4

Algoritma Dasar

> "It is a capital mistake to
> theorize before one has data."
>
> Arthur Conan Doyle

Sebelum masuk ke algoritma *machine learning* yang cukup modern/ matematis, kami akan memberi contoh algoritma yang lebih mudah yaitu **Naive Bayes**, **K-means**, dan **K-nearest-neighbor**. Algoritma-algoritma ini tergolong *non-parametrik*. Bab ini akan memuat contoh sederhana *supervised* dan *unsupervised learning*. Mudah-mudahan bab ini memberikan kamu gambaran aplikasi *machine learning* sederhana.

4.1 Naive Bayes

Naive Bayes adalah algoritma *supervised learning* yang sangat sederhana [26]. Idenya mirip dengan probabilitas bayesian pada bab 2. Secara formal, persamaan *Naive Bayes* untuk klasifikasi diberikan pada persamaan 4.1 dimana c_i adalah suatu nilai kelas, C adalah pilihan kelas (himpunan), t adalah fitur (satu fitur, bukan *feature vector*) dan F adalah banyaknya fitur. Kita memprediksi kelas berdasarkan probabilitas kemunculan nilai fitur pada kelas tersebut.

Pertama, kita hitung *likelihood* suatu *feature vector* diklasifikasikan ke kelas tertentu berdasarkan bagaiman probabilitas korespondensi fitur-fiturnya terhadap kelas tersebut (persamaan 4.1). Kemudian, kita normalisasi *likelihood* semua kelas untuk mendapatkan probabilitas *class-assignment* (softmax– persamaan 4.2). Akhirnya, kita pilih kelas dengan probabilitas tertinggi (persamaan 4.3).

$$\text{likelihood}(c_i) = P(c_i) \prod_{f=1}^{F} P(t_f|c_i) \tag{4.1}$$

id	outlook	temperature	humidity	windy	play (class)
1	sunny	hot	high	false	no
2	sunny	hot	high	true	no
3	overcast	hot	high	false	yes
4	rainy	mild	high	false	yes
5	rainy	cool	normal	false	yes
6	rainy	cool	normal	true	no
7	overcast	cool	normal	true	yes
8	sunny	mild	high	false	no
9	sunny	cool	normal	false	yes
10	rainy	mild	normal	false	yes
11	sunny	mild	normal	true	yes
12	overcast	mild	high	true	yes
13	overcast	hot	normal	false	yes
14	rainy	mild	high	true	no

Tabel 4.1: Contoh dataset *play tennis* (UCI machine learning repository)

	outlook		temperature			humidity			windy		play (class)	
	yes	no		yes	no		yes	no		yes	no	yes	no
sunny	2	3	hot	2	3	high	3	4	false	6	2	9	5
overcast	4	0	mild	4	2	normal	6	1	true	3	3		
rainy	3	2	cool	3	1								

Tabel 4.2: Frekuensi setiap nilai atribut

	outlook		temperature			humidity			windy		play (class)	
	yes	no		yes	no		yes	no		yes	no	yes	no
sunny	2/9	3/5	hot	2/9	3/5	high	3/9	4/5	false	6/9	2/5	9/14	5/14
overcast	4/9	0/5	mild	4/9	2/5	normal	6/9	1/5	true	3/9	3/5		
rainy	3/9	2/5	cool	3/9	1/5								

Tabel 4.3: Probabilitas setiap nilai atribut

$$P_{assignment}(c_i) = \frac{\text{likelihood}(c_i)}{\sum_{c_j \epsilon C} \text{likelihood}(c_j)} \tag{4.2}$$

$$\hat{c_i} = \arg\max_{c_i \epsilon C} P_{assignment}(c_i) \tag{4.3}$$

Agar mendapatkan gambaran praktis, mari kita bangun model Naive Bayes untuk Tabel 4.1. Tabel ini disebut sebagai **dataset**, yaitu memuat *entry* data (tiap baris disebut sebagai ***instance***). Kita anggap Tabel 4.1 sebagai ***training data***. Untuk menghitung probabilitas, pertama-tama kita hitung terlebih dahulu frekuensi nilai atribut seperti pada Tabel 4.2, setelah itu kita bangun model probabilitasnya seperti pada Tabel 4.3.

