Clustering

Sewoong Oh

CSE/STAT 416
University of Washington

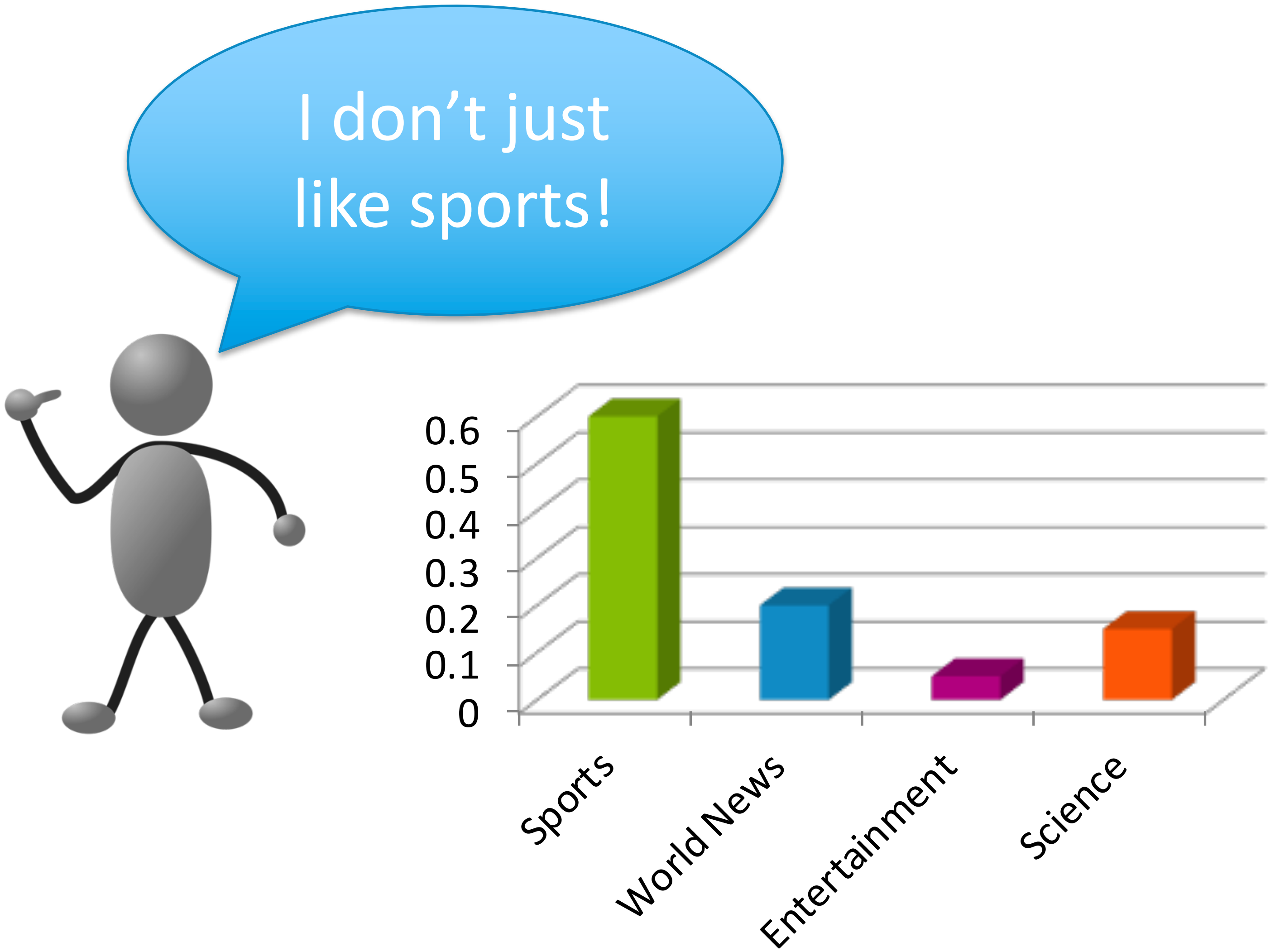
Clustering

SPORTS

WORLD NEWS

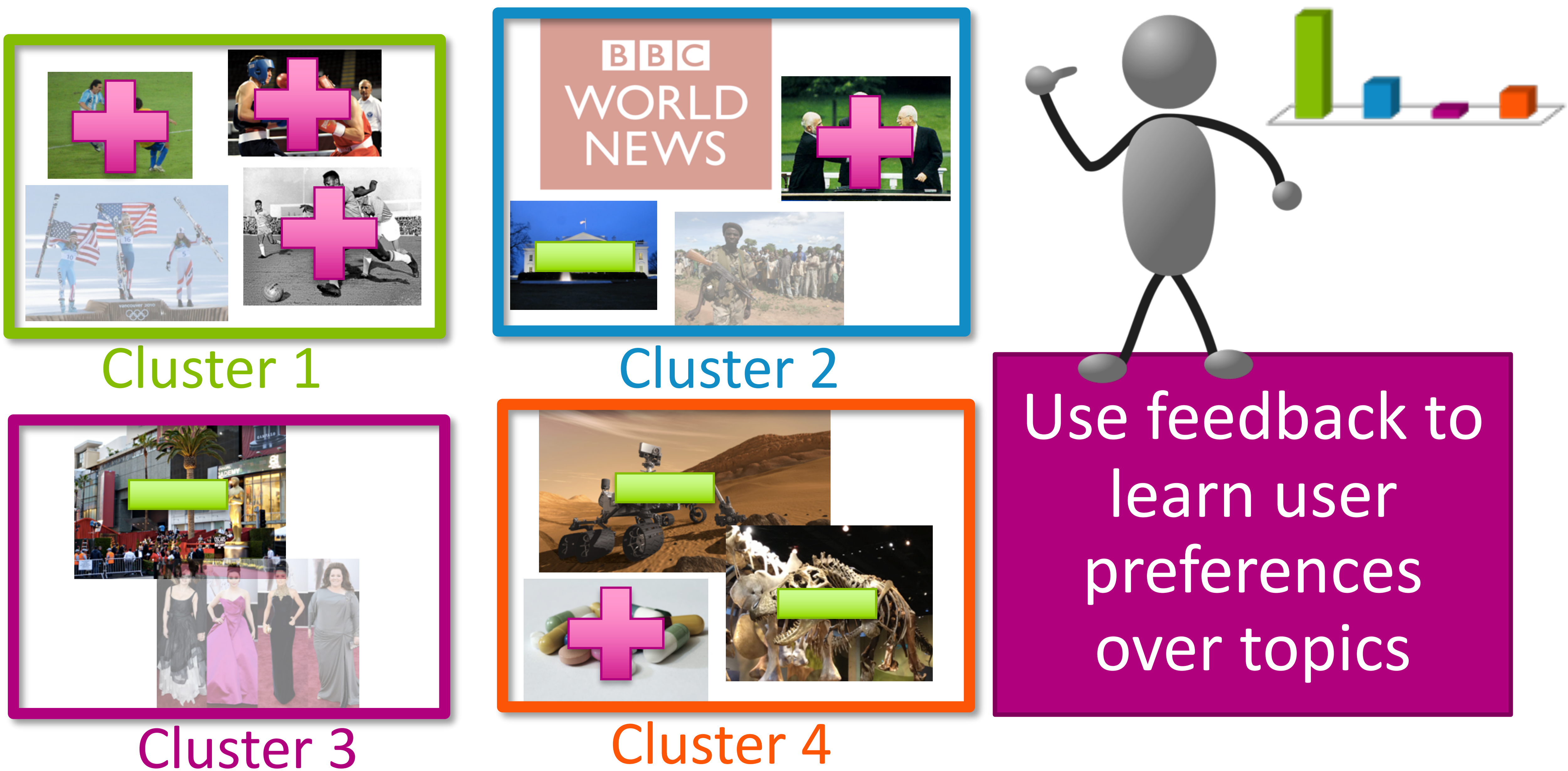
Why is clustering useful?

- User preference is important to learn, but challenging
- If we know a user's preference, we can recommend better



How do we learn a persons preference

- When the topics are not even pre-defined
- Let alone knowing which article falls into which group
- clustering: learns this from user feedback (rating, up/down)

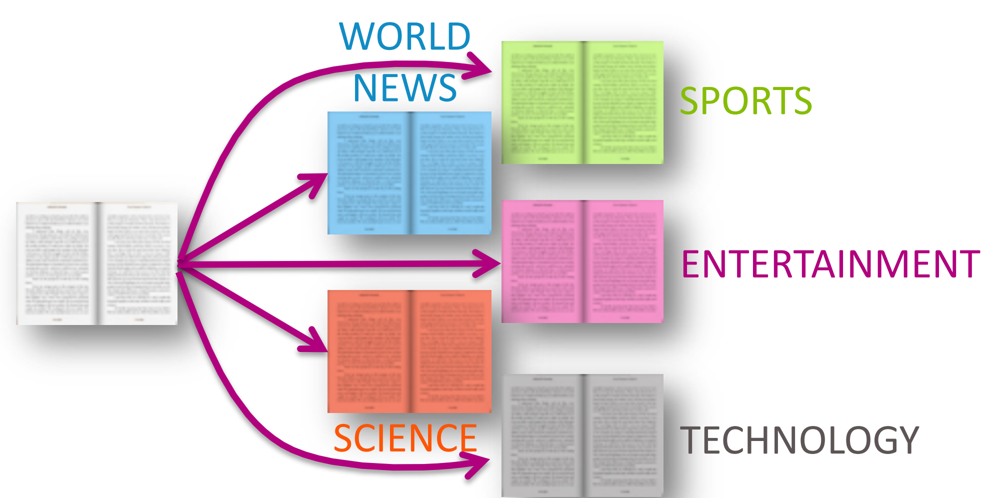


Clustering

- What if labels are known?

Training set of labeled docs

- Then we can use multiclass classification methods

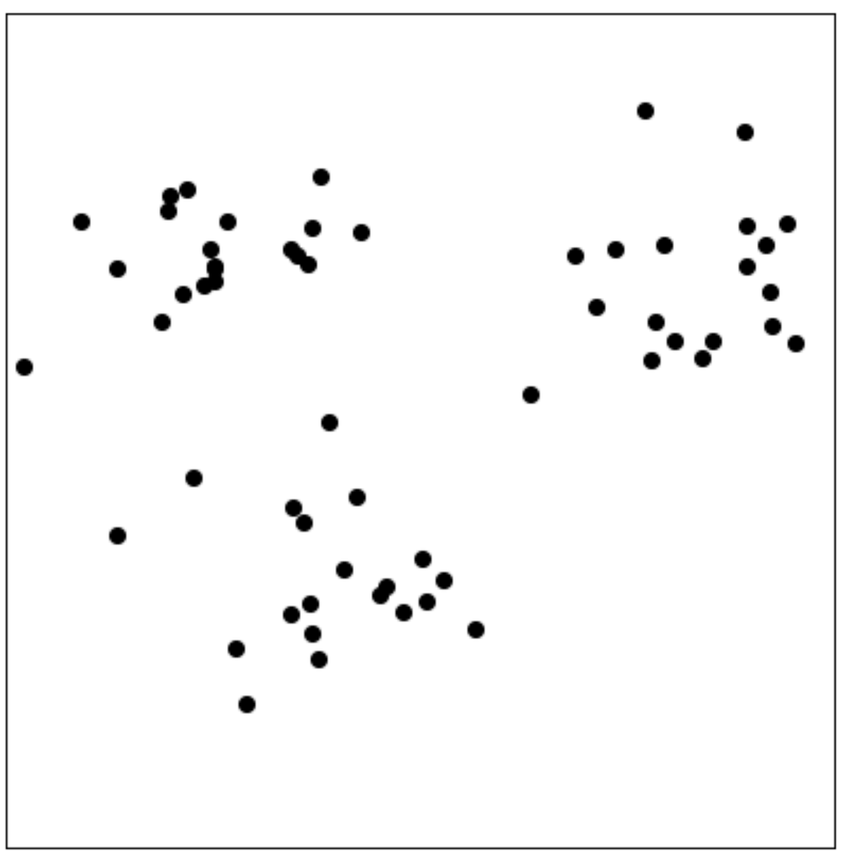


Example of supervised learning

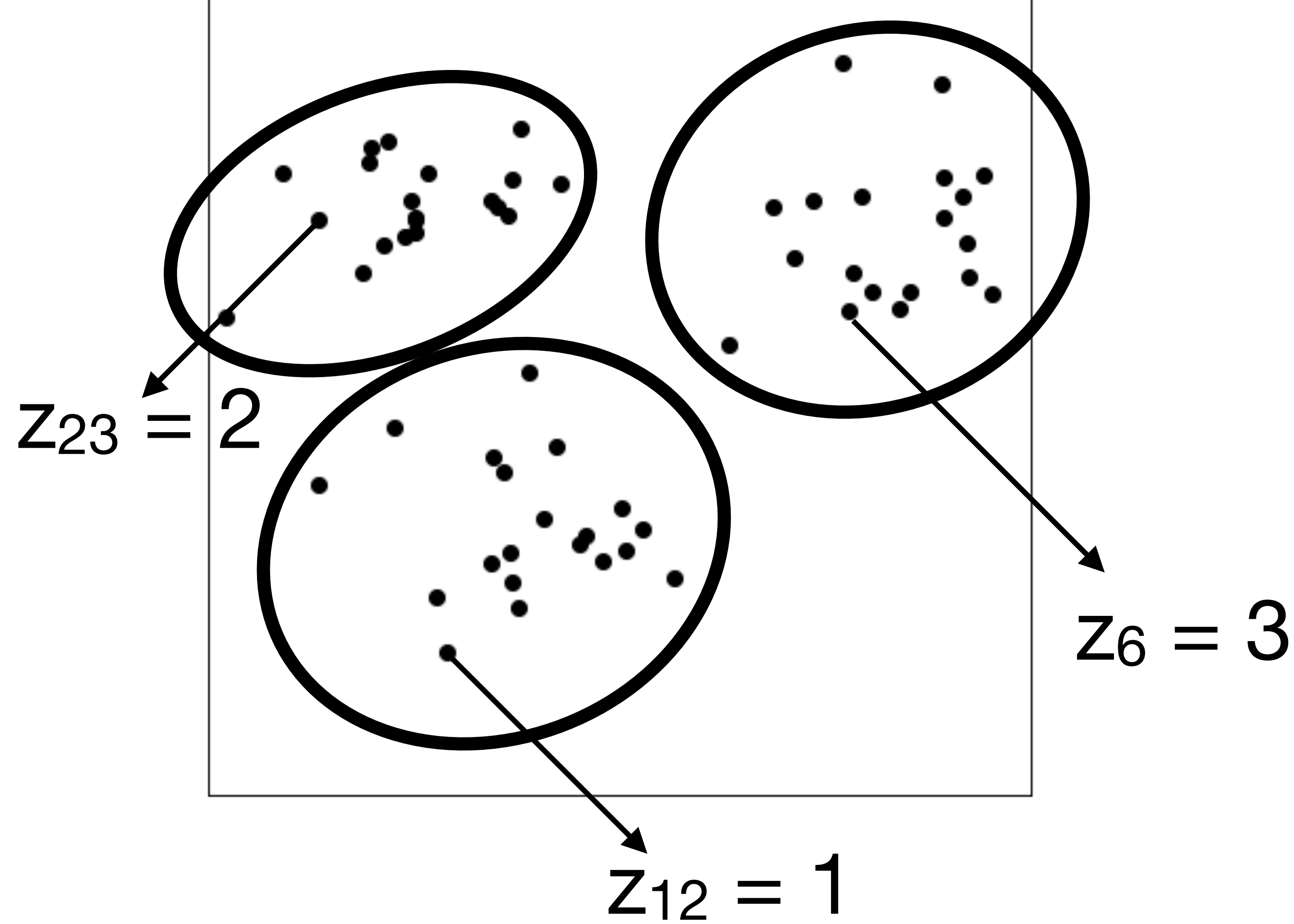
Clustering

- What if labels are unknown?
  - We need to uncover the structure (or pattern) from just x
  - ***Cluster*** is one of the most important patterns in real data
  - Finding clusters help, personalized medicine, targeted advertisement, scientific discovery, many other machine learning tasks

