Least-Squares Problems and Best-Fit

Consider solving $A\mathbf{x} = \mathbf{b}$.

- If the system is consistent, then $\mathbf{b}$ is in $Col(A)$.
- If the system is inconsistent, then $\mathbf{b}$ is *not* in $Col(A)$.

Let us find $\hat{\mathbf{x}}$ such that $A\hat{\mathbf{x}}$ is the *closest* vector in $Col(A)$ to $\mathbf{b}$; that is, $\|\mathbf{b} - A\hat{\mathbf{x}}\| \leq \|\mathbf{b} - A\mathbf{x}\|$ for all $\mathbf{x}$ in $\mathbb{R}^n$. Such a solution $\hat{\mathbf{x}}$ is called a *least-squares solution* since $\hat{\mathbf{x}}$ minimizes $\|\mathbf{b} - A\mathbf{x}\| = \|\mathbf{z}\| = \sqrt{\sum \mathbf{z}_i^2}$.

Let $\hat{\mathbf{b}} = \text{proj}_{Col(A)}\mathbf{b}$. Then $\hat{\mathbf{b}}$ is the closest vector in $Col(A)$ to $\mathbf{b}$, and $A\hat{\mathbf{x}} = \hat{\mathbf{b}}$ is *consistent.*

1 Least-Squares Solutions. The normal equations

We wish to find *least-squares solutions* $\hat{\mathbf{x}}$ to $A\mathbf{x} = \mathbf{b}$. Such least-squares solutions satisfy $A\hat{\mathbf{x}} = \hat{\mathbf{b}}$, where $\hat{\mathbf{b}} = \text{proj}_{Col(A)}\mathbf{b}$.

By the *Orthogonal Decomposition Theorem*, $\mathbf{z} = \mathbf{b} - \hat{\mathbf{b}}$ is orthogonal to $Col(A)$. Thus, $\mathbf{z} = \mathbf{b} - A\hat{\mathbf{x}}$ is orthogonal to each column $\mathbf{a}_j$ of the matrix A.

$$\mathbf{a}_j \cdot (\mathbf{b} - A\hat{\mathbf{x}}) = 0 \text{ for each } j.$$

$$\mathbf{a}_j^T(\mathbf{b} - A\hat{\mathbf{x}}) = 0 \text{ for each } j.$$

$$A^T(\mathbf{b} - A\hat{\mathbf{x}}) = \mathbf{0}.$$

Rearranging, we have that

$$(A^TA)\hat{\mathbf{x}} = A^T\mathbf{b}.$$

These are called the **normal equations**.

Theorem 1.1 *The set of least-squares solutions of $A\mathbf{x} = \mathbf{b}$ is given by the solutions of the normal equations $(A^TA)\hat{\mathbf{x}} = A^T\mathbf{b}$.*

Unique solution to the Least-Squares Problem

Theorem 1.2 *Let A be an $m \times n$ matrix. The following statements are logically equivalent:*

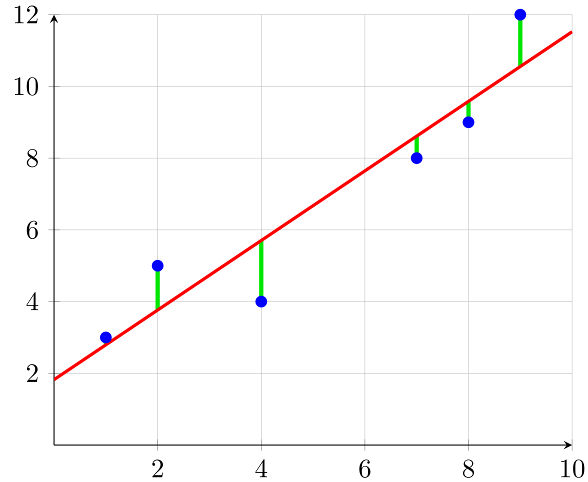
- *The equation $A\mathbf{x} = \mathbf{b}$ has a unique least-squares solution for each $\mathbf{b}$ in $\mathbb{R}^m$.*
- *The columns of A are linearly independent.*
- *The matrix A^TA is invertible.*

When these statements are satisfied, the least-squares solution $\hat{\mathbf{x}}$ is given by

$$\hat{\mathbf{x}} = (A^TA)^{-1}A^T\mathbf{b}.$$

2 Best-Fit Line

Suppose we have n data points $(x_1, y_1), \dots, (x_n, y_n)$. What is the line that *best fits* this data?



We want a line of the form $y = \beta_0 + \beta_1 x$.
Which line *best fits* the data?
We want to minimize the sum of squares

$$\sqrt{\sum_{i=1}^{n} \Big(y_i - (\beta_0 + \beta_1 x_i) \Big)^2}.$$

$y_i - (\beta_0 + \beta_1 x_i)$ is called the *residual*.
We can formulate as a least-squares problem.

