IF3170 Artificial Intelligence

Regression

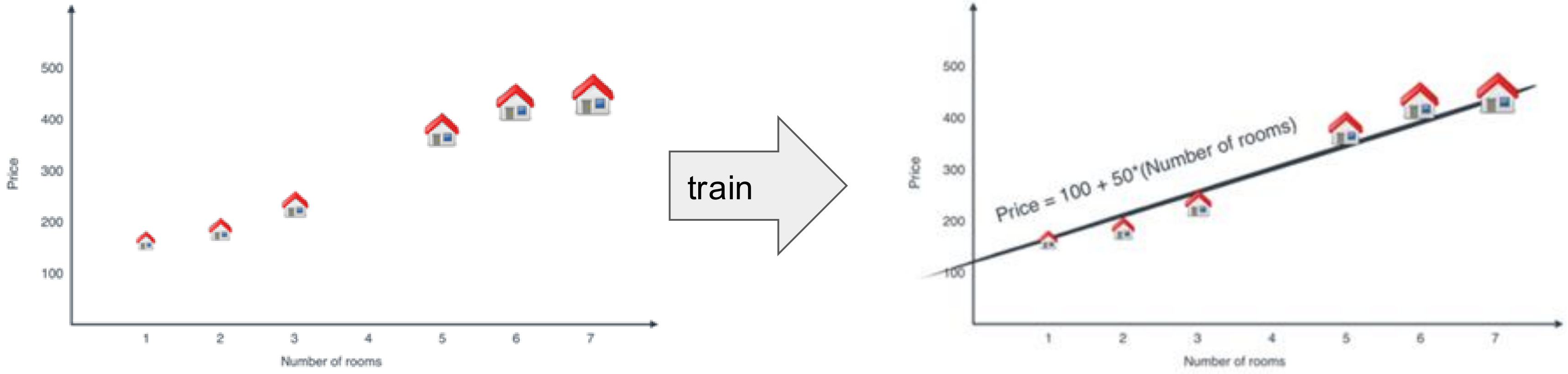
Masayu Leylia Khodra

Capaian Pembelajaran

- Mahasiswa dapat menjelaskan dan menerapkan penggunaan konsep supervised learning
- Mahasiswa dapat menjelaskan dan menerapkan teknik linear regression untuk persoalan pembelajaran mesin yang sesuai.

Regression: Introduction

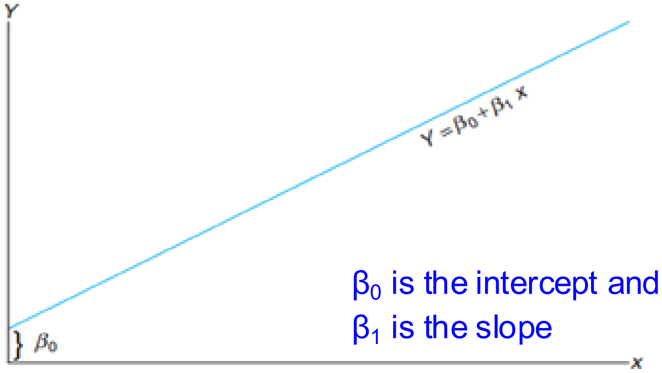
As an example, we are a real estate agent, and in charge of selling a new house. We don’t know the price, and we want to infer it by comparing it with other houses. We look at features of the house which could influence the house, i.e. number of rooms. At the end of the day, what we want is a formula on the feature which gives us the price of the house, or at least an estimate for it.



https://livebook.manning.com/book/grokking-machine-learning/3-1-the-problem-we-need-to-predict-the-price-of-a-house/v-4/44

