Regresi dan Optimasi

Ali Akbar Septiandri

August 4, 2017

Universitas Al Azhar Indonesia

Table of contents

Regresi Linear

Prediksi hubungan antara dua variabel

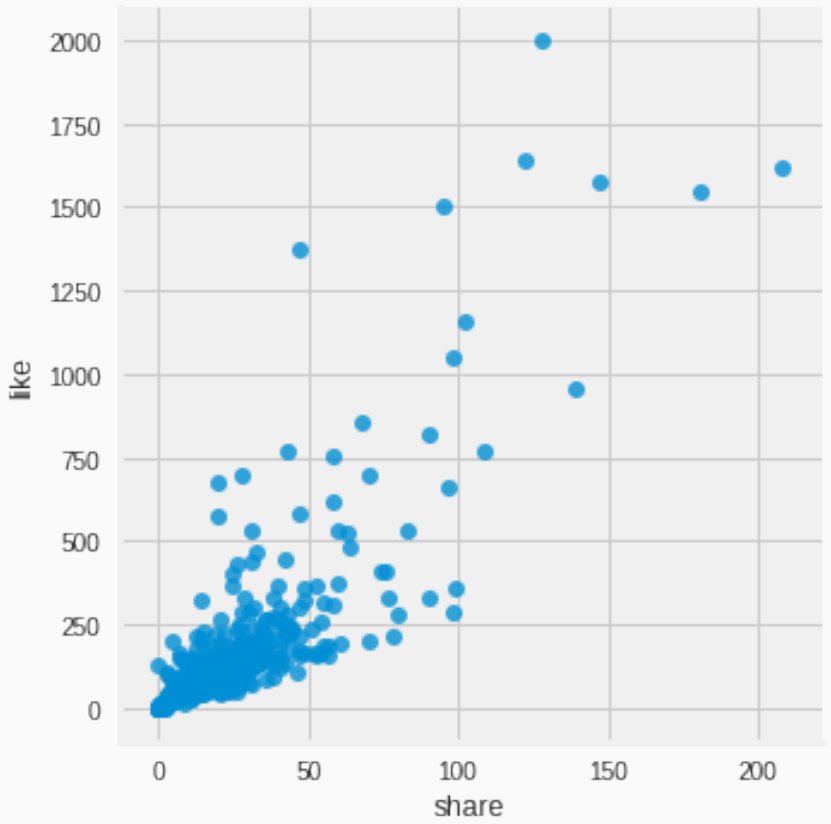


Figure 1: Data hubungan antara ‘share’ dengan ‘like’ pada Facebook

Prediksi hubungan antara dua variabel

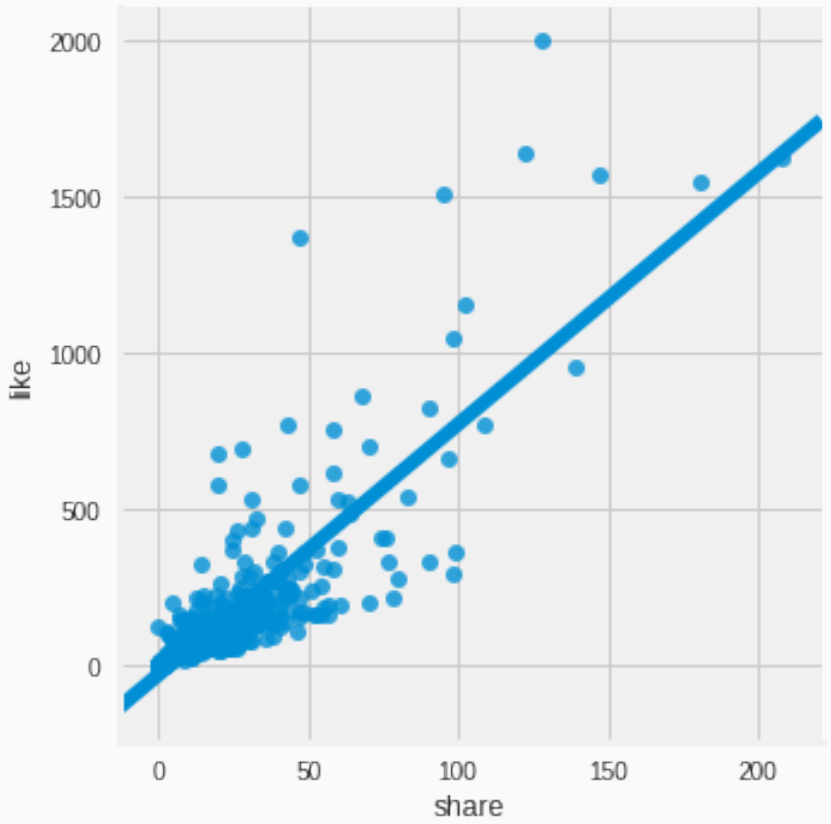


Figure 1: Data hubungan antara ‘share’ dengan ‘like’ pada Facebook

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

$$y = w_0 + w_1 x_1$$

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);
```

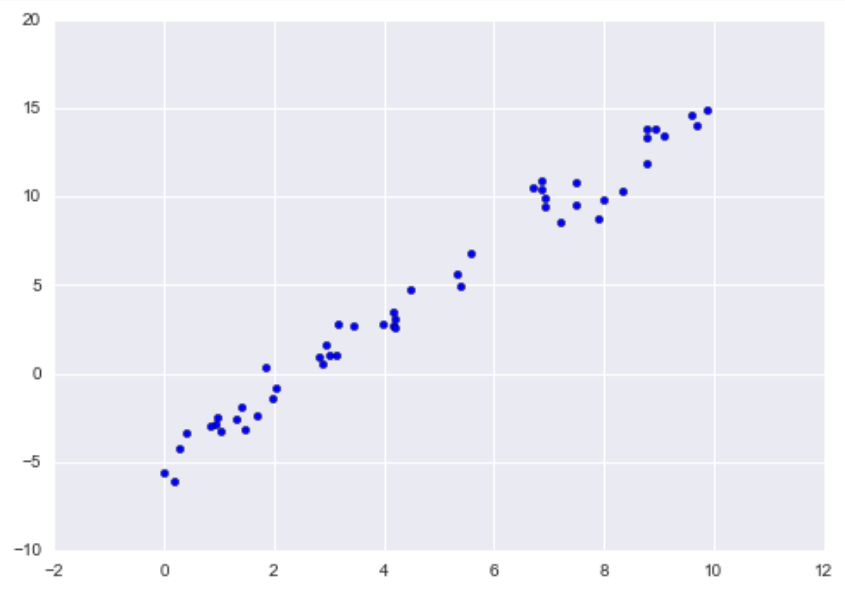


Figure 2: Data yang dimunculkan secara acak [VanderPlas, 2016]

Ordinary least squares (OLS) regression

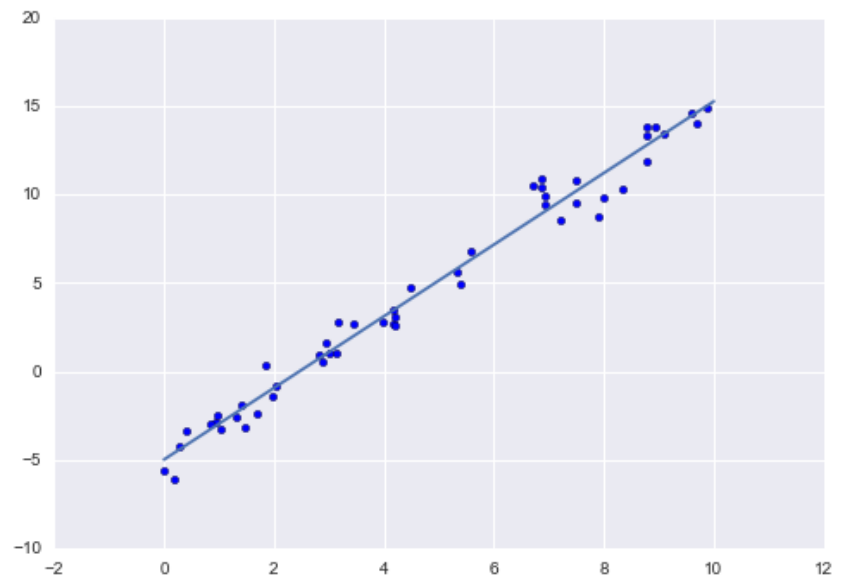


Figure 3: Hasil pencocokan garis [VanderPlas, 2016]

