Artificial Intelligence
Kuliah 6: k-Means

Ali Akbar Septiandri
Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Al Azhar Indonesia
aliakbars@live.com

1 Clustering

1.1 Unsupervised Learning

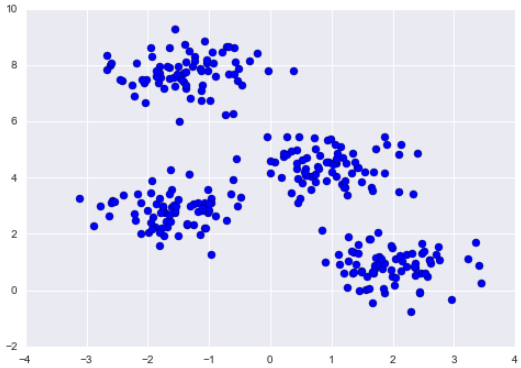
Dalam beberapa kuliah terakhir, Anda telah belajar tentang *supervised learning* yang membutuhkan label untuk dapat melatih model. Dalam notasi matematis, Anda ingin menghasilkan:

$$f : \mathcal{X} \to \mathcal{Y}$$

Namun, ada kalanya Anda butuh metode untuk dapat menemukan kelompok atau subpopulasi dalam data. Dengan kata lain, Anda perlu mencari **objek yang serupa** dalam data Anda. Apa yang menjadi kesamaan objek-objek tersebut? Untuk mengerjakan hal ini, kita dapat menggunakan ide yang telah dipelajari dalam materi k-Nearest Neighbours, yaitu konsep kedekatan antarobjek.

Sebagai contoh, Anda ingin mengelompokkan mahasiswa teknik informatika di suatu universitas. Maka, Anda dapat mengelompokkannya berdasarkan kesamaan kelas yang diambil, tapi bisa juga dilihat dari sisi usianya. Ketika Anda telah menemukan kelompok-kelompok tersebut, maka ada kemungkinan untuk menemukan **pencilan** dalam data yang Anda punya. Bisa jadi, pencilan ini adalah mahasiswa yang usianya masih sangat muda karena kelas akselarasi. Namun, bisa juga dia menjadi pencilan karena IPK-nya jauh di atas teman-teman yang lainnya. Jadi, perlu diingat bahwa dalam proses pengelompokan atau ***clustering*** tersebut, Anda boleh jadi menggunakan lebih dari satu dimensi.

Contoh yang lain: Jika dalam bidang Cartesian Anda mendapatkan data seperti pada Gambar 1, menurut Anda, berapa subpopulasi yang Anda bisa temukan?



Gambar 1: Contoh data dalam dua dimensi

Sebagian besar orang mungkin akan melihat ada empat subpopulasi dalam data tersebut seperti pada Gambar 2. Namun, bagaimana kalau misalnya kita melihat data yang di tengah menjadi satu kelompok sehingga ada tiga subpopulasi? Bukankah tidak ada cara pengelompokan yang pasti?

Gambar 2: Contoh data yang telah dikelompokkan menjadi empat

Hal inilah yang menyebabkan *clustering* menjadi salah satu cabang dari ***unsupervised learning***. Pada akhirnya, kelompok yang Anda hasilkan tersebut *tidak memerlukan label*. Anda hanya tahu bahwa kelompok tersebut punya kesamaan dari fitur atau atribut yang ada.

1.2 k-Means

Algoritma k-Means adalah salah satu cara untuk melakukan *clustering* dengan memanfaatkan konsep ***centroid***. Centroid adalah titik pusat yang merupakan rata-rata dari nilai objek dalam tiap dimensi. Dalam algoritma k-Means, ada sejumlah k centroid yang dapat kita tetapkan di awal. Nilainya untuk tiap dimensi diinisialisasi secara acak. Lalu, nilai centroid ini selalu dimutakhirkan sesuai dengan objek yang paling dekat dengan centroid tersebut. Kumpulan objek terdekat dengan centroid tersebutlah yang kemudian dianggap sebagai satu *cluster*. Untuk melakukan hal ini, k-Means memanfaatkan algoritma umum yang dikenal juga sebagai ***Expectation-Maximization (EM)***.

Algorithm 1 Expectation-Maximization

1: Inisialisasi k *centroid* secara acak
2: **while** belum konvergen **do**
3: E-step: Masukkan tiap titik/objek ke *centroid* terdekat

$$\arg\min_j D(x_i, c_j)$$

4: M-step: Ubah nilai *centroid* menjadi rata-rata dari tiap titik/objek
5: **for all** a **do**

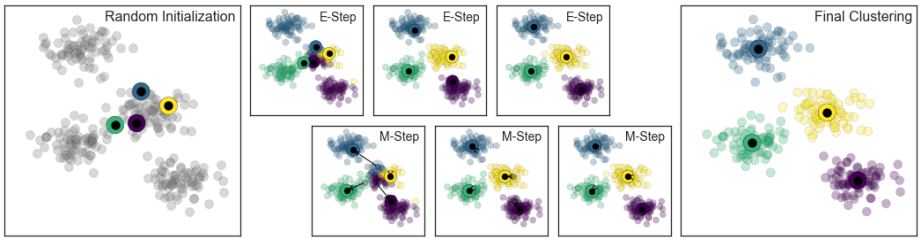
$$c_j(a) = \frac{1}{n_j} \sum_{x_i \to c_j} x_i(a)$$

