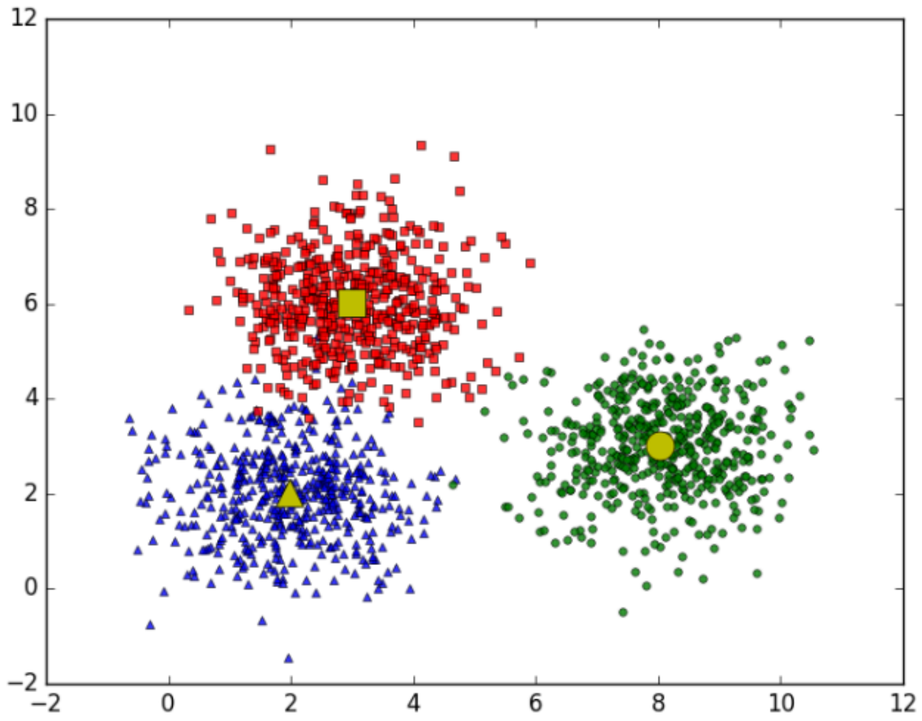
K-MEANS CLUSTERING

K-MEANS CLUSTERING là thuật toán Unsupervised learning giúp phân cụm phân hoạch dữ liệu thành K cụm khác nhau.



Bài toán với 3 clusters.

Bài toán thực tế:

Công ty có nhiều dữ liệu về khách hàng chưa được gán nhãn(số năm là khách hàng; số tiền khách hàng đã chi trả cho công ty; độ tuổi; giới tính; thành phố; nghề nghiệp; ...)
→ Muốn phân loại thành từng nhóm khách hàng khác nhau để tiện cho việc bán hàng cũng như chính sách ưu đãi

Thuật toán + ký hiệu

Tóm tắt thuật toán

Đầu vào: Dữ liệu X và số lượng cluster cần tìm K.
Đầu ra: Các center M và label vector cho từng điểm dữ liệu Y.

1. Chọn K điểm bất kỳ làm các center ban đầu.
2. Phân mỗi điểm dữ liệu vào cluster có center gần nó nhất.
3. Nếu việc gán dữ liệu vào từng cluster ở bước 2 không thay đổi so với vòng lặp trước nó thì ta dừng thuật toán.
4. Cập nhật center cho từng cluster bằng cách lấy trung bình cộng của tất các các điểm dữ liệu đã được gán vào cluster đó sau bước 2.
5. Quay lại bước 2.