- Input: $x_1, \ldots, x_N$

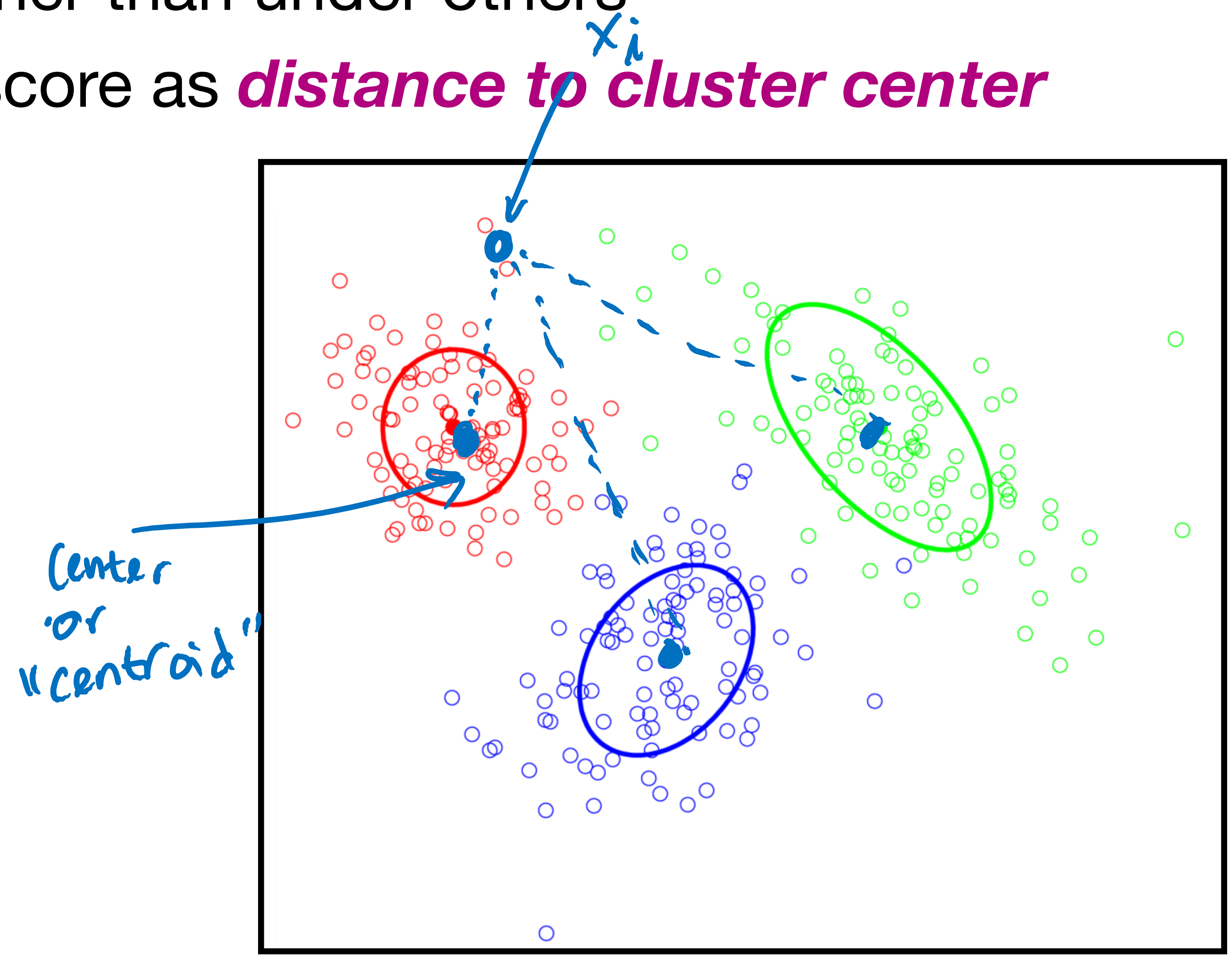


- Output: cluster label for each point z_i in $\{1,2,\ldots,k\}$



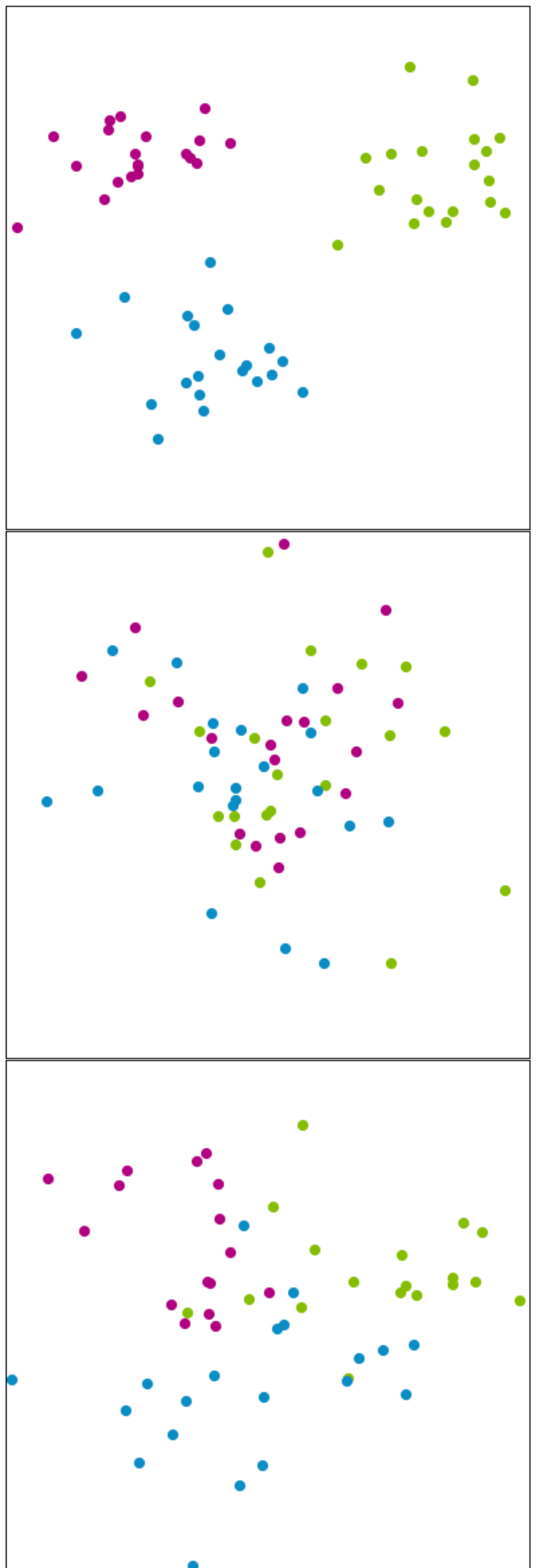
How is a cluster defined?

- In its simplest form, a cluster (on raw data) is defined by
  - The location of the ***center***
  - shape and size of the ***spread***
- An important step in defining what it means to be a cluster is
- Assign each observation x_i (doc) to cluster k (topic label) if
  - **Score** under cluster k is higher than under others
  - For simplicity, often define score as ***distance to cluster center*** (ignoring shape)



Clustering when distance of raw data captures the clusters

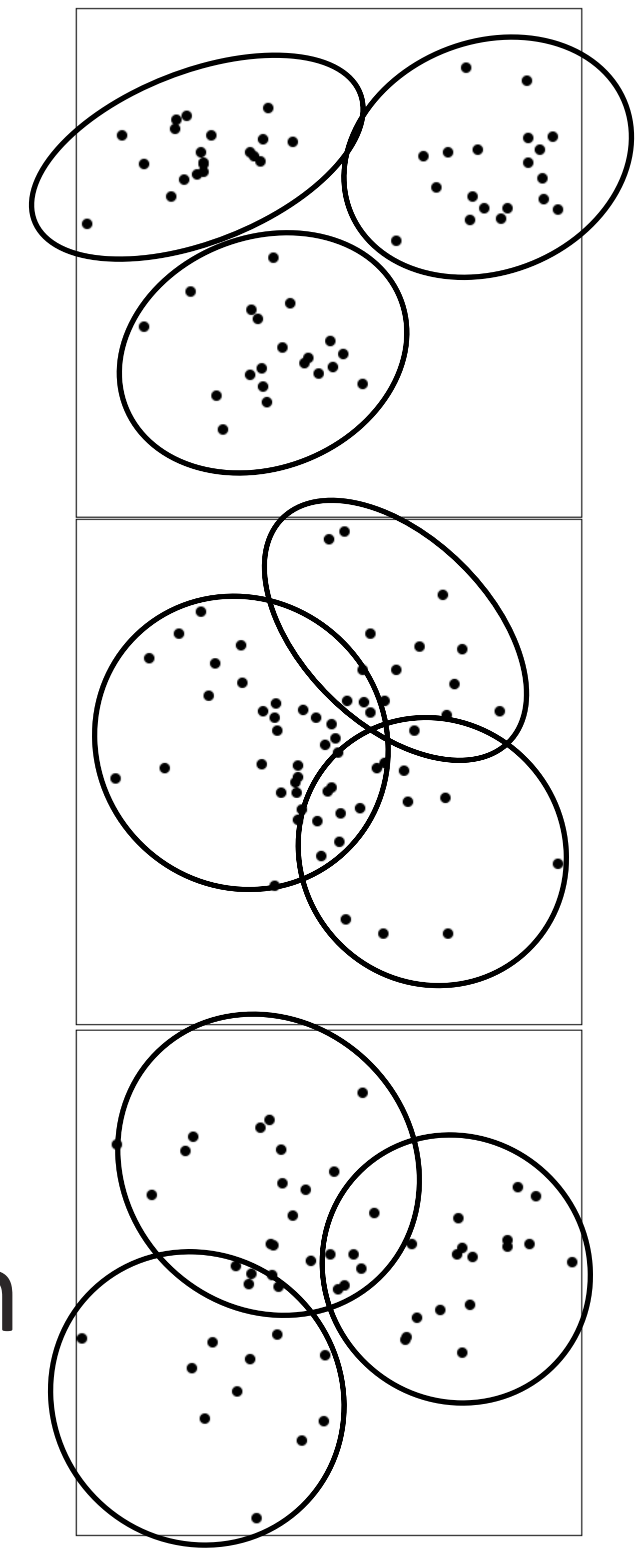
- Suppose the ground truth about the clusters is as follows.
- But data we are given do not have the ground truth labels



Easy

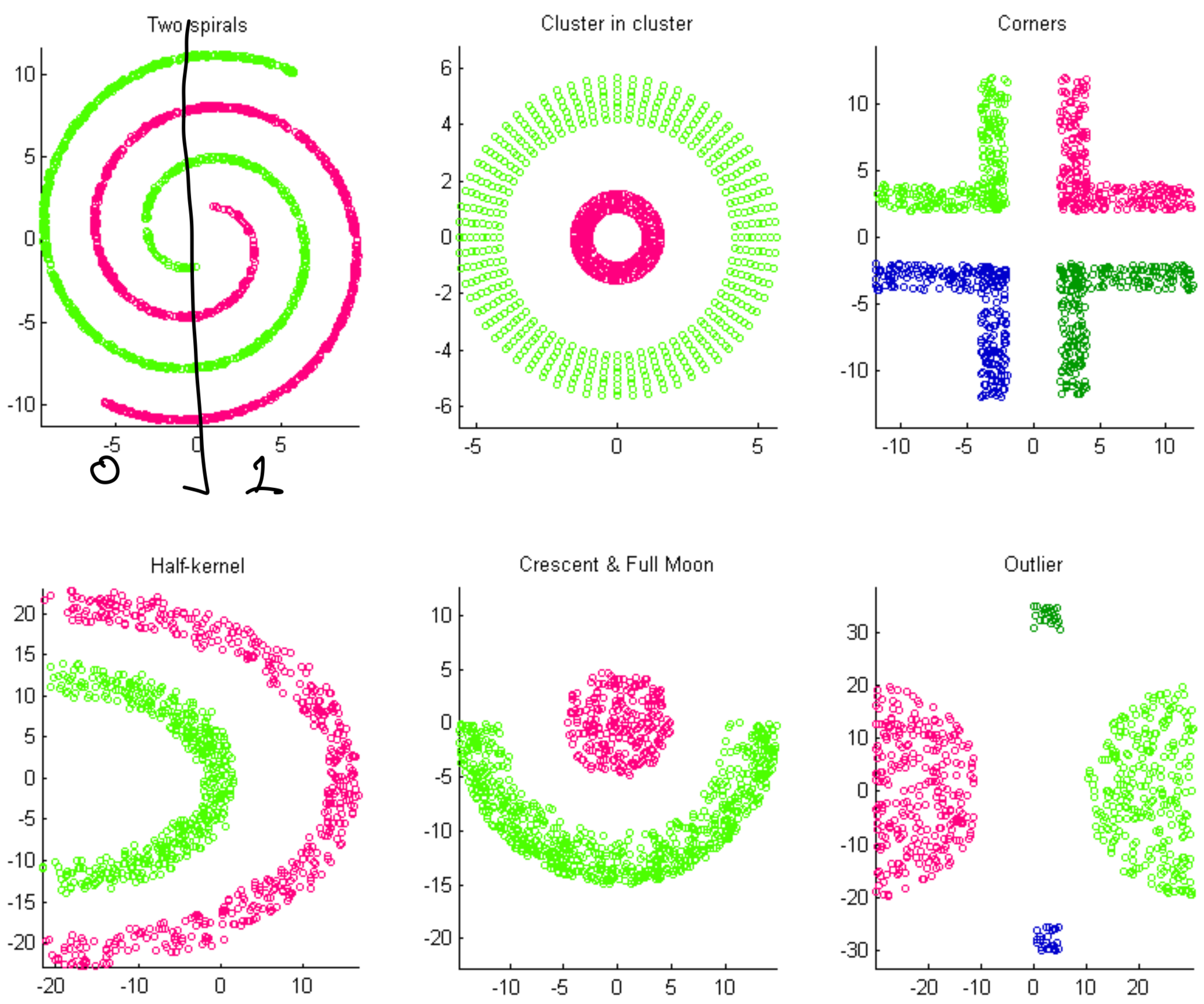
Impossible

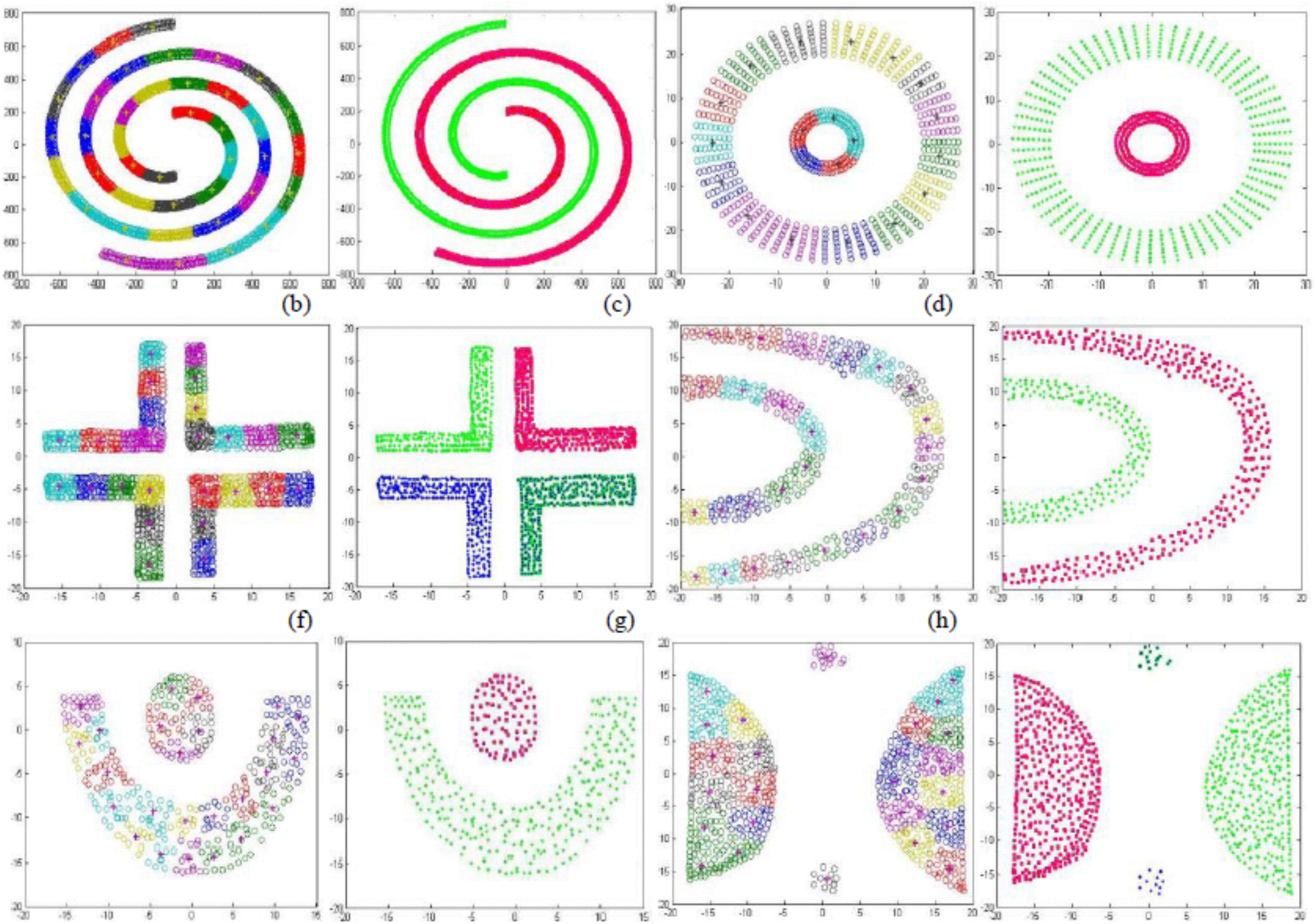
In between



The structure we are looking for can be quite complicated

- If the distance in the raw data does not reflect cluster structure




(b)
(c)
(d)
(f)
(g)
(h)

K-means clustering

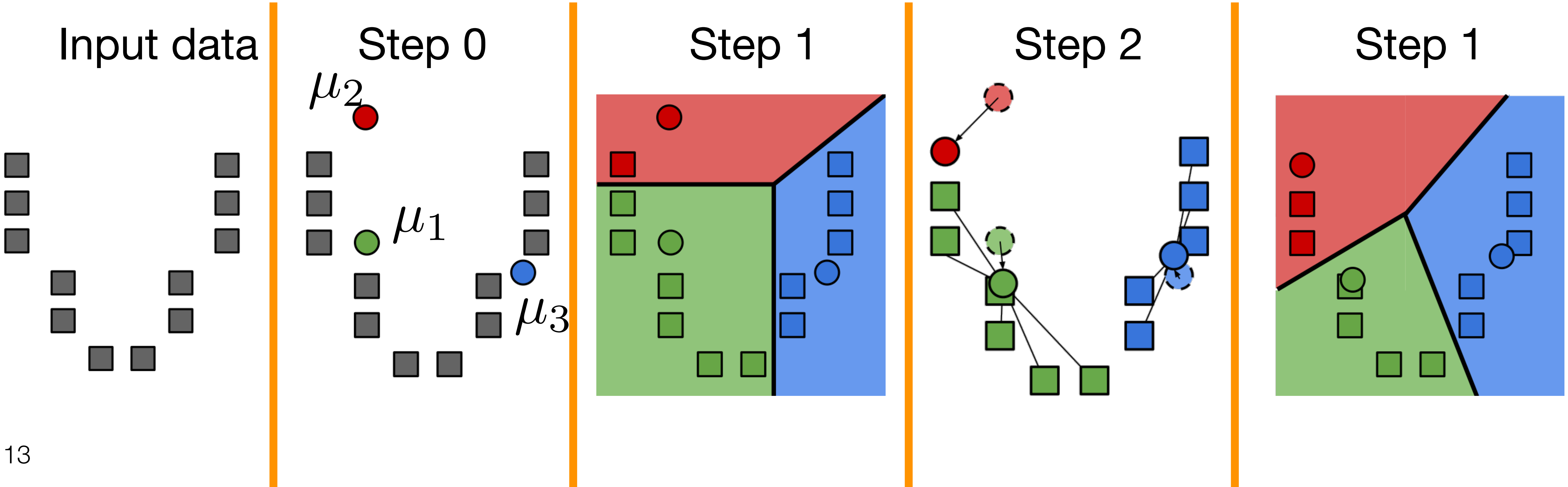
K-means algorithm

- k-means uses the **Score** between a data point $\boldsymbol{x_i}$, for some $\boldsymbol{i}$ in {***1,…,N***} and a center μ_j, for some cluster index $\boldsymbol{j}$ in {***1,…,k***} which is

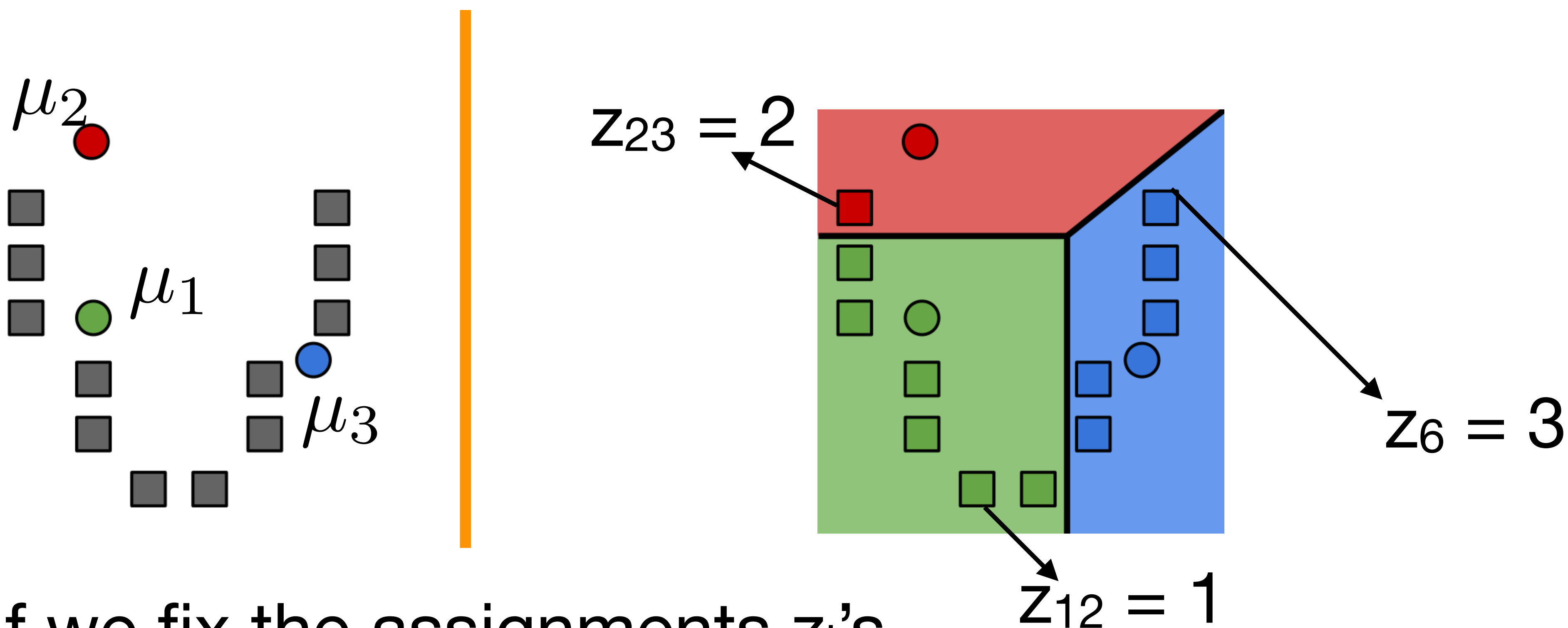
$$\text{score}(x_i, \mu_j) = \text{distance}(x_i, \mu_j)$$

