Klasifikasi: Regresi Logistik

Ali Akbar Septiandri

October 3, 2017

Universitas Al Azhar Indonesia

Daftar isi

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

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

Batas keputusan

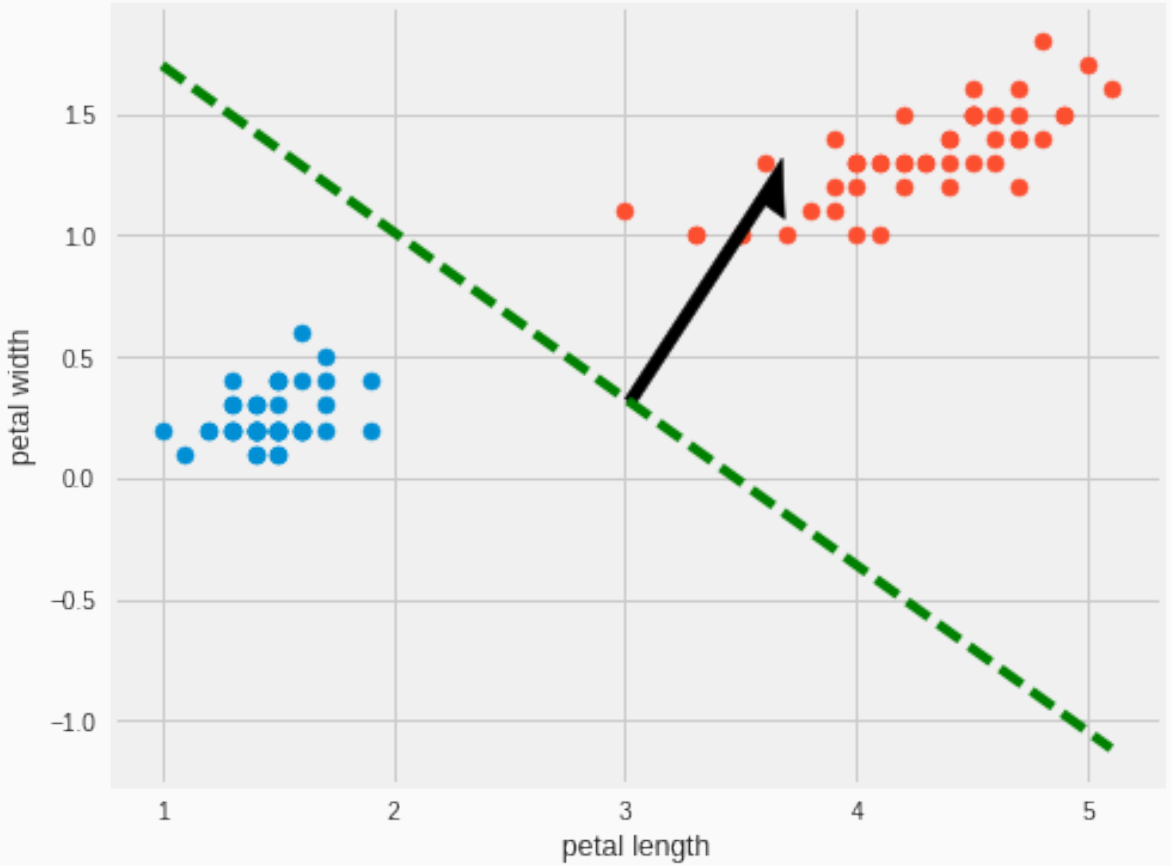

Figure 1: Batas keputusan (hijau) yang dibentuk dari vektor bobot **w** (hitam) untuk prediktor dengan dua variabel