Ký hiệu:

N điểm dữ liệu là X=[x1,x2,...,xN]
Cần tìm m1,m2,...,mK là các center để phân loại thành k cụm dữ liệu(k<N)
Với mỗi điểm xi đặt yi=[yi1,yi2,...,yiK] là label vector của nó
→ ví dụ một điểm dữ liệu có label vector là [1,0,0,...,0] thì nó thuộc vào cluster 1

$$y_{ik} \in \{0,1\}, \quad \sum_{k=1}^{K} y_{ik} = 1 \quad (1)$$

Hàm mất mát và bài toán tối ưu

Hàm mất mát

Giả sử với điểm xi được phân vào cụm có center là mk
→ Tính toán khoảng cách từ xi → mk:

$$\|\mathbf{x}_i - \mathbf{m}_k\|_2^2$$

Khi đó yik =1 → $y_{ik}\|\mathbf{x}_i - \mathbf{m}_k\|_2^2 = \sum_{j=1}^{K} y_{ij}\|\mathbf{x}_i - \mathbf{m}_j\|_2^2$

Sai số cho toàn bộ dữ liệu xi thuộc cụm mk :

$$\mathcal{L}(\mathbf{Y}, \mathbf{M}) = \sum_{i=1}^{N}\sum_{j=1}^{K} y_{ij}\|\mathbf{x}_i - \mathbf{m}_j\|_2^2$$

bài toán tối ưu:

$$\mathbf{Y}, \mathbf{M} = \arg\min_{\mathbf{Y},\mathbf{M}} \sum_{i=1}^{N}\sum_{j=1}^{K} y_{ij}\|\mathbf{x}_i - \mathbf{m}_j\|_2^2 \quad (2)$$

$$\text{subject to: } y_{ij} \in \{0,1\} \ \forall i,j; \quad \sum_{j=1}^{K} y_{ij} = 1 \ \forall i$$

Thuật toán tối ưu

Một cách đơn giản để giải bài toán là xen kẽ giải Y và M khi biến còn lại được cố định.

Cố định M, tìm Y

Khi các centers là cố định, bài toán tìm label vector cho toàn bộ dữ liệu có thể được chia nhỏ thành bài toán tìm label vector cho từng điểm dữ liệu xi như sau:

$$\mathbf{y}_i = \arg\min_{\mathbf{y}_i} \sum_{j=1}^{K} y_{ij} \|\mathbf{x}_i - \mathbf{m}_j\|_2^2 \quad (3)$$

$$\text{subject to: } y_{ij} \in \{0, 1\} \ \forall j; \quad \sum_{j=1}^{K} y_{ij} = 1$$

Vì chỉ có một phần tử của label vector yi bằng 1

$$j = \arg\min_{j} \|\mathbf{x}_i - \mathbf{m}_j\|_2^2$$

Cố định Y, tìm M

$$\mathbf{m}_j = \arg\min_{\mathbf{m}_j} \sum_{i=1}^{N} y_{ij} \|\mathbf{x}_i - \mathbf{m}_j\|_2^2.$$

Sử dụng đạo hàm để tìm mj: Đặt l(mj) = $\sum_{i=1}^{N} y_{ij} \|\mathbf{x}_i - \mathbf{m}_j\|_2^2.$

→ Đạo hàm ta được:

$$\frac{\partial l(\mathbf{m}_j)}{\partial \mathbf{m}_j} = 2 \sum_{i=1}^{N} y_{ij} (\mathbf{m}_j - \mathbf{x}_i)$$