We would like to solve:

$$\begin{aligned} y_1 &= \beta_0 + \beta_1 x_1 \\ y_2 &= \beta_0 + \beta_1 x_2 \\ y_3 &= \beta_0 + \beta_1 x_3 \\ &\vdots \\ y_n &= \beta_0 + \beta_1 x_n \end{aligned}$$

As a matrix equation:

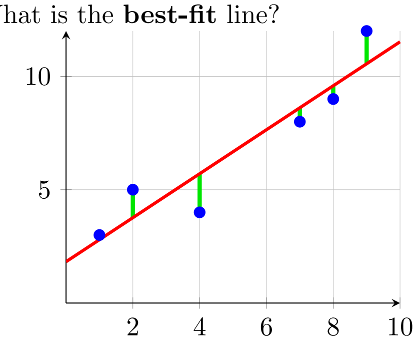
$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}$$

$$\mathbf{y} = A\boldsymbol{\beta}$$

- A is called the *model matrix* (or design matrix).
- $\mathbf{y}$ is the vector of *observed responses*.
- $\boldsymbol{\beta}$ is the vector of *regression coefficients*.

Computing the *least-squares* solution of $\mathbf{y} = A\boldsymbol{\beta}$ is equivalent to finding values for the regression coefficients β_0 and β_1 that minimize the *sum of square residuals*.

Example 1 Suppose we have data $(1,3),(2,5),(4,4),(7,8),(8,9),(9,12)$. What is the **best-fit** line?



We want a least-squares solution of $\mathbf{y} = A\boldsymbol{\beta}$.

$$\begin{bmatrix} 3 \\ 5 \\ 4 \\ 8 \\ 9 \\ 12 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 4 \\ 1 & 7 \\ 1 & 8 \\ 1 & 9 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}.$$

We solve the *normal equations* by computing $\boldsymbol{\beta} = (A^TA)^{-1}A^T\mathbf{y} = \begin{bmatrix} 1.824 \\ 0.970 \end{bmatrix}$.

So $y = 1.824 + 0.970x$ is the *best-fit line*.

We can quantify the *error* by $\mathbf{y} = A\boldsymbol{\beta} + \boldsymbol{\varepsilon} = \hat{\mathbf{y}} + \mathbf{z}$, $\|\mathbf{z}\| \approx 2.699$.

Example 2: Predicting Wealth Based on Literacy Rate

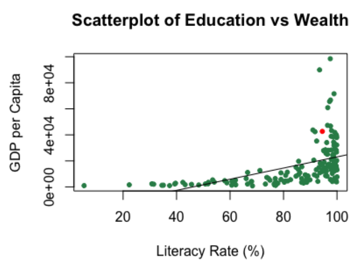
Data scientists often begin with a simple model, and then determine whether predictions increase when new predictors are added. Let's first consider the following potential relationship:

- Let y denote the Gross Domestic Product (GDP) per capita of a country (in thousands of dollars).
- Let x_1 denote the literacy rate of the country's population (as a percentage).
- We collect a dataset that consists of n observations.

- Based on our data, what is the best model of the form $y = \beta_0 + \beta_1 x_1$?

$$\begin{aligned} 2.079 &= \beta_0 + \beta_1(31.4) \\ 13.440 &= \beta_0 + \beta_1(98.1) \\ 11.324 &= \beta_0 + \beta_1(81.4) \\ &\vdots \\ 3.537 &= \beta_0 + \beta_1(88.7) \end{aligned}$$

$$\begin{bmatrix} 2.07 \\ 13.44 \\ 11.324 \\ \vdots \\ 3.537 \end{bmatrix} = \begin{bmatrix} 1 & 31.4 \\ 1 & 98.1 \\ 1 & 81.4 \\ \vdots & \vdots \\ 1 & 88.7 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}$$



Interpreting the Results

Let's imagine that we randomly select four countries record the most recent data for each country's GDP per capita and literacy rate. $\widehat{\text{Wealth}} = -1.938 + 0.126(\text{Education})$

$$A = \begin{bmatrix} 1 & 31 \\ 1 & 98 \\ 1 & 81 \\ 1 & 89 \end{bmatrix} \quad \text{and} \quad \mathbf{y} = \begin{bmatrix} 2 \\ 13 \\ 11 \\ 4 \end{bmatrix}$$

$$A^T A = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 31 & 98 & 81 & 89 \end{bmatrix} \begin{bmatrix} 1 & 31 \\ 1 & 98 \\ 1 & 81 \\ 1 & 89 \end{bmatrix} = \begin{bmatrix} 4 & 299 \\ 299 & 25{,}047 \end{bmatrix}$$

