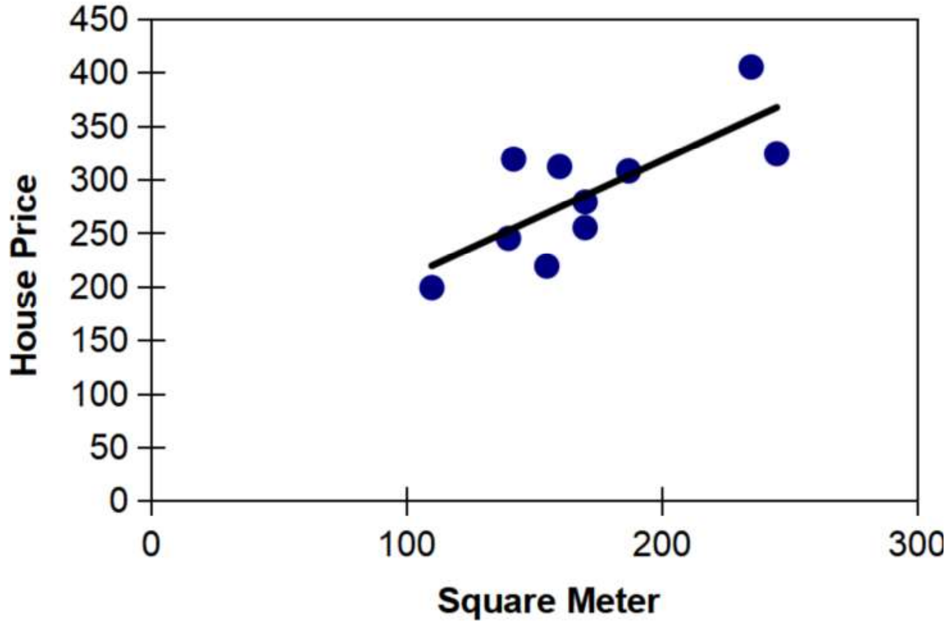
0

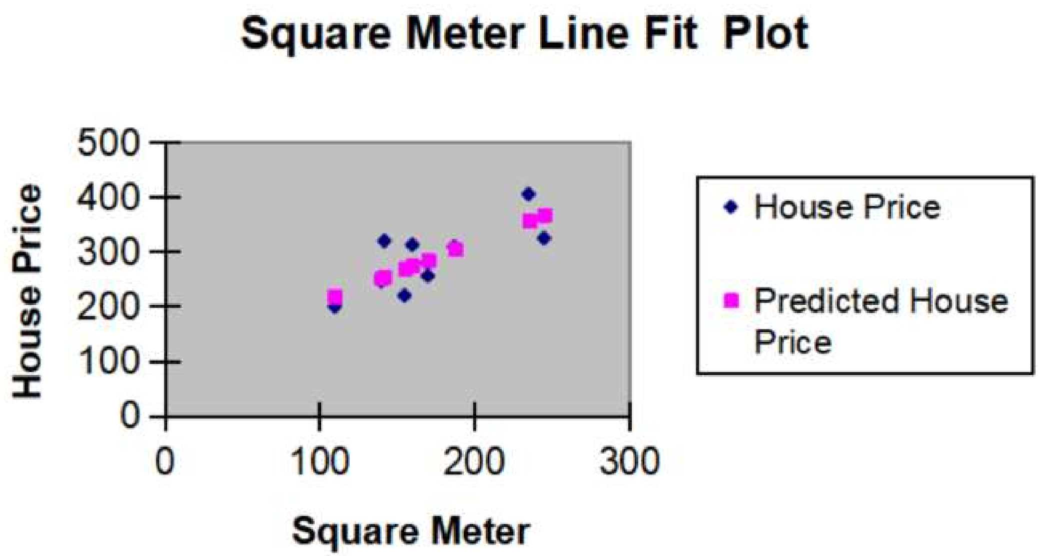
Linear Regression Example: Using Excel Data Analysis Function




Excel Options
General
Formulas
Proofing
Save
Language
Accessibility
Advanced
Customize Ribbon
Quick Access Toolbar
Add-ins
Trust Center
View and manage Microsoft Office Add-ins.
Add-ins
Name
Location
Type
Active Application Add-ins
Analysis ToolPak
C:\...Office16\Library\Analysis\ANALYS32.XLL
Excel Add-in
Kingsoft MSO2PdfPlugins Addin
C:\....0.11254\office6\kmso2pdfplugins64.dll
COM Add-in
Solver Add-in
C:\...Office16\Library\SOLVER\SOLVER.XLAM
Excel Add-in
Inactive Application Add-ins
Analysis ToolPak - VBA
C:\...ice16\Library\Analysis\ATPVBAEN.XLAM
Excel Add-in
Date (XML)
C:\...Microsoft Shared\Smart Tag\MOFL.DLL
Action
Euro Currency Tools
C:\...root\Office16\Library\EUROTOOL.XLAM
Excel Add-in
Microsoft Actions Pane 3
XML Expansion Pack
Microsoft Power Map for Excel
C:\...ap Excel Add-in\EXCELPLUGINSHELL.DLL
COM Add-in
Document Related Add-ins
No Document Related Add-ins
Disabled Application Add-ins
Add-in: Analysis ToolPak
Publisher: Microsoft Office
Compatibility: No compatibility information available
Location: C:\Program Files\Microsoft Office\root\Office16\Library\Analysis\ANALYS32.XLL
Description: Provides data analysis tools for statistical and engineering analysis
Manage: Excel Add-ins
Go...
OK
Cancel

Linear Regression





Linear regression

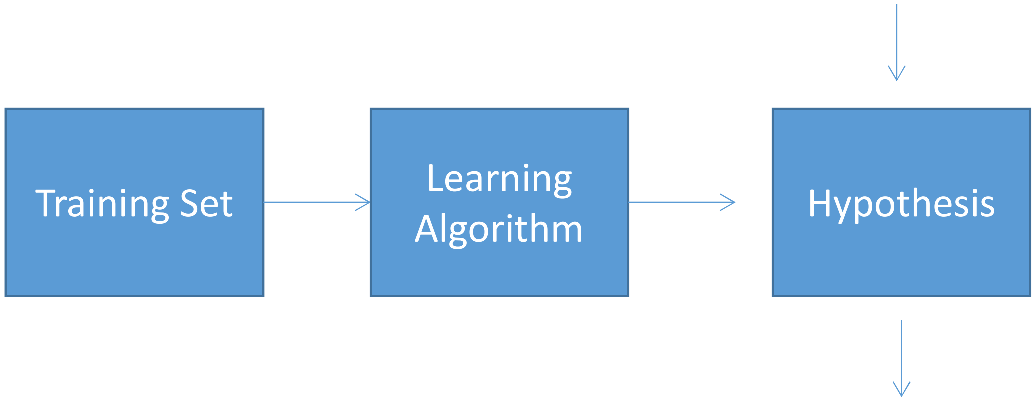
- m = Number of training samples

- X = independent (explanatory) variable
- Y = dependent (response) variable

- (X,Y) training examples
- (X_i, Y_i) i^{th} training example

Regression Model

Size of house x



Estimated price y

hypothesis maps from x’s to y’s

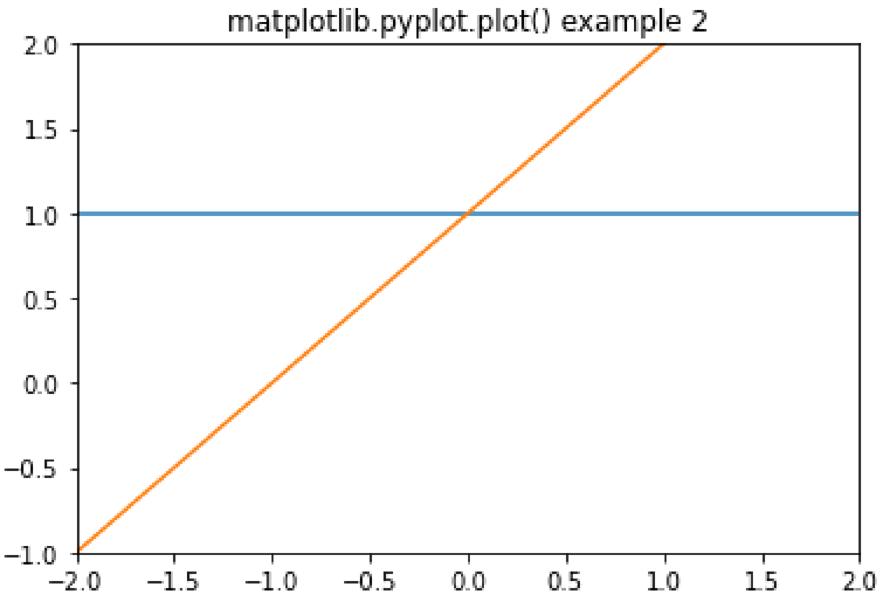
$\hat{y} = b_0 + b_1 X$
where
$\hat{y}$ represents predicted averageof Y at a given X
b_0 represents the line’s intercept
b_1 represents the line’s slope

