# Linear classification

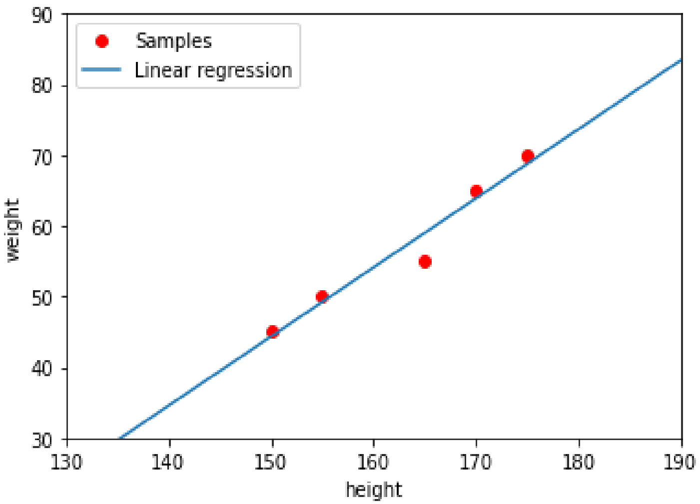
# Review: Linear regression

## Python codes

- Visualizing codes

a = 0.976
b = −102.209

$$133275a + 815b = 46875 \quad \text{Eq (3)}$$
$$815a + 5b = 285 \quad \text{Eq (4)}$$




```
from matplotlib import pyplot as plt
plt.plot([170,155,150,175,165],[65,50,45,70,55],'ro')
plt.plot([0,190],[-102.209,83.421])
plt.show()
```

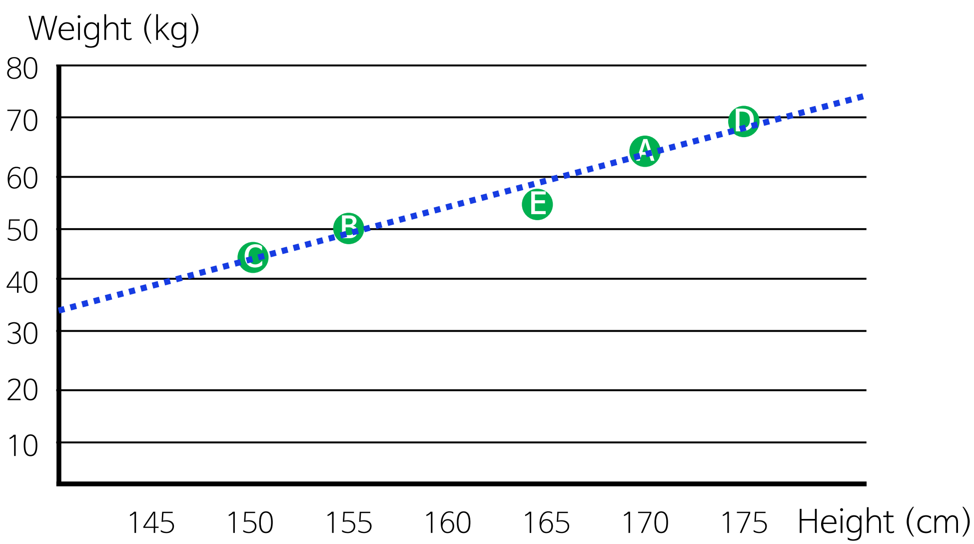
# Linear regression

## Another way to perform least square method

- Utilizing Pseudo inverse

Training dataset : 5 people's heights, weights

| index | height (cm) | weight (kg) |
|---|---|---|
| A | 170 | 65 |
| B | 155 | 50 |
| C | 150 | 45 |
| D | 175 | 70 |
| E | 165 | 55 |




$$y_1 = ax_1 + b$$
$$y_2 = ax_2 + b$$
$$y_3 = ax_3 + b$$
$$y_4 = ax_4 + b$$
$$y_5 = ax_5 + b$$

$$\longrightarrow \quad \mathbf{Y} = \mathbf{Ax}$$

# Linear regression

## Another way to perform least square method

- Utilizing Pseudo inverse

$$y_1 = ax_1 + b$$
$$y_2 = ax_2 + b$$
$$y_3 = ax_3 + b$$
$$y_4 = ax_4 + b$$
$$y_5 = ax_5 + b$$

$$\longrightarrow$$

$$\mathbf{Y} = \mathbf{A}\mathbf{x}$$
$$\mathbf{A}\mathbf{x} = \mathbf{Y}$$

where $\mathbf{Y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_5 \end{pmatrix}$, $\mathbf{A} = \begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_5 & 1 \end{pmatrix}$, $\mathbf{x} = \begin{pmatrix} a \\ b \end{pmatrix}$

$$\mathbf{A}^{-1}\mathbf{A}\mathbf{x} = \mathbf{A}^{-1}\mathbf{Y}$$
$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{Y}$$

Is it possible?