Pada Algorithm 1, nilai a adalah dimensi yang digunakan, c_j adalah *cluster* ke-j, x_i adalah titik/objek dalam data, dan n_j adalah jumlah titik/objek yang masuk ke dalam *cluster* j. Seperti halnya pada algoritma k-NN, kita dapat menggunakan Euclidean distance sebagai metode untuk menghitung $D(x_i, c_j)$. Algoritma EM dapat divisualisasikan seperti pada Gambar 3.

1.3 Menentukan Nilai k

Sayangnya, algoritma seperti k-Means tidak dapat menemukan nilai k-nya sendiri. Ini salah satunya disebabkan oleh sifat algoritma ini sebagai *unsupervised learning*. Jadi, nilai k biasanya merupakan masukan dari pengguna.

Salah satu cara untuk menentukan k adalah dengan menggunakan jumlah kelas aslinya. Dalam kasus deteksi digit seperti pada dataset MNIST, kita dapat menggunakan $k = 10$. Namun, di kebanyakan



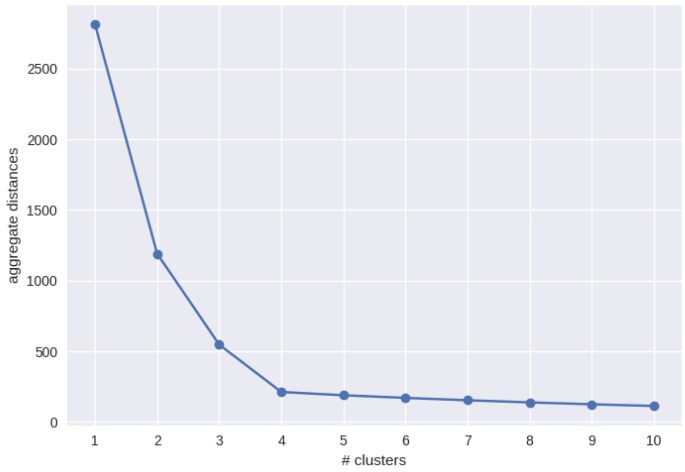
Gambar 3: Konvergensi *cluster* tercapai hanya dalam tiga iterasi [1]

kasus, tidak dimungkinkan untuk mendapatkan jumlah kelas seperti ini. Oleh karena itu, kita dapat menggunakan ***elbow method*** dari ***scree plot***.

Untuk menghasilkan *scree plot*, kita perlu menghitung nilai total jarak antara tiap titik dengan *centroid* terdekatnya. Secara matematis, nilai yang juga dikenal sebagai ***inertia*** ini didefinisikan sebagai

$$V = \sum_{j} \sum_{x_i \to c_j} D(c_j, x_i)^2$$

sehingga menghasilkan plot seperti pada Gambar 4. Nilai $k = 4$ diambil karena perubahan arah tertinggi terjadi di nilai ini. Secara matematis, nilai perubahan ini juga dapat dihitung dengan mencari turunan kedua dari fungsi untuk nilai V yang telah didefinisikan di atas.

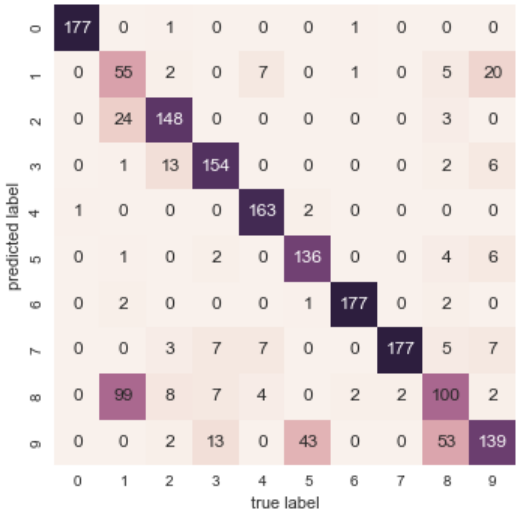


Gambar 4: Secara visual, scree plot menunjukkan nilai optimal $k = 4$

2 Mengevaluasi Clustering

Ada dua cara untuk mengevaluasi algoritma *clustering* secara umum: ekstrinsik dan intrinsik. Meski metode *clustering* tidak punya *metric* yang pasti, tetapi ada beberapa pendekatan yang dapat dilakukan untuk mengukur seberapa baik pembagian *cluster* yang telah kita lakukan. Evaluasi ekstrinsik dilakukan dengan menggunakan *clusters* yang telah dihasilkan untuk tugas yang lain, misalnya untuk menemukan pencilan atau menggunakannya sebagai atribut untuk melakukan klasifikasi.

