

Regresi Linear

Ali Akbar Septiandri

September 20, 2017

Universitas Al Azhar Indonesia

1. Ordinary Least Squares
2. Error Minimisation
3. Non-linear Functions

Model-based reflex agents

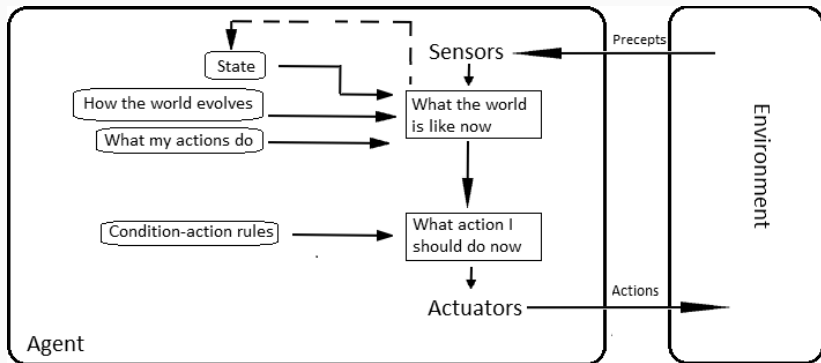


Figure 1: Model yang mengandalkan riwayat persepsi dan dampaknya

Ordinary Least Squares

Prediksi hubungan antara dua variabel

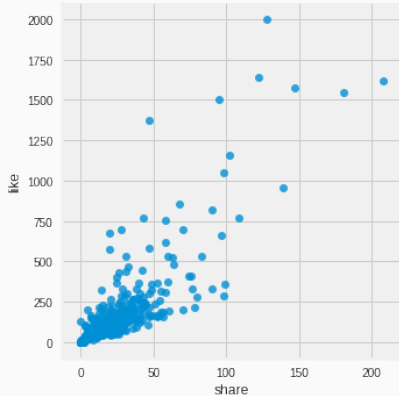


Figure 2: Data hubungan antara 'share' dengan 'like' pada Facebook

Prediksi hubungan antara dua variabel

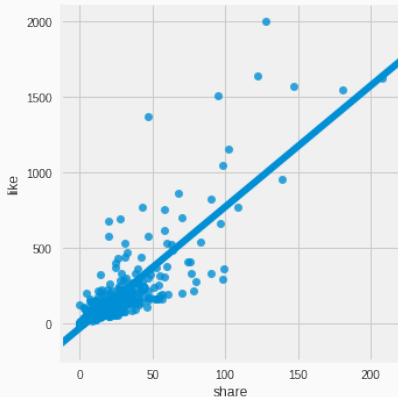


Figure 2: Data hubungan antara 'share' dengan 'like' pada Facebook

Simple linear regression

Fungsi linear

Kasus paling sederhana adalah mencocokkan garis lurus ke sekumpulan data

$$y = ax + b$$

dengan a adalah *slope*, gradien, atau kemiringan; sedangkan b dikenal dengan nama *intercept* atau bias.

Notasi lain

$$y = w_0 + w_1x_1$$

dengan w adalah bobot atau koefisien.

Linear regresi dari fungsi yang diketahui

Example

```
rng = np.random.RandomState(1)
x = 10 * rng.rand(50)
y = 2 * x - 5 + rng.randn(50)
plt.scatter(x, y);
```

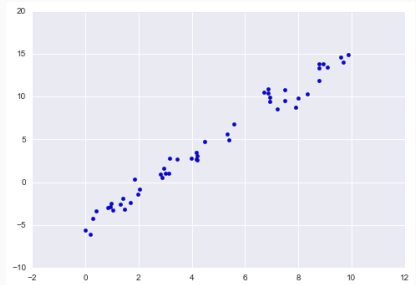


Figure 3: Data yang dimunculkan secara acak [VanderPlas, 2016]

Ordinary least squares (OLS) regression

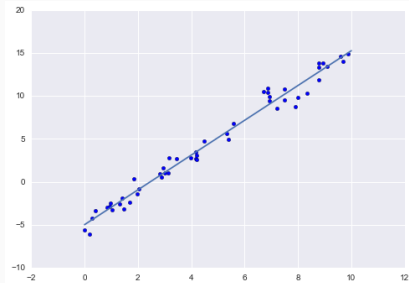


Figure 4: Hasil pencocokan garis [VanderPlas, 2016]

Model slope: 2.02720881036

Model intercept: -4.99857708555

Bagaimana kalau ada lebih dari dua variabel
yang ingin kita lihat hubungannya?