No! $\mathbf{A}$ is not invertible.

not a square matrix

# Linear regression

## Another way to perform least square method

- Then, how about this?

$$\mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{x} = \mathbf{A}^{\mathrm{T}}\mathbf{Y}$$

$$\mathbf{A}^{\mathrm{T}} = \begin{pmatrix} x_1 & x_2 & \cdots & x_5 \\ 1 & 1 & \cdots & 1 \end{pmatrix}, \qquad \mathbf{A} = \begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_5 & 1 \end{pmatrix},$$

$$\mathbf{A}^{\mathrm{T}}\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

$\mathbf{A}^{\mathrm{T}}\mathbf{A}$ is a square matrix

If $\det(\mathbf{A}^{\mathrm{T}}\mathbf{A}) \neq 0$

We can get an inverse matrix $(\mathbf{A}^{\mathrm{T}}\mathbf{A})^{-1}$

which is a Pseudo inverse of $\mathbf{A}$

# Linear regression

## Another way to perform least square method

- Then, how about this?

$$\mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{x} = \mathbf{A}^{\mathrm{T}}\mathbf{Y}$$

$$\left(\mathbf{A}^{\mathrm{T}}\mathbf{A}\right)^{-1}(\mathbf{A}^{\mathrm{T}}\mathbf{A})\mathbf{x} = \left(\mathbf{A}^{\mathrm{T}}\mathbf{A}\right)^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{Y}$$

$$\mathbf{x} = \left(\mathbf{A}^{\mathrm{T}}\mathbf{A}\right)^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{Y}$$

$$\binom{a}{b} = \left(\mathbf{A}^{\mathrm{T}}\mathbf{A}\right)^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{Y}$$

# Linear regression

## Another way to perform least square method

- Proof.

$$y_1 = ax_1 + b$$
$$y_2 = ax_2 + b$$
$$y_3 = ax_3 + b$$
$$\vdots$$
$$y_n = ax_n + b$$

Residual

$$\mathrm{R} = \sum_{i=1}^{n} [y_i - ax_i - b]^2$$

$$\frac{\partial \mathrm{R}}{\partial a} = 0, \qquad \frac{\partial \mathrm{R}}{\partial b} = 0$$

# Linear regression

## Another way to perform least square method

- Proof.

## Another way to perform least square method

- Python Programming.

In [17]:
```
import numpy as np


A = [[170, 1],
     [155, 1],
     [150, 1],
     [175, 1],
     [165, 1]]

Y = [[65], [50], [45], [70], [55]]
```

In [23]:
```
At = np.transpose(A)

print(At)
```

```
[[170 155 150 175 165]
 [  1   1   1   1   1]]
```

# Linear regression

## Another way to perform least square method

- Python Programming.

In [24]:
```
AtA = np.dot(At,A)
AtY = np.dot(At,Y)

print(AtA)
print(AtY)
```

```
[[133275    815]
 [   815      5]]
[[46875]
 [  285]]
```

In [25]:
```
inv_AtA = np.linalg.inv(AtA)
X = np.dot(inv_AtA,AtY)
```

In [26]:
```
print(X)
```

```
[[   0.97674419]
 [-102.20930233]]
```

# Linear regression

## Limitations

- It's simple, but the **computational complexity grows exponentially according to the dimension of data features**

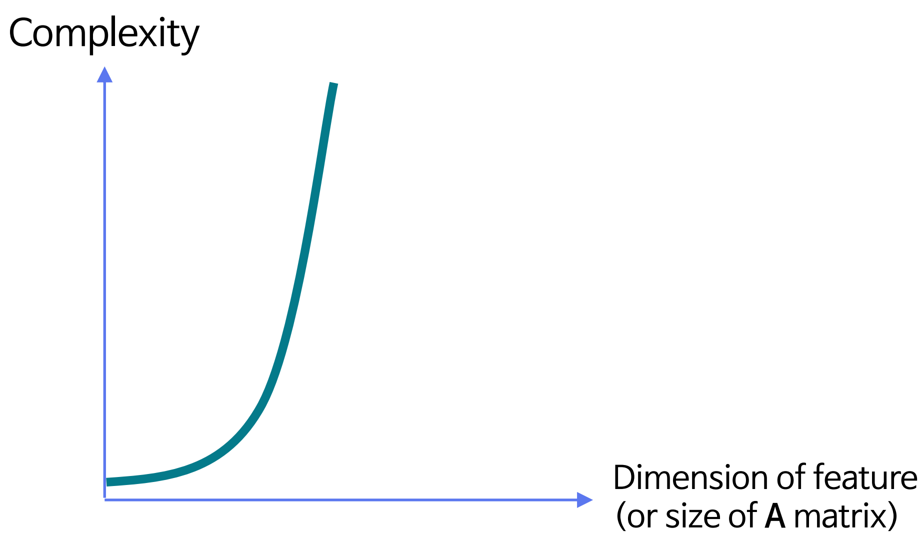
$$\mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{x} = \mathbf{A}^{\mathrm{T}}\mathbf{Y}$$