```
Model slope:  2.02720881036
Model intercept:  -4.99857708555
```

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

Multidimensional linear regression

Model

$$y = w_0 x_0 + w_1 x_1 + w_2 x_2 + ... + w_D x_D = \sum_{j=0}^{D} w_j x_j$$

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

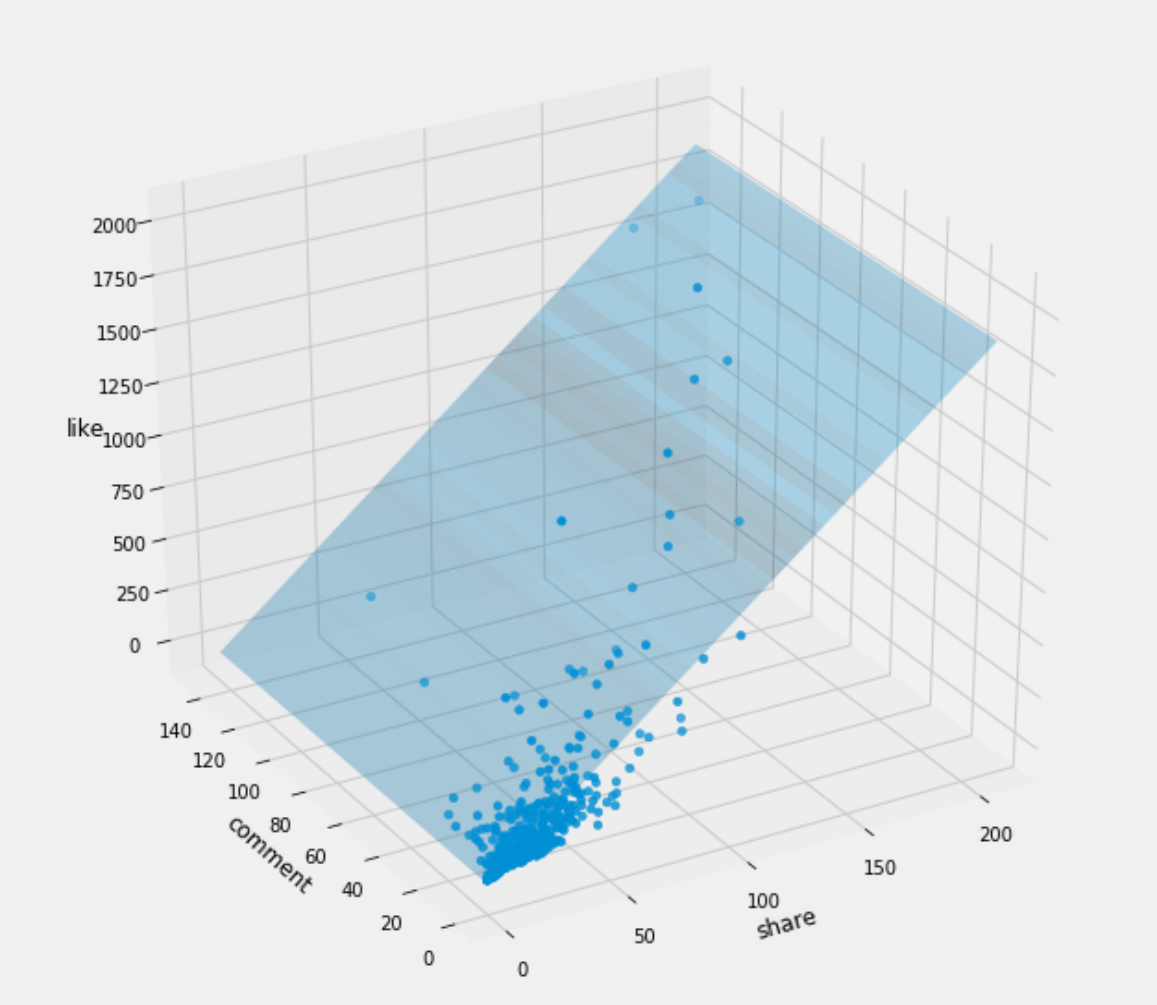


Figure 4: Hubungan antara ‘share’, ‘comment’, dan ‘like’ pada foto di Facebook

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

$$\begin{aligned}\hat{y} &= \mathbf{w} \cdot \phi(x) \\ &= \sum_{j=1}^{D} w_j \phi_j(x) \\ &= -20.24(1) + 6.65(147) + 3.53(58) = 1162.05\end{aligned}$$

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

Bagaimana cara mendapatkan nilai w?

(Supervised) Learning

- Kita ingin mencari $f : \mathcal{X} \to \mathcal{Y}$

  X: data masukan

  Y: data keluaran

  dari data latih yang i.i.d.[1]

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