- Smaller score is better

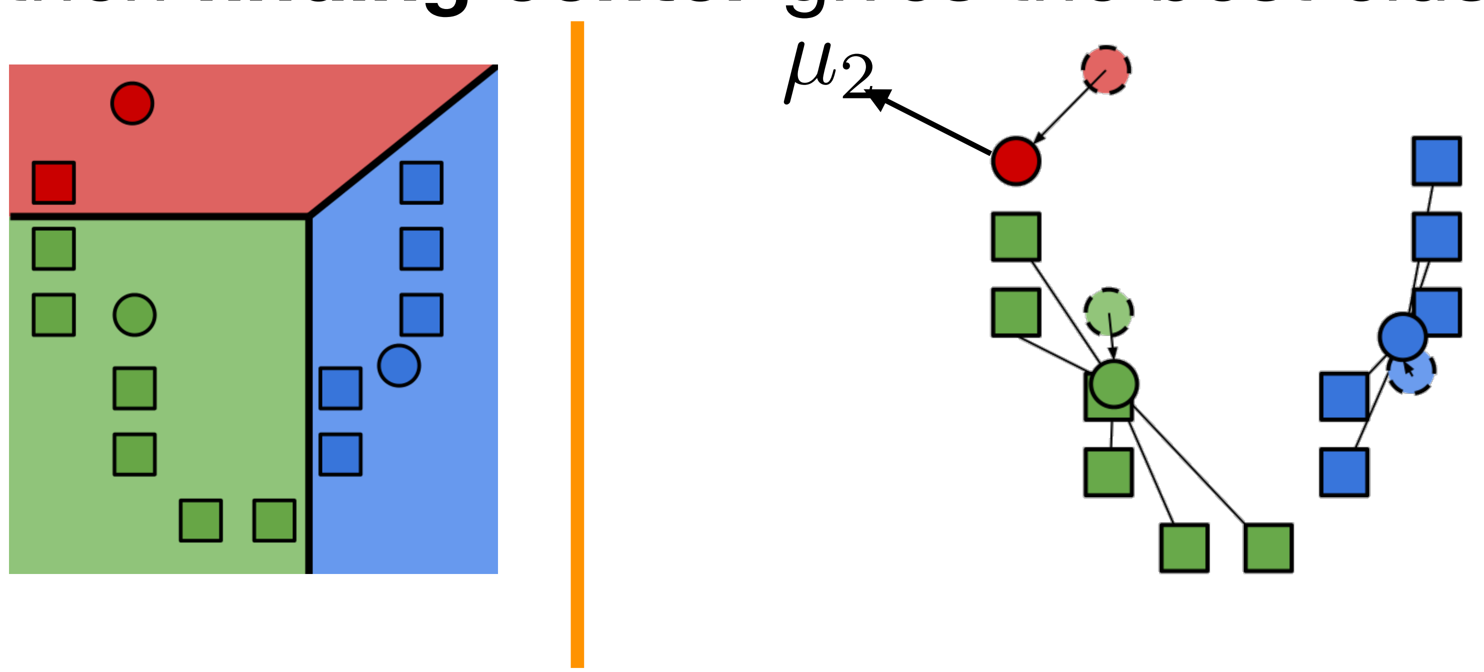
- Step 0: initialize cluster centers
- Repeat
  - Step 1: closest cluster to each **data point**
  - Step 2: update **cluster center** as the mean of assigned points



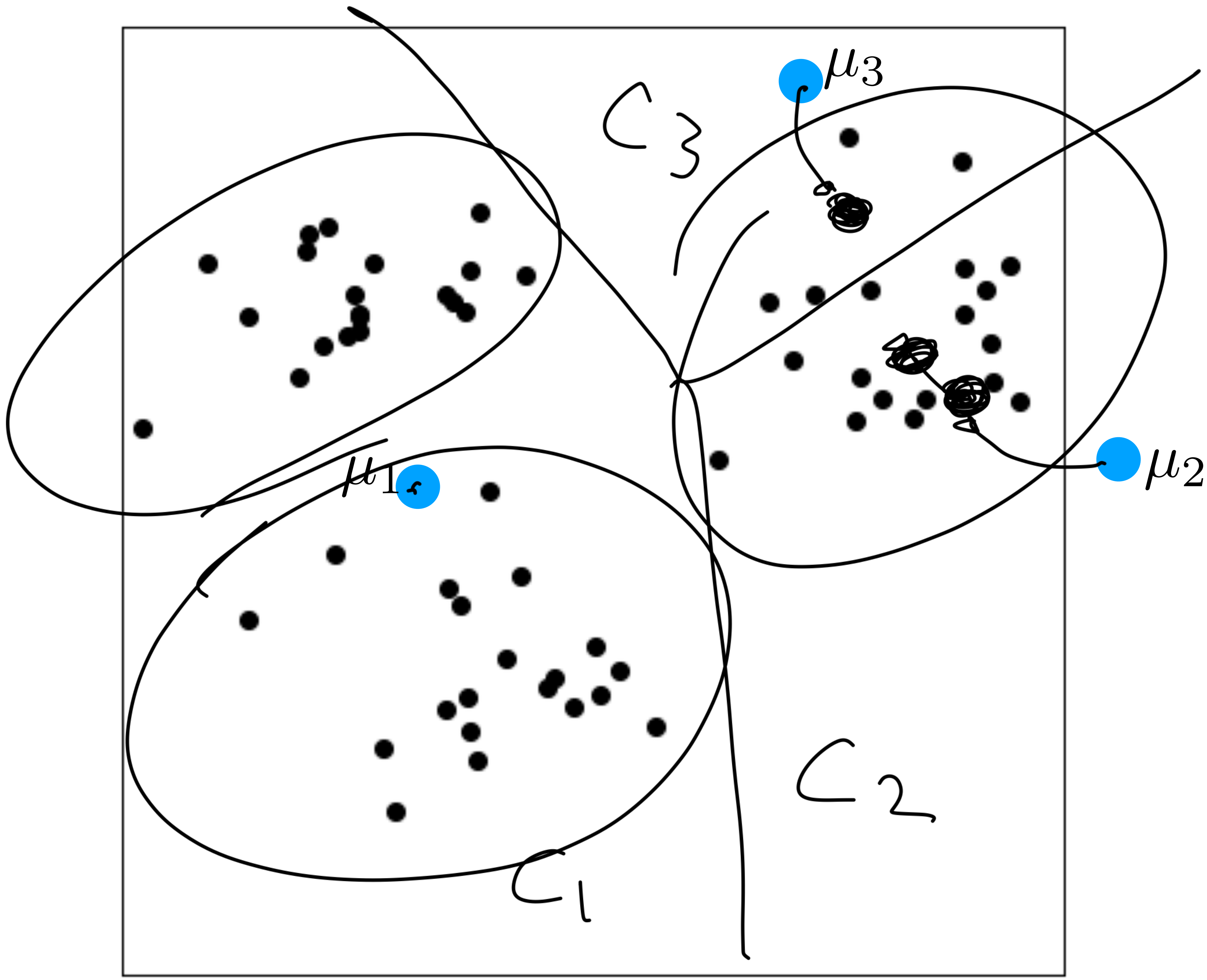
- idea: given that we use Euclidean distance as score
  - If we fix the current centers μ_j,
    then the ***nearest neighbor clustering*** gives the best cluster assignments z_i's

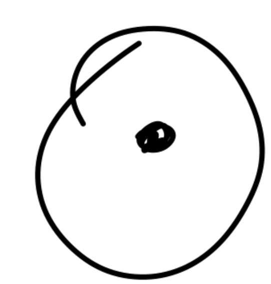


  - If we fix the assignments z_i's,
    then **finding center** gives the best cluster centers μ_j



K-means



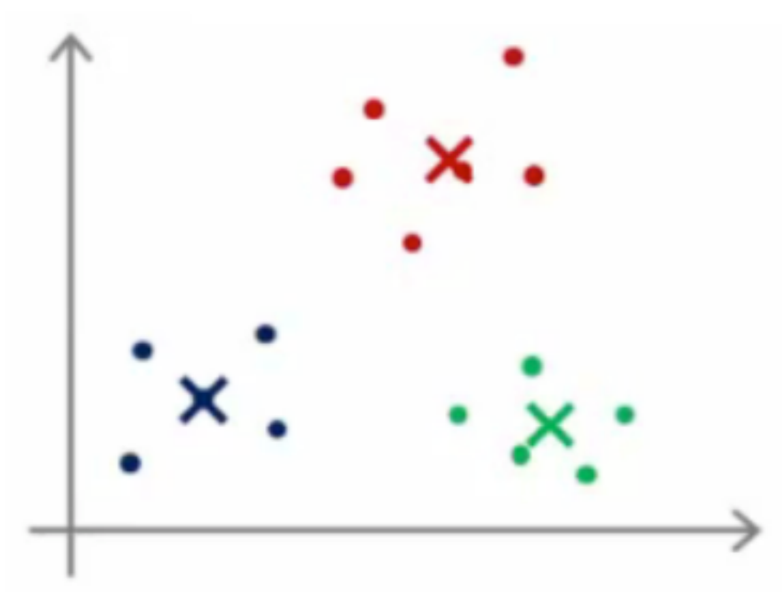


- If I give you a set of centers and assignments, can you tell if it resulted from running k-means until termination?

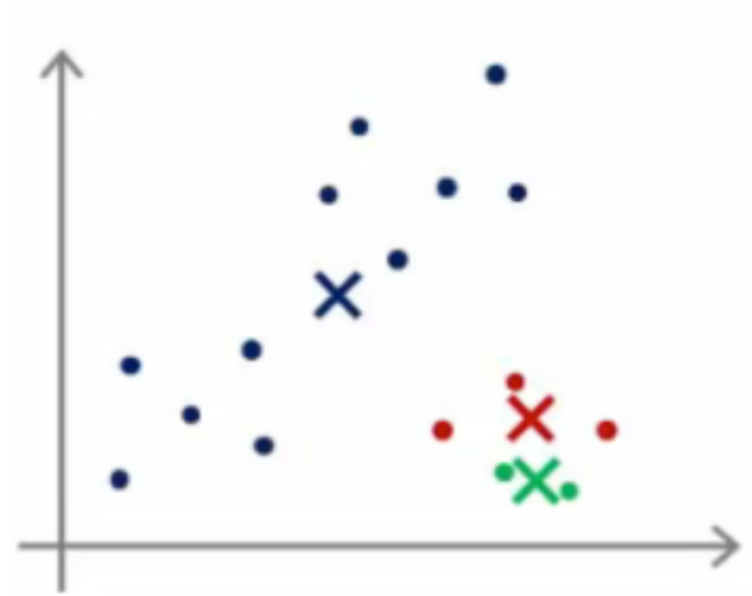
Which clustering can result from k-means?

- Can the algorithm run indefinitely? No.
- Let's say we ran k-means algorithm with some initial centers, until the center did not change any more.

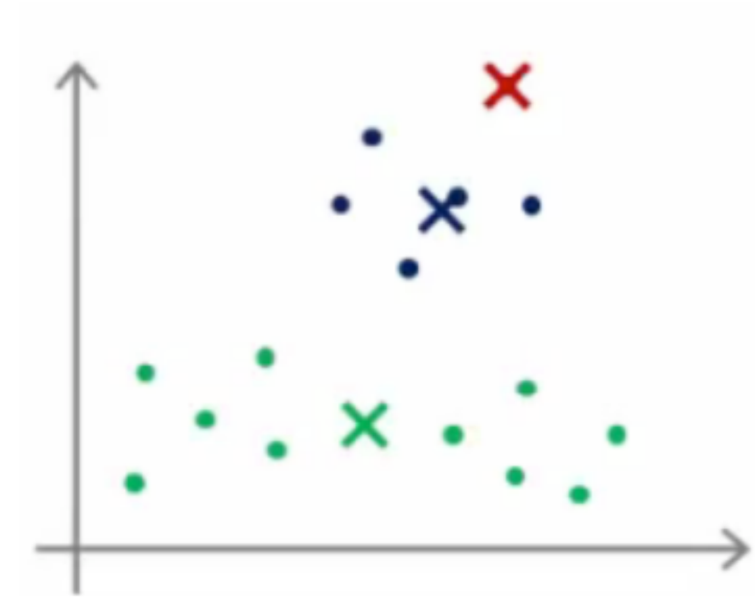
Clustering A



Clustering B

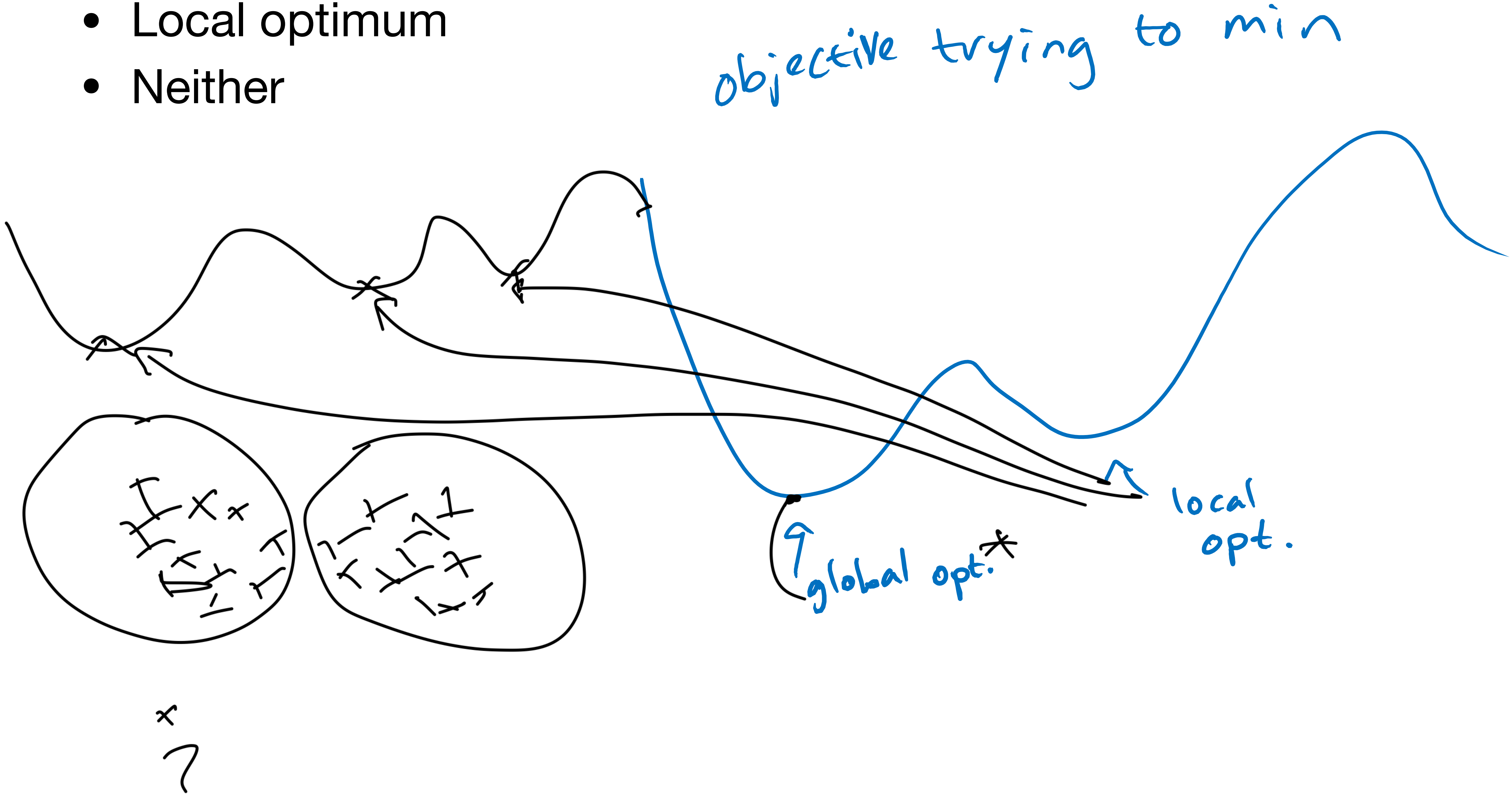


Clustering C



Convergence of k-means

- Global optimum
- Local optimum
- Neither

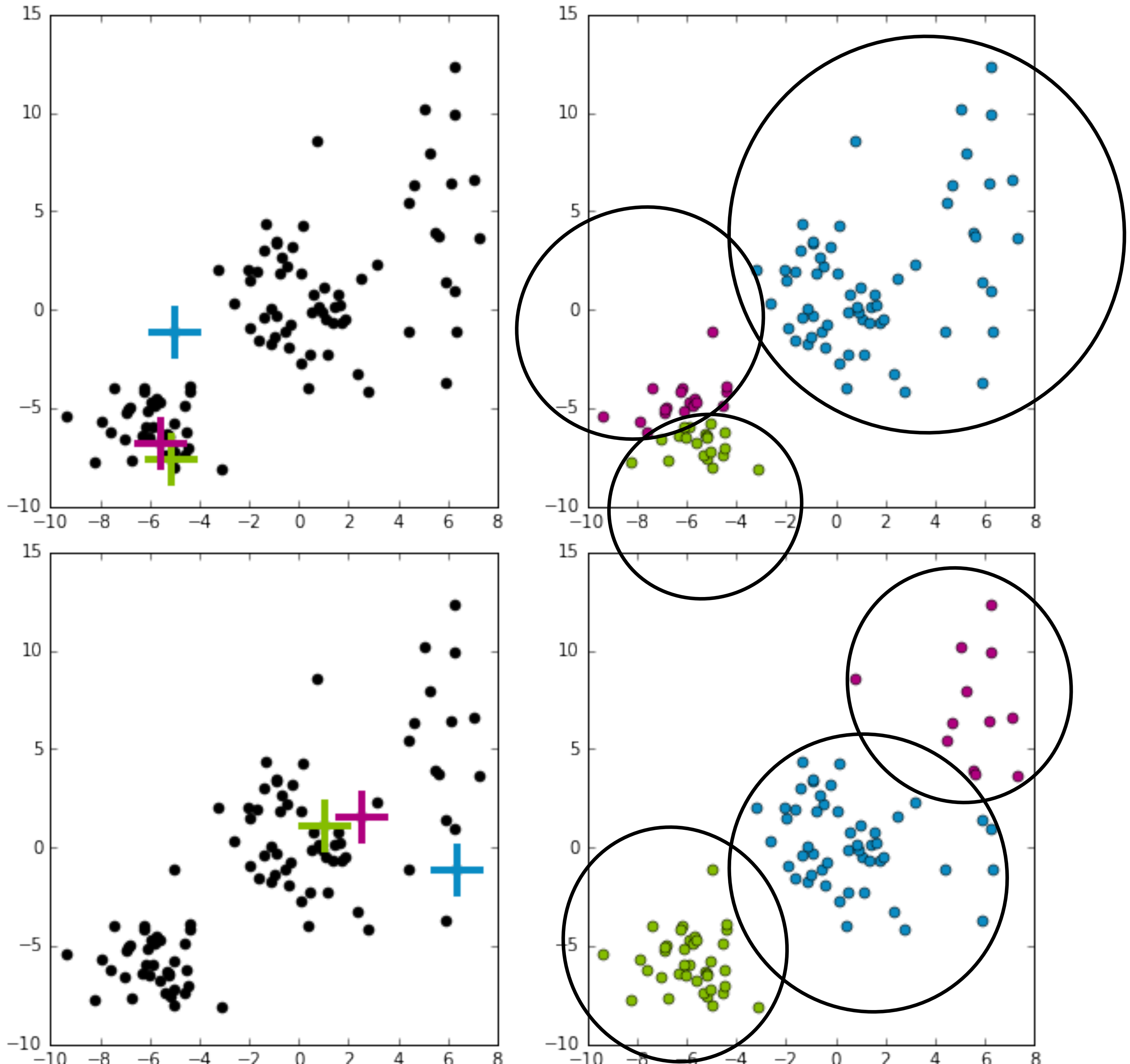


Where k-mean converges, depends on the initialization

Initial position of centers | final converged assignment

Trial 1

Trial 2



K-means ++
A smart initialization

k-means++

- Initialization of k-means algorithm is critical to quality of local optima found
- k-means++ proposes
  - Smart initialization
  - Followed by standard k-means algorithm