$$\left(\mathbf{A}^{\mathrm{T}}\mathbf{A}\right)^{-1}(\mathbf{A}^{\mathrm{T}}\mathbf{A})\mathbf{x} = \left(\mathbf{A}^{\mathrm{T}}\mathbf{A}\right)^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{Y}$$

$$\mathbf{x} = \left(\mathbf{A}^{\mathrm{T}}\mathbf{A}\right)^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{Y}$$

$$\begin{pmatrix} a \\ b \end{pmatrix} = \left(\mathbf{A}^{\mathrm{T}}\mathbf{A}\right)^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{Y}$$

Can you calculate the inverse matrix of 10000 x 10000 matrix?



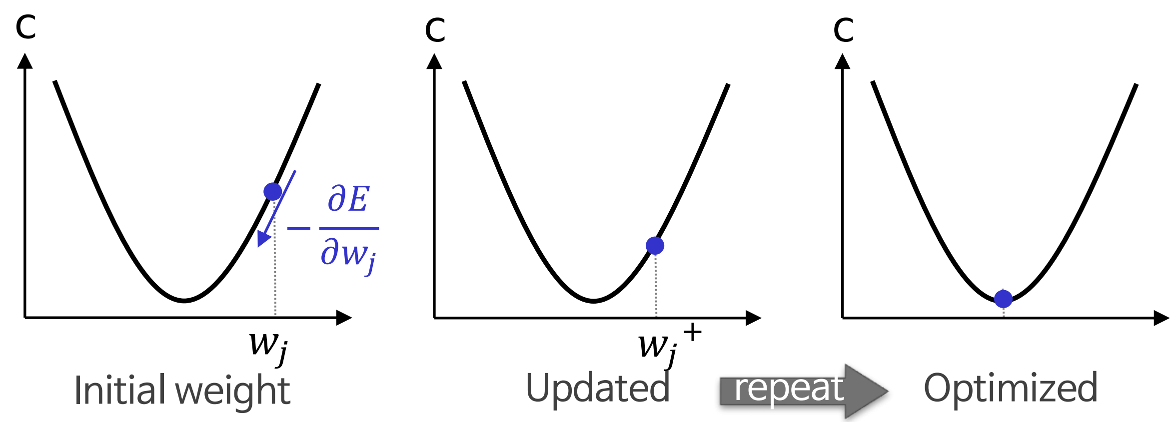
# Linear regression

## Limitations

- Possible solution : gradient descent

*Cost function (= residual)*

$$C = \frac{1}{2}\sum_{n=1}^{N}(t_n - y)^2$$




- Parameters update

Chain rule

$$\frac{\partial C}{\partial w_j} = \frac{\partial C}{\partial y}\frac{\partial y}{\partial z}\frac{\partial z}{\partial w_j} = -\sum_{n=1}^{N}(t_n - y)\; y(1-y)\; x_j$$

$$w_j{}^+ = w_j - \mu\frac{\partial C}{\partial w_j} = w_j + \mu\sum_{n=1}^{N}(t_n - y)y(1-y)x_j$$