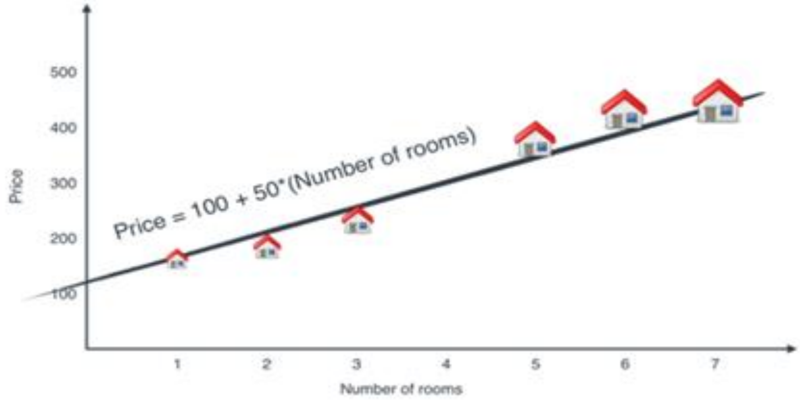
Linear Relationship

- The price of the house is dependent variable or response, and number of rooms is independent variable or regressor or predictor.
- A reasonable form of a relationship between the response Y and the regressor x is the linear relationship: $Y = \beta_0 + \beta_1 x$



β_0 is the intercept and
β_1 is the slope

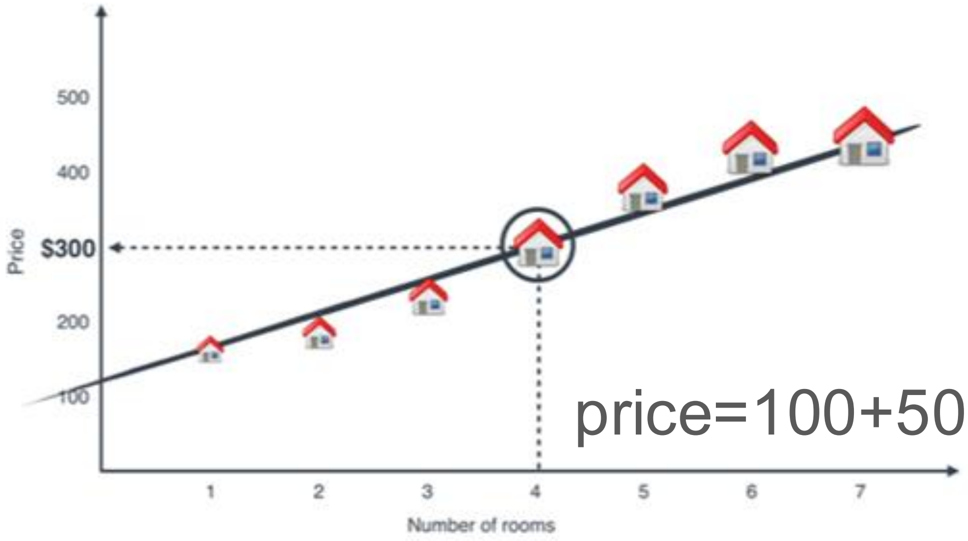
Figure 11.1: A linear relationship; β_0: intercept; β_1: slope.



https://livebook.manning.com/book/grokking-machine-learning/3-1-the-problem-we-need-to-predict-the-price-of-a-house/v-4/44

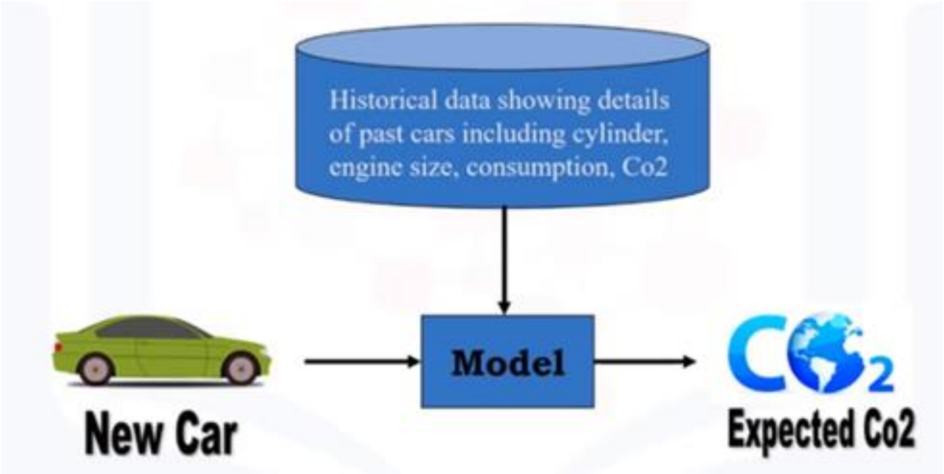
Regression: Predict Response Y based on Regressor x