$b_{i:}$ Parameter

how to choose b_i

Hypothesis

- $\hat{y} = b_0 + b_1 X$

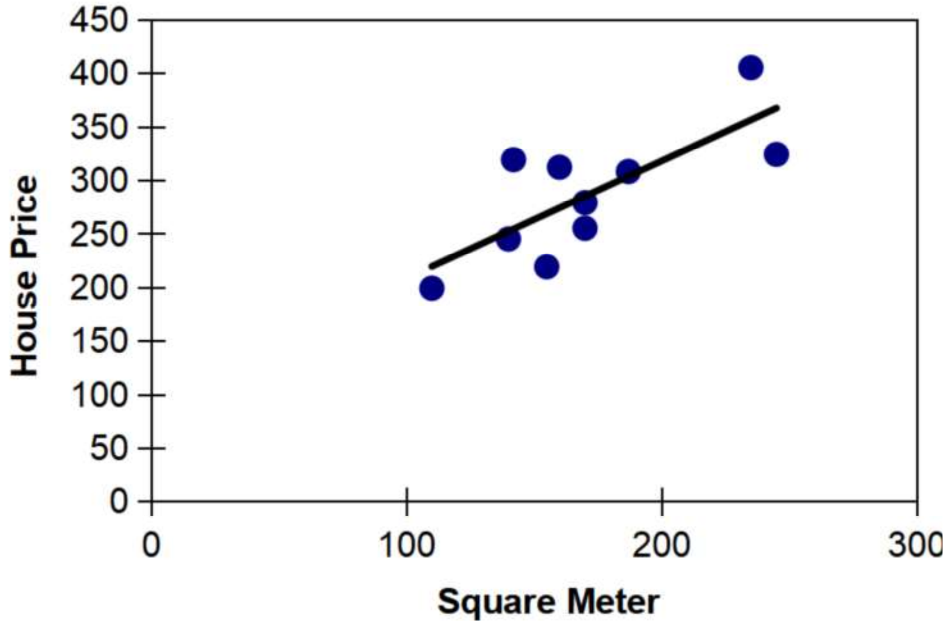



```
#implementation of matplotlib function

import matplotlib.pyplot as plt
import numpy as np

# Fixing random state for reproducibility
np.random.seed(19680801)

# create random data
xdata = np.arange(-3, 3, 0.5)



# create some y data points
ydata1 = 0*xdata +1


ydata2 = 1*xdata + 1

# plot the data
plt.plot(xdata, ydata1, color ='tab:blue')
plt.plot(xdata, ydata2, color ='tab:orange')


# set the limits
plt.xlim([-2, 2])
plt.ylim([-1, 2])

plt.title('matplotlib.pyplot.plot() example 2')
```

How formulas determine best line

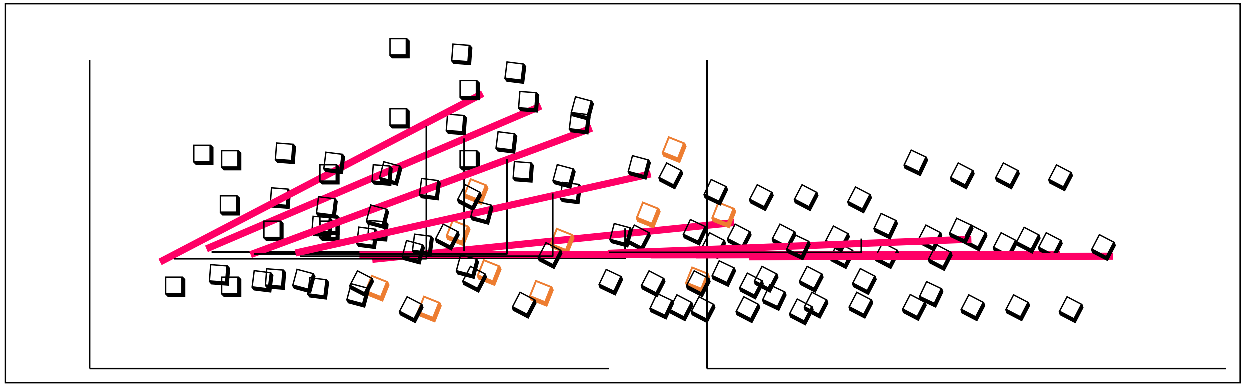
Idea: choose b so that $h_b(x)$ is close to y for our training examples (x,y)

- Distance of points from line = *residuals* (dotted)
- Minimizes sum of square residuals
- *Least squares regression line*



Testing the Slope

When no linear relationship exists between two variables, the regression line should be horizontal.



Linear relationship.
Different inputs (X) yield different outputs (Y).

The slope is not equal to zero

No linear relationship.
Different inputs (X) yield the same output (Y).

The slope is equal to zero

Linear Regression

- Any straight line can be represented by an equation of the form $Y = b_1X + b_0$, where b_1 and b_0 are constants.
- The value of b_1 is called the slope constant and determines the direction and degree to which the line is tilted.
- The value of b_0 is called the Y-intercept and determines the point where the line crosses the Y-axis.

Least Square

$$y_i = b_0 + b_1 x_i + e_i$$
$$\hat{y}_i = b_0 + b_1 x_i$$

cost function

$$\min_{b_0, b_1} J(b_0, b_1) = \sum_{i=1}^{n} (e_i)^2$$
$$= \sum_{i=1}^{n} (y_i - b_0 - b_1 x_i)^2$$

The b_0 and b_1 terms are population parameters, which are unknown. The term e_i represents any unmodelled components of the linear model, measurement error, and is simply called the error term.

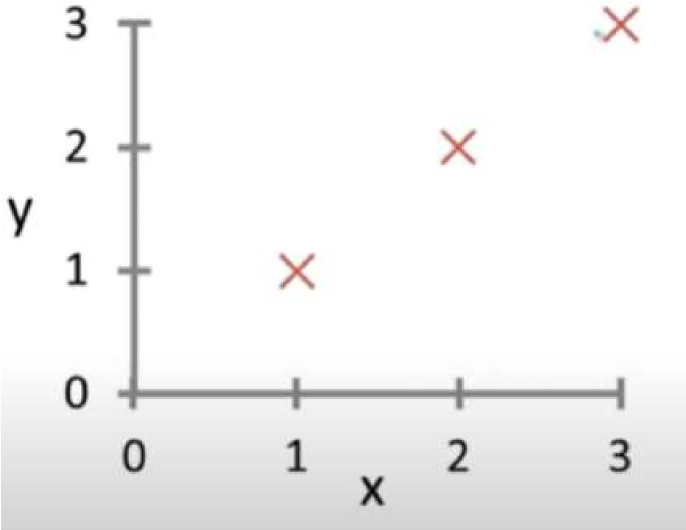
Exercise

$$\min_{b_0, b_1} J(b_0, b_1) = \sum_{i=1}^{n} (e_i)^2$$
$$= \sum_{i=1}^{n} (y_i - b_0 - b_1 x_i)^2$$

