Lecture 7: K-Means

Feng Li

Shandong University

fli@sdu.edu.cn

November 26, 2020
Outline

1. Clustering
2. K-Means Method
4. Kernel $K$-Means
5. Hierarchical Clustering
Clustering

- Usually an unsupervised learning problem
- Given: $N$ unlabeled examples \( \{x_1, \cdots, x_N\} \); no. of desired partitions $K$
- Goal: Group the examples into $K$ “homogeneous” partitions

Loosely speaking, it is classification without ground truth labels

A good clustering is one that achieves:
- High within-cluster similarity
- Low inter-cluster similarity
Similarity can be Subjective

- Clustering only looks at similarities, no labels are given
- Without labels, similarity can be hard to define
- Goal: Group the examples into $K$ “homogeneous” partitions

Thus using the right distance/similarity is very important in clustering
Also important to define/ask: “Clustering based on what”?

Feng Li (SDU)
Clustering: Some Examples

- Document/Image/Webpage Clustering
- Image Segmentation (clustering pixels)
- Clustering web-search results
- Clustering (people) nodes in (social) networks/graphs
- ... and many more...
Types of Clustering

- **Flat or Partitional clustering**: Partitions are independent of each other.

- **Hierarchical clustering**: Partitions can be visualized using a tree structure (a dendrogram).
**K-Means Clustering Problem**

- Given a set of observations $\mathbf{X} = \{x_1, x_2, \cdots, x_N\}$ ($x_i \in \mathbb{R}^D$), partition the $N$ observations into $K$ sets ($K \leq N$) $\{C_k\}_{k=1,\cdots,K}$ such that the sets minimize the within-cluster sum of squares:

$$
\arg \min_{\{C_k\}} \sum_{i=1}^{K} \sum_{x \in C_i} \|x - \mu_i\|^2
$$

where $\mu_i$ is the mean of points in set $C_i$

- How hard is this problem?
  - The problem is NP hard, but there are good heuristic algorithms that seem to work well in practice, e.g., K-means algorithm and mixtures of Gaussians.
K-Means Algorithm (Lloyd, 1957)

- In each iteration
  - (Re)-Assign each example $x_i$ to its closest cluster center (based on the smallest Euclidean distance)

$$\mathcal{C}_k = \{x_i \mid \|x_i - \mu_k\|^2 \leq \|x_i - \mu_{k'}\|^2, \text{ for } \forall k' \neq k\}$$

($\mathcal{C}_k$ is the set of examples assigned to cluster $k$ with center $\mu_k$)

- Update the cluster means

$$\mu_k = \text{mean}(\mathcal{C}_k) = \frac{1}{|\mathcal{C}_k|} \sum_{x \in \mathcal{C}_k} x$$

- Stop when cluster means or the “loss” does not change by much
K-means: Initialization (assume $K = 2$)
K-means iteration 1: Assigning points
K-means iteration 1: Recomputing the centers
K-means iteration 2: Assigning points
K-means iteration 2: Recomputing the centers
K-means iteration 3: Assigning points
K-means iteration 3: Recomputing the centers
K-means iteration 4: Assigning points
K-means iteration 4: Recomputing the centers
What Loss Function is K-means Optimizing?

- Let $\mu_1, \mu_2, \cdots, \mu_K$ be the $K$ cluster centroids (means).
- Let $z_{i,k}$ be an indicator

\[
  z_{i,k} = \begin{cases} 
    1, & x_i \in C_k \\
    0, & \text{otherwise}
  \end{cases}
\]

- Assume

\[
  z_i = [z_{i,1}, z_{i,2}, \cdots, z_{i,k}]^T
\]

with $\sum_{k=1}^{K} z_{i,k} = 1$, represents a length $K$ one-hot encoding of $x_i$.
Define the distortion or “loss” for the cluster assignment of $x_i$

$$\ell(\{\mu_k\}_{k=1,\ldots,K}, x_i, z_i) = \sum_{k=1}^{K} z_{i,k} \|x_i - \mu_k\|^2$$

Total distortion over all points defines the K-means “loss function”

$$L(\mu, X, Z) = \sum_{i=1}^{N} \sum_{k=1}^{K} z_{i,k} \|x_i - \mu_k\|^2 = \|X - Z\mu\|^2$$

where $X$ is $N \times D$, $Z$ is $N \times K$ and $\mu$ is $K \times D$
What Loss Function is K-means Optimizing? (Contd.)