Multidimensional linear regression

Model

$$y = w_0x_0 + w_1x_1 + w_2x_2 + \dots + w_Dx_D = \sum_{j=0}^D w_jx_j$$

dengan $x_0 = 1$

Notasi matriks-vektor

$$y = \mathbf{w} \cdot \phi(x)$$

dengan $\phi(x)$ adalah vektor fitur (*feature vector*)

Regresi linear untuk dua variabel

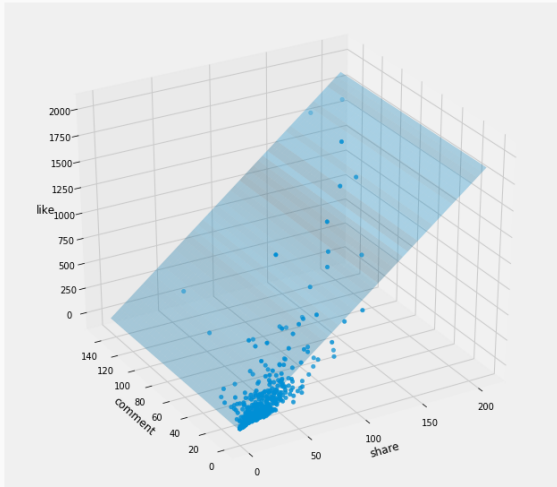


Figure 5: Hubungan antara 'share', 'comment', dan 'like' pada foto di Facebook

Prediktor linear (contoh)

Vektor bobot $\mathbf{w} \in \mathbb{R}^D$

bias: -20.24

share: 6.65

comment: 3.53

Vektor fitur $\phi(x) \in \mathbb{R}^D$

bias: 1

share: 147

comment: 58

$$\hat{y} = \mathbf{w} \cdot \phi(x)$$

$$= \sum_{j=1}^D w_j \phi_j(x)$$

$$= -20.24(1) + 6.65(147) + 3.53(58) = 1162.05$$

Jadi, *diprediksi* bahwa untuk foto dengan *share* = 147 dan *comment* = 58, foto tersebut akan mendapatkan ≈ 1162.05 *likes*.

Bagaimana cara mendapatkan nilai w ?

Error Minimisation

(Supervised) Learning

- Kita ingin mencari $f : \mathcal{X} \rightarrow \mathcal{Y}$
 \mathcal{X} : data masukan
 \mathcal{Y} : data keluaran
 dari data latih yang i.i.d.¹

$$\mathcal{D} = (x_1, y_1), \dots, (y_N, y_N)$$

- Objektif: Meminimalkan *generalisation error* dengan menggunakan *loss function* ℓ , contohnya:

$$\ell(y, f(x)) = (y - f(x))^2$$

yang juga dikenal dengan nama *squared loss*

¹independent and identically distributed

Meminimalkan error pada data latih

- Untuk meminimalkan *generalisation error*

$$w^* = \arg \min_w \mathbb{E}_{X,Y}[(Y - w^T X)^2]$$

- Kita tidak punya data untuk seluruh kemungkinan pasangan nilai X dan Y !
- Ide: Minimalkan error pada data latih

$$\hat{\mathbf{w}} = \arg \min_{\mathbf{w}} \frac{1}{|\mathcal{D}_{train}|} \sum_{(x,y) \in \mathcal{D}_{train}} \ell(y, x, \mathbf{w})$$

