Massive Data Analysis

Instructor: Dr.Gholampour

Fall 2024

HW 3

Arman Yazdani - 400102255

Contents

1 K-means

1.1 Minimizing the cost function

To show that minimizing the cost function requires selecting the mean of each cluster as its center, we can use differentiation. Suppose μ_j is the center of cluster C_j. The cost function is defined as follows:

$$L = \sum_{j=1}^{k} \sum_{x \in C_j} \|x - \mu_j\|^2$$

To minimize this cost function, we need to set its derivative with respect to μ_j to zero:

$$\frac{\partial L}{\partial \mu_j} = 0$$

The derivative of the cost function with respect to μ_j is:

$$\frac{\partial L}{\partial \mu_j} = \sum_{x \in C_j} 2(\mu_j - x)$$

Setting the derivative to zero, we get:

$$\sum_{x \in C_j} 2(\mu_j - x) = 0$$

Simplifying this equation, we have:

$$\sum_{x \in C_j} (\mu_j - x) = 0$$

$$\mu_j \sum_{x \in C_j} 1 - \sum_{x \in C_j} x = 0$$

$$\mu_j |C_j| - \sum_{x \in C_j} x = 0$$

Therefore:

$$\mu_j = \frac{1}{|C_j|} \sum_{x \in C_j} x$$

This equation shows that to minimize the cost function, the mean of each cluster should be selected as its center.

1.2 Convergence

To prove that the k-means algorithm always converges, we can use the fact that the cost function L decreases or remains constant in each iteration. This cost function is the sum of the squared Euclidean distances between the samples and the cluster centers, and its value cannot be negative. Therefore, the k-means algorithm converges to a point after a finite number of iterations.

To prove the convergence of the k-means algorithm, we can consider the following steps:

1. **Assigning samples to clusters**: In each iteration, each sample is assigned to the nearest cluster center. This step reduces the cost function because the samples are moved closer to the centers.

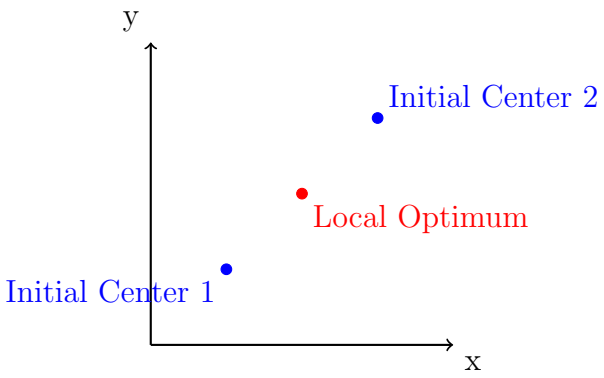
2. **Updating cluster centers**: In each iteration, the cluster centers are updated to the mean of the samples assigned to them. This step also reduces the cost function because the cluster centers are moved to more optimal positions.

Since the cost function decreases in each iteration and its value cannot be negative, the k-means algorithm converges to a point after a finite number of iterations.

However, the k-means algorithm may converge to a local optimum. This means that the algorithm may converge to a point that is a local minimum of the cost function but not the global minimum. To illustrate this, we can use a simple example:

Suppose there are two clusters in the data, and the initial cluster centers are chosen randomly. If the initial centers are chosen such that both centers are in one cluster, the k-means algorithm may converge to a local optimum and fail to identify the true clusters.

In the figure below, an example of convergence to a local optimum is shown:



In this example, the initial cluster centers are chosen such that the k-means algorithm converges to a local optimum and fails to identify the true clusters.

1.3 Knee point & other methods

1.3.1 Knee point

This method is also known as the "Elbow Method". In this method, the cost function (sum of squared distances) is calculated and plotted for different values of K. The point where the reduction in the cost function significantly slows down (the knee point) is chosen as the optimal value of K. This point represents a balance between the number of clusters and the reduction in cost.