$$C = \frac{1}{2}\sum_{n=1}^{N}(t_n - y)^2$$

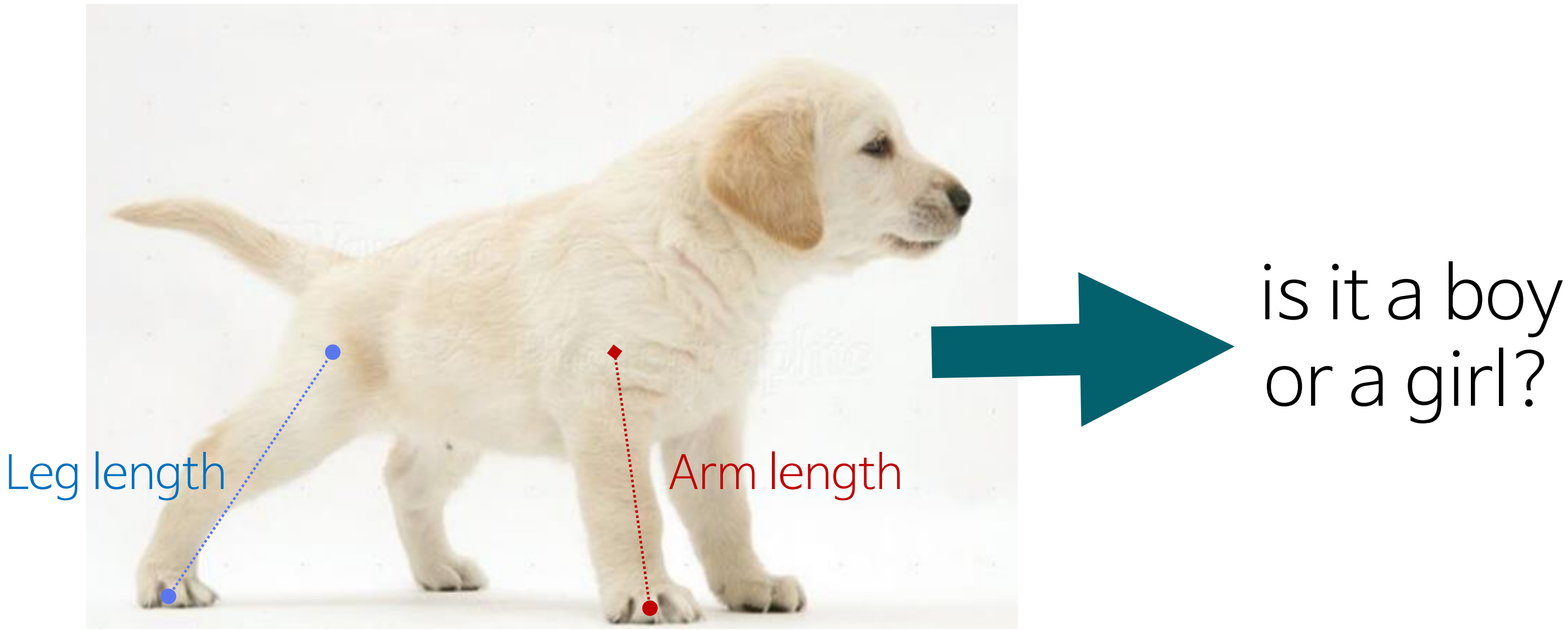
$$y = sigmoid(z)$$

$$z = \sum_{i}^{m} w_i x_i + b$$

# Linear classification

## What is the classification?

- A goal of classification is to use an data’s characteristics to identify which class the data belongs to
- Example :
  Each sample has 2-dimensional features : length of arm (x), length of leg (y)
  Train a linear classifier that separates two classes



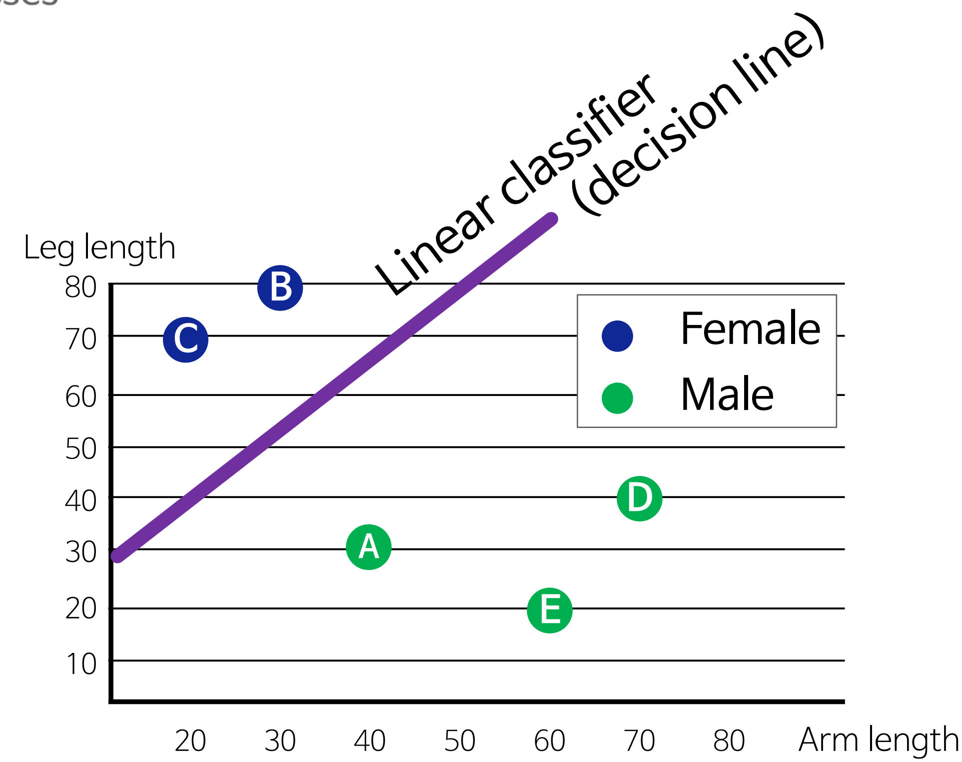
# Linear classification

## Training a linear classification model

- Example :
  Each sample has 2-dimensional features : length of arm (x), length of leg (y)
  Train a linear classifier that separates two classes