For the least-squares solution to $A\boldsymbol{\beta} = \mathbf{y}$, we solve the normal equations $A^T A\boldsymbol{\beta} = A^T\mathbf{y}$:

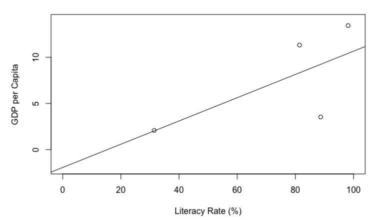
$$\boldsymbol{\beta} = \begin{bmatrix}\beta_0\\ \beta_1\end{bmatrix} = \left(\begin{bmatrix}4 & 299\\ 299 & 25{,}047\end{bmatrix}\right)^{-1}\begin{bmatrix}1 & 1 & 1 & 1\\ 31 & 98 & 81 & 89\end{bmatrix}\begin{bmatrix}2\\ 13\\ 11\\ 4\end{bmatrix} \approx \begin{bmatrix}-1.938\\ 0.126\end{bmatrix}$$

Fitting a Model for Predicting Wealth of a Nation

Let's imagine that we randomly select four countries record the most recent data for each country's GDP per capita and literacy rate.

$\widehat{\text{Wealth}} = -1.938 + 0.126(\text{Education})$

- The literacy rate in the United States in 2021 is approximately 80 %. Based on our model predict the GDP per capita of the US? (Note the actual value is \$69,734.)
- Interpret the practical meaning of the slope and vertical intercept of the linear model.



3 QR decomposition and the Least Squares

Theorem 3.1 *If A is an $(m\times n)$ matrix with linearly independent columns, then A can be factored as $A = QR$, where Q is an $(m\times n)$ matrix whose columns form an orthogonal basis for $ColA$ and R is an $(n\times n)$ upper triangular invertible matrix with positive entries on its diagonal.*

Theorem 3.2 *Given an $(m\times n)$ matrix A with linearly independent columns, let $A = QR$ be a QR decomposition of A. Then, for each $\mathbf{y}$ in $\mathbb{R}^m$, the equation $A\beta = \mathbf{y}$ has a unique least-squares solution, given by $\beta^* = R^{-1}Q^T\mathbf{y}$.*

Example 3: Find the least-squares solution of $A\beta = \mathbf{y}$ for

$$A = \begin{bmatrix}1 & 3 & 5\\ 1 & 1 & 0\\ 1 & 1 & 2\\ 1 & 3 & 3\end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix}3\\ 5\\ 7\\ -3\end{bmatrix}$$

Solution: The QR factorization of A is

$$A = QR = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} 2 & 4 & 5 \\ 0 & 2 & 3 \\ 0 & 0 & 2 \end{bmatrix}$$

Then

$$Q^T\mathbf{y} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} 3 \\ 5 \\ 7 \\ -3 \end{bmatrix} = \begin{bmatrix} 6 \\ -6 \\ 4 \end{bmatrix}$$

The least-squares solution β^* satisfies $R\beta = Q^T\mathbf{y}$ that is,

$$\begin{bmatrix} 2 & 4 & 5 \\ 0 & 2 & 3 \\ 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 6 \\ -6 \\ 4 \end{bmatrix}$$

The solution is

$$\beta^* = \begin{bmatrix} 10 \\ -6 \\ 2 \end{bmatrix}$$

4 Least-Squares via Left Inverse

$$A\boldsymbol{\beta} = \mathbf{y}$$

Let us assume we have n predictors and m observations. Thus, A is an $(m \times n)$ matrix.

If $\mathbf{y} \in \mathrm{Col}(A)$, then:

$$\boldsymbol{\beta} = (A^TA)^{-1}A^T\mathbf{y}.$$

Otherwise, let $\hat{\mathbf{y}} \in \mathrm{Col}(A)$ be the projection of $\mathbf{y}$, then:

$$\boldsymbol{\beta} = (A^TA)^{-1}A^T\hat{\mathbf{y}}.$$

5 Residuals, Sum of Squared Errors (SSE), R^2, and Pearson correlation coefficient ρ. Connections to Statistics

Given a dataset with observed values $\mathbf{y} = (y_1, y_2, \ldots, y_n)^T$ and predicted values $\hat{\mathbf{y}} = (\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n)^T$ obtained from a model, the **residuals** are defined as the differences between the observed and predicted values:

$$r_i = y_i - \hat{y}_i, \quad \text{for } i = 1, 2, \ldots, n.$$

The residual vector is given by:

$$\mathbf{r} = \mathbf{y} - \hat{\mathbf{y}}.$$

The **Sum of Squared Errors (SSE)** is a measure of the total discrepancy between the observed and predicted values. It is defined as:

$$\text{SSE} = \sum_{i=1}^{n} r_i^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2.$$

Alternatively, in matrix form:

$$\text{SSE} = \|\mathbf{r}\|^2 = \|\mathbf{y} - \hat{\mathbf{y}}\|^2.$$

A lower SSE indicates a better fit of the model to the data.