Untuk menguji kebenaran model yang telah kita bangun, kita menggunakan ***testing data***, diberikan pada Tabel 4.4. *testing data* berisi *unseen example* yaitu contoh yang tidak ada pada *training data*.

id	outlook	temperature	humidity	windy	play (class)
1	sunny	cool	high	true	no

Tabel 4.4: Contoh testing data *play tennis* [7]

$$
\begin{aligned}
\text{likelihood}(play = yes) &= P(yes)P(sunny \mid yes)P(cool \mid yes)P(high \mid yes) \\
&\quad P(true \mid yes) \\
&= \frac{9}{14} * \frac{2}{9} * \frac{3}{9} * \frac{3}{9} * \frac{3}{9} \\
&= 0.0053 \\
\text{likelihood}(play = no) &= P(no)P(sunny \mid no)P(cool \mid no)P(high \mid no) \\
&\quad P(true \mid no) \\
&= \frac{5}{14} * \frac{3}{5} * \frac{1}{5} * \frac{4}{5} * \frac{3}{5} \\
&= 0.0206 \\
P_{assignment}(play = yes) &= \frac{\text{likelihood}(play = yes)}{\text{likelihood}(play = yes) + \text{likelihood}(play = no)} \\
&= \frac{0.0053}{0.0053 + 0.0206} \\
&= 0.205 \\
P_{assignment}(play = no) &= \frac{\text{likelihood}(play = no)}{\text{likelihood}(play = yes) + \text{likelihood}(play = no)} \\
&= \frac{0.0206}{0.0053 + 0.0206} \\
&= 0.795
\end{aligned}
$$

Karena $P_{assignment}(play = no) > P_{assignment}(play = yes)$ maka diputuskan bahwa kelas untuk *unseen example* adalah $play = no$. Proses klasifikasi untuk data baru sama seperti proses klasifikasi untuk *testing data*, yaitu kita ingin menebak kelas data. Karena model berhasil menebak kelas pada *training data* dengan tepat, akurasi model adalah 100% (kebetulan contohnya hanya ada satu).

Perhatikan! Kamu mungkin berpikir kita dapat langsung menggunakan *likelihood* untuk mengklasifikasi, karena probabilitas dengan *likelihood* terbesar akan dipilih. Hal ini cukup berbahaya apabila *likelihood* untuk masing-masing kelas memiliki jarak yang cukup dekat. Sebagai contoh, menghitung probabilitas apabila (kasus abstrak) *likelihood*=$\{0.70, 0.60\}$, sehingga probabilitas kelas menjadi=$\{0.54, 0.46\}$. Karena perbedaan probabilitas kelas relatif tidak terlalu besar (contoh ini adalah penyederhanaan), kita mungkin harus

berpikir kembali untuk mengklasifikasikan *instance* ke kelas pertama. Hal ini berkaitan dengan seberapa yakin kamu mengklasifikasikan suatu sampel ke kelas tertentu. Perbedaan *likelihood* yang besar menandakan keyakinan, sementara perbedaan yang tipis menandakan ketidakyakinan.

Perhatikan, jumlah *likelihood* mungkin lebih dari 1. Sementara, probabilitas atau jumlahnya berada pada range $[0, 1]$ $(0 \leq P \leq 1)$. Pada contoh sebelumnya, nilai *likelihood* diubah menjadi bentuk probabilitas dengan menggunakan teknik *softmax*. Fungsi *softmax* berbentuk seperti pada persamaan 4.2.

