Machine Learning

Clustering

Fabio Vandin December 20^{th}, 2022

Classes of Algorithms for Clustering

1. Cost minimization algorithms
2. Linkage-based algorithms

Cost Minimization Clustering

Common approach in clustering:

- define a cost function over possible partitions of the objects
- find the partition (=clustering) of minimal cost

Assumptions:

- data points $\mathbf{x} \in \mathcal{X}$ come from a larger space $\mathcal{X}'$, that is
  input $\rightarrow$ $\mathcal{X} \subseteq \mathcal{X}'$
- distance function $d(\mathbf{x}, \mathbf{x}')$ for $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$

For simplicity: assume $\mathcal{X}' = \mathbb{R}^d$ and $d(\mathbf{x}, \mathbf{x}') = ||\mathbf{x} - \mathbf{x}'||$

k-Means Clustering

Input: data points $\overbrace{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m}^{\in \mathbb{R}^d}$; $k \in \mathbb{N}^+$

Goal: find

- partition $C = (C_1, C_2, \ldots, C_k)$ of $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m$;
- *centers* $\mu_1, \mu_2, \ldots, \mu_k$ with $\mu_i \in \mathcal{X}'$ *center* for C_i, $1 \le i \le k$

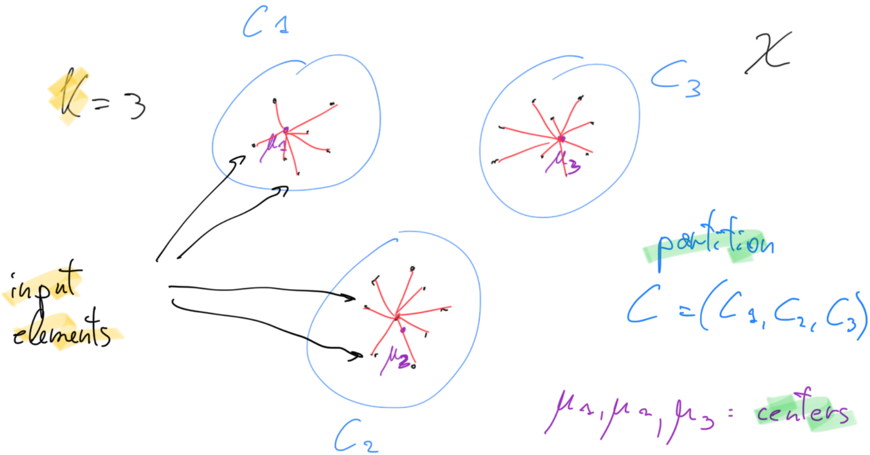
that minimizes the k-**means objective** (cost)

$$\sum_{i=1}^{k} \underbrace{\sum_{\mathbf{x} \in C_i} d(\mathbf{x}, \mu_i)^2}$$

if C_i is fixed: find $\vec{\mu}_i$ that minimizes

$$\sum_{\vec{x} \in C_i} d(\vec{x}, \vec{\mu}_i)^2$$

Example



$k = 3$
C_1
μ_1
C_3
μ_3
$\mathcal{X}$
input elements
C_2
μ_2
partition
$C = (C_1, C_2, C_3)$
μ_1, μ_2, μ_3 : centers

Other Objectives (Costs)

k-**medoids objective**:

$$\text{centers } \min_{\mu_1,\ldots,\mu_k \in \mathcal{X}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} d(\mathbf{x}, \mu_i)^2$$

input

k-**median objective**:

$$\min_{\mu_1,\ldots,\mu_k \in \mathcal{X}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} d(\mathbf{x}, \mu_i)$$

Back to k-means clustering

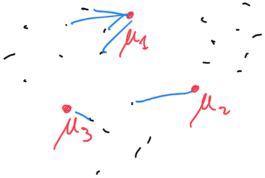
What is more difficult: finding the clusters or finding the centers?

Proposition

Given a cluster C_i, the center μ_i that minimizes $\sum_{\mathbf{x}\in C_i} d(\mathbf{x},\mu_i)^2$ is

$$\mu_i = \frac{1}{|C_i|}\sum_{\mathbf{x}\in C_i}\mathbf{x}$$

Proof: Exercise



Algorithm for k-means clustering

Naive (brute-force) algorithm to solve k-means clustering?

Try all possible partitions of the m points into k clusters, evaluate each partition, and find the best one.

