Machine Learning

Indah Agustien Siradjuddin

Linear Regression

Semester Gasal 2019-2020

Note : Pemilihan algoritma/pendekatan *Machine learning* berdasarkan permasalahan ataupun tujuan pembuatan sistem

---

Linear Regression :

1. Definisi dan Studi Kasus
2. Model Linear Regression
3. Training :

    a. Direct / Matrix Equation
    b. Gradient Descent

4. Linear Regression dengan Scikit
5. Akurasi

Linear Regression - Definisi

- Model yang paling sederhana di dalam algoritma *Machine Learning*
- Prediksi **nilai real** berdasarkan fitur atau atribut dari data pelatihan (*training*/*learning*)
- Model yang dibuat berbentuk linear
- **Supervised Machine Learning**

Linear Regression - Permasalahan *Housing*

Memprediksikan harga rumah berdasarkan luar area

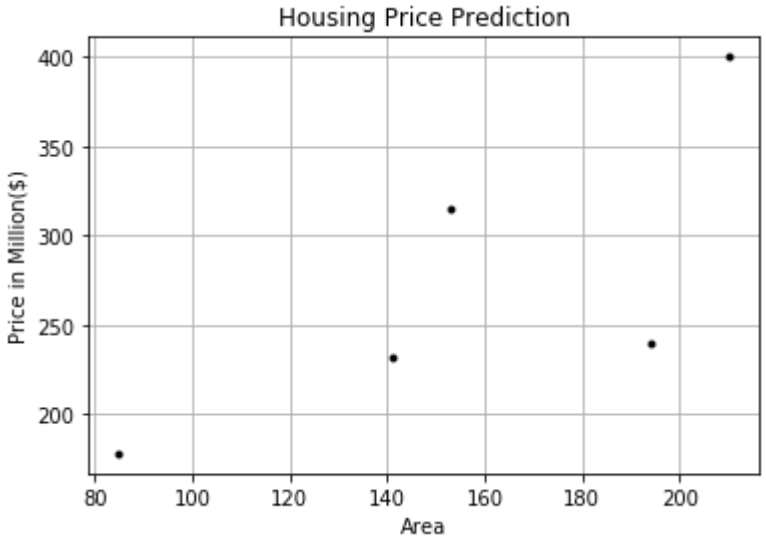
Misalkan terdapat data berikut :

Tujuan : prediksikan harga rumah jika terdapat luas rumah tertentu
Plotting data pelatihan :

In [1]:

```
import matplotlib.pyplot as plt
%matplotlib inline
X=[[210],[141],[153],[85],[194]]
Y=[[400],[232],[315],[178],[240]]

plt.title('Housing Price Prediction')
plt.xlabel('Area')
plt.ylabel('Price in Million($)')
plt.plot(X, Y, 'k.')
plt.grid(True)
plt.show()
```



Linear Regression - Model

Model Regresi Linear akan memetakan satu atribut (*univariate*) atau beberapa atribut (*multivariate*) menjadi satu nilai output (target) berupa nilai real.

Univariate Linear Regression

$$f : \Re^1 \rightarrow \Re \quad f(x;w) = w_0 + w_1 x$$

Multivariate Linear Regression

$$f : \Re^d \rightarrow \Re \quad f(x;w) = w_0 + w_1 x_1 + \ldots + w_d x_d$$

Notasi :

- $f(x;w)$ atau y : variabel output,
- x : variabel input atau nilai atribute/fitur,
- w : bobot atau *coefficients*; w_0 adalah nilai *intercept* atau *bias*

Note:

- Univariate Linear Regression untuk satu attribute/fitur
- Multivariate Linear Regression untuk beberapa attribute/fitur

Training - Buat Model Linear Regression

1. *Direct Equation* / *Matrix equation*
2. *Gradient Descent* (*stochastic/online* atau *batch/offline learning*)

Direct/Matrix Equation

Model *Univariate Linear Regression*

Linear Regression : $y = w_0 + w_1 x$
Bias atau intercept, dan *coefficient* (*weights* : w_0, w_1), dihitung melalui persamaan berikut :

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

koefisien dan intercept dari model regressi linear adalah :