The residuals and Sum of Squared Errors (SSE) provide insight into the goodness of fit of a model. Another key measure is the **coefficient of determination** R^2, which quantifies how well the model explains the variance in the observed data.

To define R^2, we introduce the following quantities:

- The **Total Sum of Squares (TSS)** measures the total variability in the observed data and is given by:

$$\text{TSS} = \sum_{i=1}^{n} (y_i - \bar{y})^2$$

where $\bar{y}$ is the mean of the observed values.

- The **Regression Sum of Squares (ESS)** or **Explained Sum of Squares** measures the variability explained by the model:

$$\text{ESS} = \sum_{i=1}^{n} (\hat{y}_i - \bar{y})^2.$$

These sums are related by:

$$\text{TSS} = \text{ESS} + \text{SSE}.$$

The coefficient of determination is defined as:

$$R^2 = 1 - \frac{\text{SSE}}{\text{TSS}}.$$

This measures the proportion of the total variance in $\mathbf{y}$ that is explained by the model. A value of R^2 close to 1 indicates a strong fit, while a value close to 0 suggests that the model does not explain much of the variance.

For simple linear regression with one predictor, the square of the **Pearson correlation coefficient** ρ between the observed values y and the predicted values $\hat{y}$ is equal to R^2:

$$R^2 = \rho^2.$$

The correlation coefficient is computed as:

$$\rho = \frac{\sum(y_i - \bar{y})(\hat{y}_i - \bar{y})}{\sqrt{\sum(y_i - \bar{y})^2}\sqrt{\sum(\hat{y}_i - \bar{y})^2}}.$$

Since R^2 is the square of ρ, it always lies between 0 and 1, while ρ can range from -1 to 1, indicating the strength and direction of the linear relationship.

While R^2 measures the goodness of fit, it does not indicate whether the observed relationship is statistically significant. To test whether the independent variable (predictor x) significantly explain the variance in dependent variable (y), we use hypothesis testing on the regression coefficients (β_0, β_1) a simple linear regression model:

$$y = \beta_0 + \beta_1 x + \varepsilon.$$

Hypothesis Testing in Regression

Consider a simple linear regression model:

$$y = \beta_0 + \beta_1 x + \varepsilon.$$

We test the null hypothesis:

$$H_0 : \beta_1 = 0 \quad \text{(no linear relationship between } x \text{ and } y\text{)}$$

against the alternative hypothesis:

$$H_a : \beta_1 \neq 0 \quad \text{(significant relationship between } x \text{ and } y\text{).}$$

To test H_0, we compute the "t-statistic" for the estimated coefficient $\hat{\beta}_1$:

$$t = \frac{\hat{\beta}_1}{\mathrm{SE}(\hat{\beta}_1)}$$

where $\mathrm{SE}(\hat{\beta}_1)$ is the standard error of $\hat{\beta}_1$. Under H_0, this follows a "t-distribution" with $n-2$ degrees of freedom.

The "p-value" is the probability of observing a test statistic as extreme as t (or more extreme) under H_0. A small p-value (typically $p < 0.05$) suggests strong evidence against H_0, meaning that the predictor is statistically significant.

Example 4 Consider the data coming from the fake experiment. Assume that $n = 20$ students were surveyed and asked the number of courses they have taken (x) and their general satisfaction with life (y). Relate y and x trying to predict life satisfaction based on the number of the courses taken. Plot the data and predicted values. Compute and plot the residuals. Estimate the sum of squared errors (SSE).

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
x	13	4	12	3	14	13	12	9	11	7	13	11	9	2	5	7	10	0	9	7
y	70	25	54	21	80	68	84	62	57	40	60	64	45	38	51	52	58	21	75	70

Listing 1: Least Squares

```
import numpy as np
import matplotlib.pyplot as plt
# null space
from scipy.linalg import null_space
import sympy as sym

#number of courses
x = [13,4,12,3,14,13,12,9,11,7,13,11,9,2,5,7,10,0,9,7]

# life happiness
y  = [70,25,54,21,80,68,84,62,57,40,60,64,45,38,51,52,58,21,75,70]

# number of students
n= len(x)

plt.figure(figsize=(6,6))
plt.plot(x,y,'ks',markersize=15)
plt.xlabel('Number of courses taken')
plt.ylabel('General life happiness')
plt.xlim([-1,15])
plt.ylim([0,100])
plt.grid()
plt.xticks(range(0,15,2))
plt.savefig('least_square.png',dpi=300)
plt.show()

# Design matrix
A = np.column_stack((x, np.ones(n)))
print(A)

# Least squares method (the normal equation); solve for the coefficients
Y = A.T @ A
beta = np.linalg.inv(Y) @ (A.T @ y)

# predicted data
pred_happiness = A@beta


plt.figure(figsize=(6,6))
```