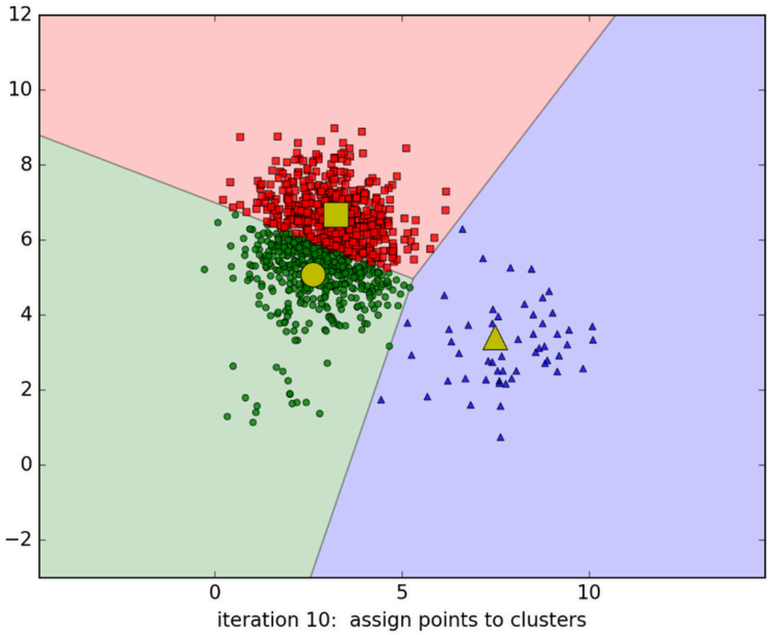
Giải đạo hàm = 0 ta được

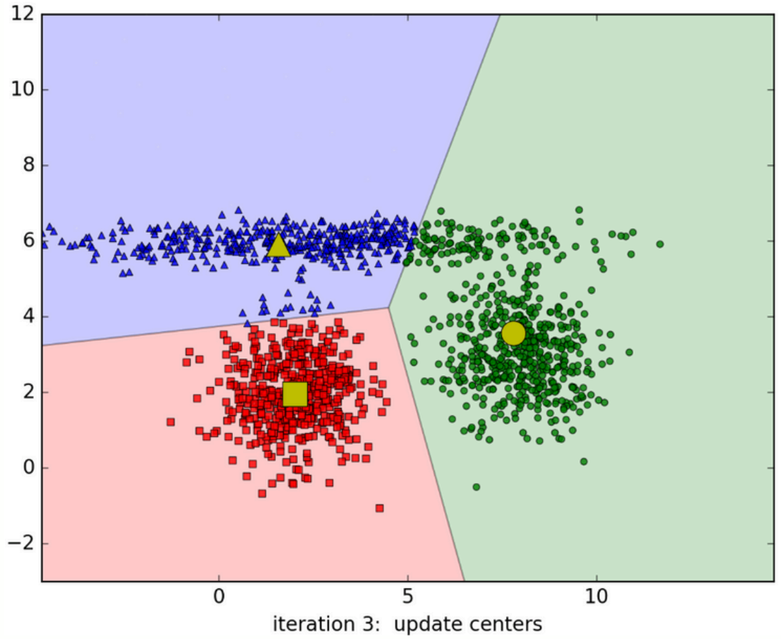
$$\mathbf{m}_j \sum_{i=1}^{N} y_{ij} = \sum_{i=1}^{N} y_{ij} \mathbf{x}_i$$

$$\Rightarrow \mathbf{m}_j = \frac{\sum_{i=1}^{N} y_{ij} \mathbf{x}_i}{\sum_{i=1}^{N} y_{ij}}$$

Mặt hạn chế của thuật toán

+ Cần biết trước số K
+ Số lượng điểm của mỗi cụm phải gần bằng nhau
+ Các cluster cần có hình dạng tròn





Tối ưu cho việc chọn k điểm

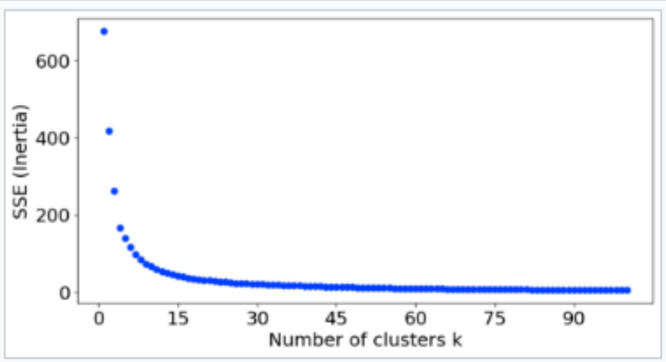
Việc chọn K ban đầu và vị trí của từng điểm là vô cùng quan trọng, quyết định tính chính xác cũng như việc thuật toán chạy nhanh hay chậm

Cách khắc phục:

- Chạy K-means clustering nhiều lần với các center ban đầu khác nhau rồi chọn cách có hàm mất mát cuối cùng đạt giá trị nhỏ nhất.
- Elbow Method: Tìm “khuỷu tay” trên đồ thị SSE(Sum of Squared Errors) vs K.
- Silhouette Score: Đo độ tách biệt và đồng nhất.