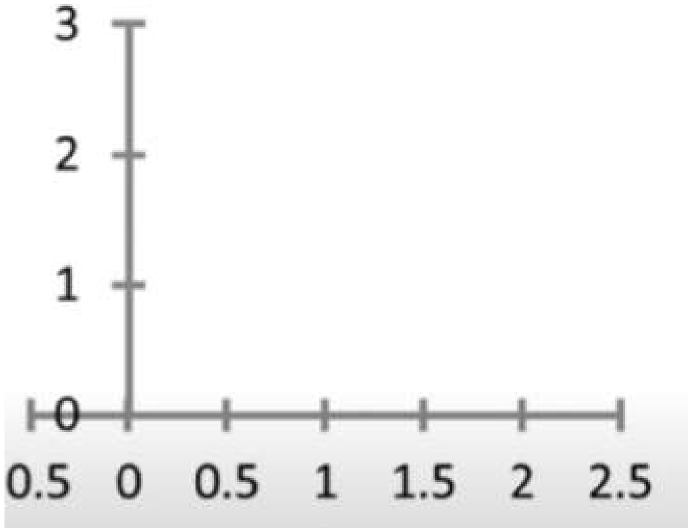
$J(b_1)$

- $h_b(x)$ for fixed b_1, , this is a function of x , b_0=0

function of the parameter b_1



$J(b_1)$



b_1

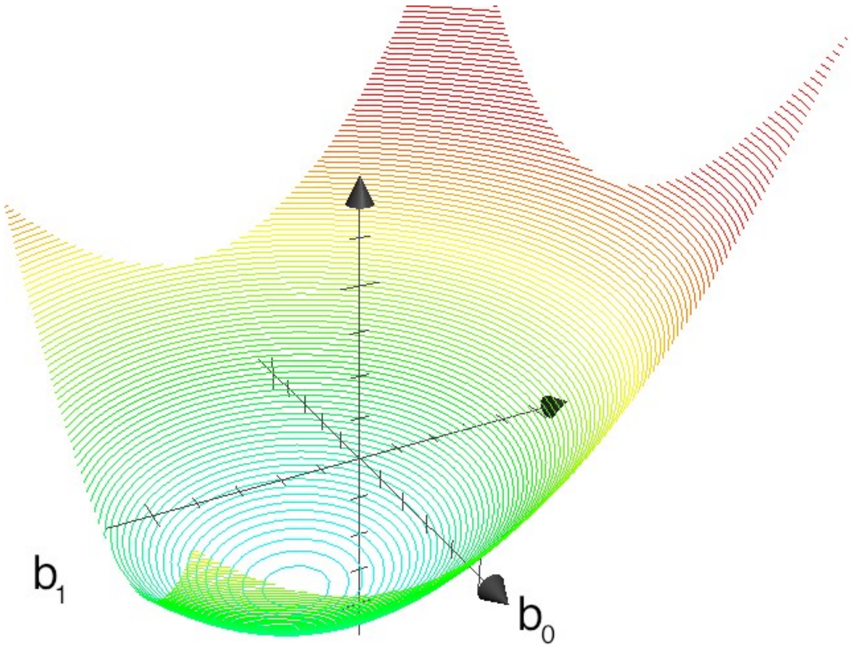
Interpretation of Slope

- b_1 = expected change in Y per unit X
- Keep track of units!
- e.g., b_1 of –0.64 predicts d*ecrease* of 0.64 units of Y for each unit X

Linear Regression

- **Regression** is a statistical procedure that determines the equation for the straight line that best fits a specific set of data.
- The Pearson correlation (Pearson's correlation) measures the degree to which a set of data points form a straight line relationship.

Least Square



$$\frac{\partial J(b_0, b_1)}{\partial b_0} = -2\sum_{i}^{n}(y_i - b_0 - b_1 x_i) = 0$$

$$\frac{\partial J(b_0, b_1)}{\partial b_1} = -2\sum_{i}^{n}(x_i)(y_i - b_0 - b_1 x_i) = 0$$

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

$$b_1 = \frac{\sum_i (x_i - \bar{\mathrm{x}})(y_i - \bar{\mathrm{y}})}{\sum_i (x_i - \bar{\mathrm{x}})^2}$$

Linear regression

$$b_0 = \bar{y} - b_1\bar{x}$$
$$b_1 = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2} = \frac{s_{xy}}{s_x^2} = r_{xy}\frac{s_y}{s_x}$$

Since $-1 < r_{xy} < 1$ then we get that if x is some measurement and y is a followup measurement from the same item, then we expect that y (on average) will be closer to the mean measurement than it was to the original value of x. This phenomenon is known as **regressions toward the mean**.

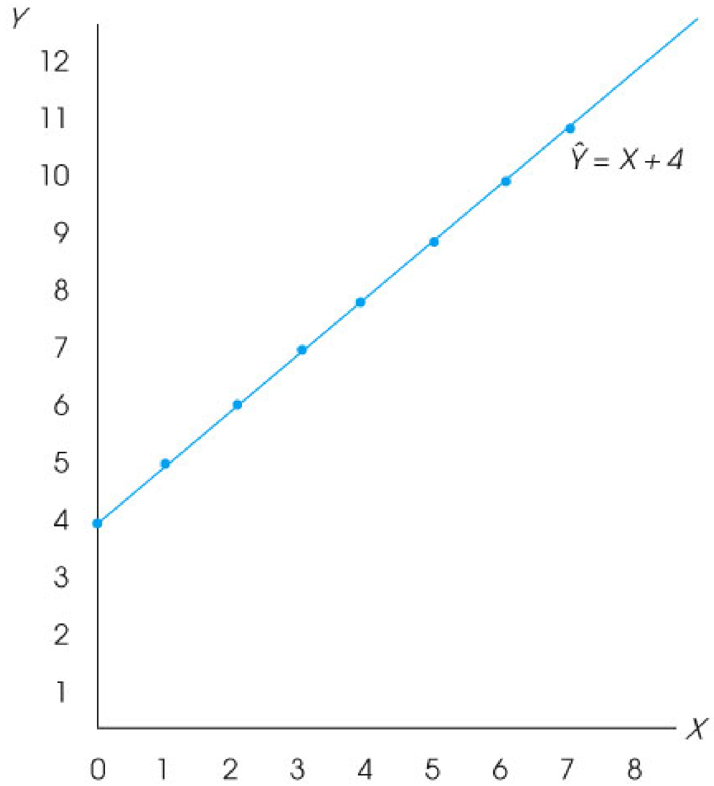
r_{xy} as the sample correlation coefficient between x and y

s_x and s_y as the uncorrected sample standard deviation of x and y

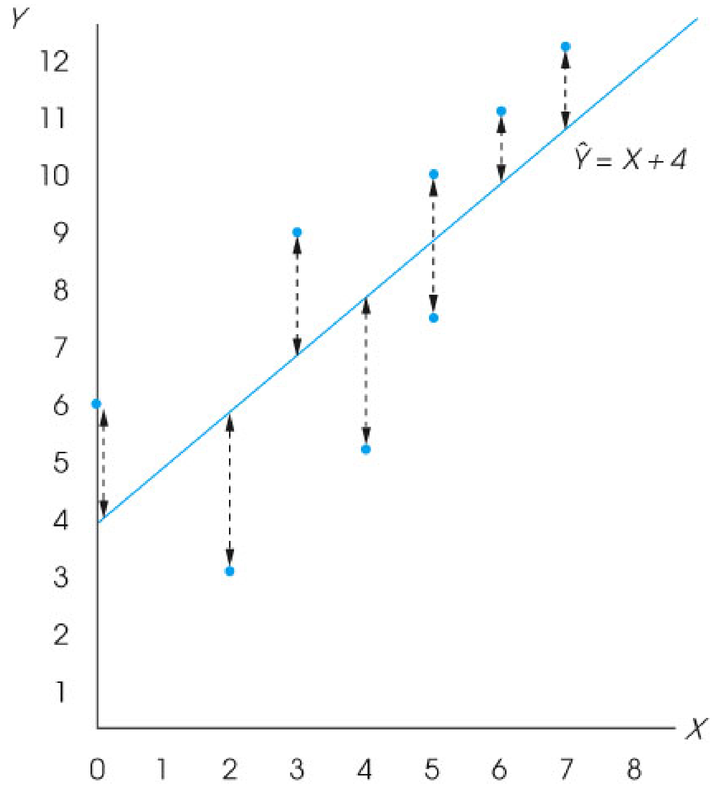
s_x^2 and S_{xy} as the sample variance and sample covariance, respectively