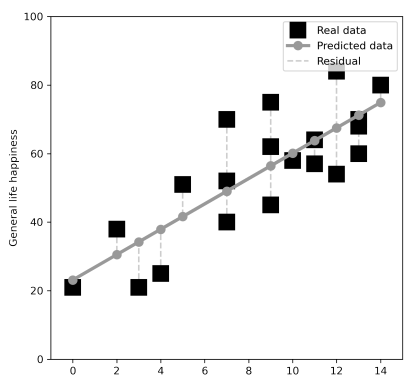
```
# plot the data and predicted values
plt.plot(x,y,'ks',markersize=15)
plt.plot(x,pred_happiness,'o-',color=[.6,.6,.6],linewidth=3,markersize=8)

# plot the residuals (errors)
# zip is a built-in Python function that allows to iterate over multiple iterabl
for k,y,yHat in zip(x,y,pred_happiness):
    plt.plot([k,k],[y,yHat],'--',color=[.8,.8,.8],zorder=-10)

# it can be done with a loop instead of zip
#for i in range(n): # Loop over indices
#    k = x[i] # Extract corresponding elements
#    y_val = y[i]
#    yHat_val = pred_happiness[i]
#    plt.plot([k, k], [y_val, yHat_val], '--', color=[.8, .8, .8], zorder=-10)

# compute SSE
SSE = np.sum((pred_happiness - y) ** 2)
print(SSE)

plt.xlabel('Number of courses taken')
plt.ylabel('General life happiness')
plt.xlim([-1,15])
plt.ylim([0,100])
plt.xticks(range(0,15,2))
plt.legend(['Real data','Predicted data','Residual'])
plt.savefig('Figure_11_04.png',dpi=300)
plt.show()
```



Simple Linear Regression (several ways to look at the same example)

Example 5: Let us consider the following data set

x	1	2	3	4	5	6	7	8	9	10
y	5	10	10	15	14	15	19	18	25	23

We want to analyze this dataset and perform linear regression using different methods. We will:

(a) Apply the Linear Algebra Least Squares Method.

(b) Use QR Decomposition to solve the linear regression problem.

(c) Determine Pearson correlation coefficient ρ, R^2 and p-value.

(d) Implement a scikit-learn Linear Regression model with a train/test split (machine learning).

After completing each method, you will compare the results, evaluate the model's performance, and visualize the data and the regression lines.

Solutions:

(a) Apply the Linear Algebra Least Squares Method.

$$A\vec{\beta} = \vec{y}, \quad \vec{\beta} = (A^T A)^{-1} A^T \vec{y}$$

Listing 2: Linear Algebra Least Squares

```
import pandas as pd
import numpy as np

# Import points
df = pd.read_csv('dogs.csv')
print(df.head())
print(' ')

# extract input variables x
x = df['x'].to_numpy()  # predictor variable
# make x a column
x=x.reshape(-1,1)
# print(x)

# extract output variables y
y=df.values[:,-1]
# make y a column
y=y.reshape(-1,1)
# print(y)

# model matrix A
ones=ones = np.full((len(x),1), 1)
A = np.append(ones, x, axis=1)
```

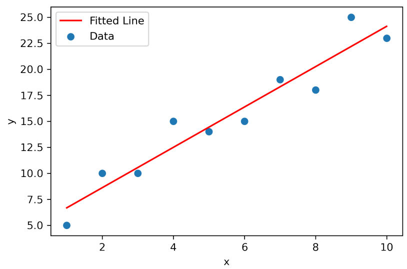
```
print("A=", A)

left = A.transpose() @ A  # compute product A^T A
right = A.transpose() @ y  # compute A^T y
beta = np.linalg.inv(left) @ right  # find solution beta = (A^T A)^(-1) A^T y
# m is the slope and b is the intercept
m=beta[1]
b=beta[0]
print(f"m={m},b={b}")

# Plot data points and fitted line
plt.scatter(x, y, label='Data')
plt.plot(x, m * x + b, color='red', label='Fitted Line')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.show()
```

```
   x   y
0  1   5
1  2  10
2  3  10
3  4  15
4  5  14

A= [[ 1  1]
 [ 1  2]
 [ 1  3]
 [ 1  4]
 [ 1  5]
 [ 1  6]
 [ 1  7]
 [ 1  8]
 [ 1  9]
 [ 1 10]]
m=[1.93939394],b=[4.73333333]
```



