Model Linear

Ali Akbar Septiandri

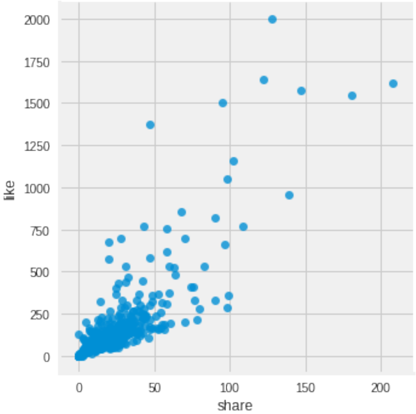
Universitas Al Azhar Indonesia

September 30, 2018

Bahan Bacaan

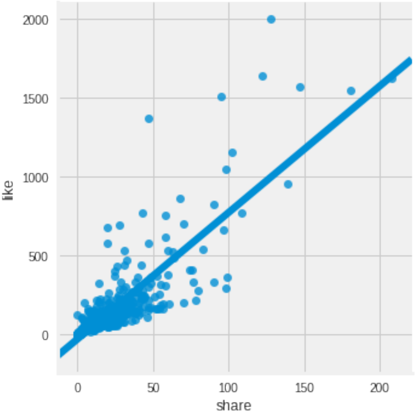
1. Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2016). Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann. (Section 4.6 Linear Models)
2. VanderPlas, J. (2016). Python Data Science Handbook. (In Depth: Linear Regression) `http://nbviewer.jupyter.org/github/jakevdp/PythonDataScienceHandbook/blob/master/notebooks/05.06-Linear-Regression.ipynb`
3. Murray, I. (2016). MLPR class notes. (Linear Regression; Regression and Gradients; Logistic Regression) `http://www.inf.ed.ac.uk/teaching/courses/mlpr/2016/notes/` (graduate level)

Prediksi hubungan antara dua variabel



Gambar: Data hubungan antara ‘share’ dengan ‘like’ pada Facebook

Prediksi hubungan antara dua variabel



Gambar: 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*, sedangkan b dikenal dengan nama *intercept*.

Notasi lain

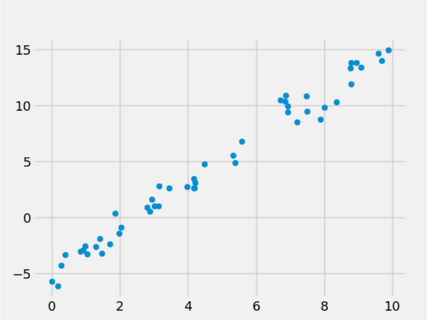
$$y = w_0 + w_1 x_1$$

dengan w adalah bobot atau koefisien.

Simple Linear Regression

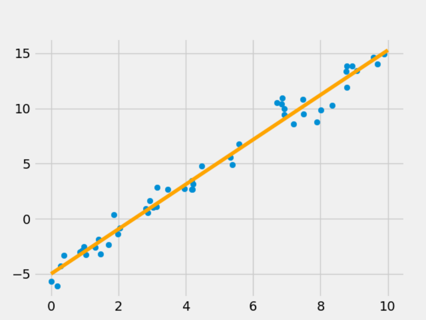
Example

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



Gambar: Data yang dimunculkan secara acak [VanderPlas, 2016]

Mencocokkan Garis



Gambar: 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

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

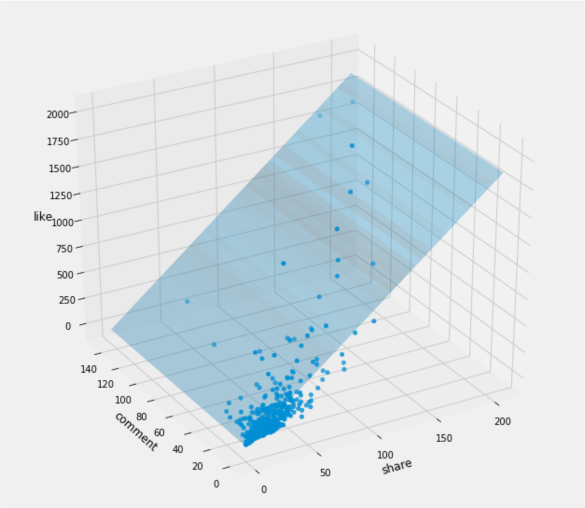
dengan $x_0 = 1$

Notasi matriks-vektor

$$y = \phi\mathbf{w}$$

dengan $\phi = (1, \mathbf{x}^T)$

Regresi linear untuk dua variabel



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*.

Kita sudah tahu nilai y dan ϕ,
tapi berapa nilai $\mathbf{w}$?

