k-means

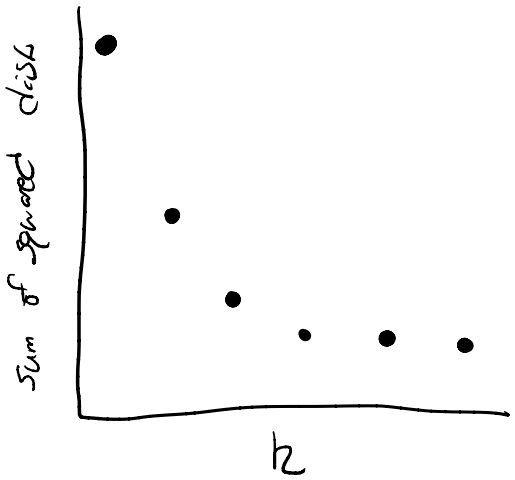
- if we know cluster centers → easy to assign
- if we know assignment → easy to compute centers

⇒ iterative algorithm

1) choose k
2) choose k initial cluster centers
3) repeat until cluster centers stop changing:
   - assign all points to cluster with nearest center
   - re-compute cluster centers as mean of point assigned to that cluster

- considerations
  - choosing initial centers
    a) select k points at random
       ↳ quality depends a lot on initialization,
         can get unlucky resulting in poor clustering
    b) initialize randomly several times
       ↳ choose clustering that performs best
- can end up with clusters with 0 elements
  ↳ add in step before recompute
    assign cluster with 0 elements
    point chosen at random

- considerations (cont.)
  - how to choose k
    - try different values of k
    - plot k vs $\sum_{k} \sum_{i \in k} (x_i - c_k)^2$
      - ↳ look for elbow



    - if have classes, look at cluster purity
  - often "bad" clusters
    - ↳ outlies mishape clusters
      - idea: junk cluster
        - ↳ anything too far from nearest center added to junk
        - ↳ don't estimate center of junk
      - or otherwise remove outliers

Modified Versions

Soft k-means: assign points to cluster centers with weights

let $a_{i,j} = \frac{e^{-\|x_i - c_j\|^2}}{2\sigma^2}$ σ = chosen scale

compute weights $w_{i,j} = \frac{a_{i,j}}{\sum_k a_{i,k}}$

recompute centers as weighted average

$$C_j = \frac{\sum_i w_{i,j} x_i}{\sum_i w_{i,j}}$$

K-medoids

- if we only have distance, but not no actual data point values
- Ex: distance between cities without knowing city locations
- use data items as cluster centers instead of averages
- update medoids by choosing data item to minimize sum of distances to points in cluster