Fungsi sigmoid/logistik

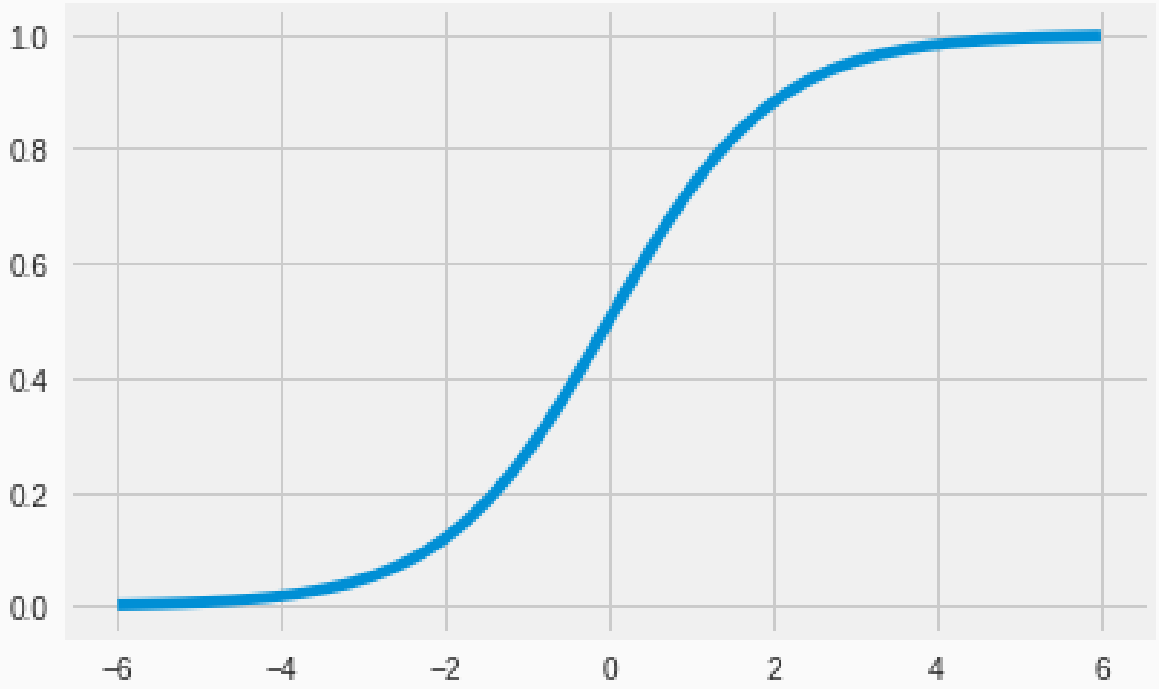

Figure 2: Fungsi sigmoid/logistik $\sigma(z) = \frac{1}{1+exp(-z)}$

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 $\mathbf{w}$

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*?

Menuruni permukaan fungsi error

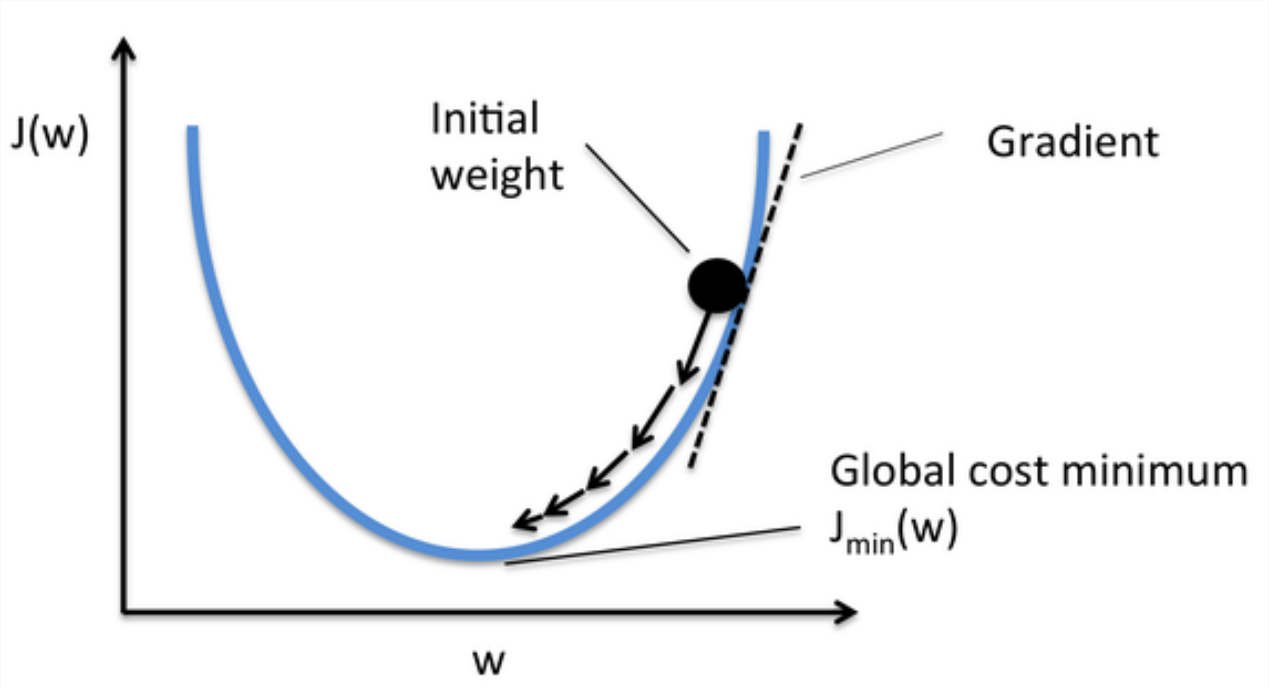

Figure 3: Menuruni lembah fungsi error J(w) [Raschka, 2015]

Mengapa melakukan optimasi?

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

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

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 **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)

Regresi linear dengan gradient descent

https://github.com/aliakbars/uai/blob/gh-pages/images/line.gif

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

Ikhtisar

- 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*
- *Gradient descent* bisa dilakukan secara *batch*, *online*, atau di antaranya
- Metode optimasi merupakan bidang matematika yang kompleks sehingga tidak perlu eksplorasi lebih jauh, gunakan pustaka yang ada

Pertemuan berikutnya

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

Referensi

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

Terima kasih