Figure 6. Our task is now to predict the price of the house with 4 rooms. Using the model (line), we deduce that the predicted price of this house is $300.



https://livebook.manning.com/book/grokking-machine-learning/3-1-the-problem-we-need-to-predict-the-price-of-a-house/v-4/44

Regression Modeling: Example

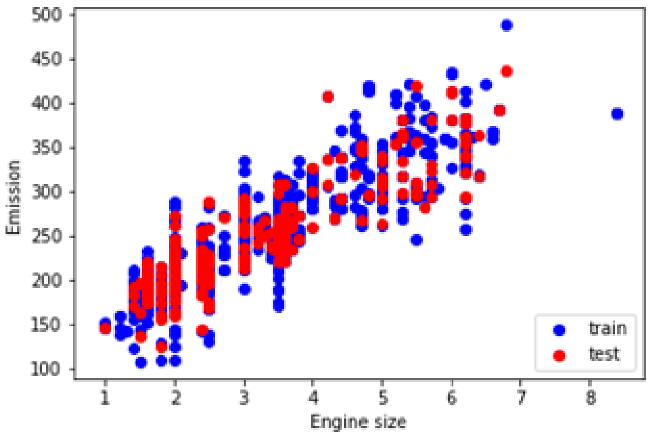



```
1  df.shape
```

(1067, 13)

```
1  df[['ENGINESIZE','CO2EMISSIONS']].sample(5)
```

	ENGINESIZE	CO2EMISSIONS
697	2.0	196
872	3.8	253
614	2.4	200
351	5.4	382
426	3.5	258

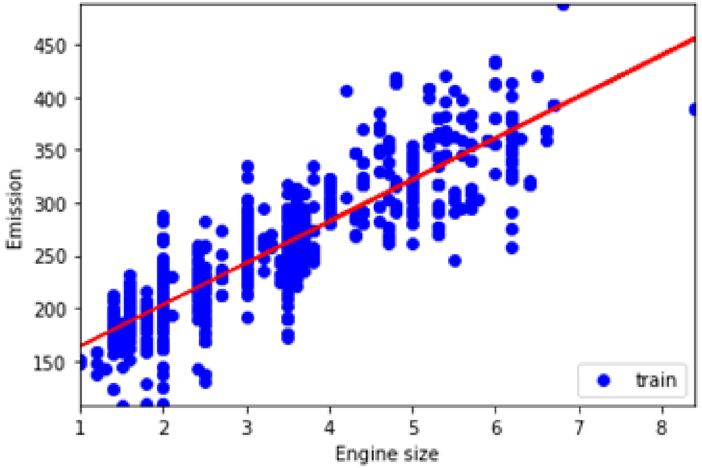


Sumber: DTS - Thematic Academy 2021 - Regresi

Simple Linear Regression: Training

```python
from sklearn import linear_model
regr = linear_model.LinearRegression()
train_x = np.asanyarray(train[['ENGINESIZE']])
train_y = np.asanyarray(train[['CO2EMISSIONS']])
regr.fit (train_x, train_y)
# The coefficients
print ('Coefficients: ', regr.coef_)
print ('Intercept: ',regr.intercept_)
```

```
Coefficients:  [[39.43522758]]
Intercept:  [124.41641136]
```