1.3.2 Silhouette + Bayesian Information Criterion

In addition to the Elbow Method, two other methods for estimating the optimal value of K are:

1. **Silhouette Method**: In this method, the silhouette score is calculated for each sample. The silhouette score measures how similar a sample is to its own cluster compared to other clusters. The optimal value of K is chosen such that the average silhouette score for all samples is maximized. This method helps evaluate the quality of clustering.
2. **Bayesian Information Criterion (BIC)**: In this method, the Bayesian Information Criterion is calculated for different values of K. BIC is a statistical criterion that balances model complexity and data fitting. The optimal value of K is chosen such that the BIC value is minimized. This method helps select a model with appropriate complexity.

1.4 Penalty term for high # of clusters

1.4.1 Paradigm

One common approach is to add a penalty term to the cost function that depends on the number of clusters. For example, the new cost function can be defined as follows:

$$L' = \sum_{j=1}^{k} \sum_{x \in C_j} ||x - \mu_j||^2 + \lambda k$$

- λ: a penalty parameter that determines the penalty for having more clusters.
- C_j set of samples that are closer to the center μ_j than to any other cluster.

1.4.2 Advantages

1. **Reduced Number of Clusters**: By adding a penalty to the cost function, the algorithm will be more inclined to reduce the number of clusters.
2. **Simplified Model**: With fewer clusters, the model becomes simpler and easier to interpret.

1.4.3 Disadvantages

1. **Potential Loss of Detail**: Reducing the number of clusters may lead to a loss of detail and accuracy in clustering, as samples may be grouped into larger, less precise clusters.
2. **Penalty Parameter Selection**: Choosing an appropriate value for the penalty parameter (λ) can be challenging and requires careful tuning.
3. **Over-Simplification**: If the penalty parameter is too large, the algorithm may produce too few clusters, leading to oversimplified clustering results.

2 fixed-CURE

If all representative points were moved by a fixed distance towards the cluster centers, there could be both advantages and disadvantages:

2.1 Advantages

1. **Simplified Calculations**: Moving all points by a fixed distance towards the cluster center could simplify the calculations and reduce the execution time.
2. **Faster Convergence**: This method might lead to faster convergence of the algorithm as all points move uniformly towards the center.

2.2 Disadvantages

1. **Loss of Cluster Shape**: Moving all points by a fixed distance might result in the loss of the shape and structure of the clusters, leading to less accurate clustering.
2. **Increased Influence of Outliers**: This method might increase the influence of outliers, as outliers would also move by the same fixed distance towards the center, potentially affecting the final clustering result.
3. **Reduced Clustering Accuracy**: By moving all points by a fixed distance, the representative points may not accurately reflect the positions of the clusters, leading to reduced clustering accuracy.

Therefore, the current CURE method, which moves the representative points by a certain fraction of the distance between their initial positions and the cluster center, helps preserve the shape and structure of the clusters and reduces the influence of outliers. This method contributes to the accuracy and quality of the clustering process and avoids the issues that might arise from moving all points by a fixed distance.

3 SVD

3.1 A^T&A

matrix A is decomposed as follows:

$$A = U\Sigma V^T$$

By transposing:

$$A^T = V\Sigma^T U^T$$

- U is an orthogonal matrix whose columns are the eigenvectors of the matrix AA^T.
- Σ is a diagonal matrix containing the singular values of the matrix A.
- V is an orthogonal matrix whose columns are the eigenvectors of the matrix A^TA

3.2 Orthogonal matrix: Q

The SVD of matrix A is given by:

$$A = U\Sigma V^T$$

Now, let's compute A^TA Using the properties of transposes and orthogonal matrices:

$$A^TA = (U\Sigma V^T)^T(U\Sigma V^T)$$

Since U and V are orthogonal matrices, $U^TU = I$ and $V^TV = I$.Therefore:

$$A^TA = V\Sigma^T\Sigma V^T$$

Let $Q = V$, thus:

$$Q^TA^TAQ = V^T(V\Sigma^T\Sigma V^T)V$$

V is orthogonal:

$$Q^TA^TAQ = \Sigma^T\Sigma$$

Since Q is orthogonal, any quadratic form of the type $B = Q^TCQ$ preserves the eigenvalues and the structure of C. In this case:

$$Q^TA^TAQ \quad \text{is a similarity transformation of} \quad A^TA.$$

Similarity transformations preserve eigenvalues, and since A^TA is symmetric, the eigenvalues are unchanged. Furthermore, the symmetric structure of A^TA is also preserved.