- 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*

[1] independent 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

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

Optimasi analitis

- Didefinisikan fungsi error

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

Optimasi analitis

- 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. $\frac{\partial E}{\partial w} = 0$ atau $\nabla_{\mathbf{w}} E(\mathbf{w}) = 0$

Optimasi analitis

- 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. $\frac{\partial E}{\partial w} = 0$ atau $\nabla_{\mathbf{w}} E(\mathbf{w}) = 0$
- Solusi tertutupnya:

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

Optimasi analitis

- Didefinisikan fungsi error

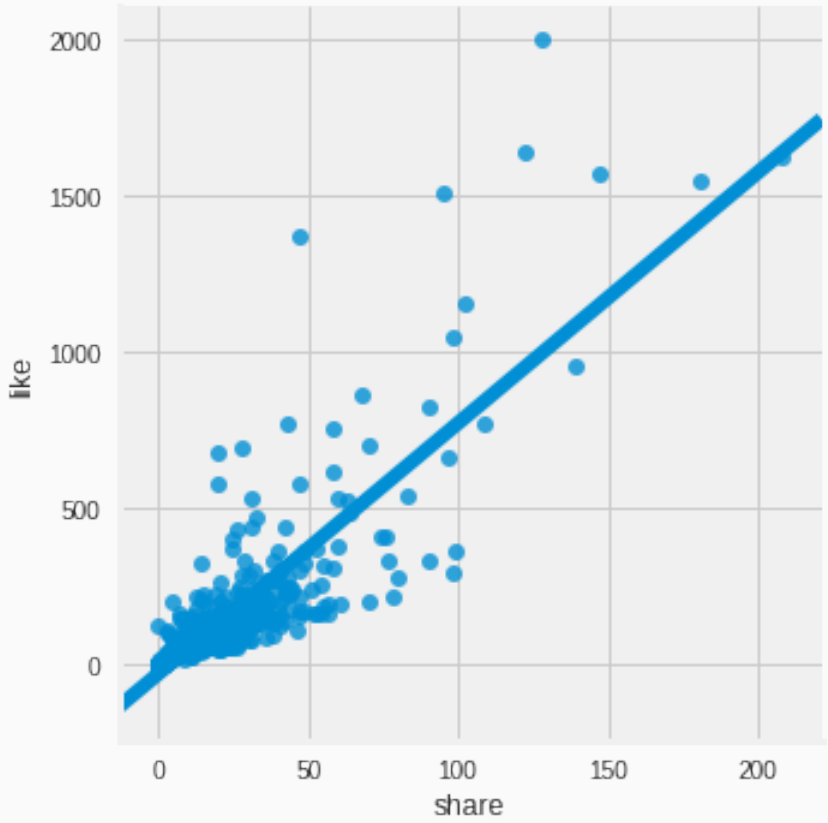
$$E(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \mathbf{w}\cdot\phi(x_i))^2$$

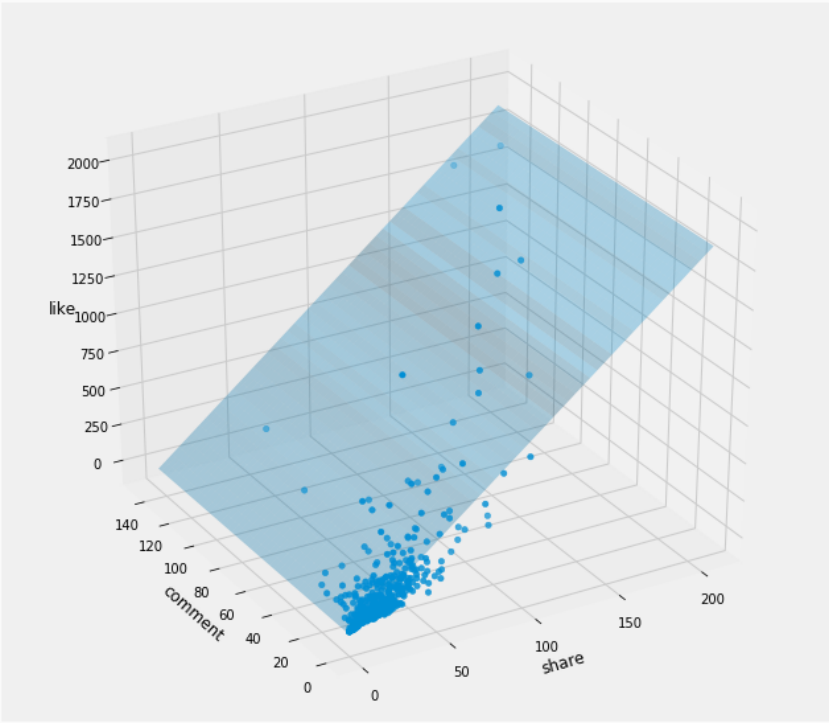