(b) Use QR Decomposition to solve the linear regression problem.

$$A\vec{\beta} = \vec{y}, \quad A = QR, \quad \vec{\beta} = R^{-1}Q^T\vec{y}$$

Listing 3: Least Squares using QR decomposition

```
import pandas as pd
import numpy as np
from numpy.linalg import qr,inv

# Import points
df = pd.read_csv('dogs.csv')
print(df.head())
print(' ')

# extract input variables x
x = df['x'].to_numpy()  # predictor variable
# make x a column
x=x.reshape(-1,1)
# print(x)

# extract output variables y
y=df.values[:,-1]
# make y a column
y=y.reshape(-1,1)
# print(y)

# model matrix A
ones=ones = np.full((len(x),1), 1)
A = np.append(ones, x, axis=1)
print(A)

# calculate coefficients using QR decomposition
Q,R=qr(A)
beta=inv(R)@Q.T@y
# m is the slope and b is the intercept
m=beta[1]
```

```
b=beta[0]
print(f"m={m},b={b}")

# Plot data points and fitted line
plt.scatter(x, y, label='Data')
plt.plot(x, m * x + b, color='red', label='Fitted Line')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.show()
```

```
   x   y
0  1   5
1  2  10
2  3  10
3  4  15
4  5  14

[[ 1  1]
 [ 1  2]
 [ 1  3]
 [ 1  4]
 [ 1  5]
 [ 1  6]
 [ 1  7]
 [ 1  8]
 [ 1  9]
 [ 1 10]]
m=[1.93939394],b=[4.73333333]
```

(c) Determine Pearson correlation coefficient ρ, R^2 and p-value.

Listing 4: Least SquaresSimple regressionStatistics

```
import pandas as pd
import matplotlib.pyplot as plt
from scipy.stats import t
from math import sqrt

# Import points
df = pd.read_csv('dogs.csv')
print(df.head())

# Extract input (x) and output (y) variables
x = df.values[:, :-1]
y = df.values[:, -1]
```

```
# Statistics

# compute correlation coefficient
correlations=df.corr(method='pearson')
r=correlations.iloc[0, 1]
print(f"r={r}"), print(' ')

# R-squared
R_squared=r**2
print(f"R_squared={R_squared}"), print(' ')


# Calculating the critical value from a T-distribution
# sample size
n=len(x)
lower_cv=t(n-1).ppf(0.025)
upper_cv=t(n-1).ppf(0.975)

# perform the test
test_value=r/sqrt((1-r**2)/(n-2))
print(f"test_value={test_value}"), print(' ')

if test_value < lower_cv or test_value > upper_cv:
    print("correlation proven, reject H0")
else:
    print("correlation not proven, failed to reject H0")
# calculate p-value
if test_value > 0:
    p_value=1.0-t(n-1).cdf(test_value)
else:
    p_value=t(n-1).cdf(test_value)
# two-tailed, so we multiply by 2
p_value=p_value*2
print(f"p_value={p_value}")
   x   y
0  1   5
1  2  10
2  3  10
3  4  15
4  5  14
r=0.9575860952087218

R_squared=0.9169711297370873

test_value=9.399575927136752
```

```
correlation proven, reject H0
p_value=5.976327099421752e-06
```

(d) Implement a scikit-learn Linear Regression model (machine learning).

Listing 5: Least SquaresSimple with scikit-learn

```python
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

# Import points
df = pd.read_csv('dogs.csv')
print(df.head())

# Extract input (x) and output (y) variables
x = df.values[:, :-1]
y = df.values[:, -1]

# Fit a linear model
fit = LinearRegression().fit(x, y)
m = fit.coef_.flatten()
b = fit.intercept_.flatten()
print(f"m={m}")
print(f"b={b}"),print(' ')

# Compute residuals and sum of squared residuals
ssr = 0
for i, (xdata, ydata) in enumerate(zip(x.flatten(), y)):
    y_model = m * xdata + b
    residual = ydata - y_model
    ssr += residual**2
    print(f"Point {i+1}: x={xdata}, y={ydata}, predicted={y_model}, residual={re

print(f"Sum of squared residuals: {ssr}"), print(' ')

# standard error of estimate (standard deviation)
n=len(x)
SSE=sqrt(ssr/(n-2))
print(f"standard deviation SSE={SSE}")

# Plot data points and fitted line
plt.scatter(x, y, label='Data')
plt.plot(x, m * x + b, color='red', label='Fitted Line')
plt.xlabel('x')
```

```
plt.ylabel('y')
plt.legend()
plt.show()

   x   y
0  1   5
1  2  10
2  3  10
3  4  15
4  5  14
m=[1.93939394]
b=[4.73333333]

Point 1: x=1, y=5, predicted=[6.67272727], residual=[-1.67272727]

Point 2: x=2, y=10, predicted=[8.61212121], residual=[1.38787879]

Point 3: x=3, y=10, predicted=[10.55151515], residual=[-0.55151515]

Point 4: x=4, y=15, predicted=[12.49090909], residual=[2.50909091]

Point 5: x=5, y=14, predicted=[14.43030303], residual=[-0.43030303]

Point 6: x=6, y=15, predicted=[16.36969697], residual=[-1.36969697]

Point 7: x=7, y=19, predicted=[18.30909091], residual=[0.69090909]

Point 8: x=8, y=18, predicted=[20.24848485], residual=[-2.24848485]

Point 9: x=9, y=25, predicted=[22.18787879], residual=[2.81212121]

Point 10: x=10, y=23, predicted=[24.12727273], residual=[-1.12727273]

Sum of squared residuals: [28.0969697]

standard deviation SSE=1.8740654236502023
```