Training dataset : 5 dog's lengths of legs, arms

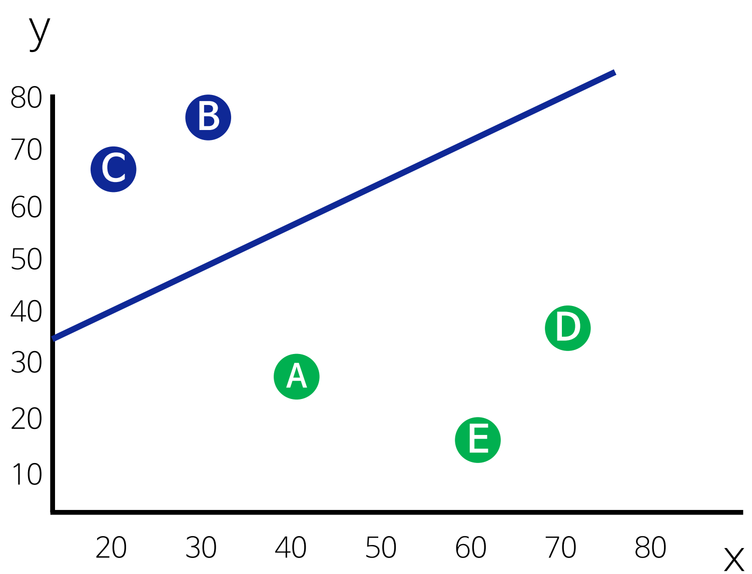
| index | Arm length | Leg length | Gender |
|---|---|---|---|
| A | 40 | 30 | M |
| B | 30 | 80 | F |
| C | 20 | 70 | F |
| D | 70 | 40 | M |
| E | 60 | 20 | M |



# Linear classification

## Training a linear classification model

- What should we train?
  : Linear function that separates positive samples and negative samples
  **Parameters: slope (a), y- intercept (b)**




A linear classification model : line

$$y = ax + b$$

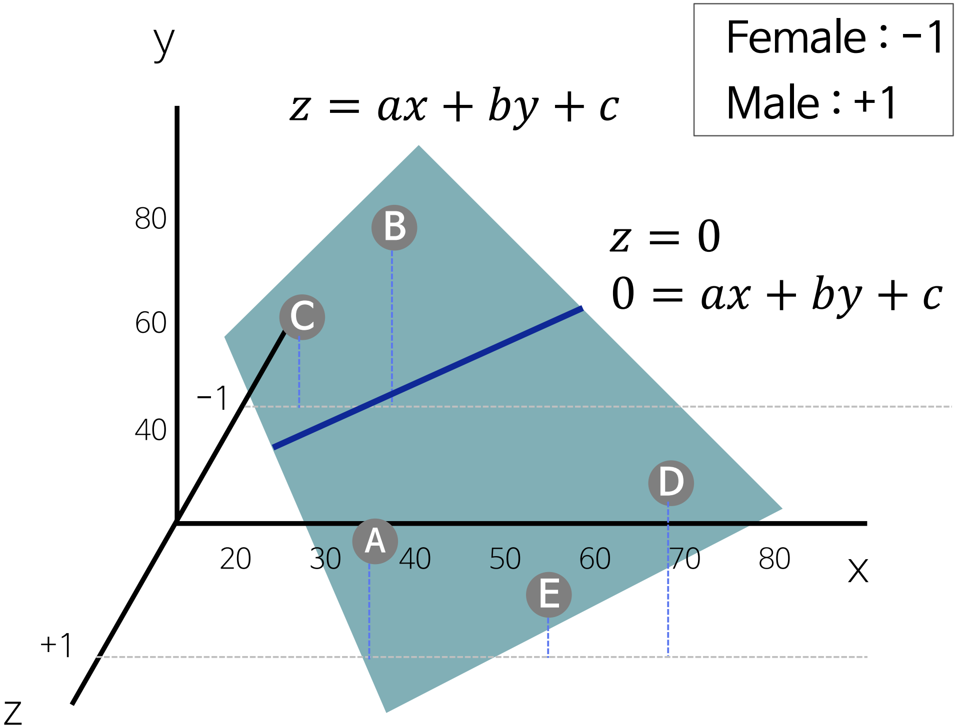
- $a$: slope of a line
- $b$: y intercept