Elbow

Cho k chạy tăng dần mỗi lần tính toán giá trị SSE(là tổng bình phương khoảng cách từ mỗi điểm xi → tâm cụm mi tương ứng. Lưu lại kết quả và vẽ lại đồ thị ta được :



Example of the typical "elbow" pattern used for choosing the number of clusters even emerging on uniform data.

Ưu điểm:

- Dễ hiểu trực quan
- Tính toán nhanh dễ triển khai

Nhược điểm:

- Khó xác định K
- Khó xác định nếu dữ liệu không phân cụm rõ ràng

Tối ưu cho việc chọn k điểm

Silhouette Score

- a(i) = trung bình khoảng cách từ i đến các điểm khác trong chính cluster đó (cohesion).
- b(i) = giá trị nhỏ nhất của trung bình khoảng cách từ i đến các điểm của các cluster khác (separation), cụ thể là cluster gần nhất.

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

Chọn K sao cho s(i) lớn nhất (-1 < s(i) < 1):

- Gần +1 → Điểm được phân về đúng cụm, cách biệt rõ với các cụm khác.
- Gần 0 → Điểm nằm ở ranh giới giữa các cụm.
- Dưới 0 → Điểm có thể bị phân sai cụm.

Ưu điểm:

- Giá trị định lượng rõ ràng
- Tổng quát với toàn bộ dữ liệu

Nhược điểm:

- Tốt nhất với các cụm hình cầu, dễ sai khi có hình dạng phức tạp
- Tốn tài nguyên tính toán, dữ liệu lớn, chậm

Simplified Silhouette:

Tính trung bình khoảng cách từ điểm i đến các tâm cụm

→ Tăng đáng kể hiệu suất tính toán, được đánh giá có hiệu năng tốt hơn Silhouette Score

Các biến thể khác

1.K-means++
2.Mini-batch K-means
3.K-medians
4.K-medoids (PAM - Partitioning Around Medoids)
5.Fuzzy C-means (FCM)

Các biến thể khác

1.K-means++:

1. Khái niệm
   - Là cải tiến của K-means giúp khởi tạo tâm cụm tốt hơn.
   - Giảm nguy cơ hội tụ vào cực tiểu cục bộ, tăng tốc độ hội tụ.
2. Ý tưởng
   - Các tâm ban đầu nên cách xa nhau để tránh phân cụm kém chất lượng.
   - Chọn tâm mới với xác suất tỉ lệ với bình phương khoảng cách tới tâm gần nhất đã chọn.

Tóm tắt thuật toán

1.Chọn ngẫu nhiên 1 điểm dữ liệu làm tâm đầu tiên.

2.Với mỗi điểm dữ liệu xi, tính D(xi) = khoảng cách từ xi tới tâm gần nhất đã chọn.

3.Chọn điểm mới làm tâm với xác suất:

$$P(x) = \frac{D(x)^2}{\sum_{x' \in X} D(x')^2}$$

4.Lặp bước 2–3 cho đến khi chọn đủ kkk tâm.

5.Chạy K-means bình thường với các tâm đã khởi tạo

Ưu điểm:

- Khởi tạo tốt
- Hội tụ nhanh hơn
- Chất lượng cụm cao hơn

Nhược điểm:

- Tốn thêm thời gian khởi tạo do phải tính khoảng cách nhiều lần.

Các biến thể khác

1.Mini-batch K-means

1. Khái niệm
   - Mini-batch K-means là một biến thể của K-means được thiết kế để chạy nhanh hơn trên tập dữ liệu lớn.
2. Ý tưởng
   - Thay vì cập nhật tâm cụm bằng toàn bộ dữ liệu mỗi vòng lặp, nó chỉ lấy một batch nhỏ dữ liệu (ngẫu nhiên) để cập nhật.

Tóm tắt thuật toán

1.Khởi tạo tâm cụm (có thể dùng K-means++ để tốt hơn).

2.Lặp cho đến khi hội tụ hoặc đạt số vòng lặp tối đa:

- Lấy ngẫu nhiên một mini-batch (ví dụ 100 điểm) từ tập dữ liệu.
- Gán từng điểm trong batch này vào cụm gần nhất.
- Cập nhật tâm cụm dựa trên điểm mới trong batch:

3.Trả về tâm cụm cuối cùng.