The equality

$$A^TA = Q^TA^TAQ$$

holds **only when** $Q = V$. In the general case, A^TA is invariant under orthogonal similarity transformations, which means the eigenvalues of A^TA remain unchanged under such transformations. However, the matrices themselves are not necessarily equal unless $Q = I$ or $Q = V$.

4 Random Sampling

4.1 CX matrix decomposition

The Frobenius norm error in the CX decomposition can be bounded as:

$$\|A - CC^{\dagger}A\|_F^2 \leq \|A - A_k\|_F^2 + \epsilon\|A\|_F^2$$

where:

- A_k is the best rank-k approximation of A (via truncated SVD),
- ϵ accounts for the approximation quality of the column selection,
- $\|A\|_F = 50$ is the Frobenius norm of A.

If A has an effective rank k, and $c \geq k$, then $A \approx A_k$, and $\|A - A_k\|_F^2$ is negligible. In this case, the error simplifies to:

$$\|A - CC^\dagger A\|_F^2 \approx \epsilon \|A\|_F^2$$

Substituting the given values:

$$\epsilon = 0.1,$$
$$\|A\|_F^2 = 50^2 = 2500$$

we obtain:

$$\|A - CC^\dagger A\|_F^2 \leq 0.1 \times 2500 = 250$$

Thus, the error bound is:

$$\|A - CC^\dagger A\|_F \leq \sqrt{250} \approx 15.81$$

For a matrix A with effective rank $k \ll n$, where the sampled columns capture the dominant singular vectors, the error $\|A - A_k\|_F$ is negligible. Therefore, the error $\|A - CC^\dagger A\|_F$ is primarily influenced by the term involving ϵ, resulting in a much tighter bound.

4.2 Approximating matrix multiplication

Generally, for multiplication approximating we have:

$$A_{m\times n} B_{n\times p} \backsimeq C_{m\times s} R_{s\times p}$$

For $t = 1$ up to s,pick a column $A^{(j_t)}$ and a row $B_{(j_t)}$ with probability:

$$\Pr(j_t) = \frac{||A^{(i)}||_2 ||B_{(i)}||_2}{\sum_{i=1}^{n} ||A^{(i)}||_2 ||B_{(i)}||_2}$$

Include $\frac{A_{(jt)}}{\sqrt{sp_{jt}}}$ as a column of C and $\frac{B_{(jt)}}{\sqrt{sp_{jt}}}$ as a row of R
For the above algorithm:

$$E\left(\|AB - CR\|_{2,F}\right) \leq \frac{1}{\sqrt{s}} \|A\|_F \|B\|_F$$

With probability at least $1 - \delta$:

$$\|AB - CR\|_{2,F} \leq O\left(\frac{\log(1/\delta)}{\sqrt{s}}\right) \|A\|_F \|B\|_F$$

Now let's replace the given parameters($\delta \leq 0.01, s = 50$):

$$E\left(\|AB - CR\|_{2,F}\right) \leq \frac{1}{\sqrt{50}} 50 \times 80 \backsimeq 565.7$$

$$\|AB - CR\|_{2,F} \leq O\left(\frac{\log(1/0.01)}{\sqrt{50}}\right) 50 \times 80 \leq 1131$$

5 Clustering

5.1 Hierarchical clustering

The transformed binary matrix is:

	a	b	c	d	e	f	g	h
A	1	1	0	1	0	0	1	0
B	1	0	1	1	0	1	0	0
C	0	0	1	0	0	1	1	1

The Jaccard distance between columns is computed as:

$$d_{\text{Jaccard}}(u, v) = 1 - \frac{|u \cap v|}{|u \cup v|}$$

Initial Clusters:

$$\{\text{a}\}, \{\text{b}\}, \{\text{c}\}, \{\text{d}\}, \{\text{e}\}, \{\text{f}\}, \{\text{g}\}, \{\text{h}\}$$