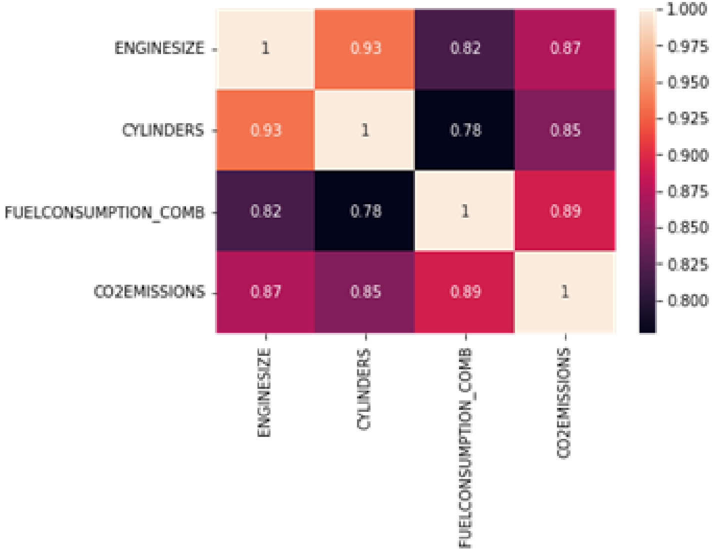
$$y=125.49+39.14x$$

Multivariate Linear Regression: Training & Testing

```python
from sklearn import linear_model
regr2 = linear_model.LinearRegression()
train_x = np.asanyarray(train[['ENGINESIZE','CYLINDERS','FUELCONSUMPTION_COMB']])
train_y = np.asanyarray(train[['CO2EMISSIONS']])
regr2.fit (train_x, train_y)
print ('Coefficients: ', regr2.coef_)
print ('Intercept: ',regr2.intercept_)
```

```
Coefficients:  [[11.36694334  7.26349823  9.58219844]]
Intercept:  [64.98474043]
```

	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_COMB	CO2EMISSIONS
0	2.0	4	8.5	196
1	2.4	4	9.6	221
2	1.5	4	5.9	136
3	3.5	6	11.1	255
4	3.5	6	10.6	244



Estimating Parameters: Least Square Estimator

- Objective function: minimize sum of the squares of the residuals/errors (SSE)

$$SSE = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} (y_i - b_0 - b_1 x_i)^2$$

$$\frac{\partial(SSE)}{\partial b_0} = -2 \sum_{i=1}^{n} (y_i - b_0 - b_1 x_i)$$

$$n b_0 + b_1 \sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i$$

$$b_0 = \frac{\sum_{i=1}^{n} y_i - b_1 \sum_{i=1}^{n} x_i}{n} = \bar{y} - b_1 \bar{x}$$

$$\frac{\partial(SSE)}{\partial b_1} = -2 \sum_{i=1}^{n} (y_i - b_0 - b_1 x_i) x_i$$

$$b_1 = \frac{n \sum_{i=1}^{n} x_i y_i - \left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} y_i\right)}{n \sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2} = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2} = \sigma_{XY}/\sigma^2_X$$

Let X be a random variable with probability distribution $f(x)$ and mean μ. The variance of X is

$$\sigma^2 = E[(X-\mu)^2] = \sum_x (x-\mu)^2 f(x), \qquad \text{if } X \text{ is discrete, and}$$

$$\sigma^2 = E[(X-\mu)^2] = \int_{-\infty}^{\infty} (x-\mu)^2 f(x)\, dx, \qquad \text{if } X \text{ is continuous.}$$

The positive square root of the variance, σ, is called the **standard deviation** of X.

Let X and Y be random variables with joint probability distribution $f(x, y)$. The covariance of X and Y is

$$\sigma_{XY} = E[(X-\mu_X)(Y-\mu_Y)] = \sum_x \sum_y (x-\mu_X)(y-\mu_y) f(x,y)$$

if X and Y are discrete, and

$$\sigma_{XY} = E[(X-\mu_X)(Y-\mu_Y)] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} (x-\mu_X)(y-\mu_y) f(x,y)\, dx\, dy$$

if X and Y are continuous.