4.2 K-means

Pada *supervised learning* kita mengetahui kelas data untuk setiap *feature vector*, sedangkan untuk *unsupervised learning* kita tidak tahu. Tujuan *unsupervised learning* salah satunya adalah melakukan ***clustering***. Yaitu mengelompokkan data-data dengan karakter mirip. Untuk melakukan pembelajaran menggunakan **K-means** [27], kita harus mengikuti langkah-langkah sebagai berikut:

1. Tentukan jumlah kelompok yang kita inginkan.
2. Inisiasi **centroid** untuk setiap kelompok (pada bab ini, secara acak). Centroid adalah data yang merepresentasikan suatu kelompok (ibaratnya ketua kelompok).
3. Hitung kedekatan suatu data terhadap *centroid*, kemudian masukkan data tersebut ke kelompok yang **centroid**-nya memiliki sifat terdekat dengan dirinya.
4. Pilih kembali **centroid** untuk masing-masing kelompok, yaitu dari anggota kelompok tersebut (semacam memilih ketua yang baru).
5. Ulangi langkah-langkah sebelumnya sampai tidak ada perubahan anggota untuk semua kelompok.

id	rich	intelligent	good looking
1	yes	yes	yes
2	yes	no	no
3	yes	yes	no
4	no	no	no
5	no	yes	no
6	no	no	yes

Tabel 4.5: Contoh dataset orang kaya

Perhatikan Tabel 4.5, kita akan mengelompokkan data pada tabel tersebut menjadi dua *clusters* (dua kelompok) yaitu k_1, k_2 menggunakan algoritma

id	perbedaan dengan $centroid_1$	perbedaan dengan $centroid_2$	assignment
2	2	2	k_1
3	1	3	k_1
4	3	1	k_2
5	2	2	k_1

Tabel 4.6: *Assignment* K-means langkah 1

id	perbedaan dengan $centroid_1$	perbedaan dengan $centroid_2$	assignment
1	2	3	k_1
3	1	3	k_1
5	3	1	k_2
6	2	2	k_1

Tabel 4.7: *Assignment* K-Means langkah 2

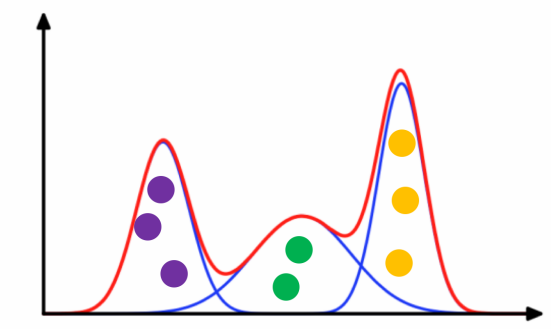
K-means. Pertama-tama kita inisiasi centroid secara acak, id_1 untuk k_1 dan id_6 untuk k_2. Kita hitung kedekatan data lainnya terhadap **centroid**. Untuk mempermudah contoh, kita hitung perbedaan data satu dan lainnya dengan menghitung perbedaan nilai atribut (nilai atributnya sama atau tidak). Apabila perbedaan suatu data terhadap kedua centroid bernilai sama, kita masukkan ke kelas dengan nomor urut lebih kecil.

Setelah langkah ini, kelompok satu beranggotakan $\{id_1, id_2, id_3, id_5\}$ sementara kelompok dua beranggotakan $\{id_4, id_6\}$. Kita pilih kembali centroid untuk masing-masing kelompok yang mana berasal dari anggota kelompok itu sendiri. Misal kita pilih secara acak,[1] centroid untuk kelompok pertama adalah id_2 sementara untuk kelompok kedua adalah id_4. Kita hitung kembali *assignment* anggota kelompok yang ilustrasinya dapat dilihat pada Tabel 4.7. Hasil langkah ke-2 adalah perubahan anggota kelompok, $k_1 = \{id_1, id_2, id_3, id_5\}$ dan $k_2 = \{id_4, id_6\}$. Anggap pada langkah ke-3 kita memilih kembali id_2 dan id_4 sebagai centroid masing-masing kelompok sehingga tidak ada perubahan keanggotaan.