Is it efficient?

Depends on the number of partitions of m points into k clusters:

- trivial upper bound: k^m
- exact count: number of ways in which we can partition a set of m objects into k subsets $\Rightarrow$ Stirling number of the second kind:

$$S(m,k) = \frac{1}{k!}\sum_{j=0}^{k}(-1)^{k-j}\binom{k}{j}j^m$$

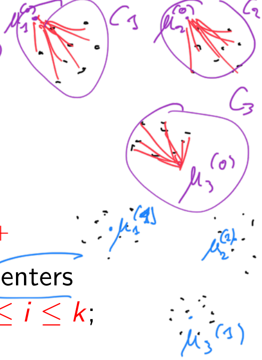
- simple bounds:
  - $S(m,k) \in O\left(\frac{k^m}{k!}\right)$
  - $S(m,k) \in \Omega\left(k^{m-k+1}\right)$

Fact

Finding the optimal solution for k-means clustering is computationally difficult (NP-hard). This is true for most optimization problems of cost minimization clusterings (including k-medoids and k-median)

Lloyd's Algorithm

$t=0$



A good practical heuristic to solve k-means

Input: data points $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$; $k \in \mathbb{N}^+$
Output: clustering $C = (C_1, C_2, \ldots, C_k)$ of $\mathcal{X}$; centers $\mu_1, \mu_2, \ldots, \mu_k$ with μ_i center for C_i, $1 \leq i \leq k$;

randomly choose $\mu_1^{(0)}, \ldots, \mu_k^{(0)}$;
for $t \leftarrow 0, 1, 2, \ldots$ **do** `/* until convergence */`
  for $i = 1, \ldots, k$: $C_i \leftarrow \{\mathbf{x} \in \mathcal{X} : i = \arg\min_j d(\mathbf{x}, \mu_j^{(t)})\}$;
  for $i = 1, \ldots, k$: $\mu_i^{(t+1)} \leftarrow \frac{1}{|C_i|} \sum_{\mathbf{x} \in C_i} \mathbf{x}$;
  if *convergence reached* **then**
    return $C = (C_1, \ldots, C_k)$ and $\mu_1^{(t+1)}, \mu_2^{(t+1)}, \ldots, \mu_k^{(t+1)}$

Notes

Convergence: commonly used criteria

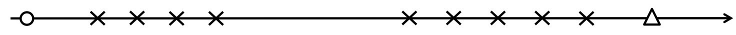
- the k-means objective for the cluster at iteration t is not lower than the k-means objective for the cluster at iteration $t-1$
- $\sum_{i=1}^{k} d(\mu_i^{(t+1)}, \mu_i^{(t)}) \leq \varepsilon$
- $\max_{1 \leq i \leq k} d(\mu_i^{(t+1)}, \mu_i^{(t)}) \leq \varepsilon$

Theorem

If the first convergence criteria above is used, then Lloyd's algorithm always terminates.

Exercize

Draw (approximately) the solution (clusters and centers) found by Lloyd algorithm for the 2 clusters ($k = 2$) problem, when the data ($x_i \in \mathbb{R}$) are the crosses in the figure below and the algorithm is initialised with center values indicated with the circle ($\circ$, cluster 1) and triangle ($\triangle$, cluster 2) shown in the figure.



Complexity of Lloyd's Algorithm

Complexity:

- Assignment of points $\mathbf{x} \in \mathcal{X}$ to clusters C_i: time $O(kmd)$
- Computation of centers μ_i: time $O(md)$

If convergence after t iterations $\Rightarrow O(tkmd)$

How many iterations are required for convergence?

Number of Iterations of Lloyd's Algorithm

- the number of iterations can be exponential in the input size: a trivial upper bound is $\approx k^m$ as before
- more sophisticated studies: upper bound $O\left(m^{kd}\right)$ ($\mathbf{x} \in \mathbb{R}^d$)
- recent studies: lower bound $2^{\Omega\left(\sqrt{m}\right)}$ in the worst-case
- in practice: much less than m iterations are required

Note: the convergence and the quality of the clustering depends on the initialization of the centers!

Effective Centers Initialization

Is there a way to choose the initial centers that is efficient but also provably leads to good clusters?

`k-means++`: simple but effective center initialization strategy proposed by D. Arthur and S. Vassilvitskii (article: D. Arthur and S. Vassilvitskii. *k-means++: the advantages of careful seeding.* Proc. of ACM-SIAM SODA 2007.)