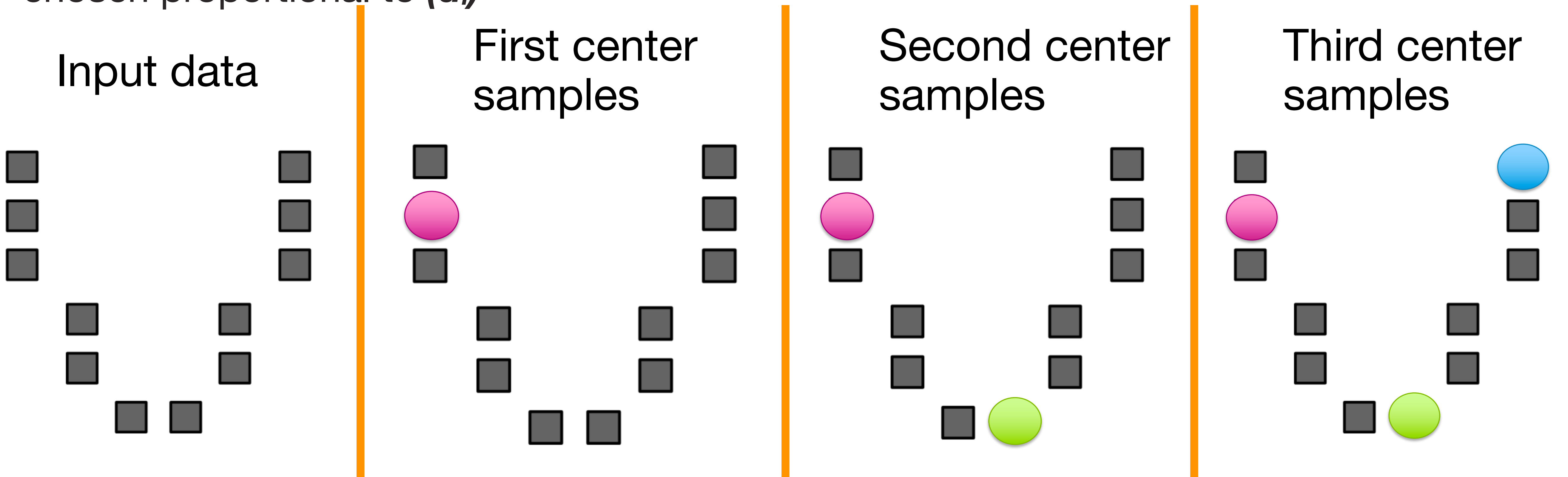
Smart initialization:

1. Choose first cluster center uniformly at random from data points

2. Repeat $\boldsymbol{k}$ times

   3. For each data point $\boldsymbol{x_i}$, compute distance $\boldsymbol{d_i}$ to nearest cluster center

   4. Choose new cluster center from amongst data points, with probability of $\boldsymbol{x_i}$ being chosen proportional to $\boldsymbol{(d_i)^2}$



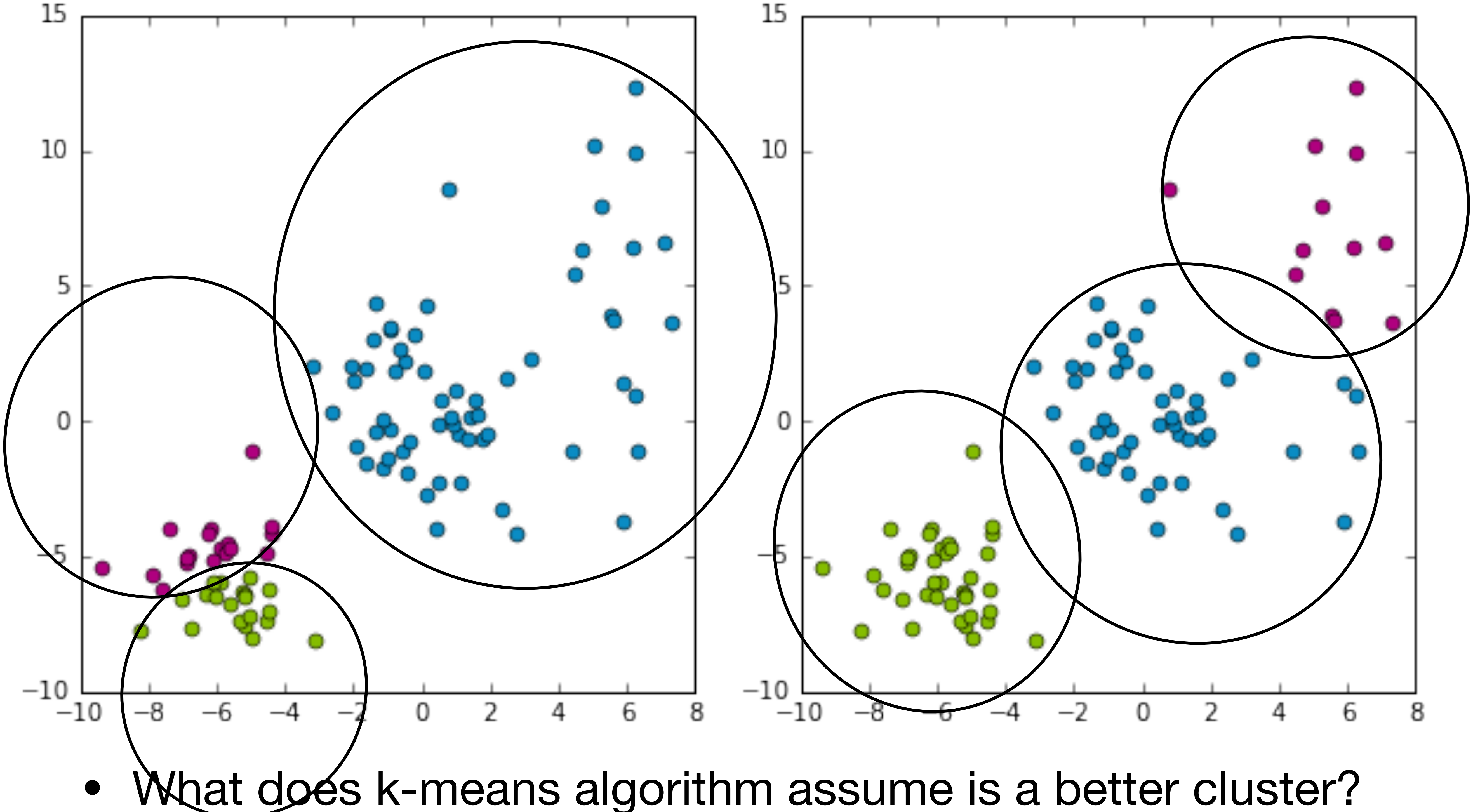
k-means++

- Compared to simple random initialization, where you pick ***k*** random data points as initial centers,
- smart initialization is computationally more costly
- But subsequent k-means algorithm converges faster

- overall, tends to find a better local optimum,
- And takes shorter time also

- insight about k-means++:
    - 1st step of randomly choosing on center tends to find one in the largest cluster, because there are more points
    - Subsequent sampling steps tend to find a center far from current centers

How do we measure which cluster is better?



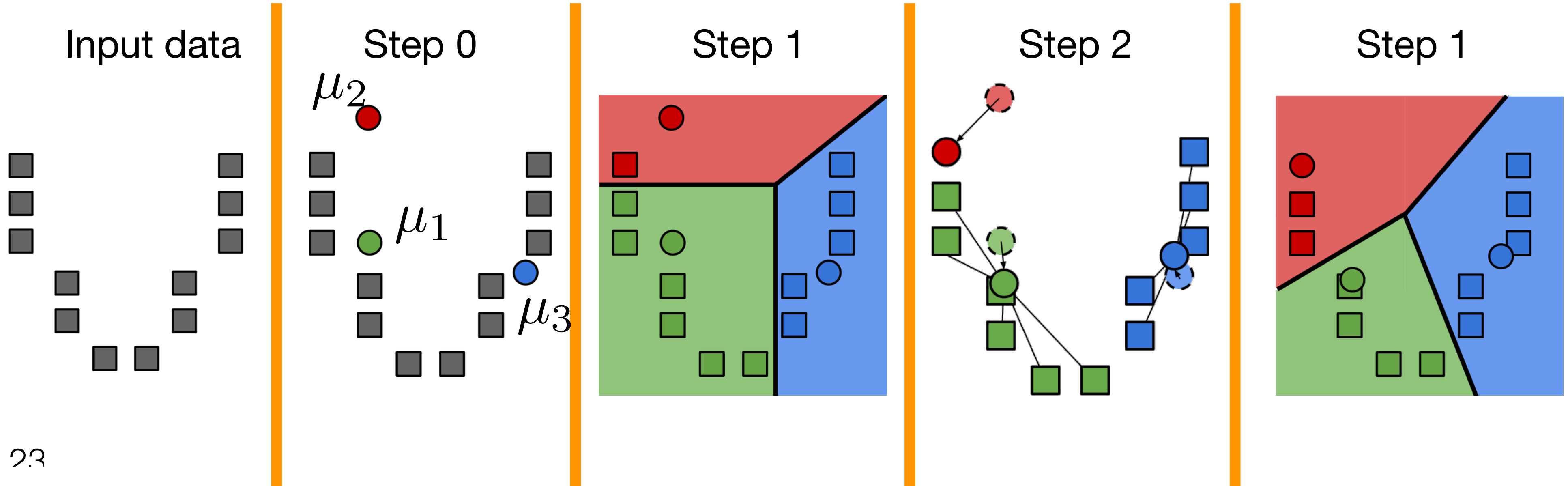
- What does k-means algorithm assume is a better cluster?
- k-means is one way of minimizing $$\sum_{j=1}^{k} \sum_{i:z_i=j} ||\mu_j - \mathbf{x}_i||_2^2$$

which is how much you pay for **heterogeneity**

K-means as coordinate descent

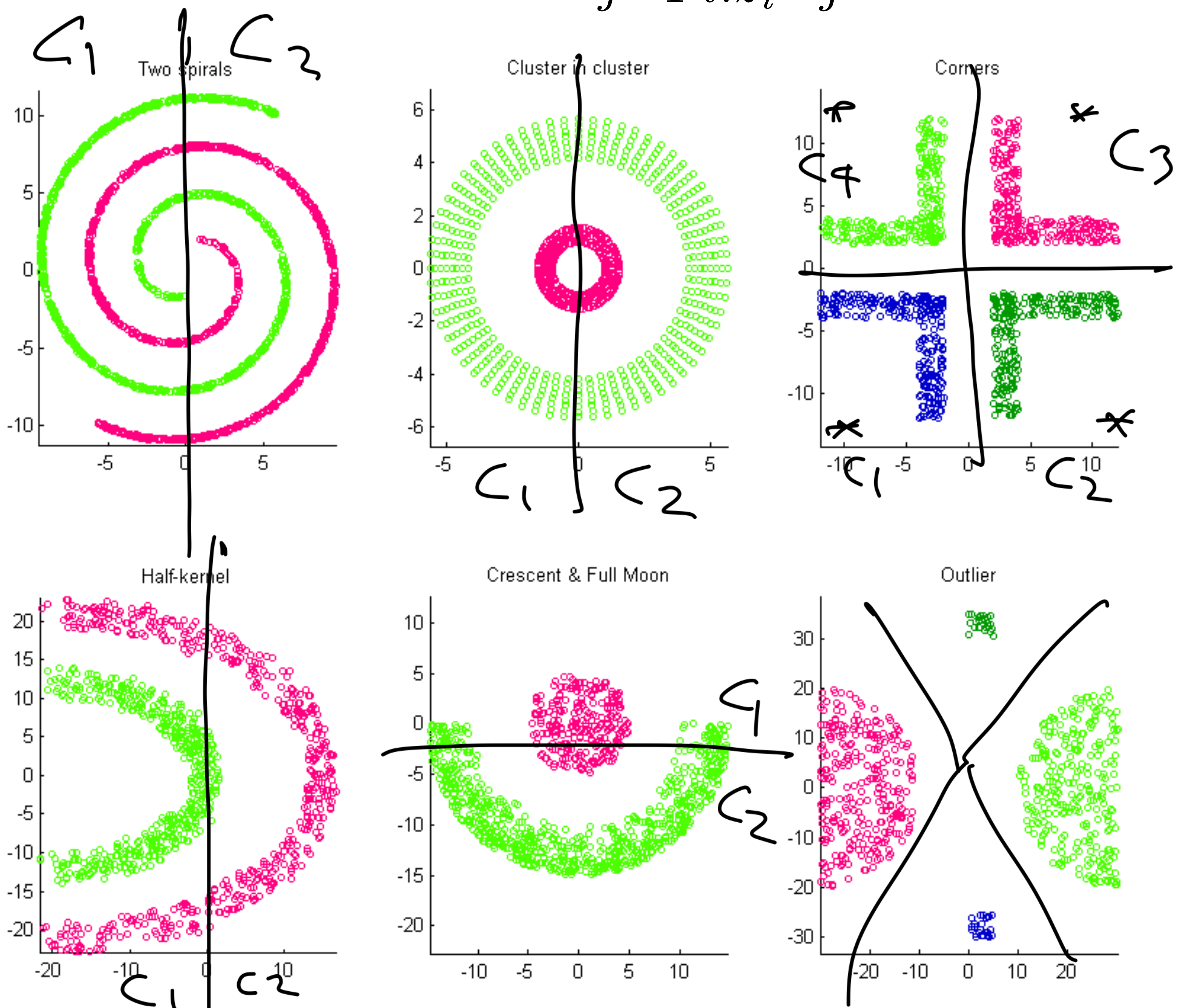
$$\min_{\mu_1,\ldots,\mu_k,z_1,\ldots,z_N} \sum_{j=1}^{k} \sum_{i:z_i=j} \left\|\mu_j - x_i\right\|_2^2$$

- k-means
  - Start with random initialization of the centers (chosen from the data points)
  - Repeat
    - Fix centers and find optimal assignments (z_i's)
    - Fix assignments and find optimal centers (μ_j's)
- Note that we make the objective **strictly** smaller every step
- The algorithm converges in finite time



Is this the best measure of clustering error?

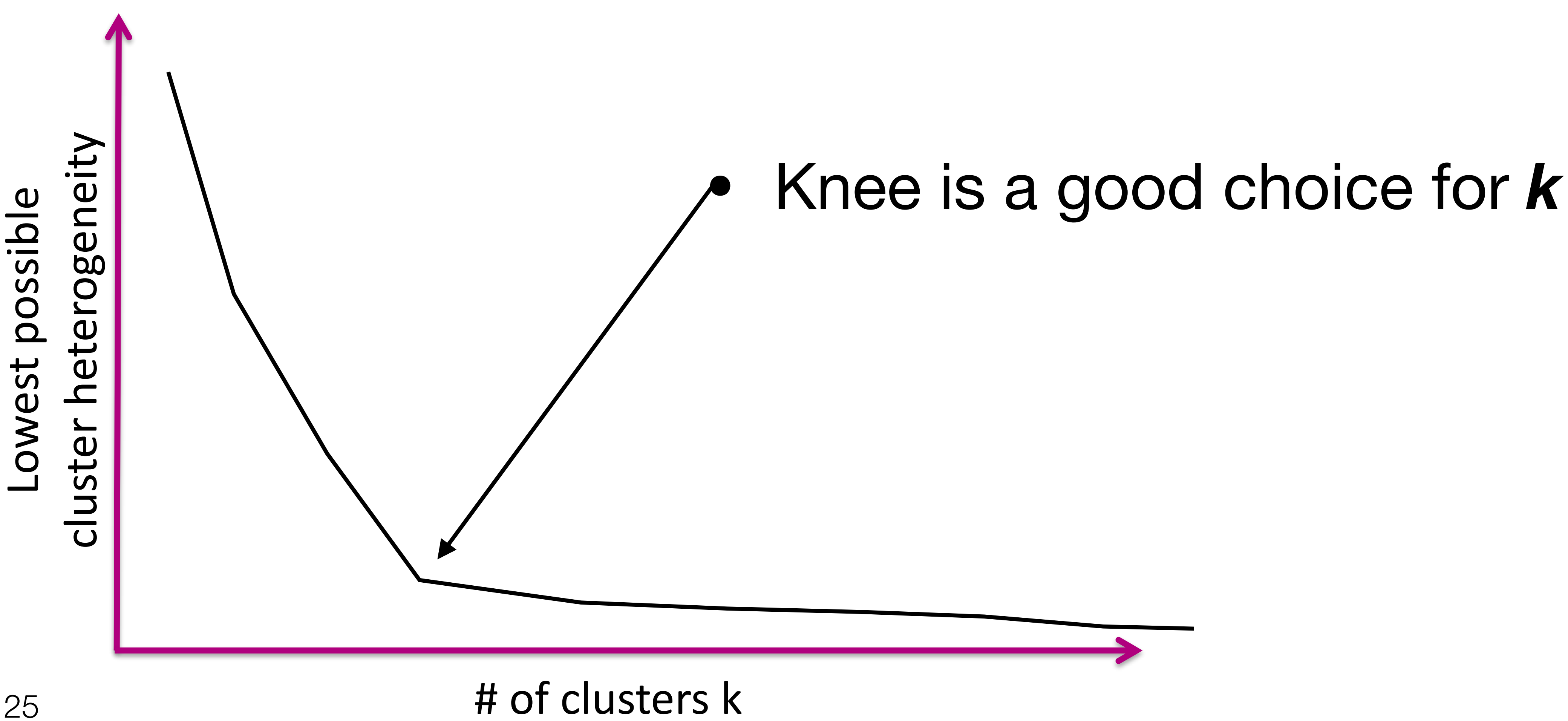
$$\min_{\mu_1,\ldots,\mu_k,z_1,\ldots,z_N} \sum_{j=1}^{k} \sum_{i:z_i=j} \|\mu_j - x_i\|_2^2$$



What *k* should we use?