- Didefinisikan fungsi error

$$E(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^N (y_i - \mathbf{w} \cdot \phi(x_i))^2$$

- Didefinisikan fungsi error

$$E(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^N (y_i - \mathbf{w} \cdot \phi(x_i))^2$$

- Nilainya dapat dioptimasi dengan mencari turunan pertama, lalu atur nilainya menjadi 0, i.e. $\frac{\partial E}{\partial \mathbf{w}} = 0$ atau $\nabla_{\mathbf{w}} E(\mathbf{w}) = 0$

- Didefinisikan fungsi error

$$E(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^N (y_i - \mathbf{w} \cdot \phi(x_i))^2$$

- Nilainya dapat dioptimasi dengan mencari turunan pertama, lalu atur nilainya menjadi 0, i.e. $\frac{\partial E}{\partial \mathbf{w}} = 0$ atau $\nabla_{\mathbf{w}} E(\mathbf{w}) = 0$
- Solusi tertutupnya:

$$\hat{\mathbf{w}} = (\phi^T \phi)^{-1} \phi^T \mathbf{y}$$

- Didefinisikan fungsi error

$$E(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^N (y_i - \mathbf{w} \cdot \phi(x_i))^2$$

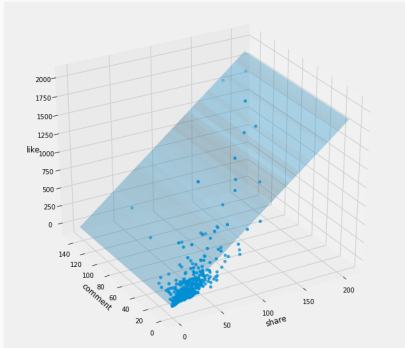
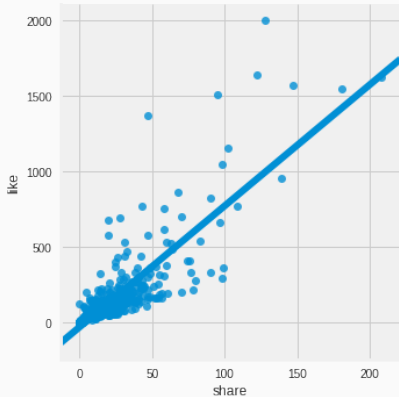
- Nilainya dapat dioptimasi dengan mencari turunan pertama, lalu atur nilainya menjadi 0, i.e. $\frac{\partial E}{\partial \mathbf{w}} = 0$ atau $\nabla_{\mathbf{w}} E(\mathbf{w}) = 0$
- Solusi tertutupnya:

$$\hat{\mathbf{w}} = (\phi^T \phi)^{-1} \phi^T \mathbf{y}$$

- Bagian $(\phi^T \phi)^{-1} \phi^T$ dikenal sebagai *pseudo-inverse*

Non-linear Functions

Perhatikan kembali



Apa kekurangan dari regresi linear sederhana seperti ini?

Non-linearity

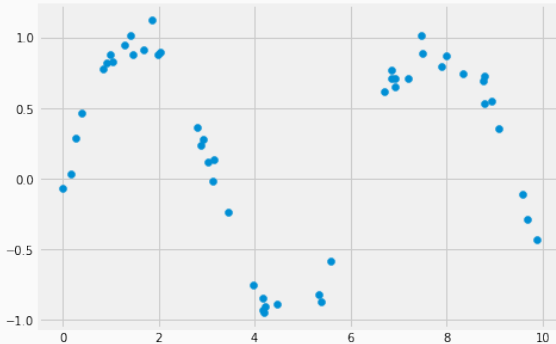


Figure 6: Data yang dihasilkan dari fungsi sin dengan *noise*

Bagaimana kalau datanya seperti ini?