Bila kamu membaca buku literatur lain, kemungkinan besar akan dijelaskan bahwa *clustering* itu memiliki **hubungan erat dengan *Gaussian Mixture Model***. Secara sederhana, **satu *cluster* (atau satu kelas)** sebenarnya seolah-olah dapat dipisahkan dengan kelas lainnya oleh distribusi gaussian. Perhatikan Gambar 4.1! Suatu *cluster* atau kelas, seolah olah "dibungkus" oleh suatu distribusi gaussian. Distribusi seluruh dataset dapat diaproksimasi dengan *Gaussian Mixture Model* (GMM).

Ingat kembali bahwa data memiliki suatu pola (dalam statistik disebut distribusi), kemudian pada bab 2 telah disebutkan bahwa GMM dipercaya dapat mengaproksimasi fungsi apapun (silahkan perdalam pengetahuan statistik

[1] Cara lain memilih akan dijelaskan pada bab 10.



Gambar 4.1: Ilustrasi hubungan Clustering, kelas, dan Gaussian

kamu untuk hal ini). Dengan demikian, *machine learning* yang mempunyai salah satu tujuan untuk menemukan pola dataset, memiliki hubungan yang sangat erat dengan distribusi gaussian karena pola tersebut dapat diaproksimasi dengan distribusi gaussian.

4.3 K-nearest-neighbor

Ide **K-nearest-neighbor** (KNN) adalah mengelompokkan data ke kelompok yang memiliki sifat termirip dengannya [28]. Hal ini sangat mirip dengan **K-means**. Bila K-means digunakan untuk *clustering*, KNN digunakan untuk klasifikasi. Algoritma klasifikasi ini disebut juga algoritma malas karena tidak mempelajari cara mengkategorikan data, melainkan hanya mengingat data yang sudah ada.[2] Pada subbab 4.2, kita telah mengelompokkan data orang kaya menjadi dua kelompok.

KNN mencari K *feature vector* dengan sifat termirip, kemudian mengelompokkan data baru ke kelompok *feature vector* tersebut. Sebagai ilustrasi mudah, kita lakukan klasifikasi algoritma KNN dengan $K = 3$ untuk data baru $\{rich = no, intelligent = yes, good\ looking = yes\}$. Kita tahu pada subbab sebelumnya bahwa kelompok satu $k_1 = \{id_1, id_2, id_3, id_5\}$ dan $k_2 = \{id_4, id_6\}$, pada Tabel 4.8. *feature vector* termirip dimiliki oleh data dengan id_1, id_5, id_6 seperti diilustrasikan pada Tabel 4.8. Kita dapat menggunakan strategi untuk mengurusi permasalahan ini, misalnya memberikan prioritas memasukkan ke kelompok yang rata-rata perbedaannya lebih kecil.[3] Dengan strategi tersebut, kita mengklasifikasikan data baru ke kelompok pertama.

[2] https://sebastianraschka.com/faq/docs/lazy-knn.html
[3] Silahkan eksplorasi cara lain juga!

id	perbedaan
1	1
2	3
3	3
4	2
5	1
6	1

Tabel 4.8: Perbedaan data baru vs data orang kaya

Soal Latihan

4.1. Data numerik

(a) Carilah suatu contoh dataset numerik.
(b) Pikirkanlah strategi untuk mengklasifikasi data numerik pada algoritma Naive Bayes dan *K-nearest-neighbor*!

4.2. K-means, KNN, GMM, EM

Buktikan bahwa K-means, K-nearest-neighbor, *Gaussian Mixture Model*, dan Expectation Maximization Algorithm memiliki hubungan! (apa kesamaan mereka).

4.3. K-means

(a) Cari tahu cara lain untuk memilih **centroid** pada algoritma K-means (selain cara acak) baik untuk data nominal dan numerik!
(b) Cari tahu cara lain untuk menghitung kedekekatan suatu data dengan **centroid** baik untuk data nominal dan numerik! Hint: *euclidian distance*, *manhattan distance*, *cosine similarity*.