Nyatanya, kita tidak bisa mencari nilai ϕ^{-1}

Loss Function

- ϕ bukan matriks bujur sangkar dan datanya mengandung *noise*
- Harus menggunakan *loss function* $O(\mathbf{w})$ yang dapat diminimalkan
- Pilihan umum: *squared error*

$$
\begin{aligned}
O(\mathbf{w}) &= \sum_{i=1}^{n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2 \\
&= (\mathbf{y} - \phi\mathbf{w})^T (\mathbf{y} - \phi\mathbf{w})
\end{aligned}
$$

Solusi

- Jawaban: Minimalkan $O(\mathbf{w}) = \sum_{i=1}^{n}(y_i - \mathbf{w}^T\mathbf{x}_i)^2$ dengan mencari turunan parsial yang diatur sama dengan 0
- Solusi analitis:

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

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

Break

Bagaimana kalau saya hanya mau memprediksi nilai biner?

Mengubah Keluaran

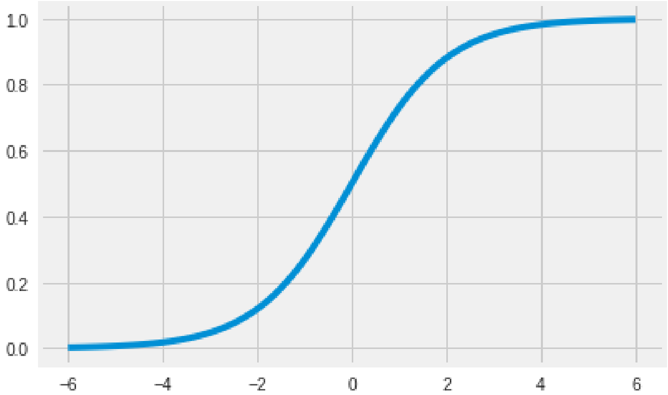
- Berdasarkan keluaran regresi linear, kita bisa memaksanya menjadi [0, 1]
- Gunakan fungsi sigmoid:

$$P(y = 1|\mathbf{x}) = f(\mathbf{x}; \mathbf{w}) = \sigma(\mathbf{w}^T\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^T\mathbf{x}}}$$

- Nilai [0, 1] dapat diartikan sebagai probabilitas dari kelas
- Karena probabilitas harus memiliki total 1, maka

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

Fungsi Sigmoid

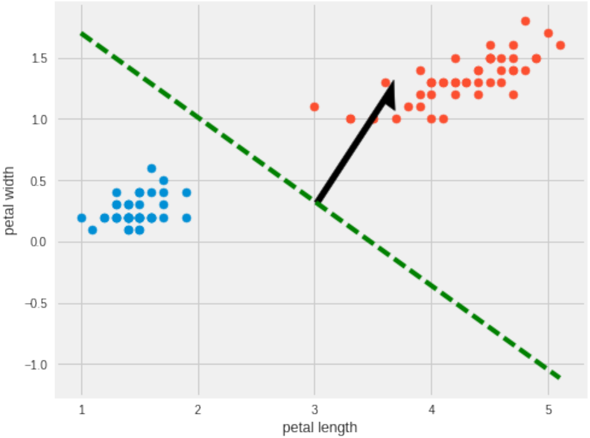


Gambar: Fungsi sigmoid/logistik $\sigma(z) = \frac{1}{1+exp(-z)}$

Decision Boundary

- $\sigma(z) = 0.5$ saat $z = 0$ sehingga batas keputusannya diberikan oleh $\mathbf{w}^T\mathbf{x} = 0$
- Batas keputusannya merupakan $M - 1$ *hyperplane* untuk masalah M dimensi
- Kita perlu mencari nilai $\mathbf{w}$

Decision Boundary



Gambar: Batas keputusan dan vektor bobot untuk klasifikasi dua kelas

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*

Mengapa dinamakan *machine learning*?

Alasan Melakukan Optimasi

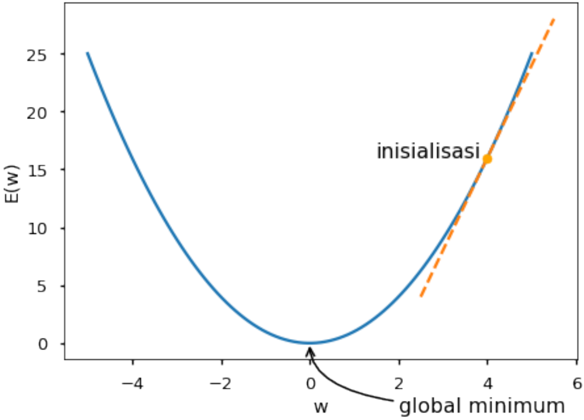
- Belajar $\rightarrow$ masalah optimasi kontinu
- Contoh: regresi linear, regresi logistik, jaringan saraf tiruan, SVM
- Salah satu caranya adalah dengan *maximum likelihood*