Let X and Y be random variables with covariance σ_{XY} and standard deviations σ_X and σ_Y, respectively. The correlation coefficient of X and Y is

$$\rho_{XY} = \frac{\sigma_{XY}}{\sigma_X \sigma_Y}.$$

Implement From Scratch: Simple Linear Regression

From Scratch

b1=covariance(X,Y)/variance(X)
b0=mean(Y)-b1*mean(X)

```
# Coefficients:  [[39.43522758]]
# Intercept:  [124.41641136]
print('cov(X,Y)=',train.CO2EMISSIONS.cov(train.ENGINESIZE))
print('var(X)=',train.ENGINESIZE.var())
b1=train.CO2EMISSIONS.cov(train.ENGINESIZE)/train.ENGINESIZE.var()
b0=train.CO2EMISSIONS.mean()-b1*train.ENGINESIZE.mean()
print('Coefficient=',b1)
print('Intercept=',b0)
```

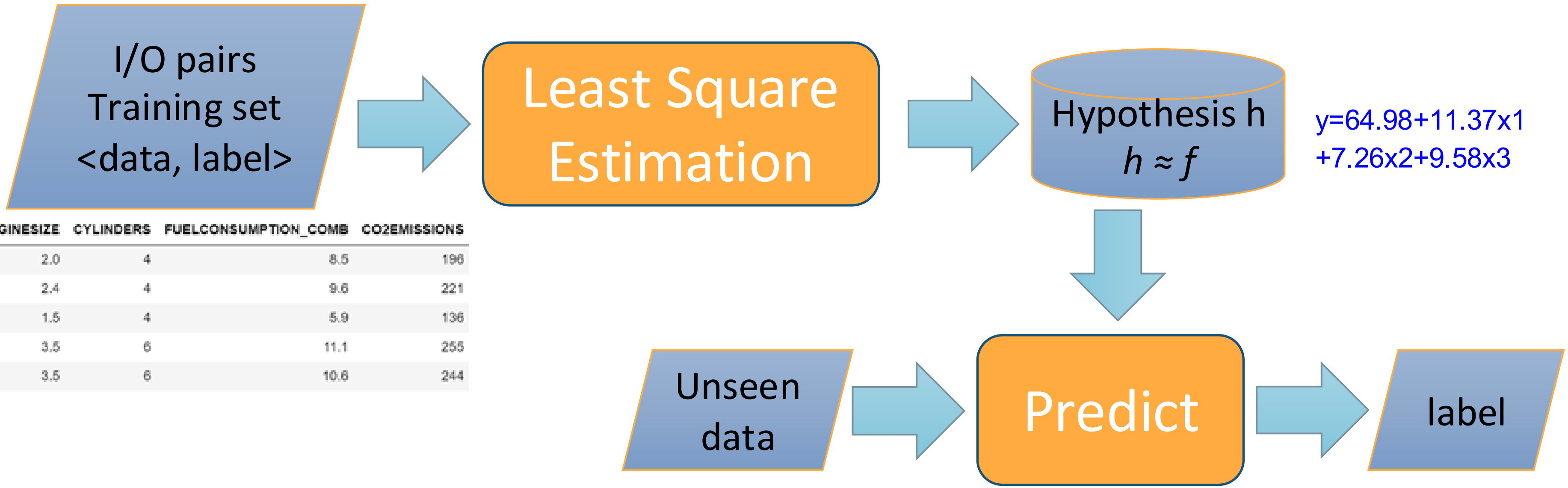
```
cov(X,Y)= 78.95604822766053
var(X)= 2.0021704722919877
Coefficient= 39.43522757943557
Intercept= 124.41641136306387
```

```
from sklearn import linear_model
regr = linear_model.LinearRegression()
train_x = np.asanyarray(train[['ENGINESIZE']])
train_y = np.asanyarray(train[['CO2EMISSIONS']])
regr.fit (train_x, train_y)
# The coefficients
print ('Coefficients: ', regr.coef_)
print ('Intercept: ',regr.intercept_)
```

```
Coefficients:  [[39.43522758]]
Intercept:  [124.41641136]
```