Underfitting

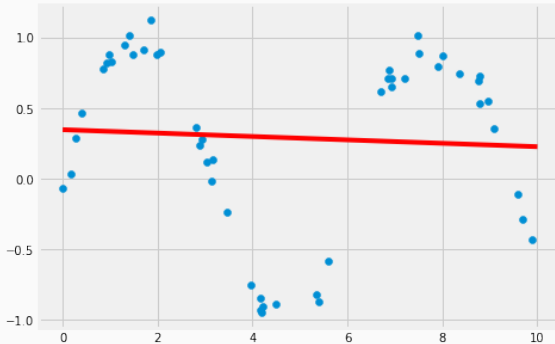


Figure 7: Hasil *fitting* regresi linear sederhana

Jika model yang dihasilkan lebih sederhana dibandingkan data yang seharusnya dicocokkan, maka model tersebut disebut mengalami **underfitting**.

Regresi linear dengan fungsi basis polinomial

Jika kita mengubah $x_p = f_p(x)$, dengan $f_p()$ adalah fungsi transformasi, maka untuk $f_p() = x^p$ dan x adalah input berdimensi satu, modelnya menjadi

$$y = w_0 + w_1x + w_2x^2 + w_3x^3 + \dots$$

Polynomial Basis Functions

In

```
from sklearn.preprocessing import PolynomialFeatures
x = np.array([2, 3, 4])
poly = PolynomialFeatures(3, include_bias=False)
poly.fit_transform(x[:, None])
```

Out

```
array([[ 2.,  4.,  8.],
       [ 3.,  9., 27.],
       [ 4., 16., 64.]])
```

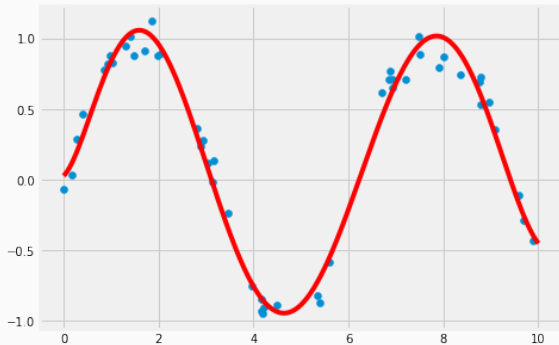


Figure 8: Hasil *fitting* fungsi basis polinomial $p = 7$

Apa yang terjadi jika p dibuat lebih besar?

Overfitting

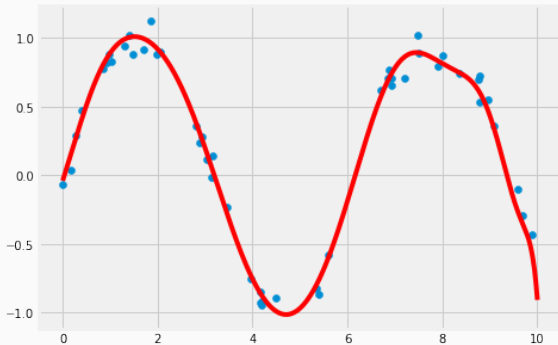


Figure 9: Hasil *fitting* fungsi basis polinomial $p = 15$

Jika model yang dihasilkan **lebih kompleks** (\sim parameternya banyak) dibandingkan data yang **seharusnya** dicocokkan, maka model tersebut disebut mengalami **overfitting**.

Kita dapat mengatasi masalah *overfitting*
pada regresi linear dengan melakukan *regularisasi*.
(*non-examinable*)

- Regresi linear dapat digunakan untuk **memprediksi nilai riil**
- Regresi linear mempunyai **solusi tertutup** untuk mencari nilai bobot
- Kasus non-linear dapat ditangani oleh regresi linear dengan melakukan transformasi terhadap fitur, e.g. dengan **fungsi basis polinomial**
- Konfigurasi parameter yang tepat dibutuhkan untuk menghindari *underfitting* dan *overfitting*

Pertemuan berikutnya

- Klasifikasi: regresi logistik
- Optimasi numerik



Jake VanderPlas (2016)

In Depth: Linear Regression

Python Data Science Handbook

Terima kasih