- Total distortion over all points defines the K-means "loss function"

\[ L(\mu, X, Z) = \sum_{i=1}^{N} \sum_{k=1}^{K} z_{i,k} \| x_i - \mu_k \|^2 = \| X - Z\mu \|^2 \]

where \( X \) is \( N \times D \), \( Z \) is \( N \times K \) and \( \mu \) is \( K \times D \)

- Let's have a closer look

\[
X - Z\mu = \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_N^T \end{bmatrix} - \begin{bmatrix} z_1^T \\ z_2^T \\ \vdots \\ z_N^T \end{bmatrix} \mu = \begin{bmatrix} x_1^T - z_1^T \mu \\ x_2^T - z_2^T \mu \\ \vdots \\ x_N^T - z_N^T \mu \end{bmatrix}
\]

where \( z_{i,\mu} \) is the centroid of the cluster \( x_i \) belongs to
Define the distortion or “loss” for the cluster assignment of $x_i$

$$\ell(\{\mu_k\}, x_i, z_i) = \sum_{k=1}^{K} z_{i,k} \|x_i - \mu_k\|^2$$

Total distortion over all points defines the K-means “loss function”

$$L(\mu, X, Z) = \sum_{i=1}^{N} \sum_{k=1}^{K} z_{i,k} \|x_i - \mu_k\|^2 = \|X - Z\mu\|^2$$

where $X$ is $N \times D$, $Z$ is $N \times K$ and $\mu$ is $K \times D$

The K-means problem is to minimize the above objective function w.r.t. $\mu$ and $Z$
Consider the K-means objective function

\[ L(\mu, X, Z) = \| X - Z\mu \|^2 \]

- It is a non-convex objective problem
  - Many local minima possible
- Also NP-hard to minimize in general (note that \( Z \) is discrete)
- The K-means algorithm we saw is a heuristic to optimize this function
- K-means algorithm alternated between the following two steps
  - Fix \( \mu \), minimize w.r.t. \( Z \) (assign points to closest centers)
  - Fix \( Z \), minimize w.r.t. \( \mu \) (recompute the center means)
- Note: The algorithm usually converges to a local minima (though may not always, and it may just convergence “somewhere”). Multiple runs with different initializations can be tried to find a good solution.
Convergence of K-means Algorithm

- Each step (updating $Z$ or $\mu$) can never increase the objective.
- When we update $Z$ from $Z^{(t-1)}$ to $Z^{(t)}$
  \[
  L(\mu^{(t-1)}, X, Z^{(t)}) \leq L(\mu^{(t-1)}, X, Z^{(t-1)})
  \]
  because the new $Z^{(t)} = \arg \min_{Z} L(\mu^{(t-1)}, X, Z)$
- When we update $\mu$ from $\mu^{(t-1)}$ to $\mu^{(t)}$
  \[
  L(\mu^{(t)}, X, Z^{(t)}) \leq L(\mu^{(t-1)}, X, Z^{(t)})
  \]
  because the new $\mu^{(t)} = \arg \min_{\mu} L(\mu, X, Z^{(t)})$
Thus the K-means algorithm monotonically decreases the objective.
Choosing $K$

- One way to select $K$ for the K-means algorithm is to try different values of $K$, plot the K-means objective versus $K$, and look at the “elbow-point”

- For the above plot, $K = 6$ is the elbow point
- Can also use information criterion such as AIC (Akaike Information Criterion)

$$AIC = 2L(\hat{\mu}, X, \hat{Z}) + K \log D$$

and choose the $K$ that has the smallest AIC (discourages large $K$)
K-means: Some Limitations

- Makes hard assignments of points to clusters
  - A point either completely belongs to a cluster or does not belong at all
  - No notion of a soft assignment (i.e., probability of being assigned to each cluster: say $K = 3$ and for some point $x_i$, $p_1 = 0.7$; $p_2 = 0.2$; $p_3 = 0.1$)
- Works well only if the clusters are roughly of equal sizes
- Probabilistic clustering methods such as Gaussian mixture models can handle both these issues (model each cluster using a Gaussian distribution)
- K-means also works well only when the clusters are round-shaped and does badly if the clusters have non-convex shapes

- Kernel K-means or Spectral clustering can handle non-convex
Basic idea: Replace the Euclidean distance/similarity computations in K-means by the kernelized versions.

\[ d(x_i, \mu_k) = \| \phi(x_i) - \phi(\mu_k) \| \]
\[ \| \phi(x_i) - \phi(\mu_k) \|^2 = \| \phi(x_i) \|^2 + \| \phi(\mu_k) \|^2 - 2\phi(x_i)^T \phi(\mu_k) \]
\[ = k(x_i, x_i) + k(\mu_k, \mu_k) - 2k(x_i, \mu_k) \]

where \( k(\cdot, \cdot) \) denotes the kernel function and \( \phi \) is its (implicit) feature map.
Kernel $K$-Means