Algorithm `k-means++`

$\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$, with $\mathbf{x}_i \in \mathbb{R}^d$ for $1 \leq i \leq m$; $k \in \mathbb{N}^+$

Given a point $\mathbf{x} \in \mathcal{X}$ and a set F, let $d(\mathbf{x}, F) = \min_{\mathbf{f} \in F} d(\mathbf{x}, \mathbf{f})$

The algorithm to compute the initial set F of centers is the following:

$\mu_1 \leftarrow$ random point from $\mathcal{X}$ chosen uniformly at random;
$F \leftarrow \{\mu_1\}$;
for $i \leftarrow 2$ **to** k **do**
  $\mu_i \leftarrow$ random point from $\mathcal{X} \setminus F$, choosing point $\mathbf{x}$ with probability $\frac{(d(\mathbf{x},\mathbf{F}))^2}{\sum_{\mathbf{x}' \in \mathcal{X} \setminus F} (d(\mathbf{x}',\mathbf{F}))^2}$;
  $F \leftarrow F \cup \{\mu_i\}$;
return F;

The following result is proved in the original paper by D. Arthur and S. Vassilvitskii.

Theorem

Let $\Phi^*_{k-means}(\mathcal{X}, k)$ be the cost of the optimal (i.e., minimum) k-means clustering of $\mathcal{X}$, and let $\Phi_{k-means}(\mathcal{X}, F_{k-means++})$ be the cost of the clustering $\mathcal{X}$ obtained by:

- using the points i $F_{k-means++}$ returned by `k-means++` as centers;
- assigning each point of $\mathcal{X}$ to its closest center.

(Note that $\Phi(\mathcal{X}, F_{k-means++})$ is a random variable.) Then

$$\mathbb{E}[\Phi_{k-means}(\mathcal{X}, F_{k-means++})] \leq 8(\ln k + 2)\Phi^*_{k-means}(\mathcal{X}, k).$$

Notes:

- the expectation $\mathbb{E}[\Phi_{k-means}(\mathcal{X}, F_{k-means++})]$ is over all possible sets $F_{k-means++}$ returned by `k-means++` (with input $\mathcal{X}$), which depends on the random choices in `k-means++`.
- `k-means++` already provides a good solution for k-means, but it makes sense to use it to initialize centers in Lloyd's algorithm (the solution can only improve in the next iterations, if the first convergence criteria is used)

Linkage-Based Clustering

General class of algorithms that follow the general scheme below.

Algorithm

1. start from the trivial clustering: each data point is a (single-point) cluster
2. **until "termination condition"**: repeatedly merge the "closest" clusters of the previous clustering

We need to specify two "parameters":

- how to define distance between clusters
- termination condition

Linkage-Based Clustering (continue)

Different distances $D(A, B)$ between two clusters A and B can be used, resulting into different linkage methods:

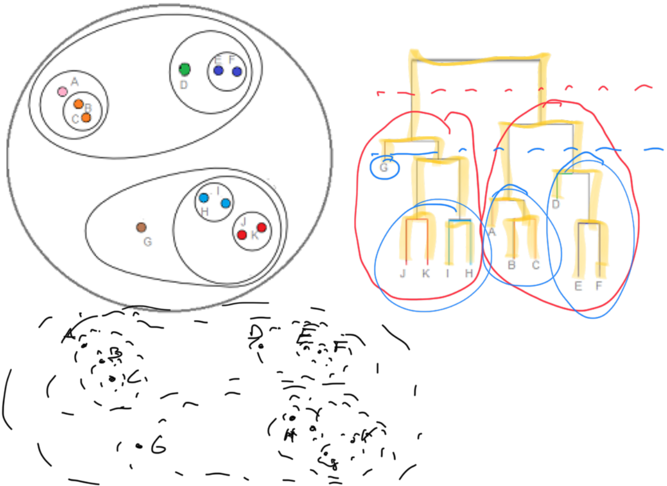
- **single linkage**: $D(A, B) = \min\{d(\mathbf{x}, \mathbf{x}') : \mathbf{x} \in A, \mathbf{x}' \in B\}$
- **average linkage**: $D(A, B) = \frac{1}{|A||B|} \sum_{\mathbf{x} \in A, \mathbf{x}' \in B} d(\mathbf{x}, \mathbf{x}')$
- **max linkage**: $D(A, B) = \max\{d(\mathbf{x}, \mathbf{x}') : \mathbf{x} \in A, \mathbf{x}' \in B\}$