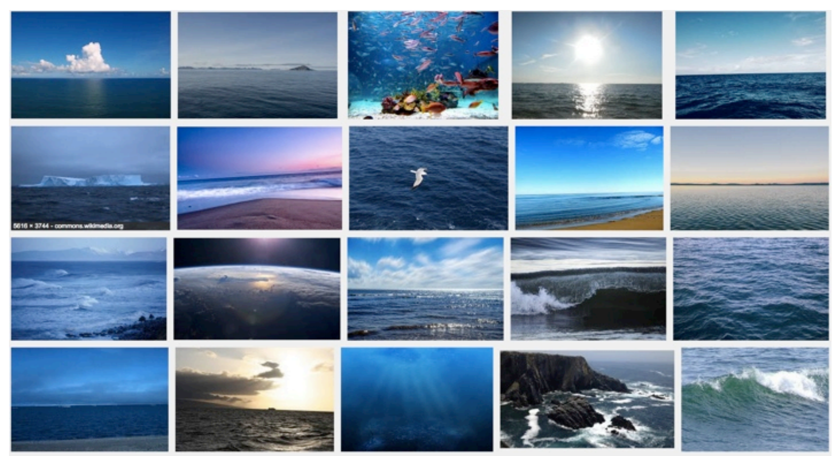
$$\sum_{j=1}^{k} \sum_{i \in C_j} \|\mu_j - x_i\|^2$$

- Increasing ***k*** eventually overfits.
- One extreme, when ***k*=*N***
  - Each data point is its own cluster
  - Heterogeneity is zero, and we get the best score under k-means



Real world examples

- For search, group as:
  - Ocean
  - Pink flower
  - Dog
  - Sunset
  - Clouds
  - …

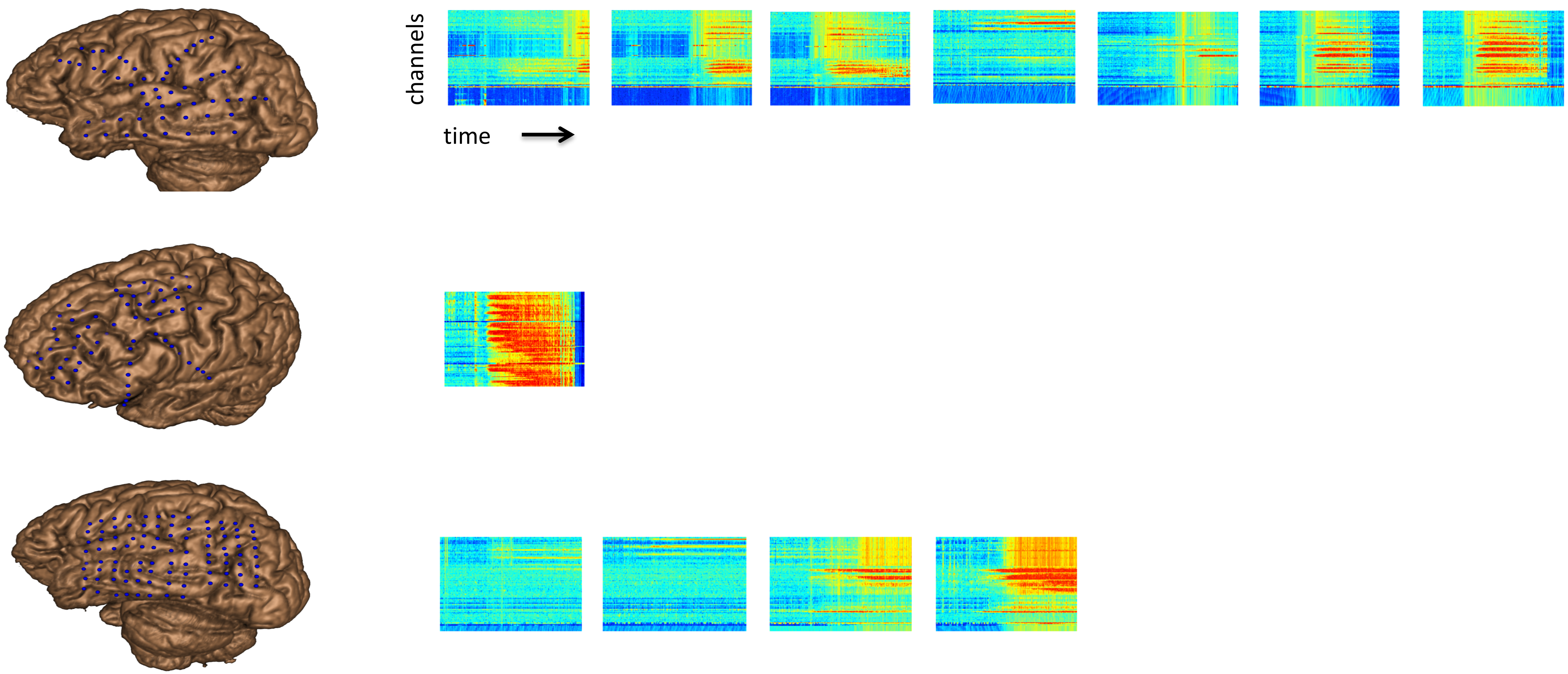


Structuring web search results

- Search terms can have multiple meanings
- Example: "cardinal"

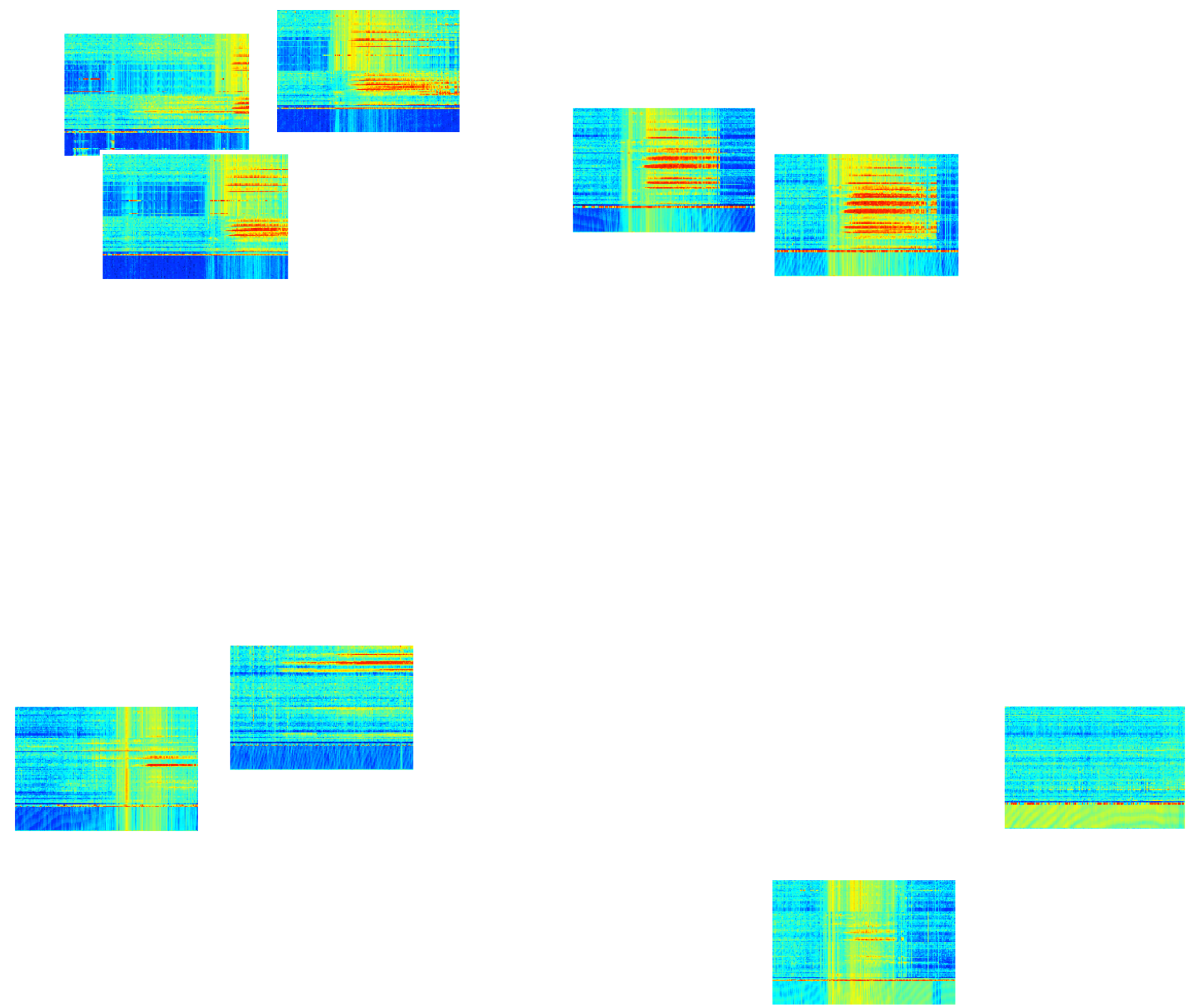
- Use clustering to structure output

- You can use it to partition patients based on medical condition, to be used in more targeted studies
- Combinations of patients and seizures are diverse



- The electrode placement is unique in each patient
- Each patient has a different number of seizures that themselves often display quite different dynamics within each seizure
- the thumbprint of each seizure with a colored box shows how a particular feature changes in each channel over the course of the seizure.

- We can place these observed signal in lower dimensional space according to their clusters, which provides important visualization and insights that can be used following clinical decisions and studies



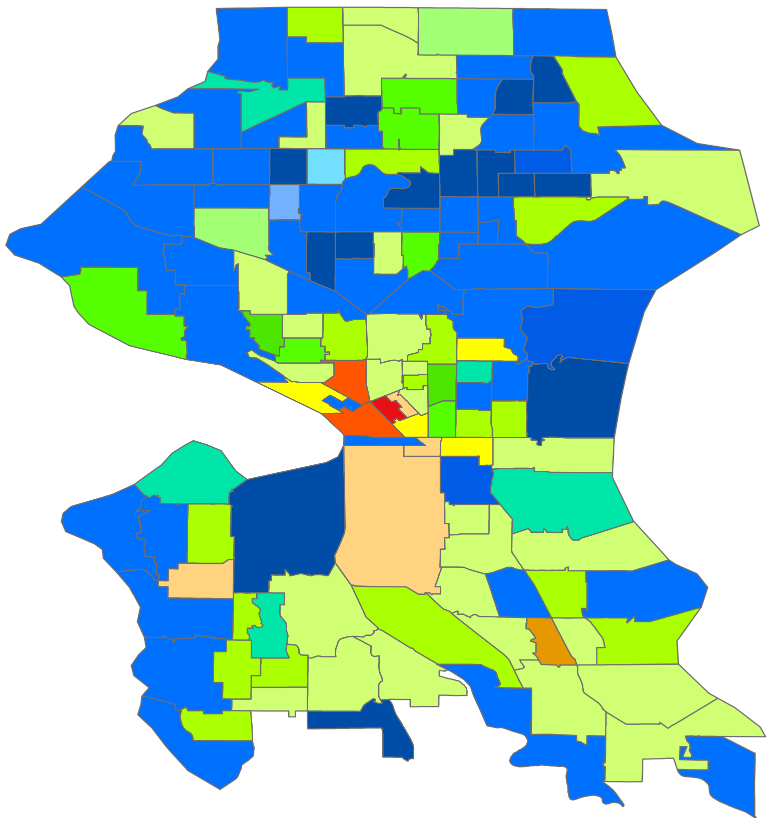
Amazon

- Discover product categories from purchase histories

- Or discovering groups of users

Discover similar neighborhoods

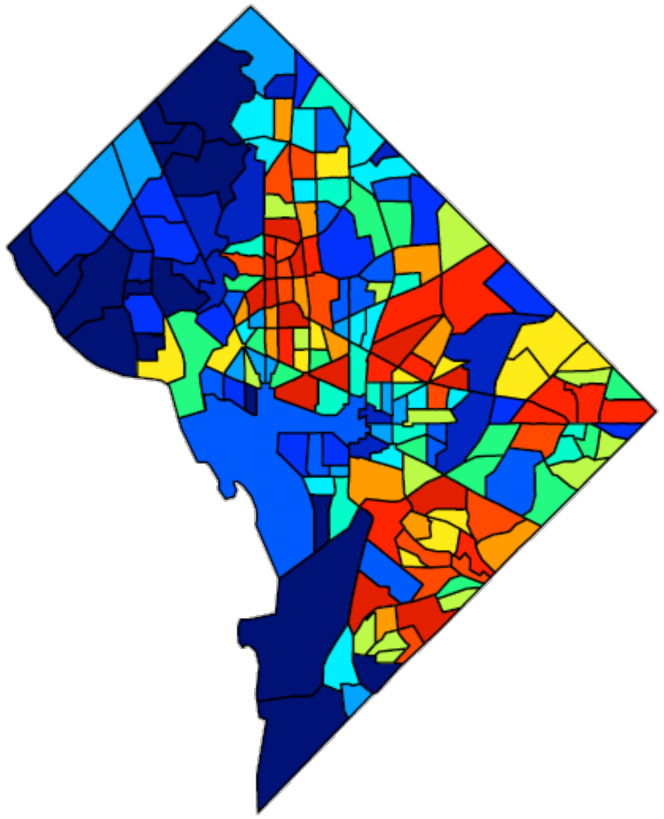
- Task 1: Estimate price at a small regional level
- Challenge:
  - Only a few (or no!) sales in each region per month
- Solution:
  - Cluster regions with similar trends and share information within a cluster



City of Seattle

Discover similar neighborhoods

- Task 2: Forecast violent crimes to better task police
- Again, cluster regions and share information!
- Leads to improved predictions compared to examining each region independently



Washington, DC

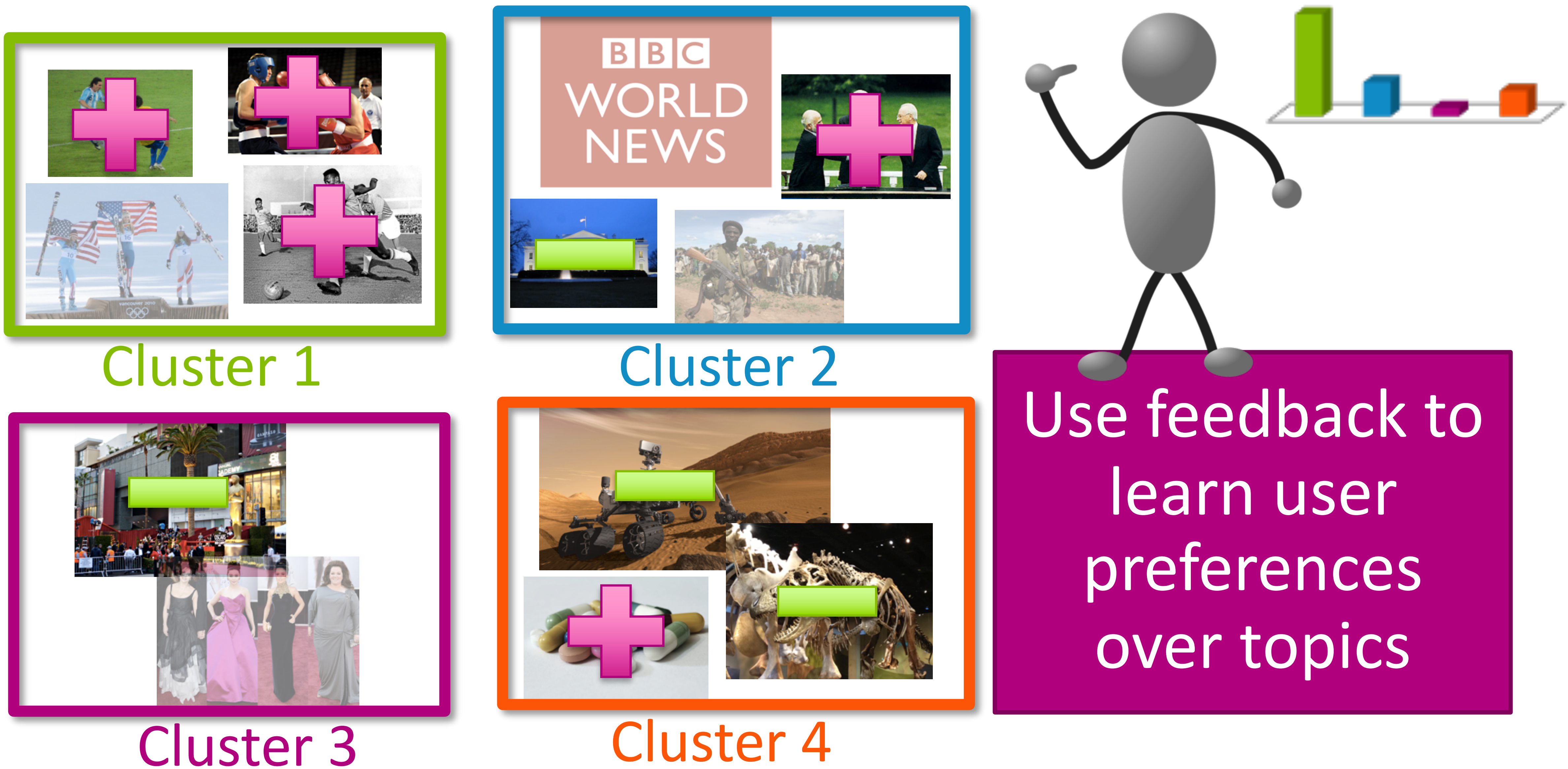
K-means explained visually

https://www.naftaliharris.com/blog/visualizing-k-means-clustering/

Limitations and failure modes of k-means

Learning user preferences

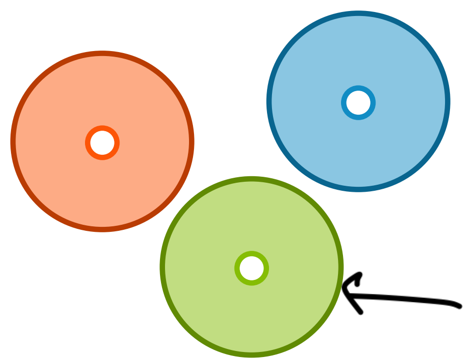
Set of clustered documents read by user



- In reality, articles are not about just one topic
- HARD clustering misses nuanced soft membership

Shapes of the clusters

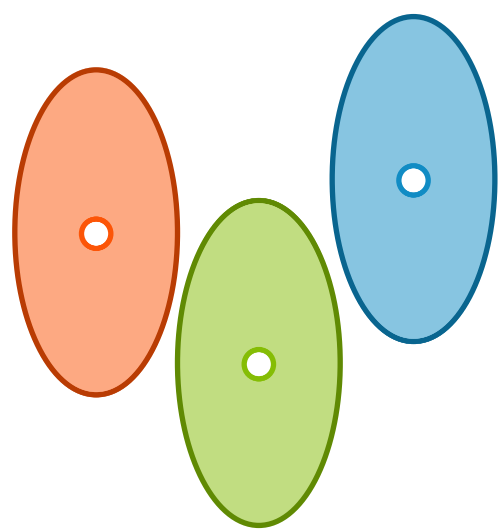
- K-means algorithm is essentially fitting or assuming spherically symmetric clusters
because we use Euclidean distance, and
all points at the same Euclidean distance are paying the same cost

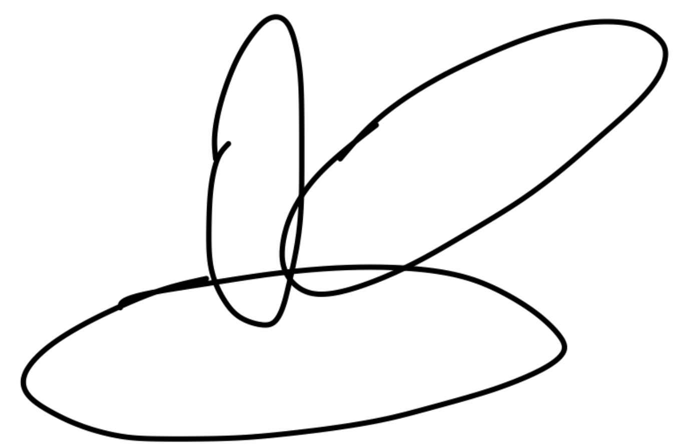


$$z_i \leftarrow \arg\min_j ||\mu_j - \mathbf{x}_i||_2^2$$

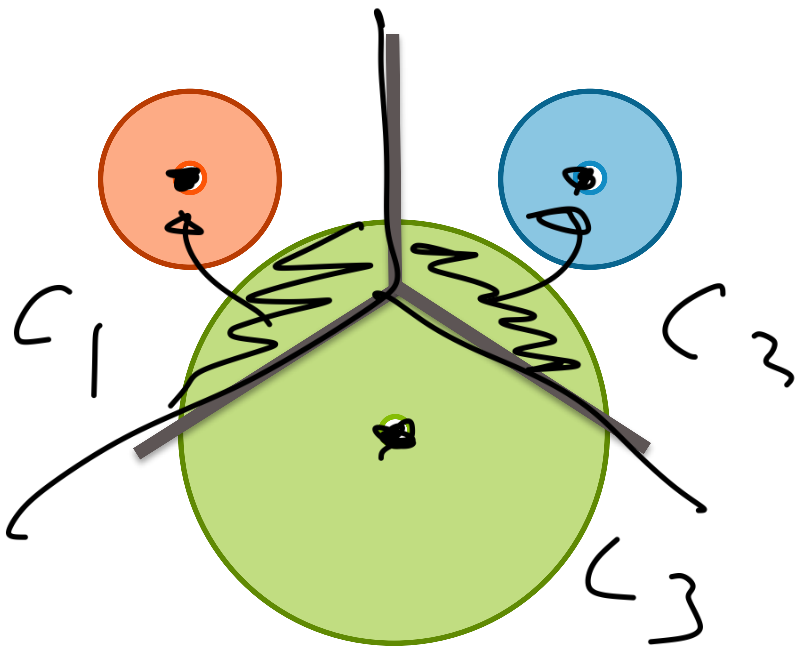
Points at same Euclidean dist.