- Note: $\phi$ does not have to be computed/stored for data $\{x_i\}$ or the cluster means $\{\mu_k\}$ because computations only depend on kernel evaluations.
Hierarchical Clustering

- Agglomerative (bottom-up) Clustering
- Divisive (top-down) Clustering
- Agglomerative is more popular and simpler than divisive (but less accurate)
Hierarchical Clustering

- Agglomerative (bottom-up) Clustering
  1. Start with each example in its own singleton cluster
  2. At each time-step, greedily merge 2 most similar clusters
  3. Stop when there is a single cluster of all examples, else go to 2
Divisive (top-down) Clustering

1. Start with all examples in the same cluster
2. At each time-step, remove the “outsiders” from the least cohesive cluster
3. Stop when each example is in its own singleton cluster, else go to 2
Metric: A measure of distance between pairs of observations

- Euclidean distance: \( \|a - b\|_2 = \sqrt{\sum_i (a_i - b_i)^2} \)
- Squared Euclidean distance: \( \|a - b\|_2^2 = \sum_i (a_i - b_i)^2 \)
- Manhattan distance: \( \|a - b\|_1 = \sum_i |a_i - b_i| \)
- Maximum distance: \( \|a - b\|_\infty = \max_i |a_i - b_i| \)
- Mahalanobis distance: \( \sqrt{(a - b)^T S^{-1} (a - b)} \) (\( S \) is the Covariance matrix)
How to compute the dissimilarity between two clusters $R$ and $S$?

- Min-link or single link: results in chaining (clusters can get very large)
  \[ d(R, S) = \min_{x_R \in R, x_S \in S} d(x_R, x_S) \]

- Max-link or complete-link: results in small, round shaped clusters
  \[ d(R, S) = \max_{x_R \in R, x_S \in S} d(x_R, x_S) \]

- Average-link: compromise between single and complex linkage
  \[ d(R, S) = \frac{1}{|R||S|} \sum_{x_R \in R, x_S \in S} d(x_R, x_S) \]
Divisive Clustering

- Bisecting K-means: Repeating 2-means algorithm until we have a desired number of clusters
- MST-based method: Build a minimum spanning tree from the dissimilarity graph, and then make new clusters by breaking the link corresponding to the largest dissimilarity
Divisive Clustering (Contd.)

- Dissimilarity analysis: In each step
  - Start with a singular cluster containing all the data $G$
  - Measure the average dissimilarity of $i \in G$ to all the other $i' \in G$

$$d_i^G = \frac{1}{n_G} \sum_{i' \in G} d_{i,i'}$$

- Remove the most dissimilar data $i^*$ and put it in its own cluster $H$

$$i^* = \arg \max_{i \in G} d_i^G, \quad G = G \setminus \{i^*\}, \quad H = \{i^*\}$$

- Repeat picking a point $i^*$ to move that maximizes the average dissimilarity to each $i' \in G$ but minimize the average dissimilarity to each $i' \in H$

$$d_i^H = \frac{1}{n_H} \sum_{i' \in H} d_{i,i'}, \quad i^* = \arg \max_{i \in G} \left( d_i^G - d_i^H \right)$$

until $d_i^G - d_i^H$ is negative
We can recursively call the algorithm on $G$ and/or $H$, or any other node in the tree. E.g., choose to split the node whose average dissimilarity is highest, or whose maximum dissimilarity is highest.

We continue the process until the average dissimilarity within each cluster is below some threshold, and/or all clusters are singletons.
Example: Agglomerative Clustering
Example: Agglomerative Clustering

![Graph showing iteration 001 with data points on a 2D plane with axes V1 and V2.](image-url)
Example: Agglomerative Clustering

Iteration 007

![Graph showing data points in V1-V2 space after iteration 007 of Agglomerative Clustering.](image)
Example: Agglomerative Clustering

iteration 017

V2

V1
Example: Agglomerative Clustering
Example: Agglomerative Clustering

iteration 022
Example: Agglomerative Clustering
Flat clustering produces a single partitioning
Hierarchical Clustering can give different partitionings depending on the level-of-resolution we are looking at
Flat clustering needs the number of clusters to be specified
Hierarchical clustering does not need the number of clusters to be specified
Flat clustering is usually more efficient run-time wise
Hierarchical clustering can be slow (has to make several merge/split decisions)
No clear consensus on which of the two produces better clustering
Thanks!

Q & A