- Nilainya dapat dioptimasi dengan mencari turunan pertama, lalu atur nilainya menjadi 0, i.e. $\frac{\partial E}{\partial w} = 0$ atau $\nabla_{\mathbf{w}} E(\mathbf{w}) = 0$
- Solusi tertutupnya:

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

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

Perhatikan kembali





Apa kekurangan dari regresi linear sederhana seperti ini?

Non-linearity

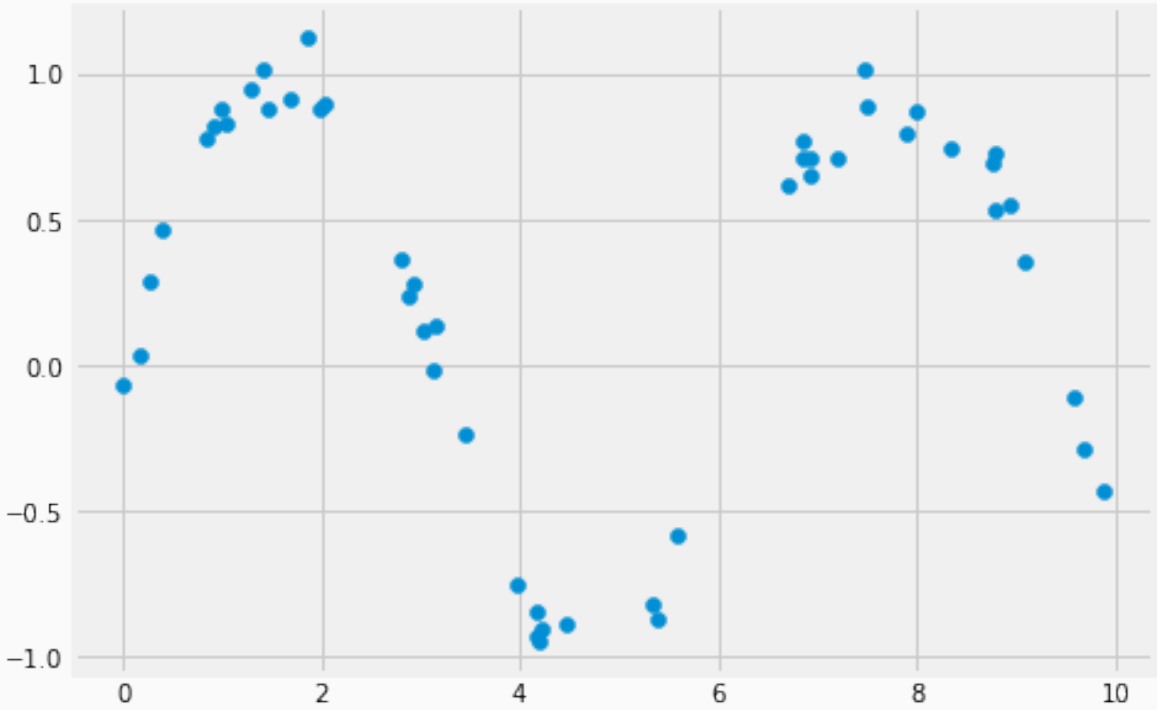


Figure 5: Data yang dihasilkan dari fungsi sin dengan *noise*

Bagaimana kalau datanya seperti ini?

Underfitting

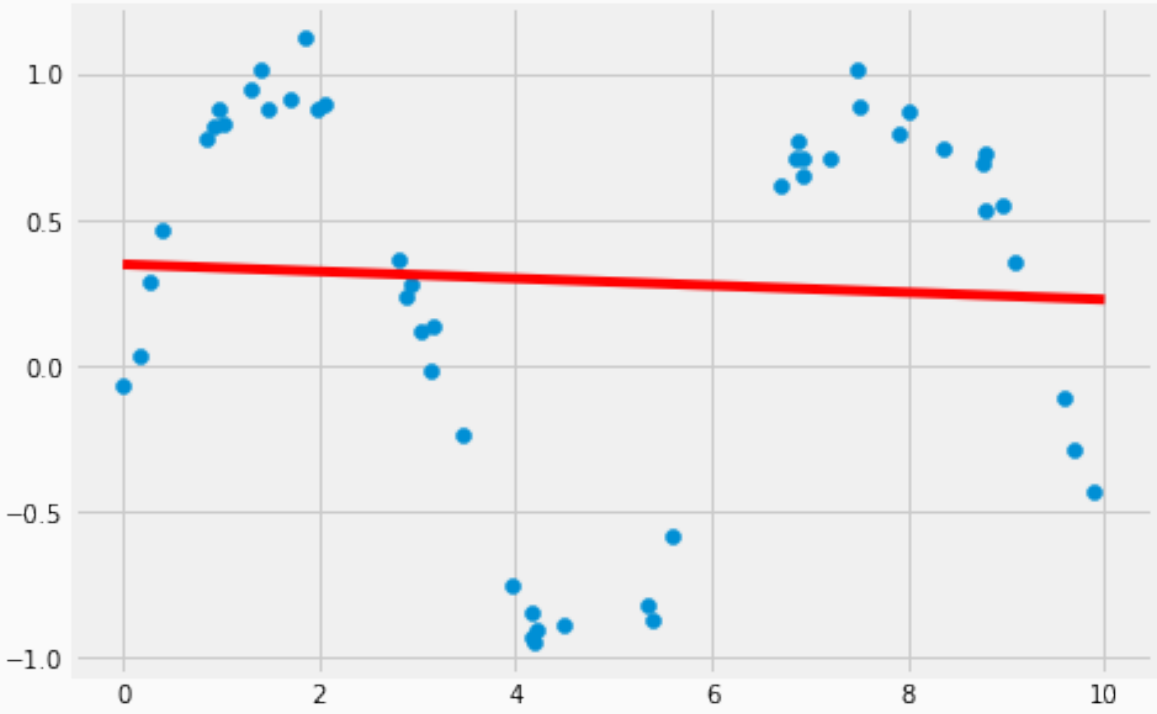


Figure 6: Hasil *fitting* regresi linear sederhana

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

Polynomial Basis Functions

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

$$y = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + ...$$

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.]])
```