- How can we resolver this? Use wighted Euclidean distance

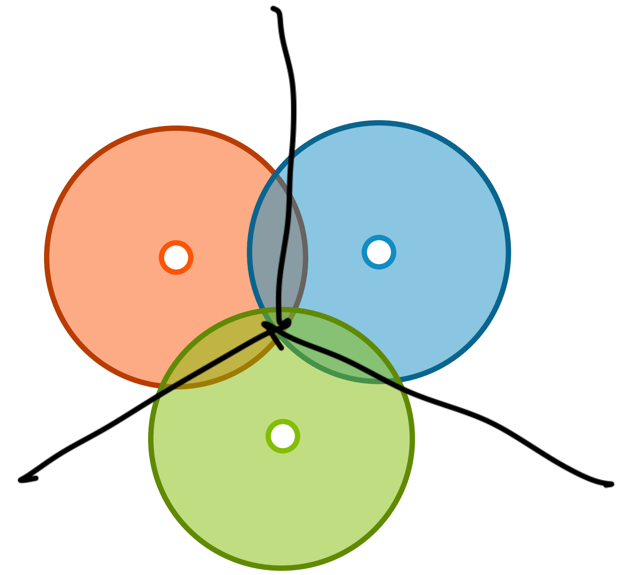




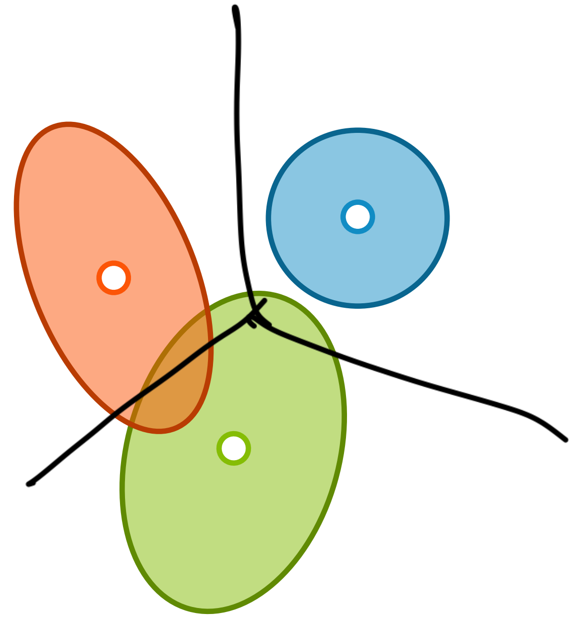
Typical failure modes



disparate cluster sizes



overlapping clusters

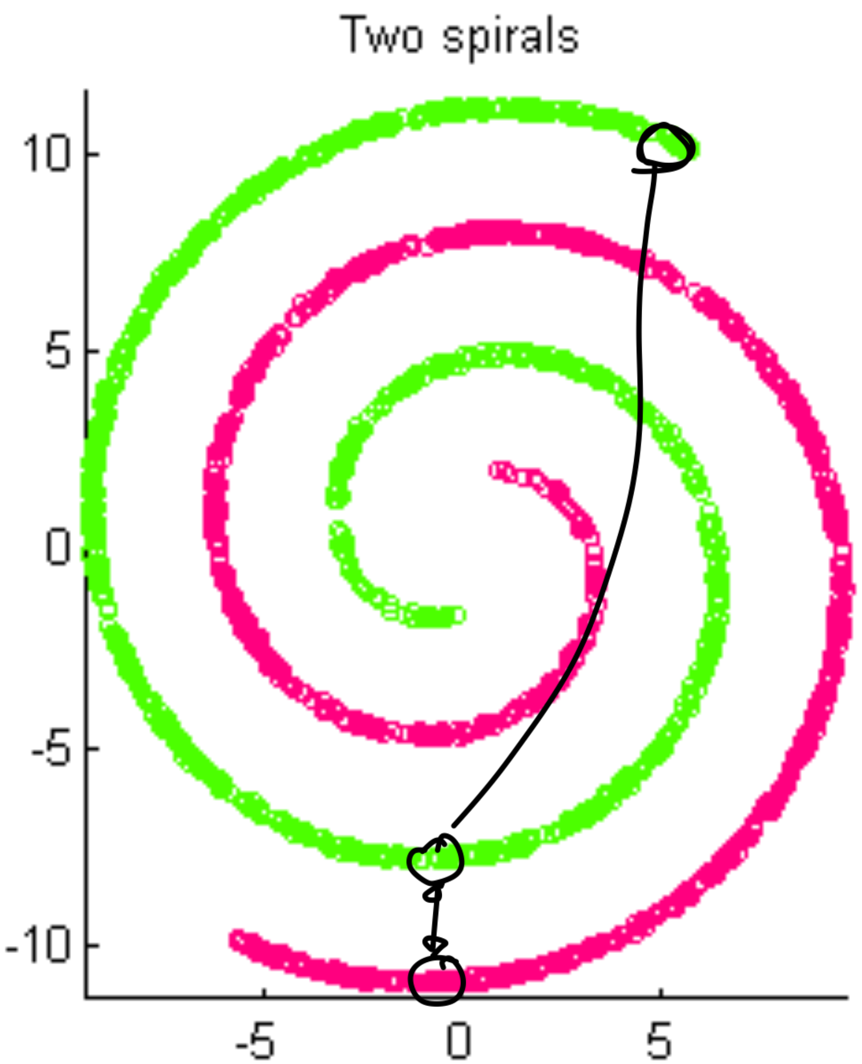


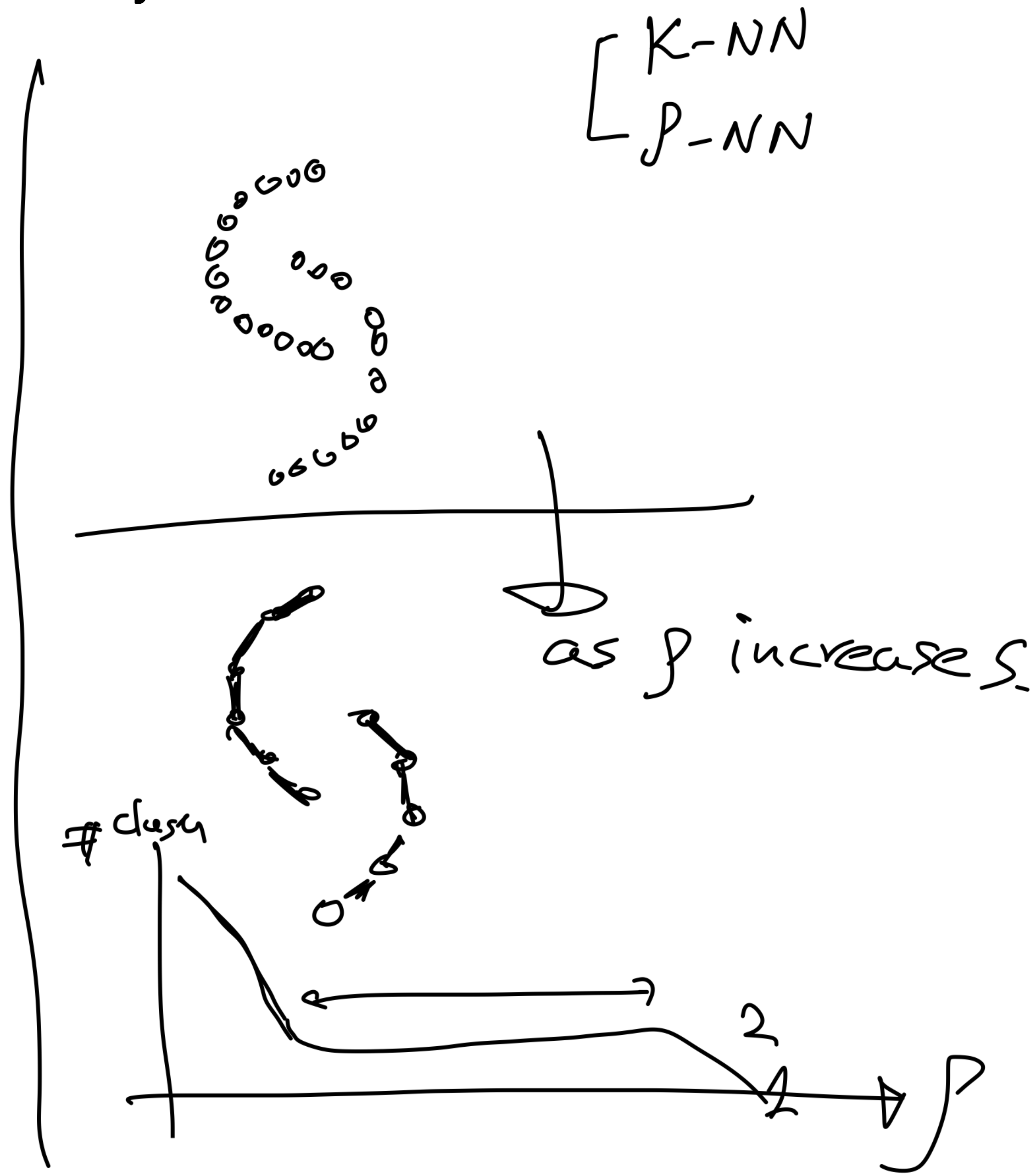
different shaped/oriented clusters

- Provides soft assignments of observations to clusters (uncertainty in assignment)
  - e.g., 54% chance document is world news, 45% science, 1% sports, and 0% entertainment
- Accounts for cluster shapes not just centers
- Enables learning weightings of dimensions
  - e.g., how much to weight each word in the vocabulary when computing cluster assignment