1.1 The pairwise Jaccard distance matrix is:

	a	b	c	d	e	f	g	h
a	0.0	0.5	0.6667	0.3333	1.0	1.0	0.5	1.0
b	0.5	0.0	0.5	0.3333	1.0	0.6667	1.0	1.0
c	0.6667	0.5	0.0	0.6667	1.0	0.3333	0.6667	0.6667
d	0.3333	0.3333	0.6667	0.0	1.0	0.6667	1.0	1.0
e	1.0	1.0	1.0	1.0	0.0	1.0	1.0	1.0
f	1.0	0.6667	0.3333	0.6667	1.0	0.0	0.5	0.5
g	0.5	1.0	0.6667	1.0	1.0	0.5	0.0	0.3333
h	1.0	1.0	0.6667	1.0	1.0	0.5	0.3333	0.0

1.2 Merge: {a} and {d} (minimum distance: 0.3333).

$$\{\text{ad}\}, \{\text{b}\}, \{\text{c}\}, \{\text{e}\}, \{\text{f}\}, \{\text{g}\}, \{\text{h}\}$$

2.1 Recalculated distances:

	ad	b	c	e	f	g	h
ad	0.0	0.3333	0.6667	1.0	0.6667	1.0	1.0
b	0.3333	0.0	0.5	1.0	0.6667	1.0	1.0
c	0.6667	0.5	0.0	1.0	0.3333	0.6667	0.6667
e	1.0	1.0	1.0	0.0	1.0	1.0	1.0
f	0.6667	0.6667	0.3333	1.0	0.0	0.5	0.5
g	1.0	1.0	0.6667	1.0	0.5	0.0	0.3333
h	1.0	1.0	0.6667	1.0	0.5	0.3333	0.0

2.2 Merge {ad} and {b} (minimum distance: 0.3333).

$$\{\text{adb}\}, \{\text{c}\}, \{\text{e}\}, \{\text{f}\}, \{\text{g}\}, \{\text{h}\}$$

3.1 Recalculated distances:

	adb	c	e	f	g	h
adb	0.0	0.5	1.0	0.6667	1.0	1.0
c	0.5	0.0	1.0	0.3333	0.6667	0.6667
e	1.0	1.0	0.0	1.0	1.0	1.0
f	0.6667	0.3333	1.0	0.0	0.5	0.5
g	1.0	0.6667	1.0	0.5	0.0	0.3333
h	1.0	0.6667	1.0	0.5	0.3333	0.0

3.2 Merge {c} and {f} (minimum distance: 0.3333).

$$\{\text{adb}\}, \{\text{cf}\}, \{\text{e}\}, \{\text{g}\}, \{\text{h}\}$$

4.1 Recalculated distances:

	adb	cf	e	g	h
adb	0.0	0.5	1.0	1.0	1.0
cf	0.5	0.0	1.0	0.5	0.5
e	1.0	1.0	0.0	1.0	1.0
g	1.0	0.5	1.0	0.0	0.3333
h	1.0	0.5	1.0	0.3333	0.0

4.2 Merge {g} and {h} (minimum distance: 0.3333) gives the **final 4 clusters**.

$$\{\text{adb}\}, \{\text{cf}\}, \{\text{gh}\}, \{\text{e}\}$$

5.2 Clusters&Users

The value of each entry in the matrix is the average of the non-empty numerical values for the corresponding user and all the items in the cluster.