Best-fit

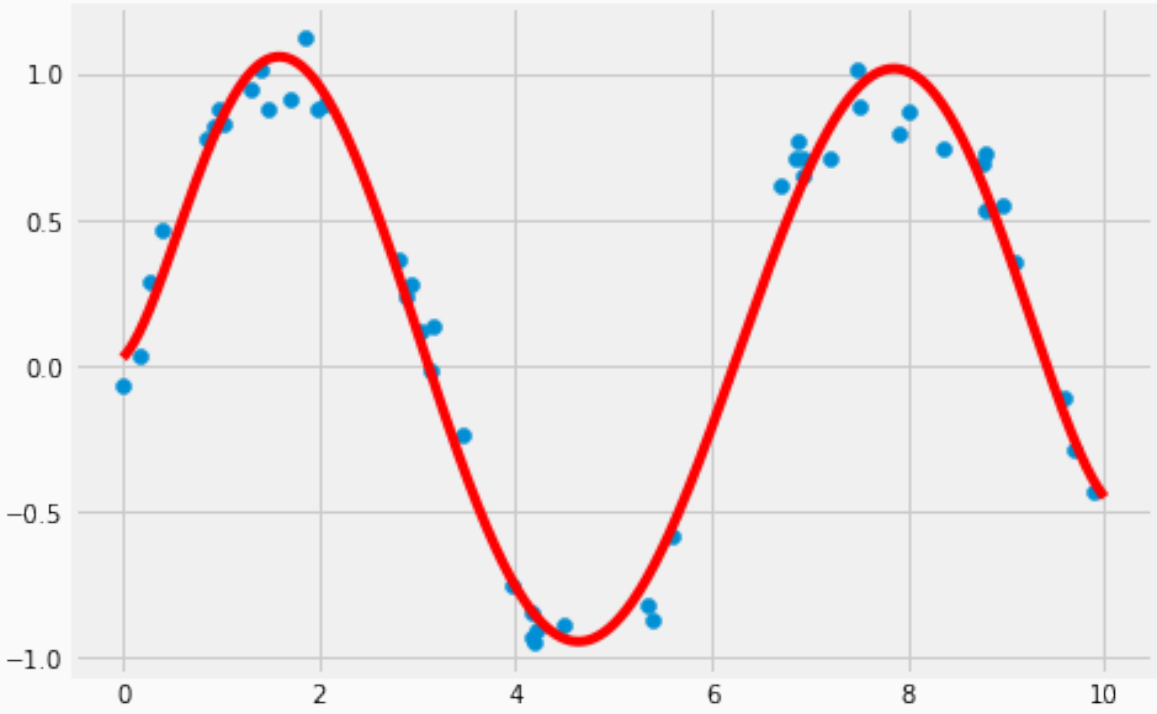


Figure 7: Hasil *fitting* fungsi basis polinomial $p = 7$

Apa yang terjadi jika p dibuat lebih besar?

Overfitting

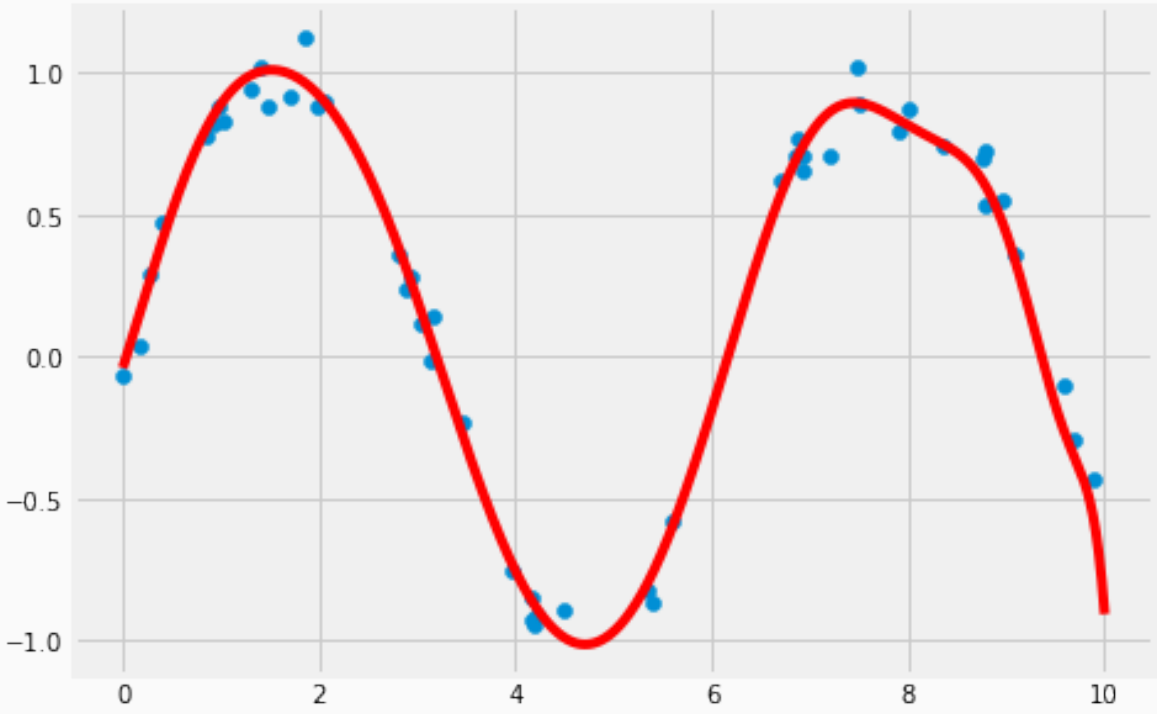


Figure 8: Hasil *fitting* fungsi basis polinomial $p = 15$

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 Logistik

Memprediksi kategori

- Apa yang harus dilakukan jika kita ingin memprediksi *kategori* alih-alih *nilai riil*?

Memprediksi kategori

- Apa yang harus dilakukan jika kita ingin memprediksi *kategori* alih-alih *nilai riil*?
- Contoh: Prediksi apakah komentar-komentar berikut termasuk *spam* atau *ham* (bukan spam) jika dilihat dari kemunculan kata-kata ‘order’ dan ‘password‘.

Memprediksi kategori

- Apa yang harus dilakukan jika kita ingin memprediksi *kategori* alih-alih *nilai riil*?
- Contoh: Prediksi apakah komentar-komentar berikut termasuk *spam* atau *ham* (bukan spam) jika dilihat dari kemunculan kata-kata 'order' dan 'password'.
- Kita asumsikan spam $= 1$ dan ham $= 0$. Bagaimana memaksa keluaran dari regresi linear $y \in (-\infty, \infty)$ menjadi $y \in \{0, 1\}$?