Linear Regression

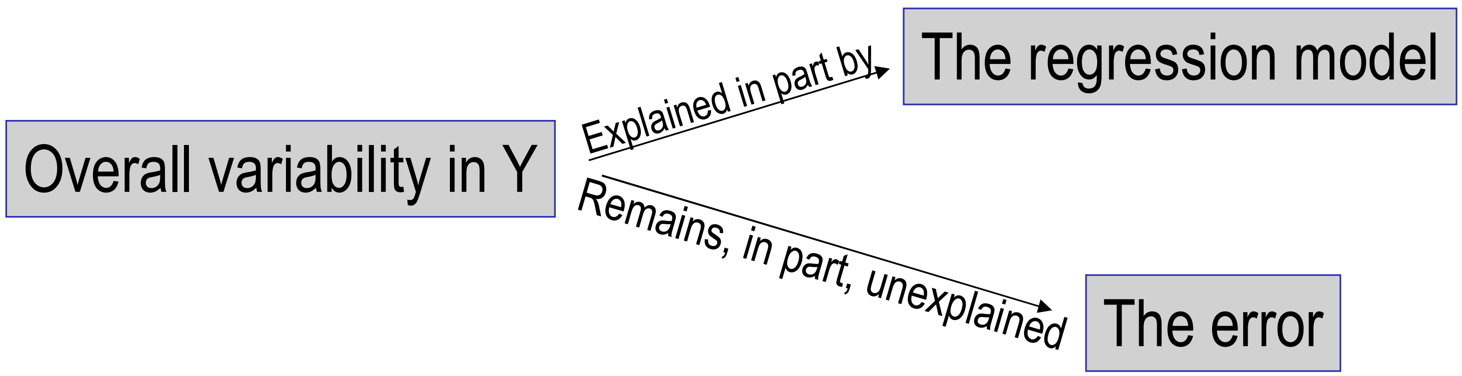
- How well a set of data points fits a straight line can be measured by calculating the distance between the data points and the line.
- The total error between the data points and the line is obtained by squaring each distance and then summing the squared values.
- The regression equation is designed to produce the minimum sum of squared errors.



Y
12
11
10
9
8
7
6
5
4
3
2
1
0 1 2 3 4 5 6 7 8
X
$\hat{Y} = X + 4$



Y
12
11
10
9
8
7
6
5
4
3
2
1
0 1 2 3 4 5 6 7 8
X
$\hat{Y} = X + 4$


Overall variability in Y
Explained in part by
The regression model
Remains, in part, unexplained
The error

Model analysis

- Analysis of variance: breaking down the data's variability into components

- Confidence intervals for the model coefficients b_0, and b_1

- Prediction error estimates for the y variable

Model analysis

- The analysis of variance is a tool to show how much variability in y the -variable is explained by:
  - Doing nothing (no model: this implies $\hat{y} = \overline{y}$)
  - The model ($\widehat{y_i}$ = $b_0 + b_1 x_i$)
  - How much variance is left over in the errors

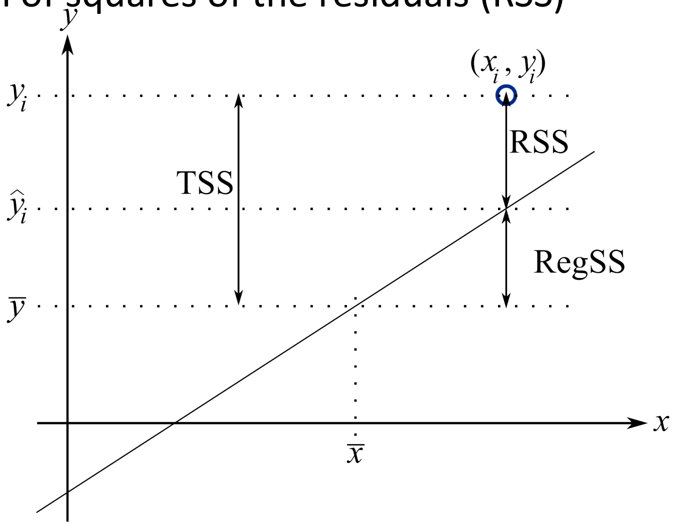
Model analysis

$$\begin{aligned}
\text{Distance relationship:} \quad (y_i - \overline{y}) &= (\hat{y}_i - \overline{y}) + (y_i - \hat{y}_i) \\
\text{Squaring both sides:} \quad (y_i - \overline{y})^2 &= (\hat{y}_i - \overline{y})^2 + 2(\hat{y}_i - \overline{y})(y_i - \hat{y}_i) + (y_i - \hat{y}_i)^2 \\
\text{Sum and simplify:} \quad \sum (y_i - \overline{y})^2 &= \sum (\hat{y}_i - \overline{y})^2 + \sum (y_i - \hat{y}_i)^2 \\
\text{Total sum of squares (TSS)} &= \text{Regression SS (RegSS)} + \text{Residual SS (RSS)}
\end{aligned}$$

The total sum of squares (TSS)

The sum of squares due to regression (RegSS)

The sum of squares of the residuals (RSS)



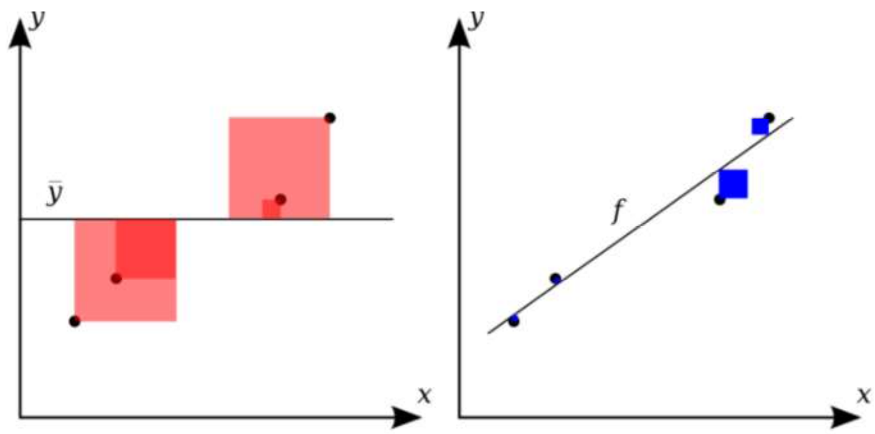
https://learnche.org/

Analysis of Variance (ANOVA) table

Type of variance	Distance	Degrees of freedom	SSQ	Mean square
Regression	$\hat{y}_i - \overline{y}$	k ($k = 2$ in the examples so far)	RegSS	RegSS/k
Error	$y_i - \hat{y}_i$	$n - k$	RSS	$\text{RSS}/(n-k)$
Total	$y_i - \overline{y}$	n	TSS	TSS/n