Diffusion maps

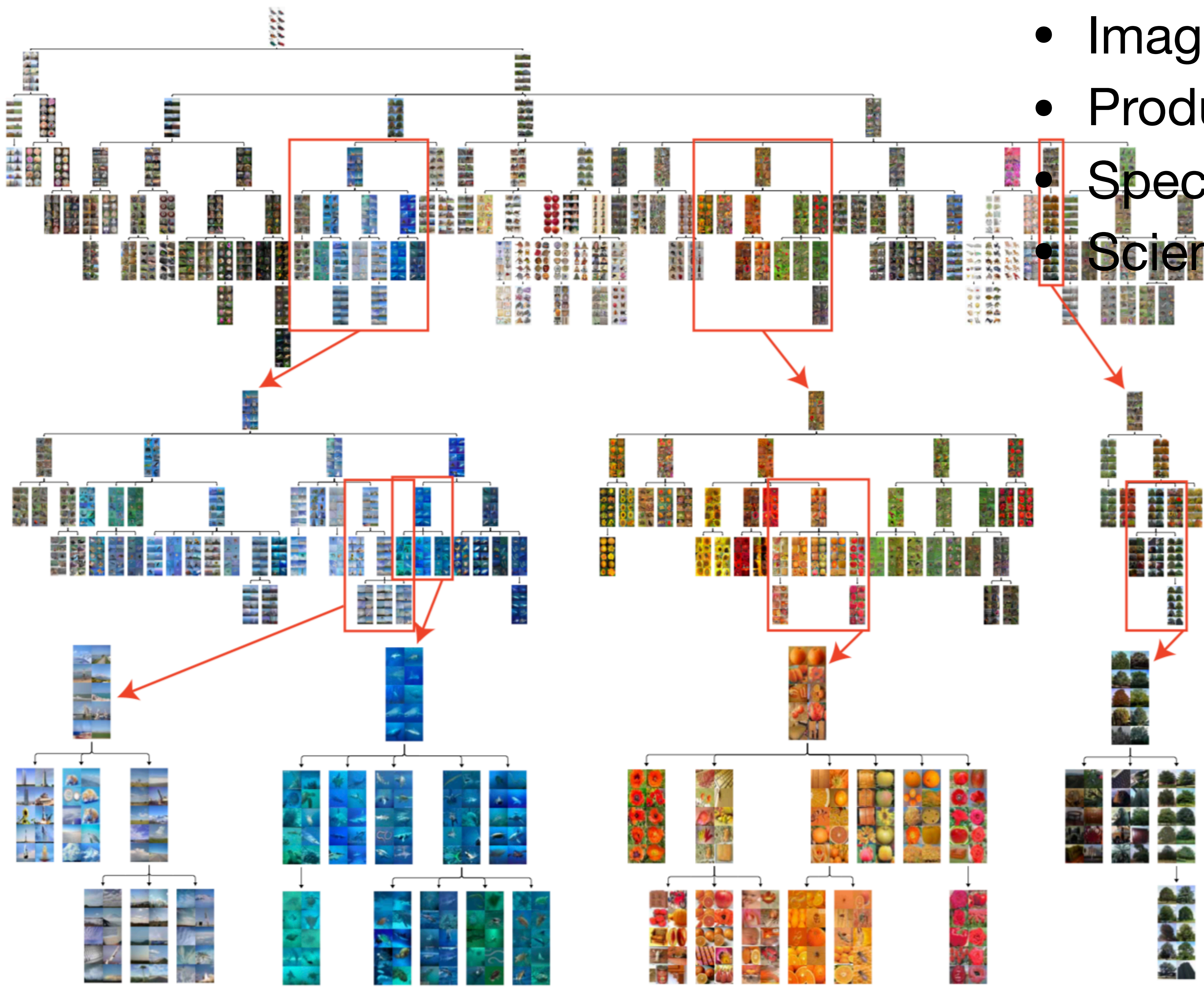
- Non-linear dimensionality reduction
-





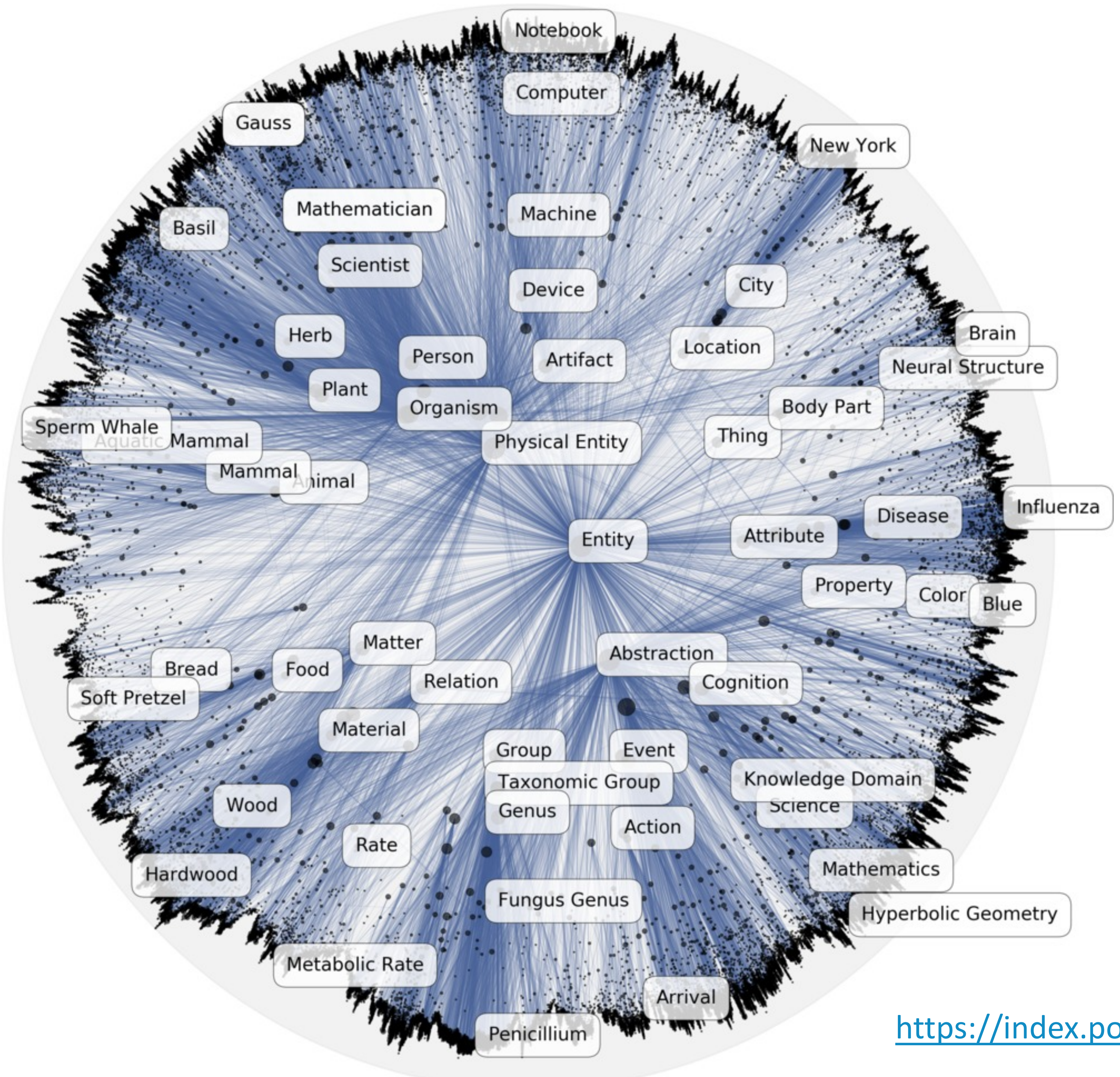
Hierarchical clustering

Lots of data are hierarchical in nature



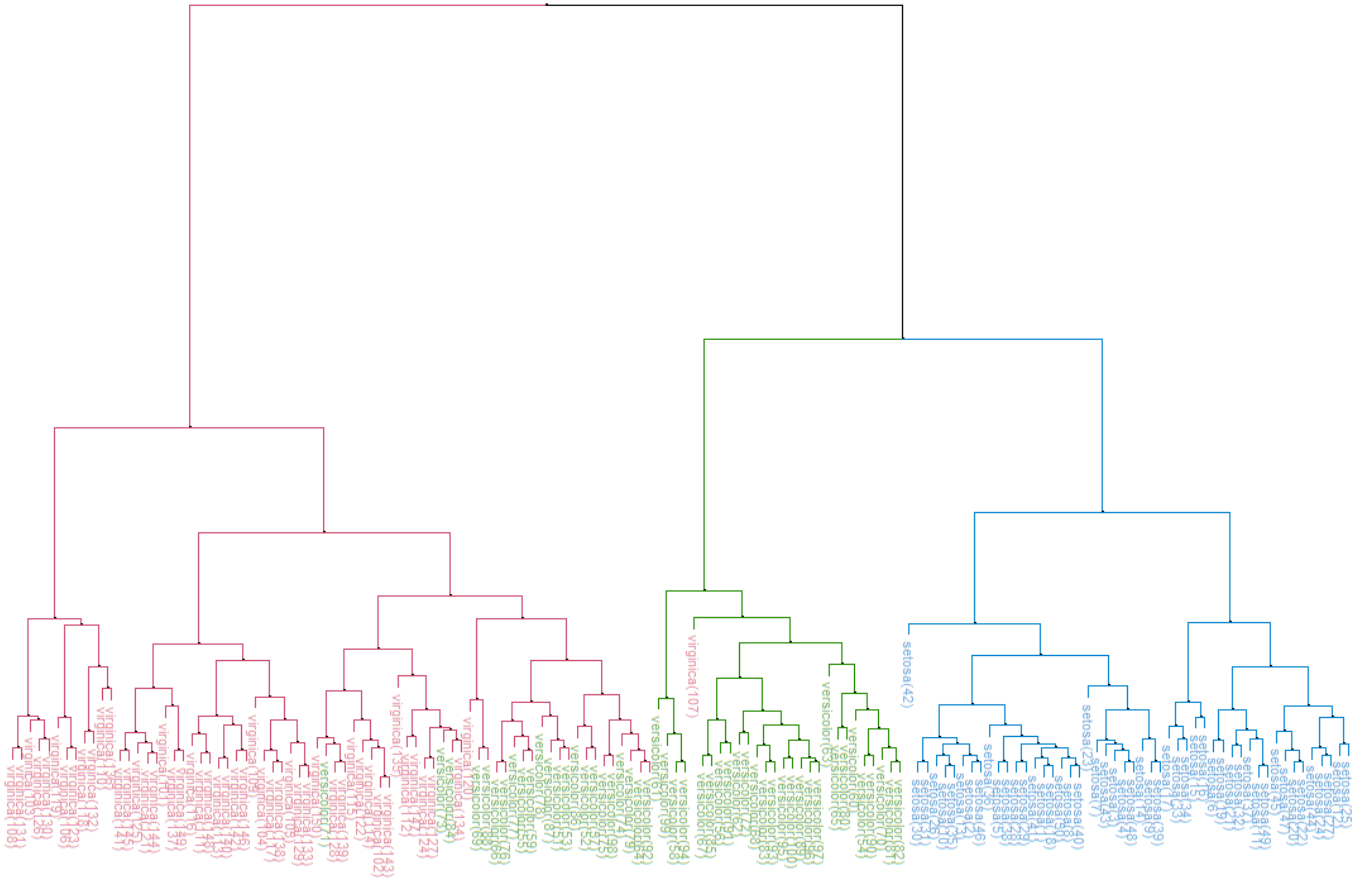
- Image types
- Product categories
- Species
- Scientific concepts

- Nouns



https://index.pocketcluster.io/facebookresearch-poincare-embeddin

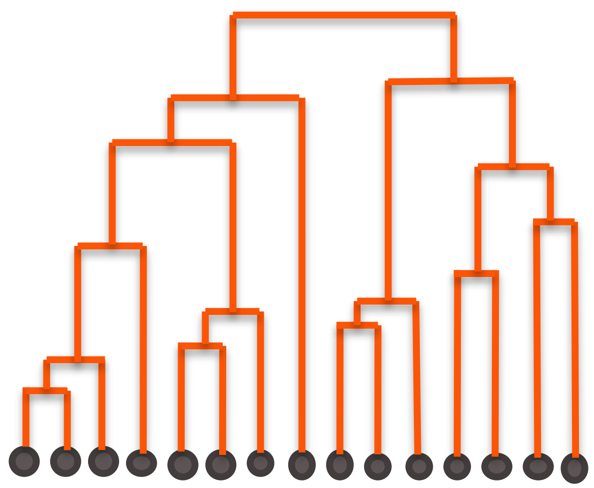
- Species



Other motivations for hierarchical clustering

- Avoid choosing # clusters beforehand

- Dendrograms help visualize different clustering granularities
  - No need to rerun algorithm

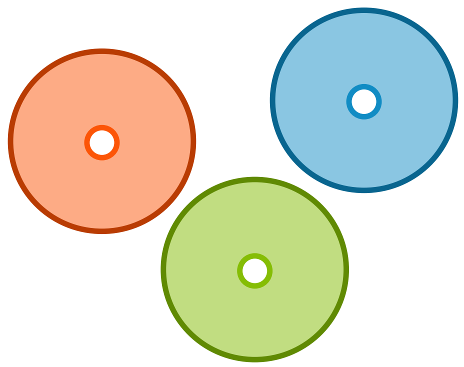


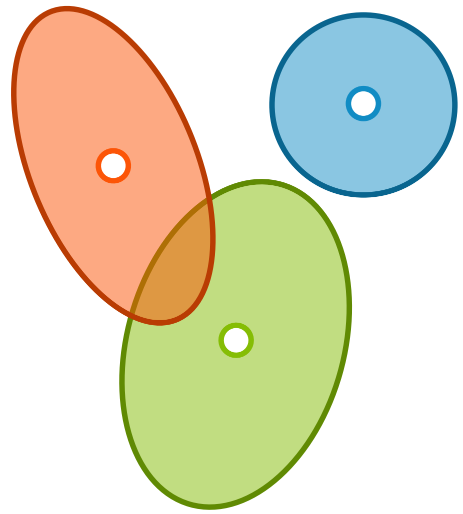
- Most algorithms allow user to choose any distance metric
  - k-means restricted us to Euclidean distance

Can often find more complex shapes than k-means or Gaussian mixture models

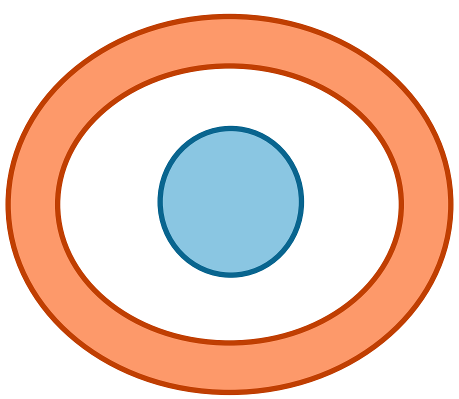
Gaussian mixtures: ellipsoids

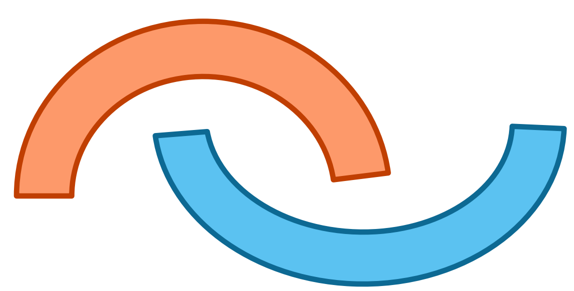
k-means: spherical clusters





What about these?





Two-types of approaches

Divisive, *a.k.a. top-down:* Start with all data in one big cluster and recursively split.

- Example: recursive k-means

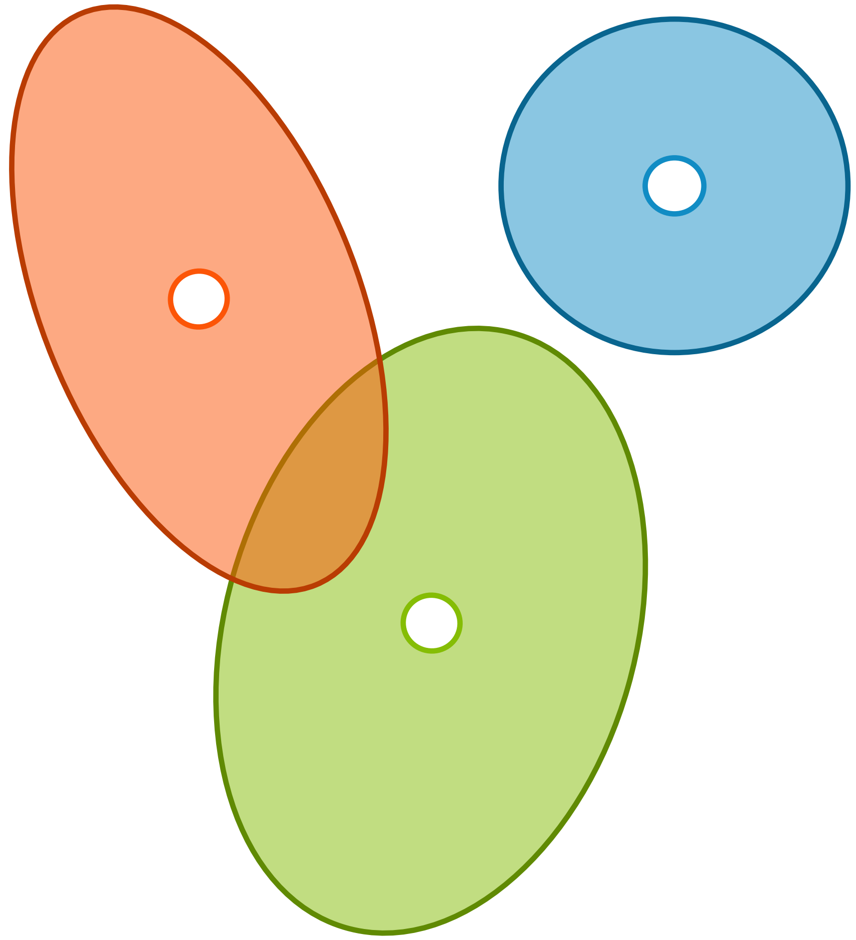
Agglomerative *a.k.a. bottom-up*: Start with each data point as its own cluster. Merge clusters until all points are in one big cluster.

- Example: single linkage

Divisive clustering

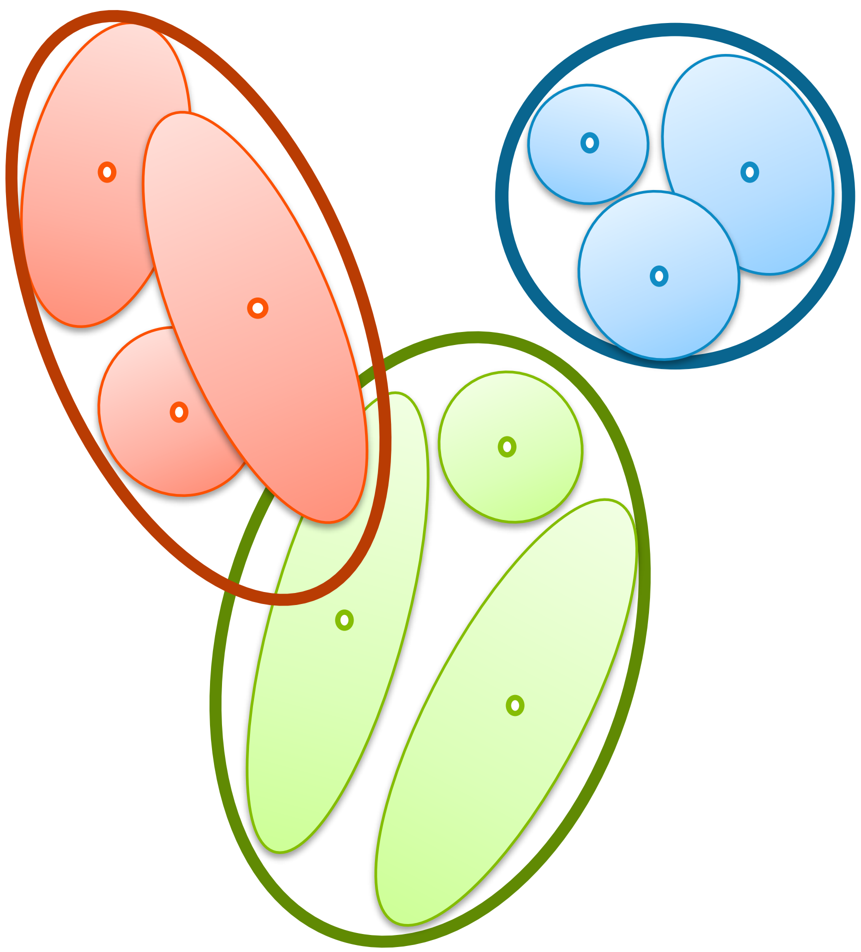
Divisive in pictures – level 1

- Cluster all the data into, say, 3 clusters first



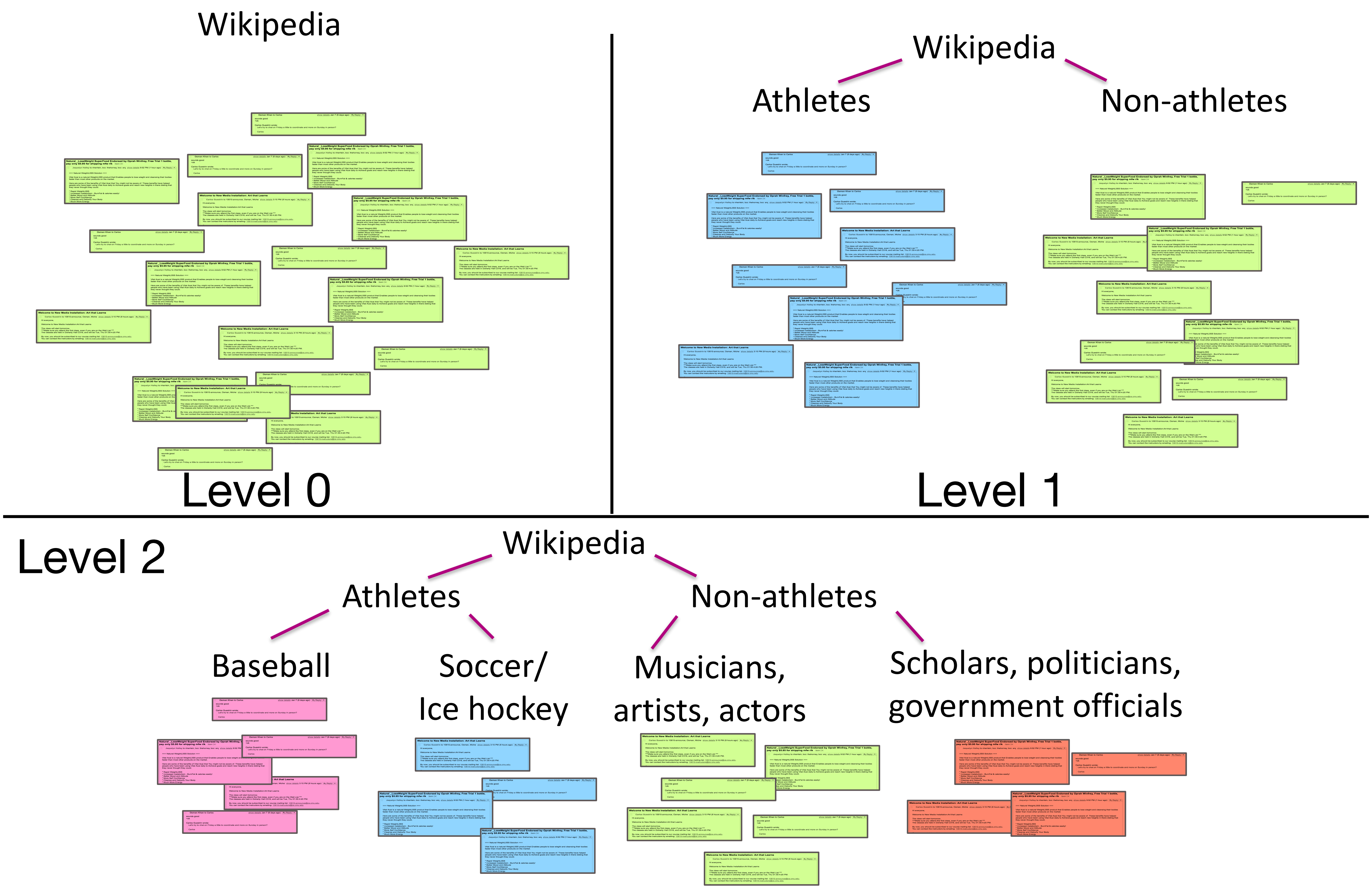
Divisive in pictures – level 2

- For data in each cluster, run a new clustering algorithm of choice



Divisive: Recursive k-means

- For example, we could run k-means, recursively



Divisive choices to be made

- Which algorithm to recurse

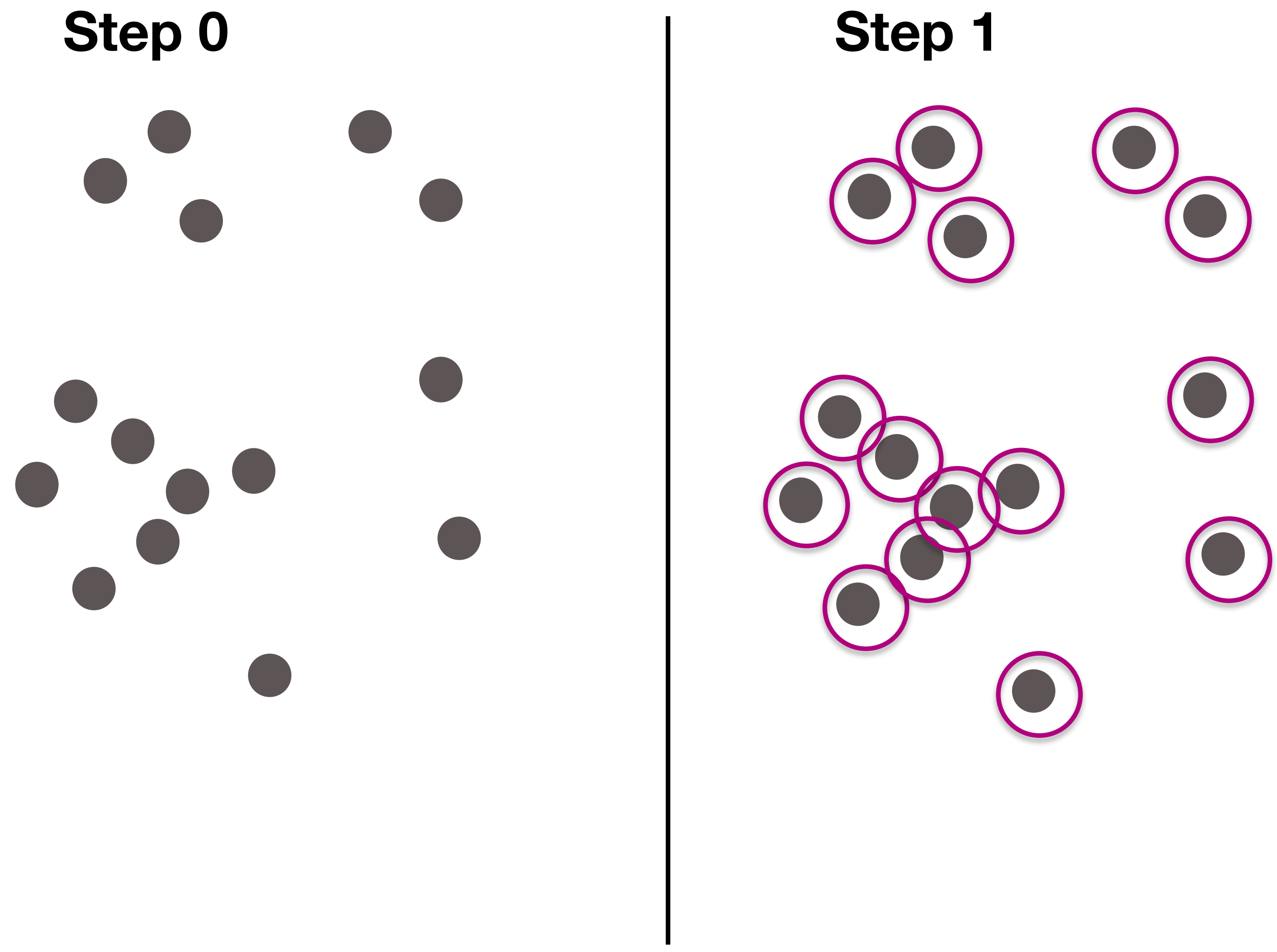
- How many clusters per split

- When to split vs. stop
  - Max cluster size:
  number of points in cluster falls below threshold
  - Max cluster radius:
  distance to furthest point falls below threshold
  - Specified # clusters:
  split until pre-specified # clusters is reached