Linear Regression: Supervised Learning

Learning a (possibly incorrect) general function from specific input-output pairs



y=64.98+11.37x1
+7.26x2+9.58x3

	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_COMB	CO2EMISSIONS
0	2.0	4	8.5	196
1	2.4	4	9.6	221
2	1.5	4	5.9	136
3	3.5	6	11.1	255
4	3.5	6	10.6	244

ENGINESIZE	CYLINDERS	FUELCONSUMPTION_COMB
2.0	4	8.5
2.4	4	9.6
1.5	4	5.9

```
test_y[0:3]
```

```
array([[196],
       [221],
       [136]],
```

y=64.98+11.37x1
+7.26x2+9.58x3

```
#test_y_ = regr2.predict(test_x)
test_y_[0:3]
```

```
array([[198.22130673],
       [213.30850235],
       [167.62411912]])
```

Linear Regression: Testing

```python
from sklearn.metrics import r2_score

test_x = np.asanyarray(test[['ENGINESIZE']])
test_y = np.asanyarray(test[['CO2EMISSIONS']])
test_y_ = regr.predict(test_x)

print("Mean absolute error: %.2f" % np.mean(np.absolute(test_y_ - test_y)))
print("Residual sum of squares (MSE): %.2f" % np.mean((test_y_ - test_y) ** 2))
print("R2-score: %.2f" % r2_score(test_y_ , test_y) )
```

y=125.49+39.14x

```
Mean absolute error: 22.50
Residual sum of squares (MSE): 840.97
R2-score: 0.73
```

y=64.98+11.37x1+7.26x2+9.58x3

```python
from sklearn.metrics import r2_score

test_x = np.asanyarray(test[['ENGINESIZE','CYLINDERS','FUELCONSUMPTION_COMB']])
test_y_ = regr2.predict(test_x)

print("Mean absolute error: %.2f" % np.mean(np.absolute(test_y_ - test_y)))
print("Residual sum of squares (MSE): %.2f" % np.mean((test_y_ - test_y) ** 2))
print("R2-score: %.2f" % r2_score(test_y_ , test_y) )
```

```
Mean absolute error: 15.20
Residual sum of squares (MSE): 443.90
R2-score: 0.87
```

$$MAE = \boxed{\frac{1}{n}} \sum \underbrace{\left| \boxed{y} - \boxed{\hat{y}} \right|}_{\text{The absolute value of the residual}}$$

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \tilde{y}_i)^2$$

Coefficient of Determination, R^2

$$R2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})},$$

Linear Regression - LSE: Summary

Target function is estimated by **hypothesis** as a linear function of X:

$h_{\boldsymbol{\theta}}(x)=\boldsymbol{\theta}_0+\boldsymbol{\theta}_1x_1+...+\boldsymbol{\theta}_nx_n=\boldsymbol{\Sigma\theta}_ix_i$, i in [0..n]

$\rightarrow h_{\boldsymbol{\theta}}(x) = \boldsymbol{\theta}^T.\mathbf{x}$

Cost function $J(\boldsymbol{\theta})$ assigns a large cost to bad predictions h(x) and small cost to good predictions h(x). For LSE, we used **sum of squares errors (SSE)**.

Find good $\boldsymbol{\theta}$ (parameters or weights): $\boldsymbol{\theta}^*=\text{argmin } J(\boldsymbol{\theta})$

Analytical solution: find $\boldsymbol{\theta}$ that solve $\nabla_J(\boldsymbol{\theta})=0$.

If $\nabla_J(\boldsymbol{\theta})=0$ is unable to solve, use iterative numerical methods. The simplest method is gradient descent.

Pertanyaan ?