Logistic function

- Cara yang banyak digunakan adalah menggunakan fungsi sigmoid/logistik, i.e.

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Logistic function

- Cara yang banyak digunakan adalah menggunakan fungsi sigmoid/logistik, i.e.

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

- Karena z bernilai $-\infty$ sampai ∞, maka $\sigma(z)$ bernilai dari 0 sampai 1 $\sim$ probabilistik

Logistic function

- Cara yang banyak digunakan adalah menggunakan fungsi sigmoid/logistik, i.e.

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

- Karena z bernilai $-\infty$ sampai ∞, maka $\sigma(z)$ bernilai dari 0 sampai 1 $\sim$ probabilistik
- $p(y = 1|\mathbf{x}) = \sigma(f(\mathbf{x})) = \sigma(\mathbf{w}^T\mathbf{x}) = \sigma(w_0x_0 + w_1x_1)$

Logistic function

- Cara yang banyak digunakan adalah menggunakan fungsi sigmoid/logistik, i.e.

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

- Karena z bernilai $-\infty$ sampai ∞, maka $\sigma(z)$ bernilai dari 0 sampai 1 $\sim$ probabilistik
- $p(y = 1|\mathbf{x}) = \sigma(f(\mathbf{x})) = \sigma(\mathbf{w}^T\mathbf{x}) = \sigma(w_0x_0 + w_1x_1)$
- Karena probabilitas jumlahnya harus 1, maka

$$p(y = 0|\mathbf{x}) = 1 - p(y = 1|\mathbf{x})$$

Fungsi sigmoid/logistik

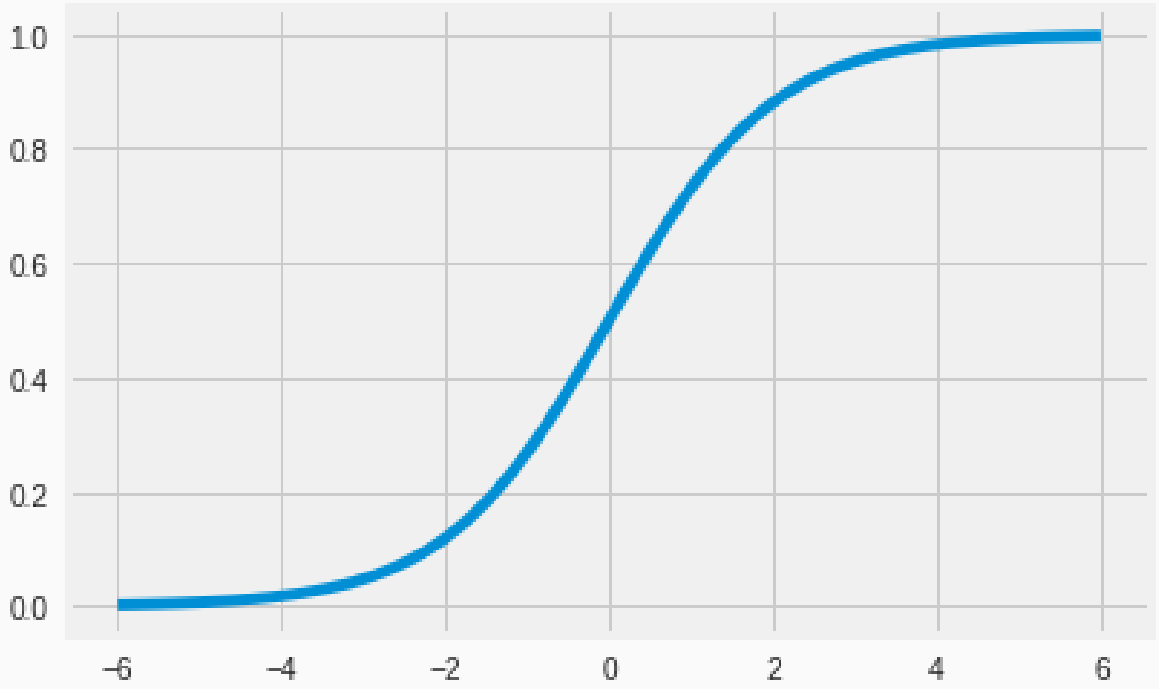


Figure 9: Fungsi sigmoid/logistik $\sigma(z) = \frac{1}{1+exp(-z)}$

Regresi logistik

- Dalam kasus satu variabel prediktor, kemiringan dari batas keputusan diatur oleh nilai w_1, sedangkan w_0 (*intercept*) hanya menggesernya

Regresi logistik

- Dalam kasus satu variabel prediktor, kemiringan dari batas keputusan diatur oleh nilai w_1, sedangkan w_0 (*intercept*) hanya menggesernya
- Batas keputusan yang dihasilkan akan berupa *hyperplane* yang akan tegak lurus terhadap vektor $\mathbf{w}$

Regresi logistik

- Dalam kasus satu variabel prediktor, kemiringan dari batas keputusan diatur oleh nilai w_1, sedangkan w_0 (*intercept*) hanya menggesernya
- Batas keputusan yang dihasilkan akan berupa *hyperplane* yang akan tegak lurus terhadap vektor $\mathbf{w}$
- Dari $\mathbf{w}$, kita bisa menggambarkan batas keputusan (*decision boundary*) ketika $p(y = 1|\mathbf{x}) = p(y = 0|\mathbf{x}) = 0.5$, i.e. $\mathbf{w}^T\mathbf{x} = 0$

Regresi logistik

- Dalam kasus satu variabel prediktor, kemiringan dari batas keputusan diatur oleh nilai w_1, sedangkan w_0 (*intercept*) hanya menggesernya
- Batas keputusan yang dihasilkan akan berupa *hyperplane* yang akan tegak lurus terhadap vektor $\mathbf{w}$
- Dari $\mathbf{w}$, kita bisa menggambarkan batas keputusan (*decision boundary*) ketika $p(y = 1|\mathbf{x}) = p(y = 0|\mathbf{x}) = 0.5$, i.e. $\mathbf{w}^T\mathbf{x} = 0$
- Kita perlu mencari nilai $\mathbf{w}$