$$w_1 = \frac{cov(x,y)}{var(x)}$$
$$w_0 = \bar{y} - w_1 \bar{x}$$

Kode *Training* Univariate Linear Regression

In [2]:

```
import simpleLinear
import numpy as np
```

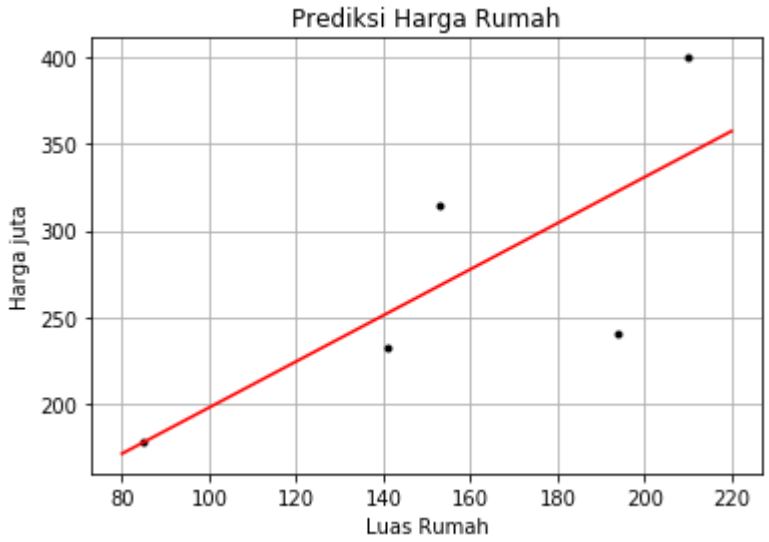
In [3]:

```
X=np.array([[210],[141],[153],[85],[194]])
Y=np.array([[400],[232],[315],[178],[240]])
meanX=simpleLinear.meanData(X)
varX=simpleLinear.varianceData(X)
covData=simpleLinear.covarianceData(X,Y)
print(covData)
coefData=covData/varX
print('coef=',coefData)
intercept=simpleLinear.meanData(Y)-coefData*meanX
print('intercept=',intercept)
```

```
3209.5
coef= 1.3326828052983435
intercept= 64.30187269027942
```

In [4]:

```
dataX=np.linspace(80,220,50)
dataY=intercept+coefData*dataX
#plot Data training and testing
plt.title('Prediksi Harga Rumah')
plt.xlabel('Luas Rumah')
plt.ylabel('Harga juta')
plt.plot(X, Y, 'k.')
plt.plot(dataX, dataY, 'r-')
plt.grid(True)
plt.show()
```



Kode *Testing* Univariate Linear Regression

Model yang dibentuk berdasarkan proses pelatihan : $y = 64.3 + 1.33x$

In [5]:

```
dataTest=int(input('Luas rumah = '))
harga=intercept+coefData*dataTest
print('harga rumah =',harga)
```

```
Luas rumah = 100
harga rumah = 197.57015322011378
```

Latihan

Berikut adalah data *training* yang terdiri dari fitur X dan target Y .

Tugas :

1. *plot* data training
2. Hitung bobot (dan bias) secara manual, dan tulis model yang dihasilkan
3. Berdasarkan model yang dibuat, hitung output, jika input adalah 9 atau 10
4. Gambar grafik model linear regression, dan plot data training (*blue dots*) dan data testing (*red dot*)

Multivariate Linear Regression

Model Multivariate Linear Regression : $f(x;w) = w_0 + w_1x_1 + \ldots + w_dx_d$ or
$y = w_0 + w_1x_1 + \ldots + w_dx_d$

Fungsi obyektif atau *loss function* diperlukan untuk perhitungan bobot atau *weight* :

Loss Function :

$$J_n(w) = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i;w))^2$$

Bobot diperoleh dengan cara meminimalkan fungsi *loss*

$$y = \begin{bmatrix} y_1 \\ \ldots \\ y_n \end{bmatrix} \quad X = \begin{bmatrix} 1 & x_1 \\ \ldots & \ldots \\ 1 & x_n \end{bmatrix} \quad w = \begin{bmatrix} w_0 \\ \ldots \\ w_1 \end{bmatrix}$$

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - f(x;w))^2 = \frac{1}{n}\left[\begin{bmatrix} y_1 \\ \ldots \\ y_n \end{bmatrix} - \begin{bmatrix} 1 & x_1 \\ \ldots & \ldots \\ 1 & x_n \end{bmatrix}\begin{bmatrix} w_0 \\ \ldots \\ w_1 \end{bmatrix}\right]^2$$

$$= \frac{1}{n}[y - Xw]^2$$

$$\frac{\partial}{\partial w}\frac{1}{n}[y - Xw]^2 = \frac{2}{n}X^T(y - Xw)$$

$$= \frac{2}{n}(X^Ty - X^TXw) = 0$$

$$w = (X^TX)^{-1}X^Ty$$

Latihan

Berikut adalah data *training* yang terdiri dari fitur X dan target Y .

Tugas :

1. Buat model *linear regression* dengan menggunakan persamaan matriks, dan hitung weight secara manual (data yang digunakan adalah dua baris terakhir)
2. Buat model *linear regression* dengan kode program (menggunakan persamaan matriks)
3. Berapa output jika input adalah 6 atau 7

Latihan

Buat model linear regression (buat kode program sendiri) dengan menggunakan persamaan matriks :

1. gunakan dataset boston (scikit)
2. Untuk mempermudah, gunakan 3 fitur pertama dari dataset boston

Gradient Descent Learning

Solusi optimal dicari dengan cara mencari nilai bobot secara iteratif, sedemikian hingga fungsi loss (misal : Mean Square Error) akan bernilai minimal.
sumber :

challenges : Penentuan parameter *learning*
Nilai paramater *learning* terlalu kecil:

sumber :