The parameters of
linear classification model
$\theta: a, b$

# Linear classification

## Training a linear classification model

- How can we train a linear model?
  : Use Least square method but consider one more dimension for the data class (−1, +1)




A linear model : plane

$z = ax + by + c$

- $a, b, c$
  parameters for a plane

The parameters of
linear classification model
$\theta: a, b, c$

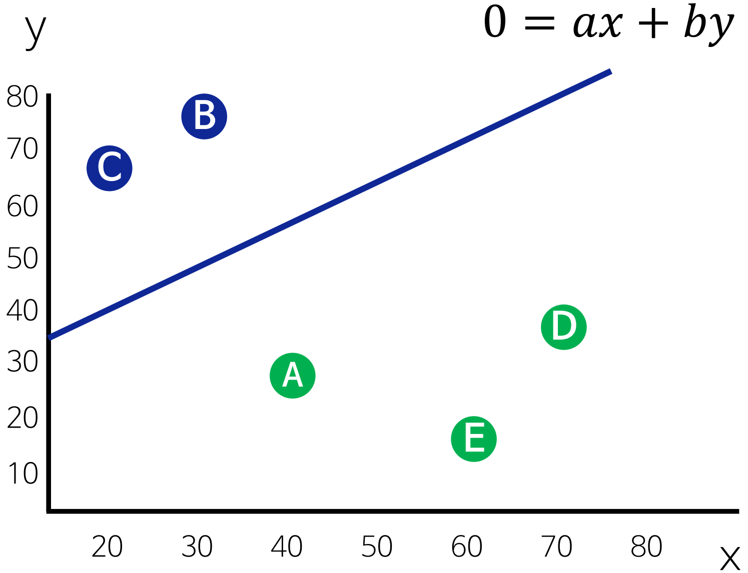
# Linear classification

## Training a linear classification model

- How can we train a linear model?
  : Use Least square method but **consider one more dimension for the data class (−1, +1)**

Find a decision line

$$0 = ax + by + c$$




A linear classification model : line

$$y = ax + b$$

- $a$: slope of a line
- $b$: y intercept

The parameters of linear classification model $\theta: a, b$

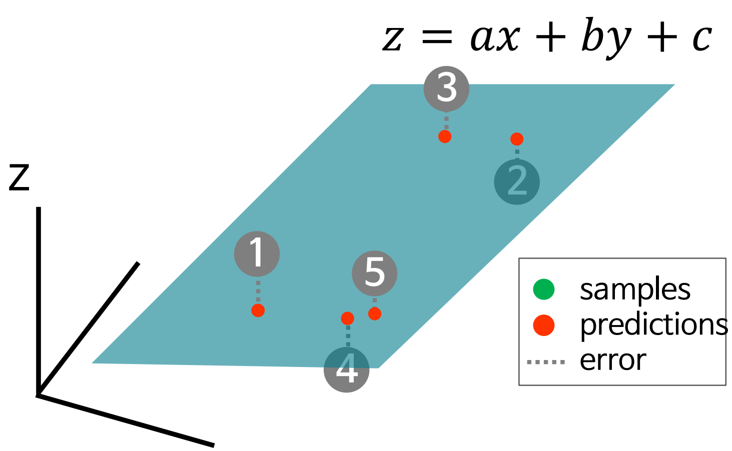
# Linear classification

## Training a linear classification model

- Least square method
  : A statistical procedure **to find the best fit for a set of data** by minimizing **the sum of the offsets** or residuals

$$z = ax + by + c$$

| index $(i)$ | arm $(x_i)$ | leg $(y_i)$ | gender $(z_i)$ | Prediction $(\overline{z_i})$ | error $(\overline{z_i} - z_i)$ |
|---|---|---|---|---|---|
| 1 | 40 | 30 | +1 | 40a+30b+c | (40a+30b+c) − (+1) |
| 2 | 30 | 80 | −1 | 30a+80b+c | (30a+80b+c) − (−1) |
| 3 | 20 | 70 | −1 | 20a+70b+c | (20a+70b+c) − (−1) |
| 4 | 70 | 40 | +1 | 70a+40b+c | (70a+40b+c) − (+1) |
| 5 | 60 | 20 | +1 | 60a+20b+c | (60a+20b+c) − (+1) |




$$\text{residual, R} = \sum_{i=1}^{5} (\overline{z_i} - z_i)^2$$

# Linear classification

## Training a linear classification model

- Least square method
  : A statistical procedure **to find the best fit for a set of data** by minimizing **the sum of the offsets** or residuals

$$z = ax + by + c$$

$$\mathrm{R} = f(a,b,c) = \sum_{i=1}^{5} (\overline{z_i} - z_i)^2 = (40a + 30b + c - 1)^2 + (30a + 80b + c + 1)^2 + (20a + 70b + c + 1)^2 + (70a + 40b + c - 1)^2 + (60a + 20b + c - 1)^2$$

$$\frac{\partial \mathrm{R}}{\partial a} = 0 \longrightarrow \square a + \square b + \square c = \square$$

$$\frac{\partial \mathrm{R}}{\partial b} = 0 \longrightarrow \square a + \square b + \square c = \square$$

$$\frac{\partial \mathrm{R}}{\partial c} = 0 \longrightarrow \square a + \square b + \square c = \square$$