Linear regression

Wikipedia



The sum of squares of residuals, also called the residual sum of squares

RegSS= $\sum_{i=1}^{n} (y_i - \hat{y}_i)^2$

The total sum of squares of
RSS= $\sum_{i=1}^{n} (y_i - \overline{y}_i)^2$

The most general defination of the coefficient of determination is

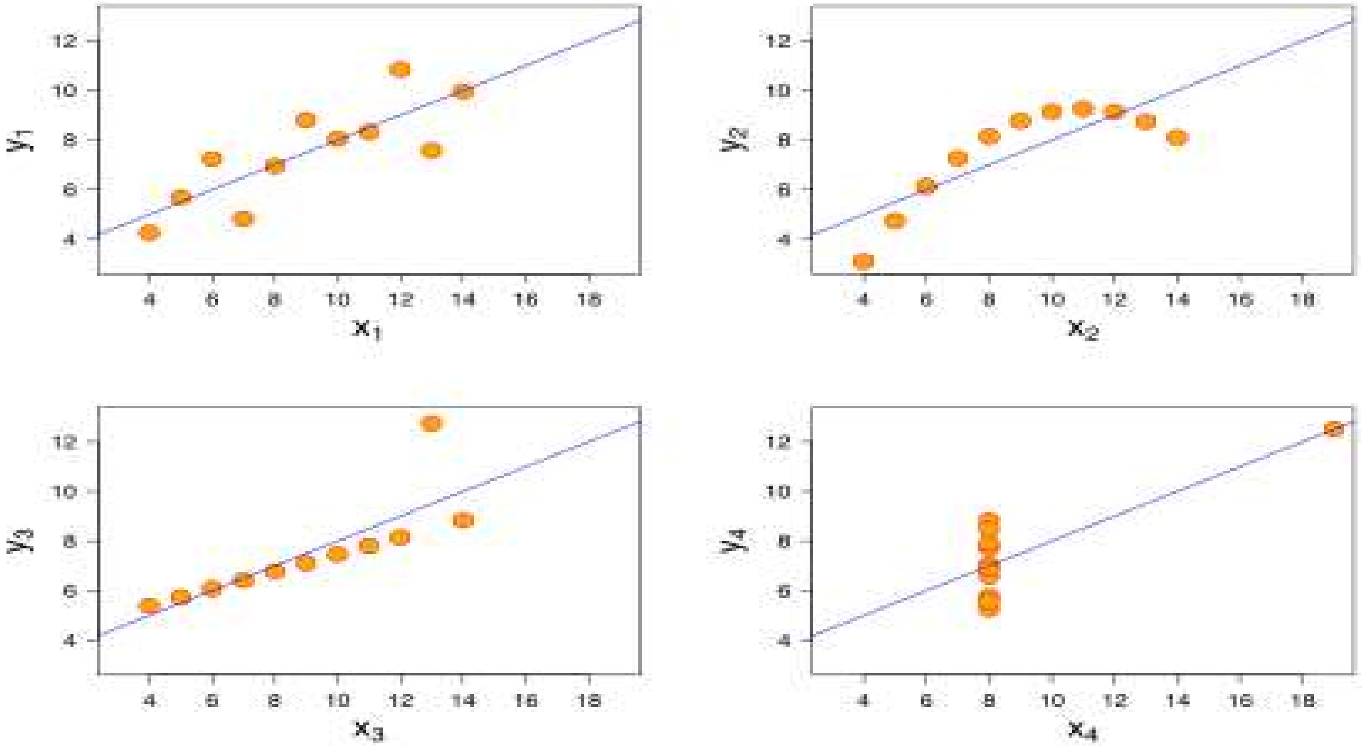
R^2= 1- $\frac{RegSS}{\text{RSS}}$

The best case R^2= 0
What would be the worst case ?

The better the linear regression (on the right) fits the data in comparison to the simple average (on the left graph), the closer the value of R^2 is to 1. The areas of the blue squares represent the squared residuals with respect to the linear regression. The areas of the red squares represent the squared residuals with respect to the average value.

Linear regression

The data sets in the Anscombe's quartet are designed to have approximately the same linear regression line (as well as nearly identical means, standard deviations, and correlations) but are graphically very different. This illustrates the pitfalls of relying solely on a fitted model to understand the relationship between variables.

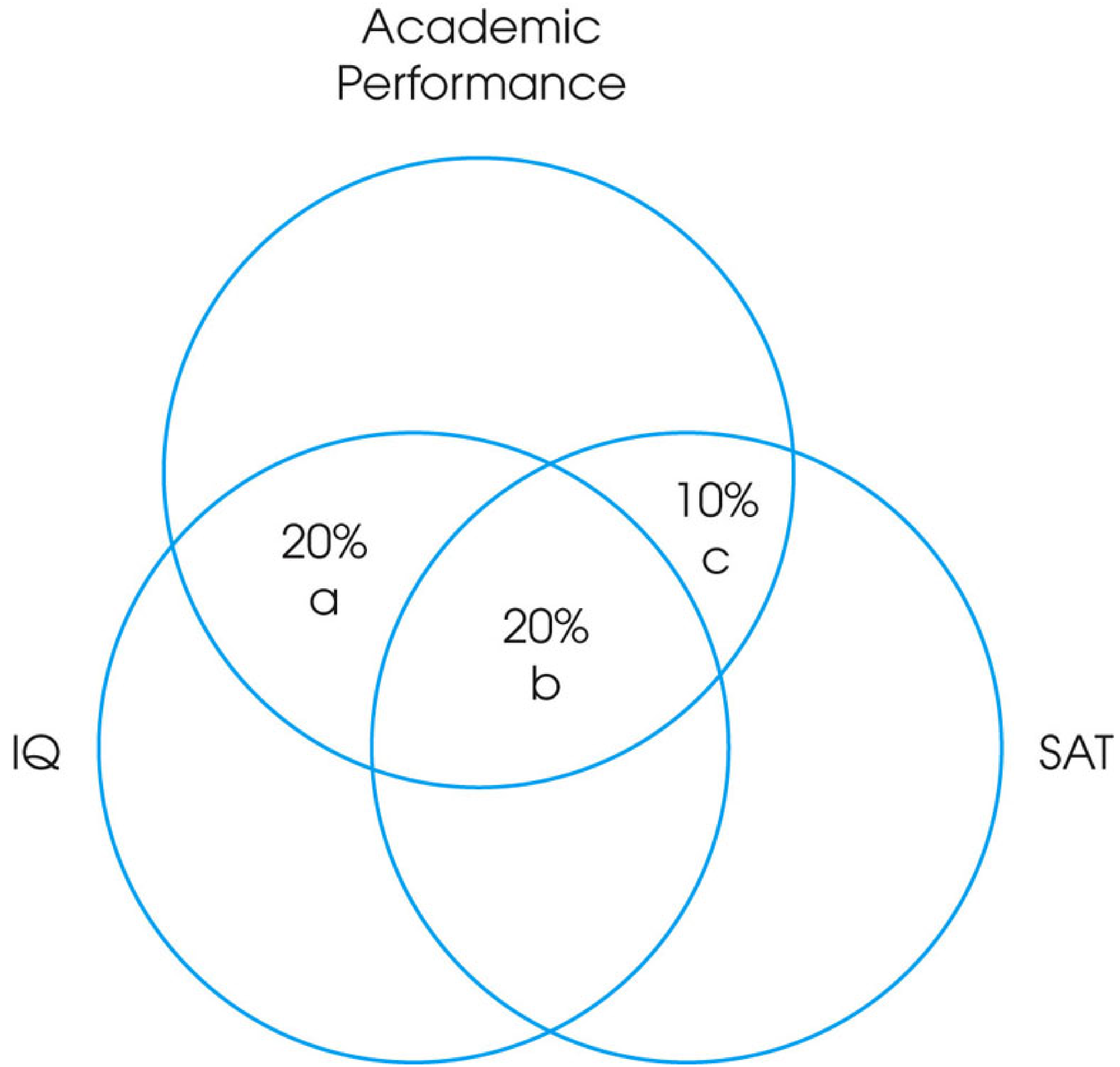


Anscombe's quartet from wikipedia

Break

Introduction to Multiple Regression

- In the same way that linear regression produces an equation that uses values of X to predict values of Y, multiple regression produces an equation that uses two different variables (X_1 and X_2) to predict values of Y.
- The equation is determined by a least squared error solution that minimizes the squared distances between the actual Y values and the predicted Y values.



The SAT is a standardized test widely used for college admissions in the United States. Since its debut in 1926, its name and scoring have changed several times; originally called the Scholastic Aptitude

Introduction to Multiple Regression with Two Predictor Variables

- For two predictor variables, the general form of the multiple regression equation is:

$$\hat{Y} = b_1X_1 + b_2X_2 + b_0$$

- The ability of the multiple regression equation to accurately predict the Y values is measured by first computing the proportion of the Y-score variability that is predicted by the regression equation and the proportion that is not predicted.

Multiple features

Size (m^2) x	Price ($1000) y
210	460
141	232
153	315
85	178
…	…

Multiple features (variables)

Size (feet^2)	Number of bedrooms	Number of floors	Age of home (years)	Price ($1000)
210	5	1	45	460
141	3	2	40	232
153	3	2	30	315
85	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.

Hypothesis:

Previously: $h_b(x) = b_0 + b_1 x$

$$h_b(x) = b_0 + b_1 x_1 + b_2 x_2 + b_3 x_3 + b_4 x_4 + b_5 x_5$$

$$h_b(x) = b^T x = [b_0\ b_1\ b_2\ b_3\ b_4\ b_5] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}$$

Multivariate linear regression

Hypothesis:

Parameters: $\boldsymbol{b}_0 \boldsymbol{b}_1, \ldots \boldsymbol{b}_n$

Cost function:

$$J(\boldsymbol{b}_0 \boldsymbol{b}_1, \ldots \boldsymbol{b}_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_{\boldsymbol{b}}(x^{(i)}) - y^{(i)})^2$$