• Cluster {adb}:

$$A : \frac{4+5+5}{3} = 4.67$$
$$B : \frac{3+3}{2} = 3.00$$
$$C : \frac{2+1}{2} = 1.50$$

• Cluster {cf}:

$$A : \text{NaN (no values available)}$$
$$B : \frac{4+2}{2} = 3.00$$
$$C : \frac{3+4}{2} = 3.50$$

• Cluster {gh}:

$$A : \frac{3+2}{2} = 2.50$$
$$B : \frac{1}{1} = 1.00$$
$$C : \frac{5+3}{2} = 4.00$$

• Cluster {gh}:

$$A : 1.00$$
$$B : 1.00$$
$$C : \text{NaN (no values available)}$$

The new utility matrix, where rows represent users and columns represent clusters, is as follows:

	{adb}	{cf}	{gh}	{e}
A	4.67	NaN	2.50	1.00
B	3.00	3.00	1.00	1.00
C	1.50	3.50	4.00	NaN

5.3 Cosine distance(Clusters)

The cosine distance between two users is given by:

$$\text{Cosine Distance}(u, v) = 1 - \cos(u, v)$$

where

$$\cos(u, v) = \frac{\sum_{i=1}^{n} u_i v_i}{\sqrt{\sum_{i=1}^{n} u_i^2} \cdot \sqrt{\sum_{i=1}^{n} v_i^2}}$$

Here, u and v are the row vectors for two users in the matrix, and n is the number of clusters (columns in the matrix).

Let's Calculate distances:

• Pair A and B: Using clusters {adb}, {gh}, {e} (valid entries for both):

$$u = [4.67, 2.50, 1.00], \quad v = [3.00, 1.00, 1.00]$$

$$\cos(A,B) = \frac{(4.67 \cdot 3.00) + (2.50 \cdot 1.00) + (1.00 \cdot 1.00)}{\sqrt{(4.67^2 + 2.50^2 + 1.00^2)} \cdot \sqrt{(3.00^2 + 1.00^2 + 1.00^2)}}$$

$$\cos(A,B) = \frac{14.01 + 2.50 + 1.00}{\sqrt{21.80 + 6.25 + 1.00} \cdot \sqrt{9.00 + 1.00 + 1.00}}$$

$$\cos(A,B) = \frac{17.51}{\sqrt{29.05} \cdot \sqrt{11.00}} = \frac{17.51}{5.39 \cdot 3.32} = \frac{17.51}{17.90} \approx 0.978$$

$$\text{Cosine Distance}(A,B) = 1 - 0.978 = 0.022$$

• Pair A and C: Using clusters {adb}, {cf}, {gh} (valid entries for both):

$$u = [4.67, 2.50], \quad v = [1.50, 3.50]$$

$$\cos(A,C) = \frac{(4.67 \cdot 1.50) + (2.50 \cdot 3.50)}{\sqrt{(4.67^2 + 2.50^2)} \cdot \sqrt{(1.50^2 + 3.50^2)}}$$

$$\cos(A,C) = \frac{7.01 + 8.75}{\sqrt{21.80 + 6.25} \cdot \sqrt{2.25 + 12.25}}$$

$$\cos(A,C) = \frac{15.76}{\sqrt{29.05} \cdot \sqrt{14.50}} = \frac{15.76}{5.39 \cdot 3.81} = \frac{15.76}{20.54} \approx 0.767$$

$$\text{Cosine Distance}(A,C) = 1 - 0.767 = 0.233$$

• Pair B and C: Using clusters {cf}, {gh} (valid entries for both):

$$u = [3.00, 1.00], \quad v = [3.50, 4.00]$$

$$\cos(B,C) = \frac{(3.00 \cdot 3.50) + (1.00 \cdot 4.00)}{\sqrt{(3.00^2 + 1.00^2)} \cdot \sqrt{(3.50^2 + 4.00^2)}}$$

$$\cos(B,C) = \frac{10.50 + 4.00}{\sqrt{9.00 + 1.00} \cdot \sqrt{12.25 + 16.00}}$$

$$\cos(B,C) = \frac{14.50}{\sqrt{10.00} \cdot \sqrt{28.25}} = \frac{14.50}{3.16 \cdot 5.31} = \frac{14.50}{16.79} \approx 0.864$$

$$\text{Cosine Distance}(B,C) = 1 - 0.864 = 0.136$$

► Summary of Results:

$$\text{Cosine Distance}(A,B) = 0.022$$

$$\text{Cosine Distance}(A,C) = 0.233$$

$$\text{Cosine Distance}(B,C) = 0.136$$