Di sisi lain, evaluasi intrinsik salah satunya dapat dilakukan dengan menggunakan kelas aslinya sebagai referensi. Misalkan kita punya *clusters* $c_1, c_2, ..., c_k$ dan kelas referensi $r_1, r_2, ..., r_n$, maka yang kita lakukan adalah menghitung akurasi dengan memasangkan r_i dengan c_j. Sebagai contoh, untuk MNIST kita bisa menghasilkan *confusion matrix* seperti pada Gambar 5. Sebagai catatan, nilai



Gambar 5: Confusion matrix dari MNIST clustering [1]

i tidak harus sama dengan j. Kita dapat melakukan pendekatan *greedy* dalam kasus ini. Dengan kata lain, kita cukup memasangkan suatu *cluster* dengan kelas yang paling banyak kecocokan nilainya.

Untuk memperjelas, dalam kasus *clustering* untuk menentukan kelompok karakter dalam sandiwara Julius Caesar, kita dapat secara *greedy* menentukan grup G_i mana yang akan dipasangkan dengan *cluster* C_j seperti pada Gambar 6. Untuk menghitung akurasinya, kita hanya perlu menghitung nilai yang dicetak biru sebagai jumlah yang benar, lalu bagi nilainya dengan jumlah seluruh karakter yang ada dalam data. Nilai yang dicetak biru merupakan irisan terbesar yang dicari untuk setiap C_j.

	G1	G2	G3	G4	G5	G6
C1	1	**7**	0	1	4	0
C2	0	0	0	0	2	**7**
C3	0	0	**2**	0	0	0
C4	**3**	1	0	0	1	0

Gambar 6: Kelompok karakter dalam sandiwara Julius Caesar

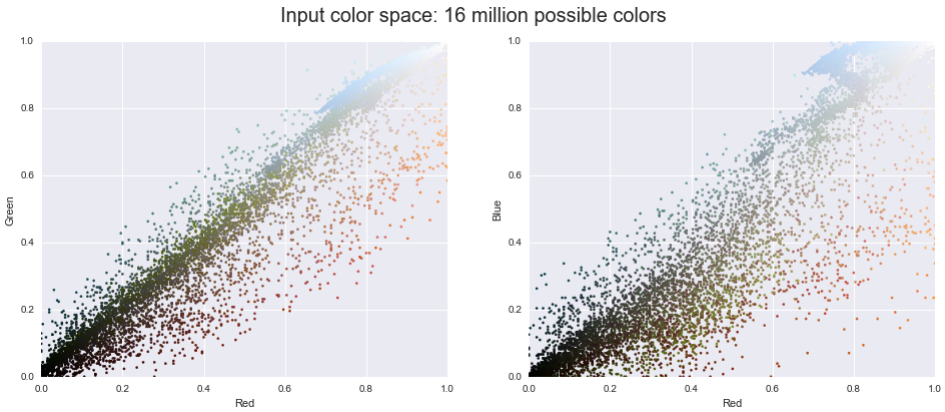
3 Aplikasi

Secara umum, algoritma untuk *clustering* dapat digunakan untuk melihat subpopulasi dalam data. Namun, lebih jauh lagi, subpopulasi ini dapat digunakan sebagai representasi fitur yang digunakan, misalnya untuk mewakili *pixels* yang ada dalam gambar [2]. Representasi inilah yang kemudian dapat digunakan untuk mengklasifikasikan gambar-gambar tersebut.

Selain itu, *clusters* yang dihasilkan juga dapat mewakili segmentasi dari pengguna suatu layanan dalam kasus pengguna layanan telekomunikasi, misalnya. Penerapan seperti ini merupakan salah satu penggunaan *clustering* yang paling sering dilakukan dalam dunia bisnis. Dengan algoritma yang cenderung sederhana, kita dapat menghasilkan kelompok untuk menerapkan metode pemasaran yang berbeda untuk meningkatkan *revenue*.

Algoritma k-Means juga dapat digunakan untuk melakukan kompresi gambar. Jika kita melihat warna *pixels* dalam kanal RGB, maka kita dapat menerapkan *clustering* untuk kemudian menggunakan nilai *centroid*-nya sebagai representasi *pixels* dengan warna yang berdekatan. Representasi yang dapat kita lihat pada Gambar 7, kemudian dapat digunakan untuk mengompresi gambar tersebut dengan faktor hingga 1 juta tanpa kehilangan bagian penting dari gambarnya [1]. Anda dapat melihat perubahan

sebelum dan setelah kompresi pada Gambar 8. Anda masih bisa melihat bentuk asli bangunannya dalam gambar yang terkompresi kan?



Gambar 7: Setiap *pixel* diwakili oleh satu titik dalam grafik ini [1]

Gambar 8: Kompresi dengan faktor hingga 1 juta menggunakan *clustering* [1]

References

[1] VanderPlas, J., 2016. *Python Data Science Handbook*. O'Reilly Media Inc.

[2] Coates, A. and Ng, A.Y., 2012. Learning feature representations with k-means. In *Neural networks: Tricks of the trade* (pp. 561-580). Springer Berlin Heidelberg.