Gradient descent:

assignment

Repeat $\{$

learning rate

derivative term of J

$$\boldsymbol{b}_j := \boldsymbol{b}_j - \alpha \frac{\partial}{\partial \boldsymbol{b}_j} J(\boldsymbol{b}_0 \boldsymbol{b}_1, \ldots \boldsymbol{b}_n)$$

with respect to b

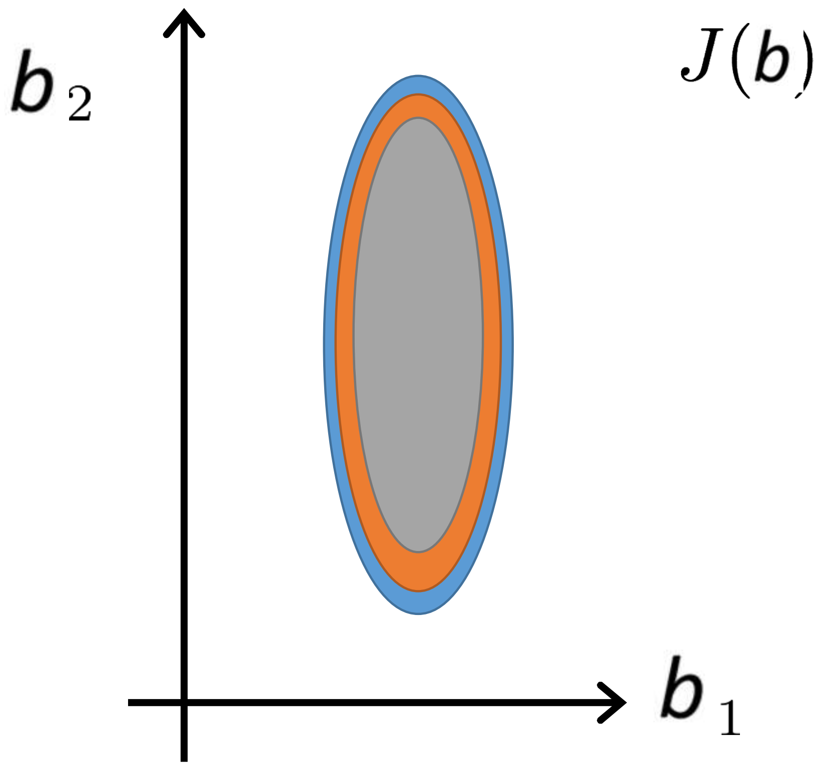
$\}$ (simultaneously update for every $j = 0, \ldots, n$)

Feature Scaling

Idea: Make sure features are on a similar scale.

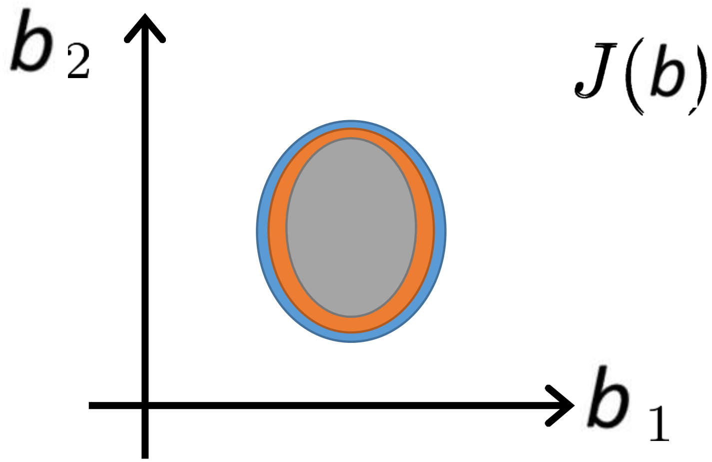
E.g. x_1 = size (0-200 m^2)

x_2 = number of bedrooms (1-5)



$$x_1 = \frac{\text{size (m}^2\text{)}}{200}$$

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



Feature Scaling

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

Normalization

Replace x_i with $x_i - \mu_i$ to make features have approximately zero mean (Do not apply to $x_0 = 1$).

E.g. $$x_1 = \frac{size - 100}{200}$$

$$x_2 = \frac{\#bedrooms - 2}{5}$$

$$-0.5 \leq x_1 \leq 0.5, -0.5 \leq x_2 \leq 0.5$$

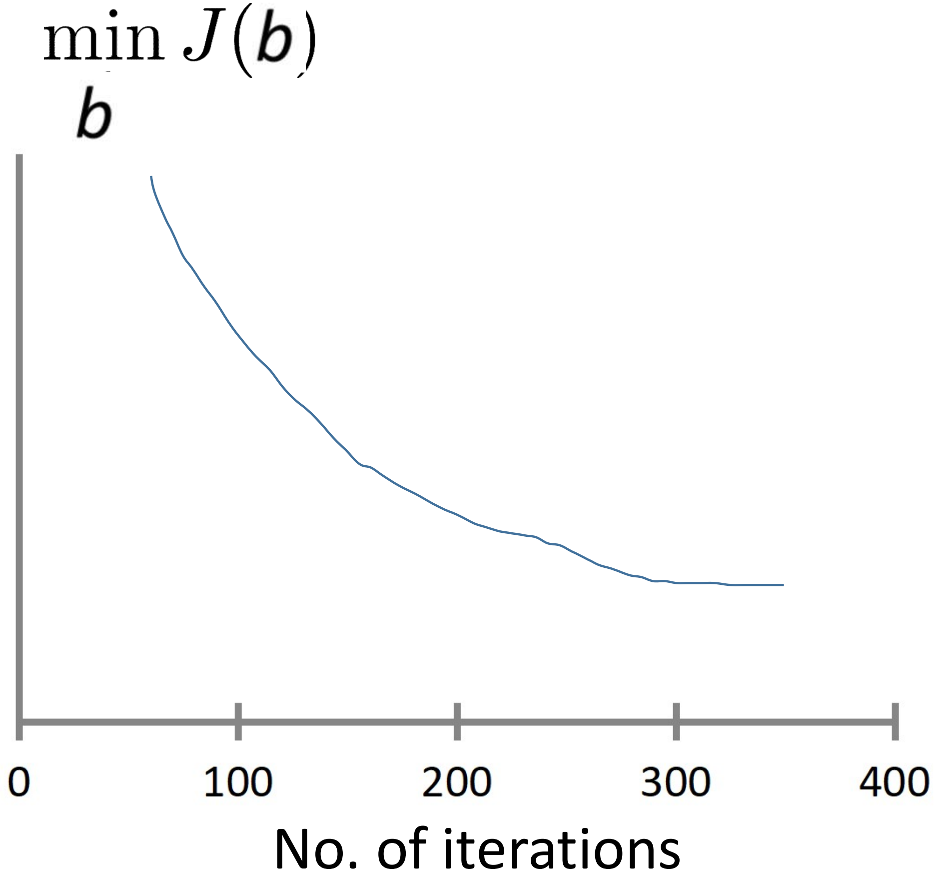
Gradient descent

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

- "Debugging": How to make sure gradient descent is working correctly.

- How to choose learning rate α.

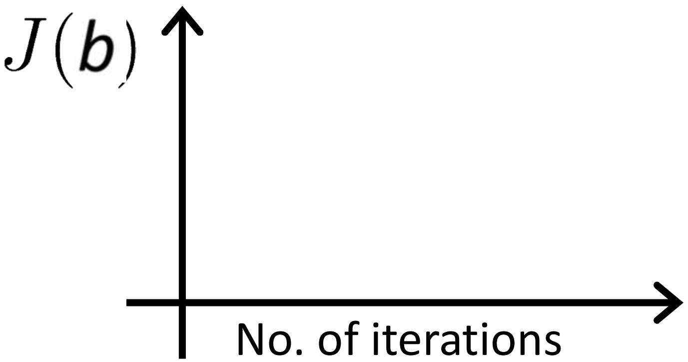
Making sure gradient descent is working correctly.



Example automatic convergence test:

Declare convergence if $J(b)$ 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, $J(\boldsymbol{b})$ should decrease on every iteration.
- But if α is too small, gradient descent can be slow to converge.