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

Cara Melakukan Optimasi

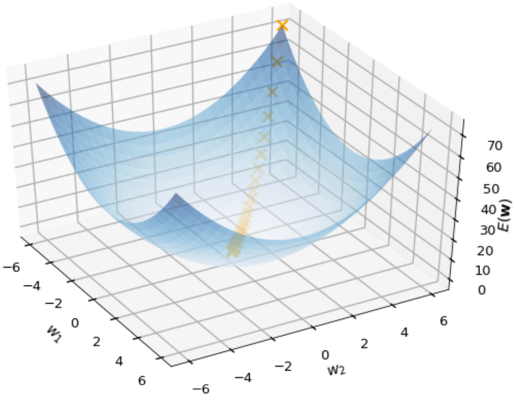
- Menggunakan fungsi galat/error $E(\mathbf{w})$ yang akan diminimalkan
- e.g. dapat berupa $-L(\mathbf{w})$
- Beda nilai $\mathbf{w}$, beda besar error
- Belajar $\equiv$ menuruni permukaan error

Menuruni Permukaan Fungsi Error



Gambar: Menuruni lembah fungsi error $E(w)$

Menuruni Permukaan Fungsi Error



Gambar: Menuruni lembah fungsi error $E(\mathbf{w})$

Gradient Descent

begin

Inisialisasi $\mathbf{w}$

while $E(\mathbf{w})$ *masih terlalu besar* **do**

Hitung $\mathbf{g} \leftarrow \frac{\partial E}{\partial \mathbf{w}}$

$\mathbf{w} \leftarrow \mathbf{w} - \eta \mathbf{g}$

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 $\mathbf{w}$ (*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 $\mathbf{g} \leftarrow \sum_{i=1}^{N} \frac{\partial E_i}{\partial \mathbf{w}}$

    $\mathbf{w} \leftarrow \mathbf{w} - \eta \mathbf{g}$

  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 $\mathbf{g} \leftarrow \frac{\partial E_j}{\partial \mathbf{w}}$

$\mathbf{w} \leftarrow \mathbf{w} - \eta \mathbf{g}$

end

return $\mathbf{w}$

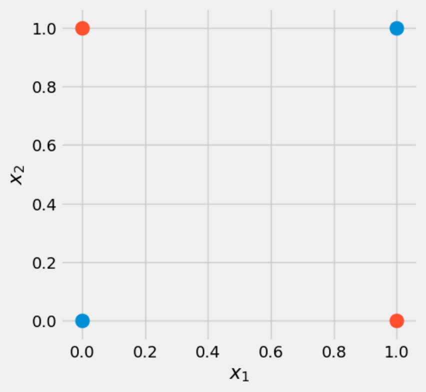
end

Algorithm 3: Stochastic gradient descent (SGD)

Kelebihan dan Kekurangan

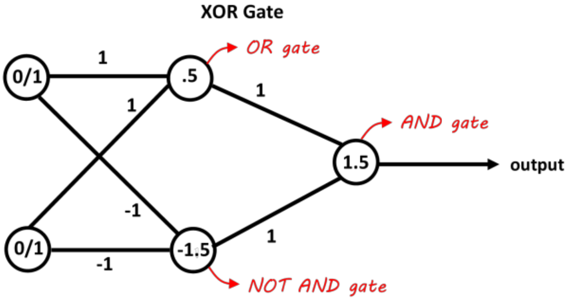
- **Batch** lebih *powerful*
- **Batch** lebih mudah dianalisis
- **Online** lebih praktikal untuk data yang besar
- **Online** dapat melompati optimum lokal

Kasus



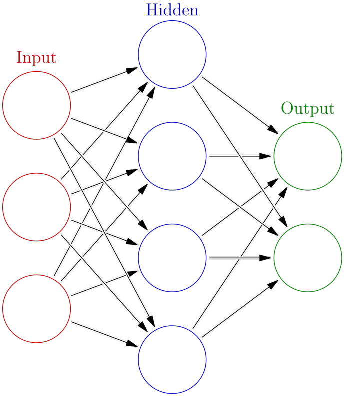
Gambar: Bagaimana kalau datanya seperti ini?

XOR Net



Gambar: Menggunakan single hidden layer untuk XOR

Deep Learning



Gambar: Pada dasarnya, *deep learning* "hanya" model linear yang ditumpuk

Pertemuan berikutnya

- Single layer networks
- Backpropagation
- Softmax
- MNIST digit recognition

Referensi

Jake VanderPlas (2016)
In Depth: Linear Regression
Python Data Science Handbook

Terima kasih