Common termination condition:

- data points are partitioned into k clusters
- minimum distance between pairs of clusters is $> r$, where r is a parameter provided in input
- all points are in a cluster $\Rightarrow$ output is a dendrogram

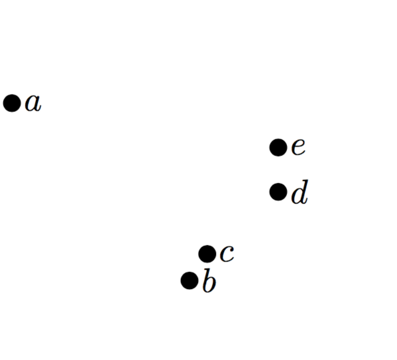
Dendrogram: Example

Dendrogram: tree, with input points $\mathbf{x} \in \mathcal{X}$ as leaves, that shows the arrangement/relation between clusters.



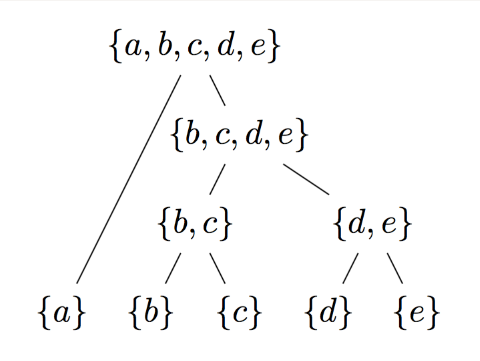
Exercize

Let the dataset $\mathcal{X}$ be as in figure below. Show the output of running the single linkage clustering algorithm when the termination condition is given by having all points in a cluster.



Solution

The output is a dendrogram:



Choice of number k of clusters

Choosing the number k of clusters (e.g., for k-means) is not easy.

Common approach:

1. run clustering algorithm for various values of k, obtaining a clustering $C^{(k)} = \{C_1^{(k)}, C_2^{(k)}, \ldots, C_k^{(k)}\}$ for each value of k considered;
2. use a *score* S to evaluate each clustering $C^{(k)}$, getting scores $S(C^{(k)})$ for each value of k
3. pick the value of k (and clustering) of maximum score: $C = \arg\max_{C^{(k)}} \{S(C^{(k)})\}$

A very common score based on distances alone: *silhouette*

Silhouette

Given a clustering $C = (C_1, C_2, \ldots, C_k)$ of $\mathcal{X}$ and a point $\mathbf{x} \in \mathcal{X}$, let $C(\mathbf{x})$ be the cluster to which $\mathbf{x}$ is assigned to. Assume $|C_i| \geq 2 \ \forall \ 1 \leq i \leq k$. Define:

$$A(\mathbf{x}) = \frac{\sum_{\mathbf{x}' \neq \mathbf{x}, \mathbf{x}' \in C(\mathbf{x})} d(\mathbf{x}, \mathbf{x}')}{|C(\mathbf{x})| - 1}$$

Given a cluster $C_i \neq C(\mathbf{x})$, let

$$d(\mathbf{x}, C_i) = \frac{\sum_{\mathbf{x}' \in C_i} d(\mathbf{x}, \mathbf{x}')}{|C_i|}$$

and $B(\mathbf{x}) = \min_{C_i \neq C(\mathbf{x})} d(\mathbf{x}, C_i)$.

Then the *silhouette* $s(\mathbf{x})$ of $\mathbf{x}$ is

$$s(\mathbf{x}) = \frac{B(\mathbf{x}) - A(\mathbf{x})}{\max\{A(\mathbf{x}), B(\mathbf{x})\}}$$

Intuition: $s(\mathbf{x})$ measures if $\mathbf{x}$ is closer to points in its "nearest cluster" than to the cluster it is assigned to.

Question: what is the range for $s(\mathbf{x})$? [-1, 1]

The silhouette of clustering $C = (C_1, C_2, \ldots, C_k)$ is

$$S(C) = \frac{\sum_{\mathbf{x} \in \mathcal{X}} s(\mathbf{x})}{|\mathbf{X}|}$$

The higher $S(C)$, the better the clustering quality.