(See the Jupyter Notebook for more on Machine Learning)

6 Multiple Regression

We can include other factors and also fit them.

Wealth	Literacy	Life Exp	Area
2	31	65	653
13	98	79	27
11	81	77	2381
4	89	61	387

$$A = \begin{bmatrix} 1 & 31 & 65 & 653 \\ 1 & 98 & 79 & 27 \\ 1 & 81 & 77 & 2381 \\ 1 & 89 & 61 & 387 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} 2 \\ 13 \\ 11 \\ 4 \end{bmatrix}.$$

We wish to find the regression coefficients $\beta_0, \beta_1, \beta_2$, and β_3 that will give us the best fitting model of the form

$$\text{wealth} = \beta_0 + \beta_1(\text{Literacy}) + \beta_2(\text{LifeExp}) + \beta_3(\text{Area}) + \epsilon.$$

Solving the normal equations, we obtain

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix} = (A^T A)^{-1} A^T \mathbf{y} \approx \begin{bmatrix} -30.458 \\ 0.067 \\ 0.467 \\ 0.00003 \end{bmatrix}$$

$$\widehat{\text{wealth}} = -30.458 + 0.067(\text{Literacy}) + 0.467(\text{LifeExp}) + 0.00003(\text{Area})$$

See the Jupyter Notebook for an example of a code for Multiple regression.

7 Fitting Other Models

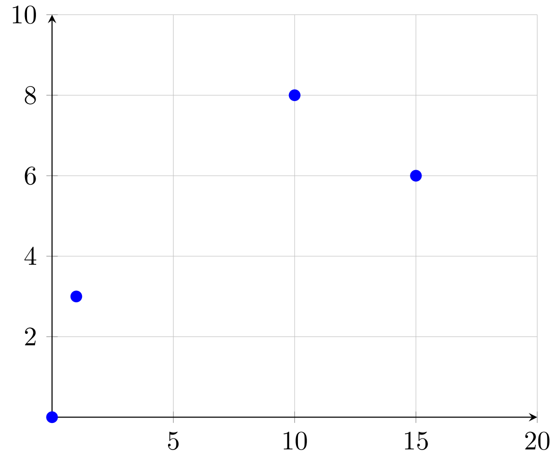
We call any model which is linear in the coefficients β's a *linear model.*

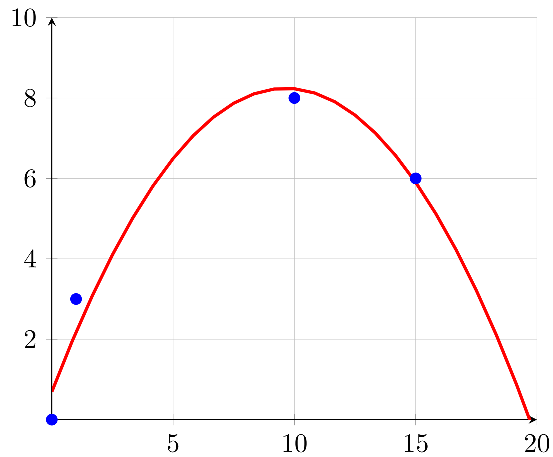
For example:

- $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$ includes two factors.
- $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2$ includes an interaction term.
- $y = \beta_0 + \beta_1 x + \beta_2 x^2$ is a linear model that includes a second-order term.

Fitting a Quadratic Polynomial

Example 7: Consider a ball thrown from $(0, 0)$. The height is measured at the following distances: $(1, 3), (10, 8), (15, 6)$. Where do we predict the ball will hit the ground?