Solving the system of equations and find parameters a,b,c

# Linear classification

## Training a linear classification model

- Detailed solving process

$$\mathrm{R} = (40a + 30b + c - 1)^2 + (30a + 80b + c + 1)^2 + (20a + 70b + c + 1)^2 + (70a + 40b + c - 1)^2 + (60a + 20b + c - 1)^2$$

$$\frac{\partial R}{\partial a} = 2 \cdot 40(40a + 30b + c - 1) + 2 \cdot 30(30a + 80b + c + 1) + 2 \cdot 20(20a + 70b + c + 1) + \\ 2 \cdot 70(70a + 40b + c - 1) + 2 \cdot 60(60a + 20b + c - 1) = 0$$

$$\frac{\partial R}{\partial b} = 2 \cdot 30(40a + 30b + c - 1) + 2 \cdot 80(30a + 80b + c + 1) + 2 \cdot 70(20a + 70b + c + 1) + \\ 2 \cdot 40(70a + 40b + c - 1) + 2 \cdot 20(60a + 20b + c - 1) = 0$$

$$\frac{\partial R}{\partial c} = 2(40a + 30b + c - 1) + 2(30a + 80b + c + 1) + 2(20a + 70b + c + 1) + \\ 2(70a + 40b + c - 1) + 2(60a + 20b + c - 1) = 0$$

# Linear classification

## Training a linear classification model

- Python programming.

```
# training data
arm=[40, 30, 20, 70, 60]
leg=[30, 80, 70, 40, 20]
gen=[1, -1, -1, 1, 1]
print(arm,leg,gen)
```

```
[40, 30, 20, 70, 60] [30, 80, 70, 40, 20] [1, -1, -1, 1, 1]
```

```
# differential

import sympy as sym
a=sym.Symbol('a')
b=sym.Symbol('b')
c=sym.Symbol('c')
R=0
```

# Linear classification

## Training a linear classification model

- Python programming.

```
# Residual
for i in range(0,5):
    R+=(arm[i]*a+leg[i]*b+c-gen[i])**2

R_a=sym.diff(R,a)
R_b=sym.diff(R,b)
R_c=sym.diff(R,c)
print("Result of differentiating with respect to a:", R_a, "= 0")
print("Result of differentiating with respect to b:", R_b, "= 0")
print("Result of differentiating with respect to c:", R_c, "= 0")
```

```
Result of differentiating with respect to a: 22800*a + 18000*b + 440*c - 240 = 0
Result of differentiating with respect to b: 18000*a + 28400*b + 480*c + 120 = 0
Result of differentiating with respect to c: 440*a + 480*b + 10*c - 2 = 0
```

# Linear classification

## Training a linear classification model

- Python programming.

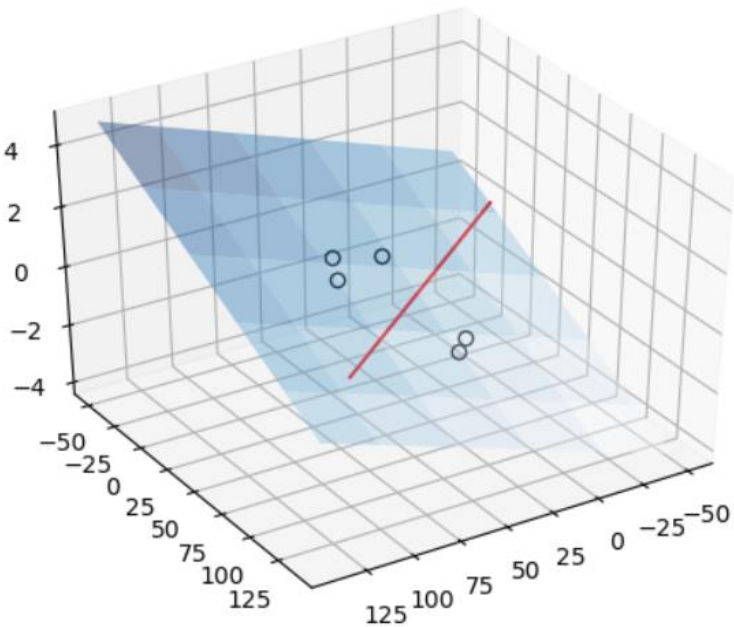
```
# Solve a,b,c using matrix operations

import numpy as np
A=[[22800,18000,440],[18000,28400,480],[440,480,10]]
B=[240,-120,2]
inv_A=np.linalg.inv(A)
result=np.dot(inv_A,B)
print("a=",result[0]," b=",result[1]," c=",result[2])
a=result[0]
b=result[1]
c=result[2]
```