Batas keputusan

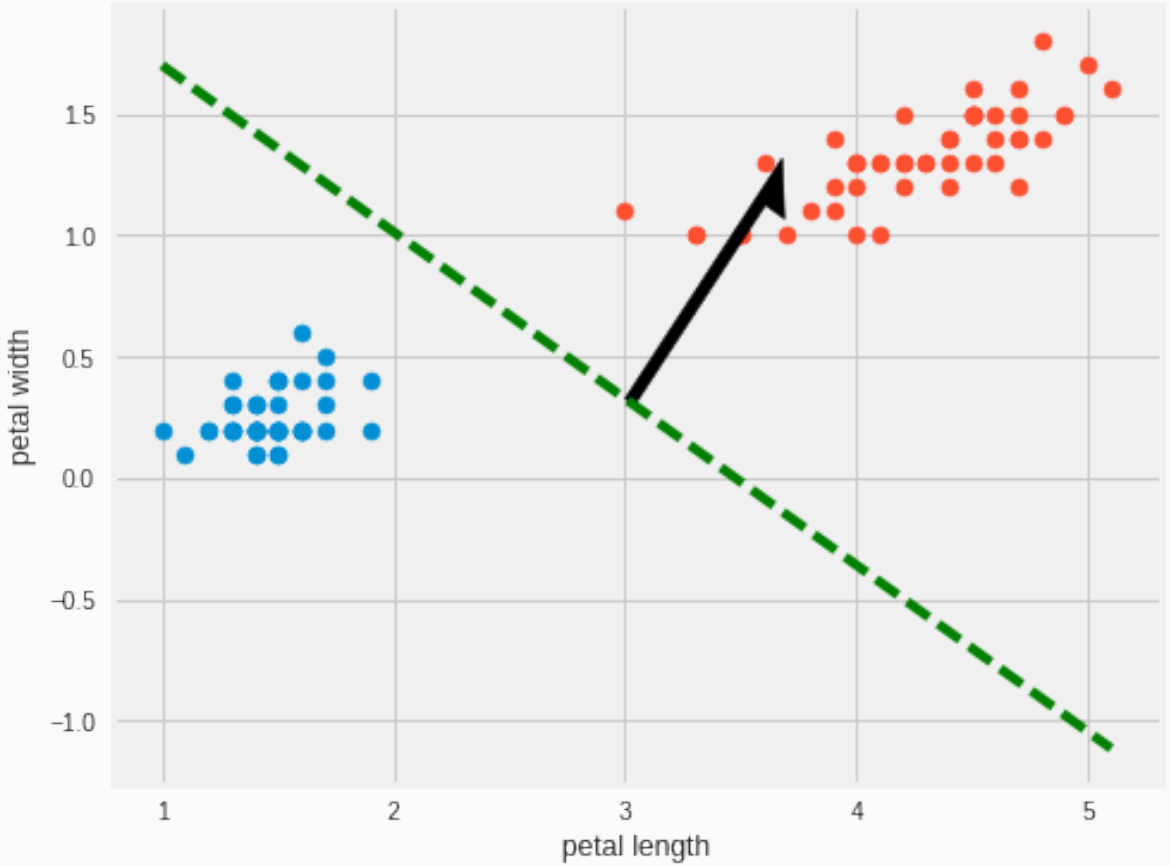


Figure 10: Batas keputusan (hijau) yang dibentuk dari vektor bobot **w** (hitam) untuk prediktor dengan dua variabel

Likelihood (non-examinable)

- Asumsi i.i.d.
- Dataset $\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_n, y_n)\}$
- *Likelihood*-nya menjadi

$$p(\mathcal{D}|\mathbf{w}) = \prod_{i=1}^{N} p(y = y_i|\mathbf{x}_i, \mathbf{w})$$
$$= \prod_{i=1}^{N} p(y = 1|\mathbf{x}_i, \mathbf{w})^{y_i}(1 - p(y = 1|\mathbf{x}_i, \mathbf{w}))^{1-y_i}$$

- *Log likelihood* $L(\mathbf{w}) = log\ p(\mathcal{D}|\mathbf{w})$

$$L(\mathbf{w}) = \sum_{i=1}^{N} y_i log\ \sigma(\mathbf{w}^T\mathbf{x}_i) + (1 - y_i) log(1 - \sigma(\mathbf{w}^T\mathbf{x}_i))$$

Solusi

- Nilai optimum untuk kasus ini unik, i.e. *convex*
- Untuk memaksimalkan nilainya, gunakan gradien

$$\frac{\partial L}{\partial w_j} = \sum_{i=1}^{N} (y_i - \sigma(\mathbf{w}^T \mathbf{x}_i)) x_{ij}$$

- Tidak ada solusi tertutup sehingga harus menggunakan *optimasi numerik*, e.g. dengan *gradient descent*

Optimasi

Mengapa dinamakan *machine learning*?

Mengapa melakukan optimasi?

- Belajar $\rightarrow$ masalah optimasi kontinu
- Contoh: regresi linear, regresi logistik, jaringan saraf tiruan, SVM
- Salah satu caranya adalah dengan *maximum likelihood*

“Berapa peluangnya kita melihat data ini jika diketahui parameternya?”

Cara melakukan optimasi

- Menggunakan fungsi galat/error $E(\mathbf{w})$ yang akan diminimalkan
- e.g. dapat berupa $-L(\mathbf{w})$
- Beda nilai $\mathbf{w}$, beda besar error
- Belajar $\equiv$ menuruni permukaan error

