Clustering + K-Means Clustering

Thorsten Joachims
What does Single-Link HAC optimize?

- Single-link HAC finds the clustering $C_1, ..., C_K$ that maximizes the following objective:

$$M_3 = \min_{x_s, x_t: c(x_s) \neq c(x_t)} \text{dissimilarity}(x_s, x_t)$$

[Maximize smallest between-cluster distance]
Observation

Say c is the clustering produced by single-link HAC. Then it always holds

$$\min_{x_s, x_t: c(x_s) \neq c(x_t)} \text{dissimilarity}(x_s, x_t) > \text{merged distances in tree}$$
Demo
Demo
What does Single-Link HAC optimize?

• Single-link HAC finds the clustering $c_1, \ldots, c_K$ that maximizes the following objective:

$$M_3 = \min_{x_s, x_t: c(x_s) \neq c(x_t)} \text{dissimilarity}(x_s, x_t)$$

[Maximize smallest between-cluster distance]
Proof:

Say $c$ is solution produced by single-link clustering

Key observation:

$$\min_{t,s: c(x_t) \neq c(x_s)} \text{dissimilarity}(x_i, x_j) > \text{Merged distances in tree}$$

Say $c' \neq c$ then,

$$\exists \ t, s \ \text{s.t.} \ c'(x_t) \neq c'(x_s) \ \text{but} \ c(x_t) = c(x_s)$$

Points merged by single link (a tree)
Summary

• Clustering: Find partitioning of the data points.

• HAC: Repeatedly merge clusters bottom up.
  • Design: Distance measure and merge criterion.
  • Efficiency: $O(n^2 \log n)$

• Single-link optimizes $M_3$ criterion.
For all $j \in [K]$, initialize cluster centroids $\hat{r}_j^0$ randomly and set $m = 1$

Repeat until convergence (or until patience runs out)

1. For each $t \in \{1, \ldots, n\}$, set cluster identity of the point

$$\hat{c}^m(x_t) = \arg\min_{j \in [K]} \|x_t - \hat{r}_j^{m-1}\|$$

2. For each $j \in [K]$, set new representative as

$$\hat{r}_j^m = \frac{1}{|\hat{C}_j^m|} \sum_{x_t \in \hat{C}_j^m} x_t$$

3. $m \leftarrow m + 1$
Demo
Demo
Demo
K-means Clustering

- For all \( j \in [K] \), initialize cluster centroids \( \hat{r}_j^0 \) randomly and set \( m = 1 \)

- Repeat until convergence (or until patience runs out)
  1. For each \( t \in \{1, \ldots, n\} \), set cluster identity of the point
     \[
     \hat{c}^m(x_t) = \arg\min_{j \in [K]} \| x_t - \hat{r}_j^{m-1} \|
     \]
  2. For each \( j \in [K] \), set new representative as
     \[
     \hat{r}_j^m = \frac{1}{|\hat{C}_j^m|} \sum_{x_t \in \hat{C}_j^m} x_t
     \]
  3. \( m \leftarrow m + 1 \)

Question: Does the K-Means algorithm always terminate, or can it run into an infinite loop?
K-means Convergence

- K-means algorithm converges to local minima of objective

\[ O(c; r_1, \ldots, r_K) = \sum_{j=1}^{K} \sum_{c(x_t) = j} \| x_t - r_j \|_2^2 \]

- Proof:

\[ C^m(x_t) = \min_{j \in \{1, \ldots, K\}} \left( x_t - r_j^m \right)^2 \]
K-means Convergence

K-means algorithm converges to local minima of objective

\[ O(c; r_1, \ldots, r_K) = \sum_{j=1}^{K} \sum_{c(x_t)=j} \|x_t - r_j\|_2^2 \]

Proof:
Clustering assignment improves objective:

\[ O(\hat{c}^{m-1}; r_1^{m-1}, \ldots, r_K^{m-1}) \geq O(\hat{c}^m; r_1^{m-1}, \ldots, r_K^{m-1}) \]

(By definition of \(\hat{c}^m(x_t)\))
Computing centroids improves objective:

\[ O(\hat{c}^m; r_1^{m-1}, \ldots, r_K^{m-1}) \geq O(\hat{c}^m; r_1^m, \ldots, r_K^m) \]

(By the fact about centroid)
Fact: Centroid is Minimizer

\[ \forall r_j, \sum_{t \in C_j} \left\| x_t - \frac{1}{|C_j|} \sum_{s \in C_j} x_s \right\|^2 \leq \sum_{t \in C_j} \left\| x_t - r_j \right\|^2 \]

\[ 0 = \frac{\partial}{\partial r_j} \sum_{t \in C_j} (x_t - r_j)^2 = \sum_{t \in C_j} 2(x_t - r_j) = -2|C_j|/r_j + 2 \sum_{t \in C_j} x_t \]

\[ \implies \frac{2}{|C_j|/r_j} = 2 \sum_{t \in C_j} x_t \]

\[ r_j = \frac{\sum_{t \in C_j} x_t}{|C_j|} \quad \text{minimum } r_j \]
Time Complexity

• Assume computing distance between two instances is $O(d)$ where $d$ is the dimensionality of the vectors.
• Reassigning clusters for $n$ points: $O(Kn)$ distance computations, or $O(Knd)$.
• Computing centroids: Each instance gets added once to some centroid: $O(nd)$.
• Assume these two steps are each done once for $i$ iterations: $O(iKnd)$.
• Linear in all relevant factors, assuming a fixed number of iterations.
Buckshot Algorithm

Problem
• Results can vary based on random seed selection, especially for high-dimensional data.
• Some seeds can result in poor convergence rate, or convergence to sub-optimal clusterings.

• First randomly take a sample of instances of size $n^{1/2}$
• Run average-link HAC on this sample
• Use the results of HAC as initial seeds for K-means.
• Overall algorithm is efficient and avoids problems of bad seed selection.
K-means: pitfalls
K-means: pitfalls
K-means: pitfalls
K-means: pitfalls
K-means: pitfalls
K-means: pitfalls
K-means: pitfalls
K-means: pitfalls

- Looks for spherical clusters
- Of same radius
- And with roughly equal number of points
Can we design algorithm that can address these shortcomings?