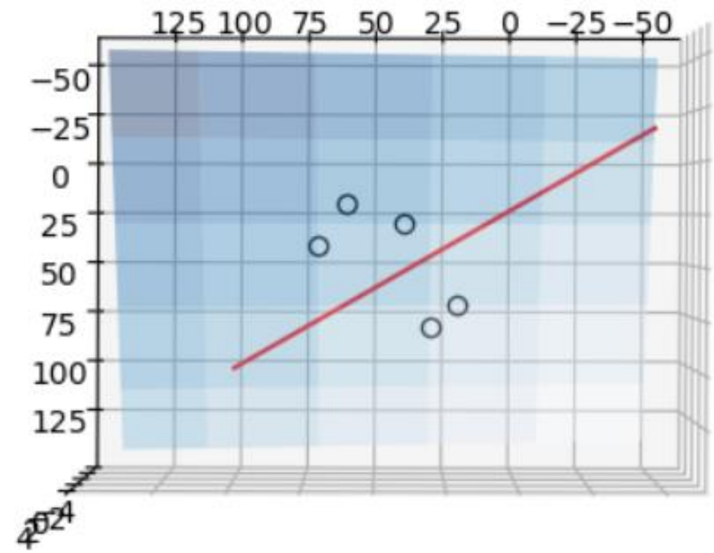
```
a= 0.016176470588235244  b= -0.030882352941176514  c= 0.9705882352941213
```

# Linear classification

## Training a linear classification model

- Python programming.



# Linear classification

## Another way to perform least square method

- Utilizing Pseudo inverse

$$z_1 = ax_1 + by_1 + c$$
$$z_2 = ax_2 + by_2 + c$$
$$z_3 = ax_3 + by_3 + c$$
$$z_4 = ax_4 + by_4 + c$$
$$z_5 = ax_5 + by_5 + c$$

$$\longrightarrow \quad \mathbf{Ax} = \mathbf{Z}$$

where $\mathbf{Z} = \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_5 \end{pmatrix}$, $\quad \mathbf{A} = \begin{pmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ \vdots & \vdots & \vdots \\ x_5 & y_5 & 1 \end{pmatrix}$, $\quad \mathbf{x} = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$

# Linear classification

## Another way to perform least square method

- Utilizing Pseudo inverse

$$\mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{x} = \mathbf{A}^{\mathrm{T}}\mathbf{Z}$$

$$\left(\mathbf{A}^{\mathrm{T}}\mathbf{A}\right)^{-1}(\mathbf{A}^{\mathrm{T}}\mathbf{A})\mathbf{x} = \left(\mathbf{A}^{\mathrm{T}}\mathbf{A}\right)^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{Z}$$

$$\mathbf{A}^{\mathrm{T}}\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

$$\mathbf{x} = \left(\mathbf{A}^{\mathrm{T}}\mathbf{A}\right)^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{Z}$$

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} = \left(\mathbf{A}^{\mathrm{T}}\mathbf{A}\right)^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{Y}$$

# Linear classification

## Training a linear classification model

- Python programming.

```
# train a linear regrssion model using Pseudo inverse of A

import numpy as np

A = [[40, 30, 1],
     [30, 80, 1],
     [20, 70, 1],
     [70, 40, 1],
     [60, 20, 1]]

Z = [[1], [-1], [-1], [1], [1]]
```

```
At = np.transpose(A)

print(At)
```

```
[[40 30 20 70 60]
 [30 80 70 40 20]
 [ 1  1  1  1  1]]
```

## Training a linear classification model

- Python programming.

```
AtA = np.dot(At,A)
AtZ = np.dot(At,Z)

print(AtA)
print(AtZ)
```

```
[[11400  9000   220]
 [ 9000 14200   240]
 [  220   240     5]]
[[120]
 [-60]
 [  1]]
```

```
inv_AtA = np.linalg.inv(AtA)
X = np.dot(inv_AtA,AtZ)
```

```
print(X)
```

```
[[ 0.01617647]
 [-0.03088235]
 [ 0.97058824]]
```

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} = \left(\mathbf{A}^{\mathrm{T}}\mathbf{A}\right)^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{Y}$$