Menuruni permukaan fungsi error

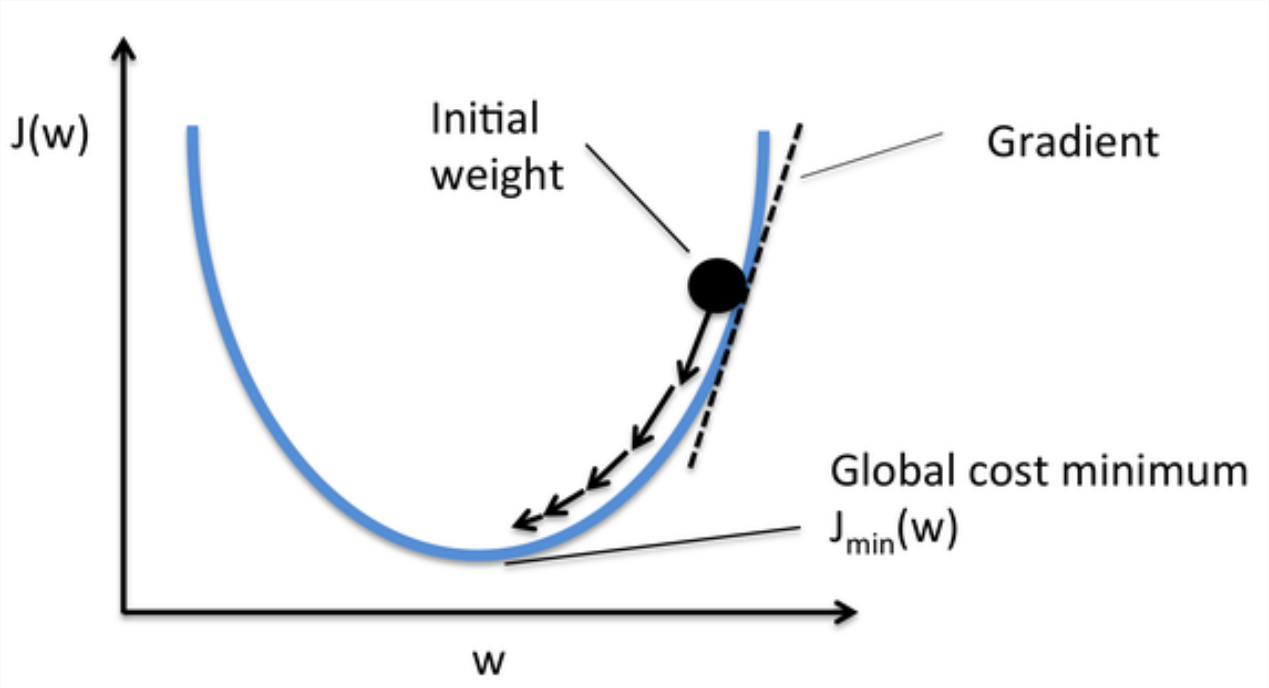


Figure 11: Menuruni lembah fungsi error J(w) [Raschka, 2015]

Gradient descent

begin
  Inisialisasi $\mathbf{w}$
  while $E(\mathbf{w})$ *masih terlalu besar* **do**
    Hitung $\mathbf{g} \leftarrow \nabla_{\mathbf{w}} E(\mathbf{w})$
    $\mathbf{w} \leftarrow \mathbf{w} - \eta \mathbf{g}$
  end
  return $\mathbf{w}$
end

Algorithm 1: Melatih dengan gradient descent

Learning rate

- η (baca: "eta") dikenal sebagai *step size* atau *learning rate* dengan nilai $\eta > 0$
- η terlalu kecil $\rightarrow$ lambat
- η terlalu besar $\rightarrow$ tidak stabil

Batch vs online

- Untuk data yang sedikit, kita bisa menjumlahkan semua error sebelum memperbarui nilai $\mathbf{w}$ (*batch*)

Batch vs online

- Untuk data yang sedikit, kita bisa menjumlahkan semua error sebelum memperbarui nilai **w** (*batch*)
- Bagaimana untuk 10 juta data?

Batch vs online

- Untuk data yang sedikit, kita bisa menjumlahkan semua error sebelum memperbarui nilai **w** (*batch*)
- Bagaimana untuk 10 juta data?
- Ternyata, kita bisa memperbarui nilai **w** untuk setiap satu data (*online*)

Gradient descent (batch)

begin
  Inisialisasi $\mathbf{w}$
  while $E(\mathbf{w})$ *masih terlalu besar* **do**
    Hitung $\mathbf{g} \leftarrow \sum_{i=1}^{N} \nabla_{\mathbf{w}} E_i(\mathbf{w})$
    $\mathbf{w} \leftarrow \mathbf{w} - \eta \mathbf{g}$
  end
  return $\mathbf{w}$
end

Algorithm 2: Melatih dengan batch gradient descent

Stochastic gradient descent

begin
  Inisialisasi $\mathbf{w}$
  while $E(\mathbf{w})$ *masih terlalu besar* **do**
    Pilih j sebagai integer acak antara 1..N
    Hitung $\mathbf{g} \leftarrow \nabla_{\mathbf{w}} E_j(\mathbf{w})$
    $\mathbf{w} \leftarrow \mathbf{w} - \eta \mathbf{g}$
  end
  return $\mathbf{w}$
end

Algorithm 3: Stochastic gradient descent (SGD)

Kelebihan dan kekurangan

- **Batch** lebih *powerful*
- **Batch** lebih mudah dianalisis
- **Online** lebih praktikal untuk data yang besar
- **Online** dapat melompati optimum lokal

Pengembangan gradient descent (non-examinable)

- "Why **Momentum** Really Works" [Goh, 2017]
- **Performance-dependent** η, e.g. "NewBOB": η berubah menjadi setengahnya saat validation set tidak menjadi lebih baik
- **Time-dependent schedules**, e.g. eksponensial: $\eta(t) = \eta(0)exp(-t/r)$ ($r \sim$ ukuran data latih)

Tentang metode optimasi

- Masih banyak metode optimasi yang tidak dibahas, e.g. linear programming, Newton's method, dll.
- Optimasi merupakan bidang matematika yang kompleks
- Masalah convex: optimum global. Non-convex: optimum lokal.
- Pahami mengapa *gradient descent* bisa mengalami masalah

Summary

- 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*
- Regresi linear dapat diubah untuk memprediksi data kategorikal dengan menggunakan fungsi sigmoid/logistik
- Proses optimasi merupakan bagian penting dari *machine learning* yang dapat dilakukan secara numerik, e.g. *gradient descent*

Pertemuan berikutnya

- Neural networks
- Gradient descent dan backpropagation
- Aplikasi pada *computer vision*

References

Jake VanderPlas (2016)

In Depth: Linear Regression

Python Data Science Handbook

Sebastian Raschka (2015)

Single-Layer Neural Networks and Gradient Descent

`http://sebastianraschka.com/Articles/2015_singlelayer_neurons.html`

Gabriel Goh (2017)

“Why Momentum Really Works”

Distill

Thank you