Ưu điểm:

- Nhanh: mỗi vòng chỉ xử lý một phần dữ liệu → tiết kiệm bộ nhớ và thời gian.
- Phù hợp dữ liệu streaming: có thể chạy khi dữ liệu đến liên tục (online learning).
- Vẫn cho kết quả gần như K-means chuẩn, nếu batch đủ đa dạng.
- Phù hợp với dữ liệu lớn, bộ nhớ xử lý thấp

Nhược điểm:

- Kết quả ít chính xác hơn K-means chuẩn (do dùng mẫu nhỏ).
- Cần chọn kích thước batch hợp lý (quá nhỏ → nhiễu; quá lớn → chậm).
- Phụ thuộc vào tính ngẫu nhiên khi chọn batch.

Các biến thể khác

1.K-medians

1. Khái niệm
   - K-medians là một biến thể của K-means, nhưng thay vì dùng trung bình (mean) để tính tâm cụm, nó dùng trung vị (median).
2. Ý tưởng
   - Dùng trung vị để tính tâm cụm giúp giảm ảnh hưởng của outlier

Tóm tắt thuật toán

1.Khởi tạo tâm cụm ngẫu nhiên
2.Gán điểm vào cụm gần nhất
3.Cập nhật tâm cụm

$$m_j = \text{median}(x_i)$$

4.Lặp lại cho đến khi hội tụ.

Ưu điểm:

- Ít nhạy với outlier hơn: vì trung vị không bị kéo lệch mạnh như trung bình.
- Tốt hơn cho dữ liệu có phân phối lệch (skewed distribution).

Nhược điểm:

- Tính trung vị trong nhiều chiều phức tạp hơn trung bình.
- Có thể hội tụ chậm hơn.
- Khoảng cách Manhattan có thể không phản ánh tốt dữ liệu dạng “tròn” như trong K-means.

Các biến thể khác

1.K-medoids (PAM - Partitioning Around Medoids)

1. Khái niệm

- K-medoids là một biến thể của K-means, nhưng sử dụng chính các điểm trong cụm làm tâm cụm

Tóm tắt thuật toán

1.Chọn k điểm ngẫu nhiên làm tâm cụm

2.Gán mỗi điểm vào tâm cụm gần nhất

3.Với mỗi cụm thay tâm cụm bằng 1 điểm khác trong cụm

Tính tổng bình phương khoảng cách từ tâm cụm mới đến các điểm giảm đi → cập nhật lại tâm cụm

4.Lặp lại cho đến khi hội tụ

Ưu điểm:

- Ít nhạy với outlier hơn
- Thích hợp khi muốn tìm tâm cụm là điểm dữ liệu thực

Nhược điểm:

- Độ phức tạp tính toán cao hơn
- Tiêu tốn nhiều tài nguyên tính toán hơn

Các biến thể khác

Fuzzy C-means (FCM)

1. Khái niệm
   - Là thuật toán cho phép điểm xi có thể được phân vào cụm ci dựa vào mức độ thuộc về (membership)
2. Hàm mục tiêu

$$J = \sum_{i=1}^{n} \sum_{j=1}^{k} u_{ij}^{m} \, \|x_i - c_j\|^2$$

Tóm tắt thuật toán

1.Khởi tạo k cụm và ma trận U = [Uij] thỏa mãn

$$\sum_j u_{ij} = 1$$

2.Cập nhật tâm cụm theo trọng số:

$$c_j = \frac{\sum_{i=1}^{n} u_{ij}^{m} \, x_i}{\sum_{i=1}^{n} u_{ij}^{m}}$$

3.Cập nhật trọng số Uij

$$u_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{\|x_i - c_j\|}{\|x_i - c_k\|} \right)^{\frac{2}{m-1}}}$$

4.Lặp lại bước 2 và 3 cho đến khi hội tụ

Ưu điểm:

- Xử lý ranh giới mờ tốt.
- Giảm “quyết định cứng” sai khi điểm nằm giữa hai cụm.

Nhược điểm:

- Nhạy outlier (vẫn tối thiểu bình phương khoảng cách)
- Nhạy khởi tạo, có thể rơi cực tiểu cục bộ.