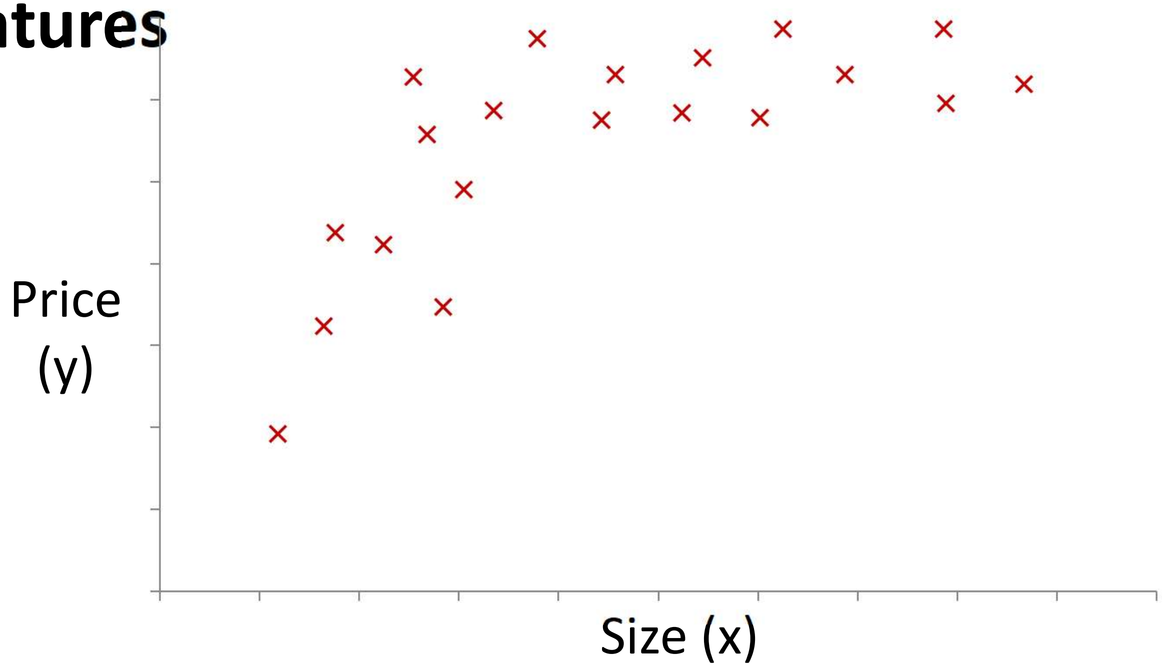
Summary:

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

To choose α, try

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

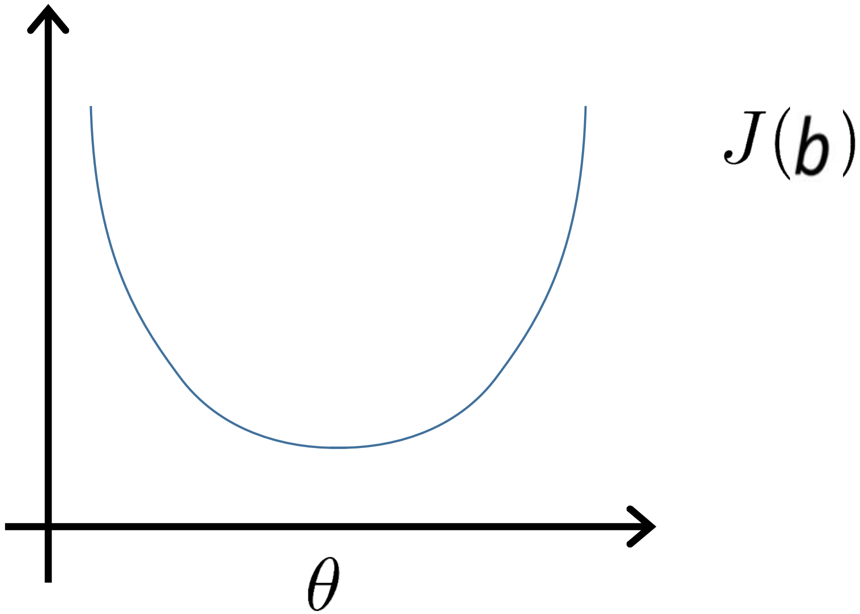
Polynomial regression
Choice of features



$$h_{b}(x) = b_0 + b_1(size) + b_2(size)^2$$

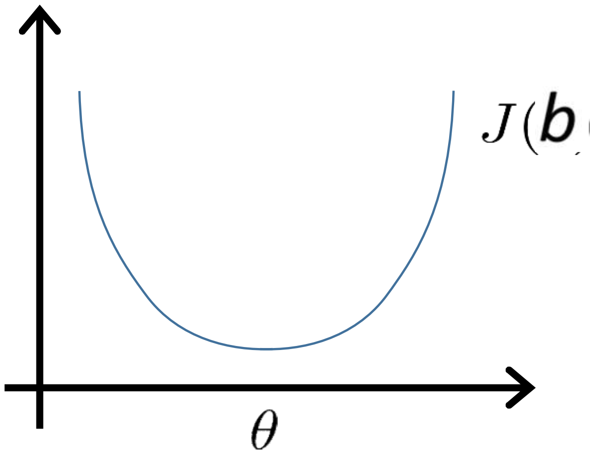
$$h_{b}(x) = b_0 + b_1(size) + b_2\sqrt{(size)}$$

Gradient Descent



Normal equation: Method to solve for b analytically.

Intuition: If 1D



$b \in \mathbb{R}^{n+1}$ $\qquad J(b_0, b_1, \ldots, b_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_b(x^{(i)}) - y^{(i)})^2$

$\frac{\partial}{\partial b_j} J(b) = \cdots = 0$ (for every j)

Solve for $b_0, b_1, \ldots, b_n$

Examples: $m = 5.$

x_0	Size (2) x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price ($1000) y
1	210	5	1	45	460
1	141	3	2	40	232
1	153	3	2	30	315
1	85	2	1	36	178
1	300	4	1	38	540

$$X = \begin{bmatrix} 1 & 210 & 5 & 1 & 45 \\ 1 & 141 & 3 & 2 & 40 \\ 1 & 153 & 3 & 2 & 30 \\ 1 & 85 & 2 & 1 & 36 \\ 1 & 300 & 4 & 1 & 38 \end{bmatrix} \qquad y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \\ 540 \end{bmatrix}$$

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

m **examples** $(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})$ **;** n **features.**

$$x^{(i)} = \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ x_2^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix} \in \mathbb{R}^{n+1}$$

E.g. If $x^{(i)} = \begin{bmatrix} 1 \\ x_1^{(i)} \end{bmatrix}$

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

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

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 $(X^TX)^{-1}$
- Slow if n is very large.

Linear regression by hand

https://www.statology.org/linear-regression-by-hand/

Weight (lbs)	Height (inches)
140	60
155	62
159	67
179	70
192	71
200	72
212	75

Linear regression by hand

https://www.statology.org/linear-regression-by-hand/

Step 1: Calculate X*Y, X2, and Y2

Weight (lbs)	Height (inches)	X*Y	X^2	Y^2
140	60	8400	19600	3600
155	62	9610	24025	3844
159	67	10653	25281	4489
179	70	12530	32041	4900
192	71	13632	36864	5041
200	72	14400	40000	5184
212	75	15900	44944	5625

Linear regression by hand

https://www.statology.org/linear-regression-by-hand/

- Step 2: Calculate ΣX, ΣY, ΣX*Y, ΣX2, and ΣY2

	Weight (lbs)	Height (inches)	X*Y	X^2	Y^2
	140	60	8400	19600	3600
	155	62	9610	24025	3844
	159	67	10653	25281	4489
	179	70	12530	32041	4900
	192	71	13632	36864	5041
	200	72	14400	40000	5184
	212	75	15900	44944	5625
Σ	**1237**	**477**	**85125**	**222755**	**32683**

Linear regression by hand

https://www.statology.org/linear-regression-by-hand/

- Step 3: Calculate b0

The formula to calculate b0 is: [(ΣY)(ΣX2) – (ΣX)(ΣXY)] / [n(ΣX2) – (ΣX)2]

In this example, b0 = [(477)(222755) – (1237)(85125)] / [7(222755) – (1237)2] = 32.783

Linear regression by hand

https://www.statology.org/linear-regression-by-hand/

- Step 4: Calculate b1

The formula to calculate b1 is: $[n(\Sigma XY) – (\Sigma X)(\Sigma Y)] / [n(\Sigma X^2) – (\Sigma X)^2]$

In this example, b1 = $[7(85125) – (1237)(477)] / [7(222755) – (1237)^2] = 0.2001$

Linear regression by hand

https://www.statology.org/linear-regression-by-hand/

- Step 5: Place b0 and b1 in the estimated linear regression equation

The estimated linear regression equation is: $\hat{y} = b0 + b1*x$

In our example, it is $\hat{y} = 0.32783 + (0.2001)*x$

Linear regression by hand

https://www.statology.org/linear-regression-by-hand/

https://www.statology.org/linear-regression-calculator/

- Here is how to interpret this estimated linear regression equation: $\hat{y} = 32.783 + 0.2001x$

- b0 = 32.7830. When weight is zero pounds, the predicted height is 32.783 inches. Sometimes the value for b0 can be useful to know, but in this example it doesn’t actually make sense to interpret b0 since a person can’t weigh zero pounds

  - b1 = 0.2001. A one pound increase in weight is associated with a 0.2001 inch increase in height.