We use a model of $y = \beta_0 + \beta_1 x + \beta_2 x^2$.

$$\begin{bmatrix} 0 \\ 3 \\ 8 \\ 6 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0^2 \\ 1 & 1 & 1^2 \\ 1 & 10 & 10^2 \\ 1 & 15 & 15^2 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix}.$$

We solve the *normal equations* by computing

$\boldsymbol{\beta} = (A^T A)^{-1} A^T \mathbf{y} = \begin{bmatrix} 0.691 \\ 1.567 \\ -0.0813 \end{bmatrix}$.

So $y = 0.691 + 1.567x - 0.0813x^2$ is the best fit.
Using the quadratic formula, height $= 0$ at 19.703.

8 Some more least squares examples

Example 8: Identify the parameters in the Newton cooling ($u(t) = temperature$) or the Price decay ($p(t) = price$) models:

$$\frac{du}{dt} = c(u_{sur} - u)$$

$$\frac{dp}{dt} = c(p_{min} - p)$$

Notice that for this model

$$u(t) = u_{min} + (u_0 - u_{min})e^{-ct}$$

where u_{min} is the long-term minimum temperature, u_0 is the initial temperature, and c is a decay constant. or (for the price decay model)

$$p(t) = p_{min} + (p_0 - p_{min})e^{-ct}$$

where p_{min} is the long-term minimum price, p_0 is the initial price, and c is a decay constant.

A discrete approximation of the Price model is

$$\frac{p_{k+1} - p_{k-1}}{2\Delta t} = (cp_{min}) + (-c)p_k, \quad p_k \approx p(k\Delta t)$$

The unknown parameters are (cp_{min}) and $(-c)$.

If there are six data measurements for the past price, then we can the least squares problem $A\beta = \mathbf{y}$, where A will be (4×2) matrix

$$A = \begin{bmatrix} p_2 & 1 \\ p_3 & 1 \\ p_4 & 1 \\ p_5 & 1 \end{bmatrix}.$$

and

$$\beta = \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix} = \begin{bmatrix} -c \\ cp_{min} \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} \frac{p_3-p_1}{2\Delta t} \\ \frac{p_4-p_2}{2\Delta t} \\ \frac{p_5-p_3}{2\Delta t} \\ \frac{p_6-p_4}{2\Delta t} \end{bmatrix}$$

The least squares problem is

$$\begin{bmatrix} p_2 & 1 \\ p_3 & 1 \\ p_4 & 1 \\ p_5 & 1 \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix} = \begin{bmatrix} \frac{p_3-p_1}{2\Delta t} \\ \frac{p_4-p_2}{2\Delta t} \\ \frac{p_5-p_3}{2\Delta t} \\ \frac{p_6-p_4}{2\Delta t} \end{bmatrix}$$

When this is solved, one can compute c and p_{min}. The solution of the continuous price model has these values and an exponential function of time. Now predicted prices for future times can be done.

Let us now consider a numerical example.

Example 8a:

A company is tracking the price of a commodity over time. The recorded (in some units) prices at specific time points are given by Price=[2080, 2000, 1950,1910,1875,1855]. Assuming that the price follows the model described above, estimate the decay ate c and the minimum price p_{min}. Using the estimated parameters compute predicted prices for times from 0 to 15 and plot the given price data along with the predicted curve. Predict the price at $t = 8$. Compute the residual and its norm.

Solution:

Listing 6: Python Least Squares

```
import numpy as np
import matplotlib.pyplot as plt

# Data
price = np.array([2080, 2000, 1950, 1910, 1875, 1855])
time = np.arange(0, 16)

# Compute y
y = []
for i in range(1, 5):
    y.append((price[i + 1] - price[i - 1]) / 2)
y = np.array(y)

# Matrix A
A = np.column_stack((price[1:5], np.ones(4)))

# Least squares method (the normal equation)
Y = A.T @ A
beta = np.linalg.inv(Y) @ (A.T @ y)

# Extract parameters
c = -beta[0]
pmin = beta[1] / (-beta[0])

# Predicted future prices
future_price = pmin + (2080 - pmin) * np.exp(-c * time)

# Plotting
plt.plot(time[:6], price, '*', label="Price Data")
plt.plot(time, future_price, label="Predicted Price Curve")
plt.title('Price Data and Predicted Price Curve')
plt.xlabel('t')
```

```
plt.ylabel('Price')
plt.legend()
plt.savefig("price.png")
plt.show()

# Predicted price at t = 8
predicted_price = future_price[8]

# Compute the norm of the Residual Vector
r = price[:6] - future_price[:6]
norm_r = np.linalg.norm(r)

# Output results
print("Predicted price at t=8:", future_price[8])
print("Residual vector r:", r)
print("Norm of r:", norm_r)
```

```
Predicted price at t=8: 1812.5554359691694
Residual vector r: [ 0.         -5.02380325  0.9662229   2.77793334 -0.99830682
 2.31868184]
Norm of r: 6.345234544175826
```