Nilai paramater *learning* terlalu besar:

sumber :

Note : Semua data fitur seharusnya punya skala yang sama --> **normalize data**

Training dengan : Stochastic dan Batch Gradient Descent

1. Stochastic Gradient Descent:
   untuk semua data dalam data training :
   $$w_j = w_j - \eta(f(x^i) - y^i)x_j^i$$
2. Batch Gradient Descent :
   $$w_j = w_j - \eta\frac{1}{n}\sum_{i=1}^{n}(f(x^i) - y^i)x_j^i$$

n=jumlah data dalam data training
$f(x_i)$ =prediksi ke-i
y_i =target output untuk data ke-i
d= jumlah fitur
x_j =fitur ke - j
$x_0 = 1$ bias/intercept
w_j =bobot untuk fitur ke - j

Latihan

Bangun model linear regression berdasarkan data berikut :

1. Hitung bobot dengan menggunakan SGD secara manual, sampai dengan lima iterasi
2. Buat Kode program secara manual
3. Buat kode program dengan menggunakan scikit

Linear Regression dengan scikit

Training/Learning

- *load training dataset (features and target)*
- buat Model
- lakukan proses training atau learning

Testing

- Load Model
- test model dengan data test / *Generalize*

In [6]:

```
import numpy as np
from sklearn.linear_model import LinearRegression

X=np.array([[210],[141],[153],[85],[194]])
Y=np.array([[400],[232],[315],[178],[240]])
print(X.shape, Y.shape)

#create Model
model = LinearRegression()

# Learning
model.fit(X, Y)
```

```
(5, 1) (5, 1)
```

Out[6]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=F
alse)
```

In [7]:

```
# Testing
print('coef=',model.coef_)
print('intercept=',model.intercept_)

newArea=np.array([[125]])
predPrice=model.predict(newArea)
print(predPrice)
```

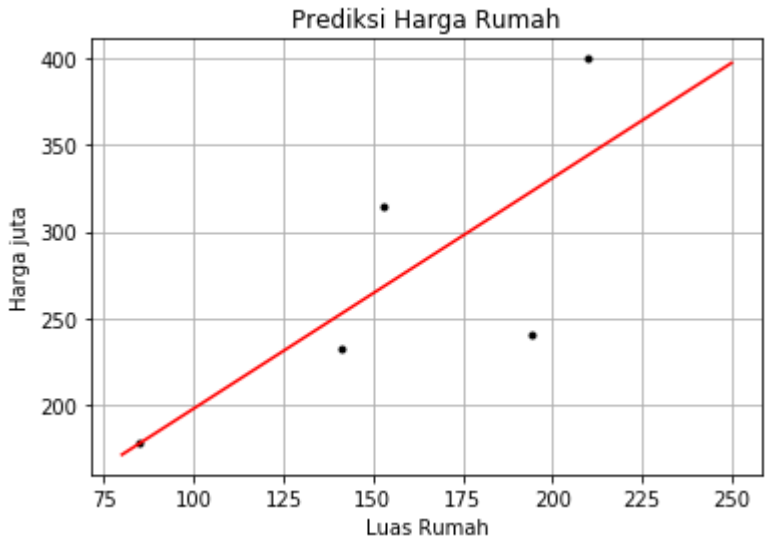
```
coef= [[1.33268281]]
intercept= [64.30187269]
[[230.88722335]]
```

In [8]:

```
import matplotlib.pyplot as plt
%matplotlib inline


dataX=np.linspace(80,250,50)
dataY=model.intercept_+model.coef_*dataX
dataY=dataY.T
#plot Data training and testing
plt.title('Prediksi Harga Rumah')
plt.xlabel('Luas Rumah')
plt.ylabel('Harga juta')
plt.plot(X, Y, 'k.')
plt.plot(dataX[:], dataY[:,0], 'r-')
plt.grid(True)
plt.show()
```

```
[[210]
 [141]
 [153]
 [ 85]
 [194]] [[400]
 [232]
 [315]
 [178]
 [240]]
```



Akurasi

Perhitungan kinerja untuk mengukur seberapa akurat model yang dihasilkan untuk proses generalisasi. *R-Square* untuk mengukur model linear regressi.
Nilai *r-squared* antara 0 s.d 1, nilai 1 berarti output dapat diprediksi secara benar tanpa ada error.

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

$$SS_{res} = \sum_{i=1}^{n} (y_i - f(x_i))^2$$

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

where
y_i =target data ke -i
$\bar{y}$ =mean target
n =jumlah data
x_i =data ke -i
$f(x_i)$ =prediksi untuk data ke - i

In [9]:

```
# Accuracy using scikit

from sklearn.linear_model import LinearRegression
model = LinearRegression()
X=[[1],[2],[4],[3],[5]]
Y=[[1],[3],[3],[2],[5]]
model.fit(X, Y)
print(model.score(X,Y))

#model.predict(data)
```

```
0.7272727272727273
```

Referensi