Programming example

https://www.w3schools.com/python/python_ml_linear_regression.asp

- In the example below, the x-axis represents age, and the y-axis represents speed. We have registered the age and speed of 13 cars as they were passing a tollbooth. Let us see if the data we collected could be used in a linear regression:

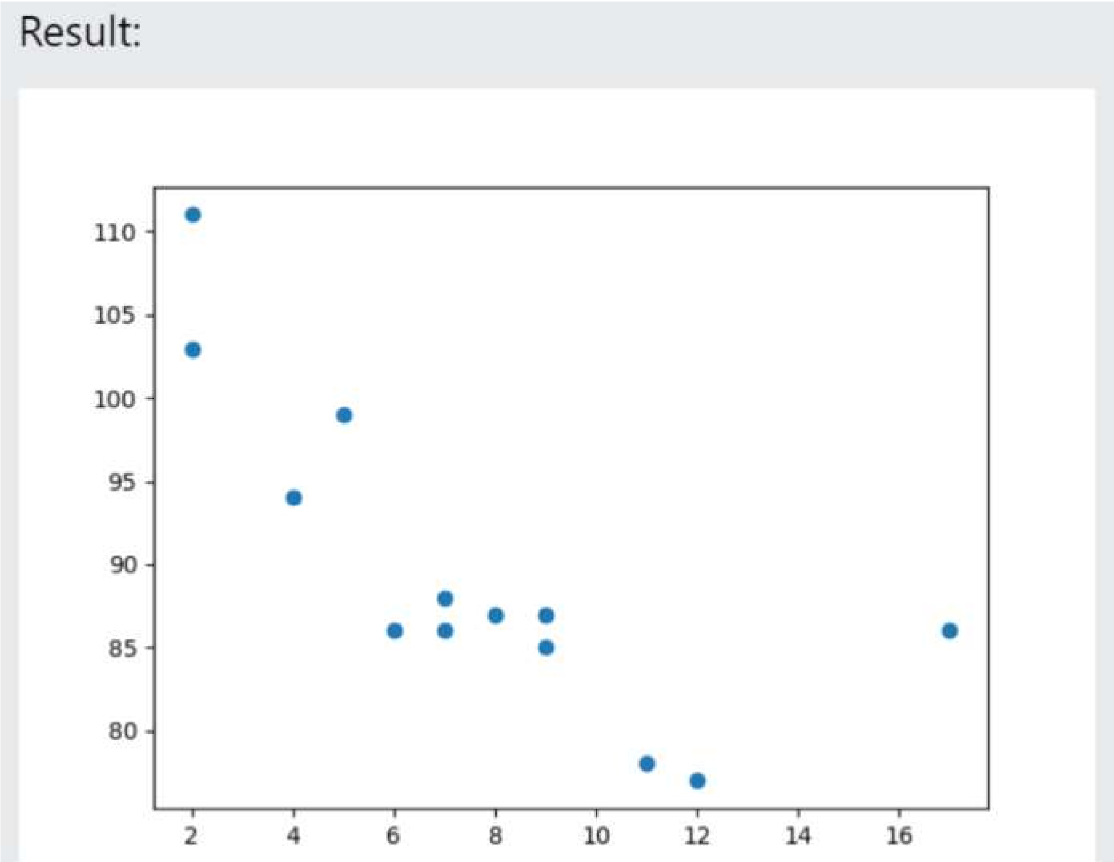
Example

Start by drawing a scatter plot:

```
import matplotlib.pyplot as plt

x = [5,7,8,7,2,17,2,9,4,11,12,9,6]
y = [99,86,87,88,111,86,103,87,94,78,77,85,86]

plt.scatter(x, y)
plt.show()
```

Programming example

https://www.w3schools.com/python/python_ml_linear_regression.asp



Programming example

https://www.w3schools.com/python/python_ml_linear_regression.asp

Example

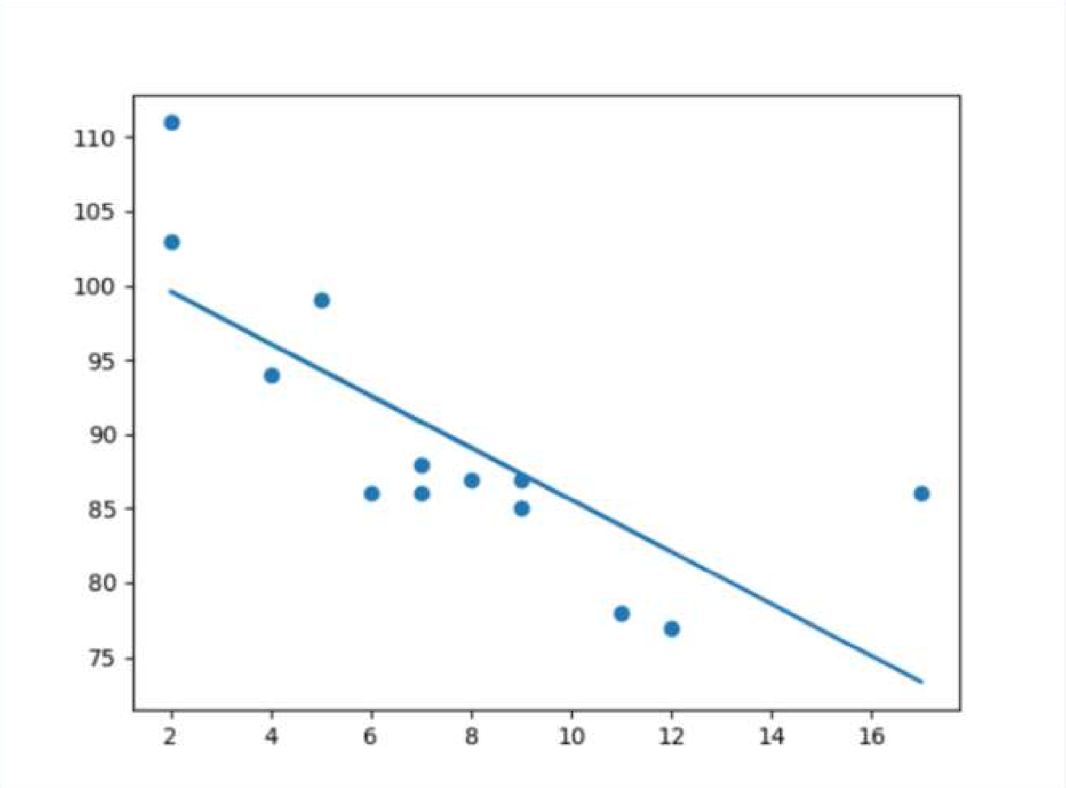
Import `scipy` and draw the line of Linear Regression:

```
import matplotlib.pyplot as plt
from scipy import stats

x = [5,7,8,7,2,17,2,9,4,11,12,9,6]
y = [99,86,87,88,111,86,103,87,94,78,77,85,86]

slope, intercept, r, p, std_err = stats.linregress(x, y)

def myfunc(x):
  return slope * x + intercept

mymodel = list(map(myfunc, x))

plt.scatter(x, y)
plt.plot(x, mymodel)
plt.show()
```

Programming example

https://www.w3schools.com/python/python_ml_linear_regression.asp

Result:



Programming example

https://www.w3schools.com/python/python_ml_linear_regression.asp

Example

How well does my data fit in a linear regression?

```
from scipy import stats

x = [5,7,8,7,2,17,2,9,4,11,12,9,6]
y = [99,86,87,88,111,86,103,87,94,78,77,85,86]

slope, intercept, r, p, std_err = stats.linregress(x, y)

print(r)
```

Programming example

https://www.w3schools.com/python/python_ml_linear_regression.asp

Example

Predict the speed of a 10 years old car:

```
from scipy import stats

x = [5,7,8,7,2,17,2,9,4,11,12,9,6]
y = [99,86,87,88,111,86,103,87,94,78,77,85,86]

slope, intercept, r, p, std_err = stats.linregress(x, y)

def myfunc(x):
  return slope * x + intercept

speed = myfunc(10)

print(speed)
```