Agglomerative clustering

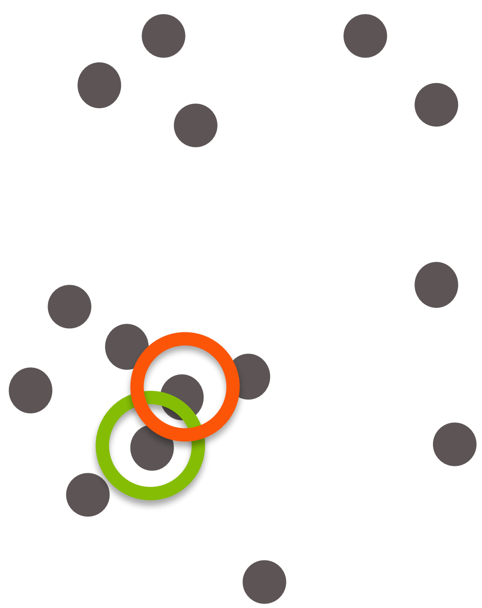
Agglomerative: Single linkage

1. Initialize each point to be its own cluster



Agglomerative: Single linkage

2. Define distance between clusters to be:



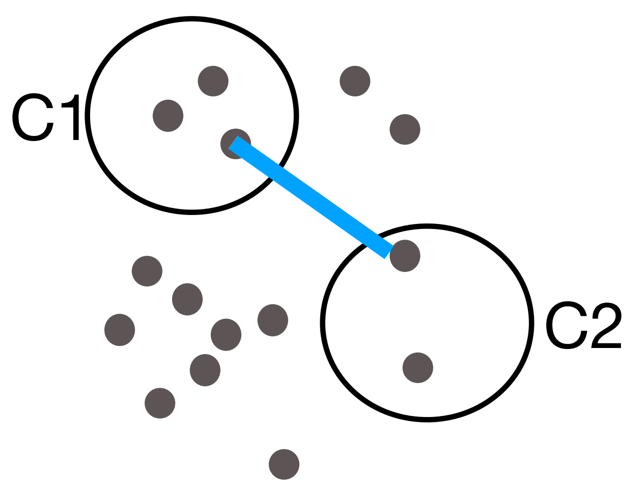
$$\text{distance}(C_1, C_2) = \min \; d(x_i, x_j)$$

Linkage criteria

specified pairwise distance function

and closest pair of clusters are merged, recursively

Single linkage means that
we use the shortest single link (or edge) between two clusters
to measure the distance between clusters

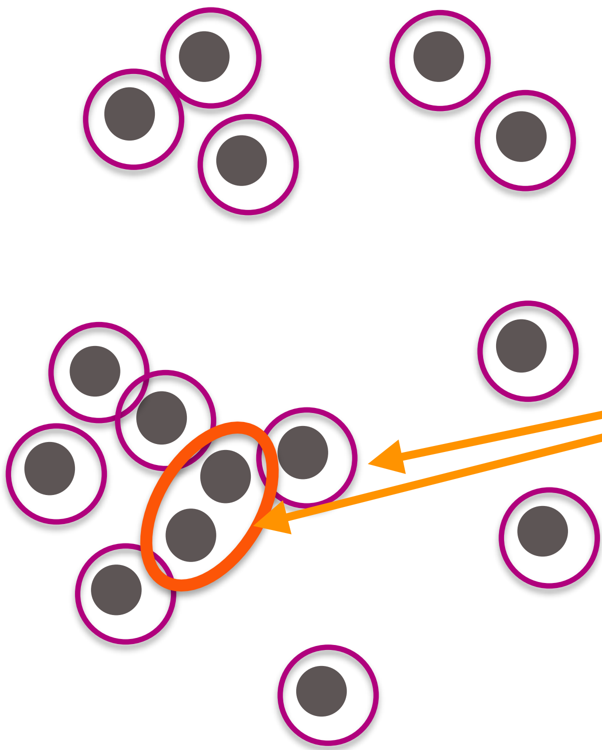


That is, the distance between C1 and C2
is the length of the two points that are closest
each coming from one of the clusters (blue line)

There are other ways to measure distance between two clusters,
which give different properties of the resulting hierarchical clusters

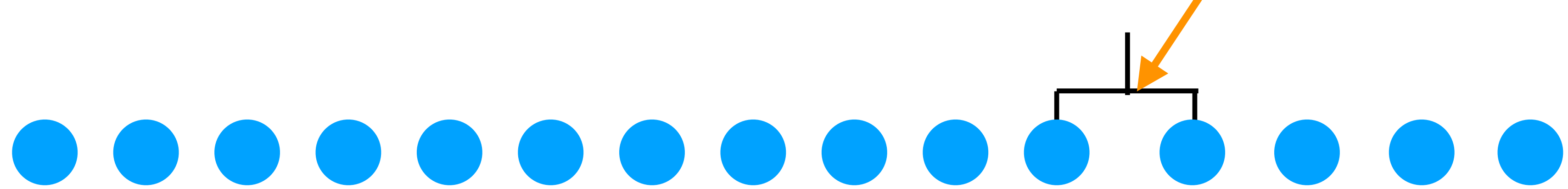
Agglomerative: Single linkage

3. Merge the two closest clusters



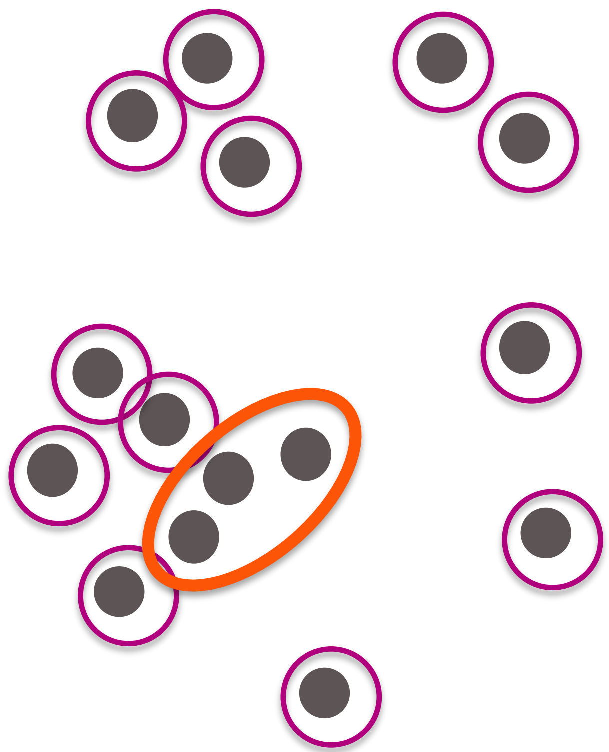
Next closest two clusters are merged next

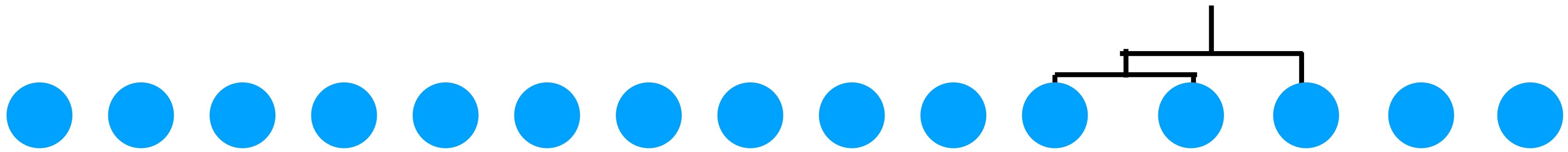
We can tract this process with a growing **dendrogram** like the one below
Each blue node corresponds to a data point,
When merge happens two clusters are joined by a branch,
The hight of the branch is the distance between those two clusters



Agglomerative: Single linkage

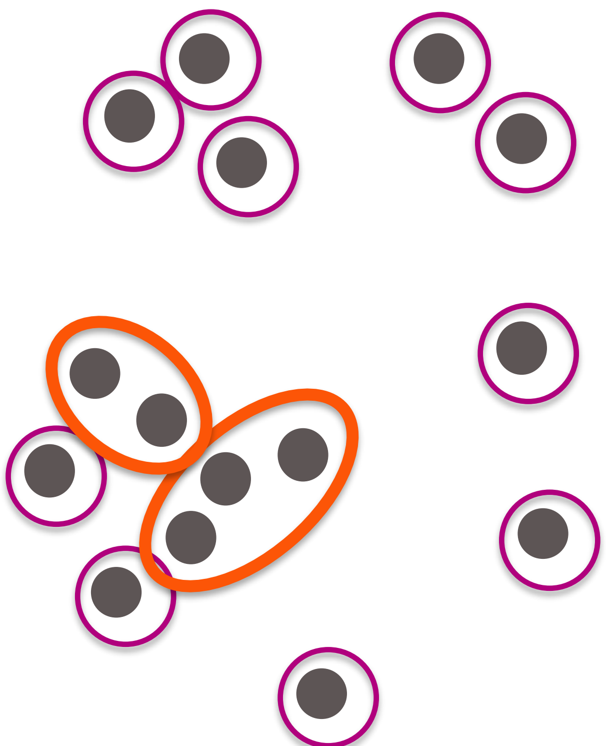
4. Repeat step 3 until all points are in one cluster





Agglomerative: Single linkage

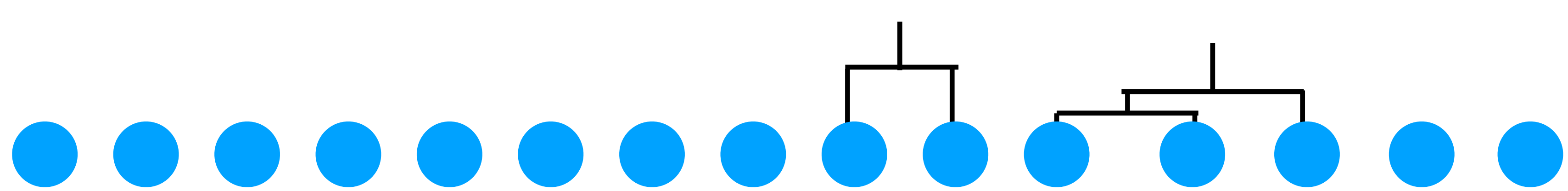
4. Repeat step 3 until all points are in one cluster



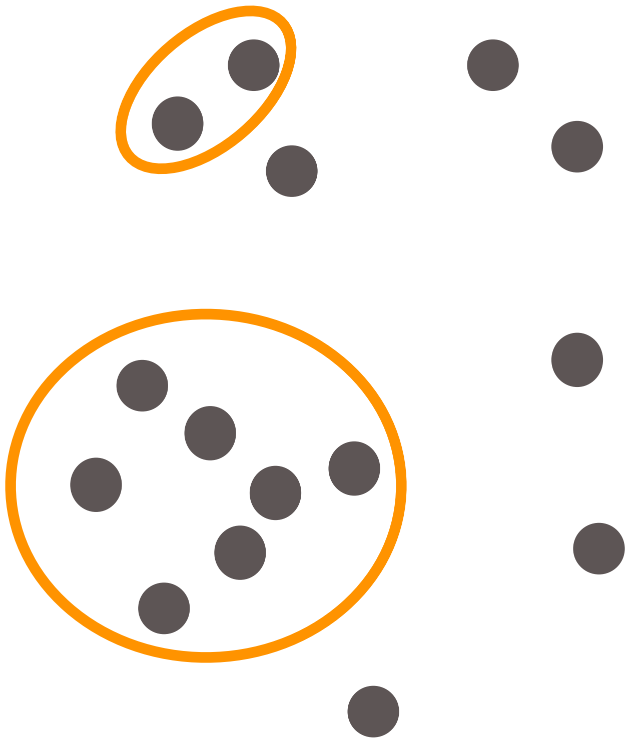
Notice how the heights of the branching (or merging) point is increasing,

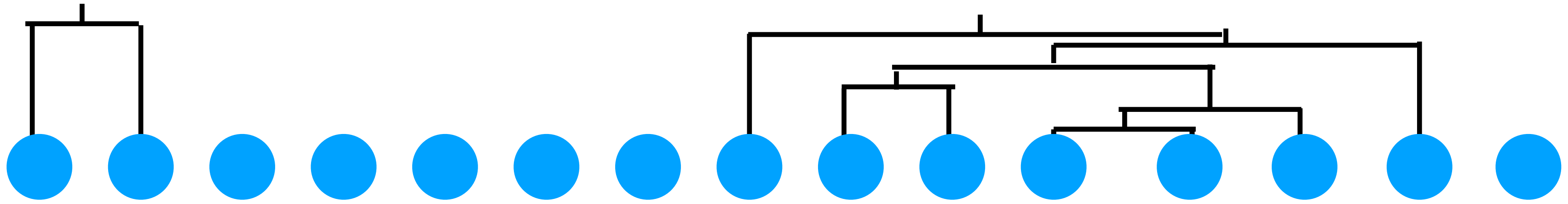
That is because the clusters that are merged later are
the ones that are further apart

If they were closer, than those two nodes in the “single linkage”
would have been merged earlier



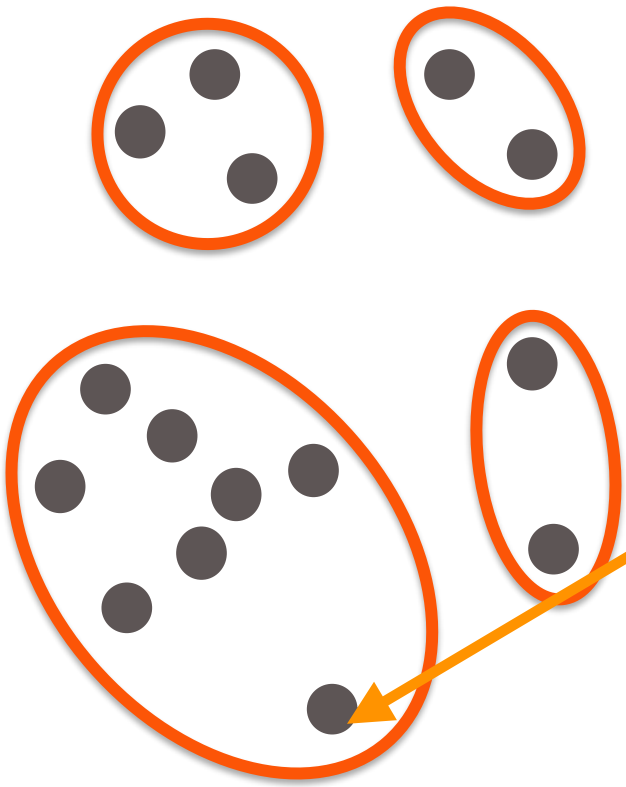
Agglomerative: Single linkage





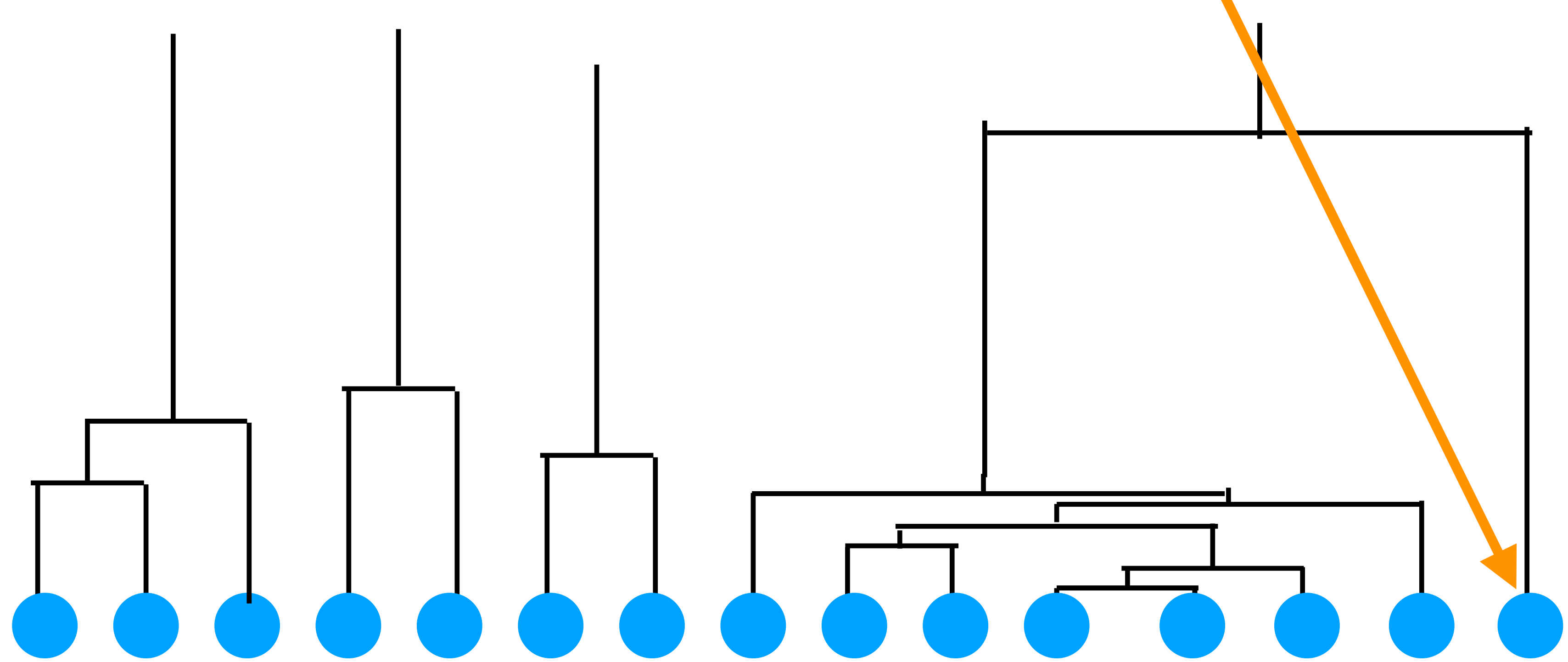
Agglomerative: Single linkage

4. Repeat step 3 until all points are in one cluster



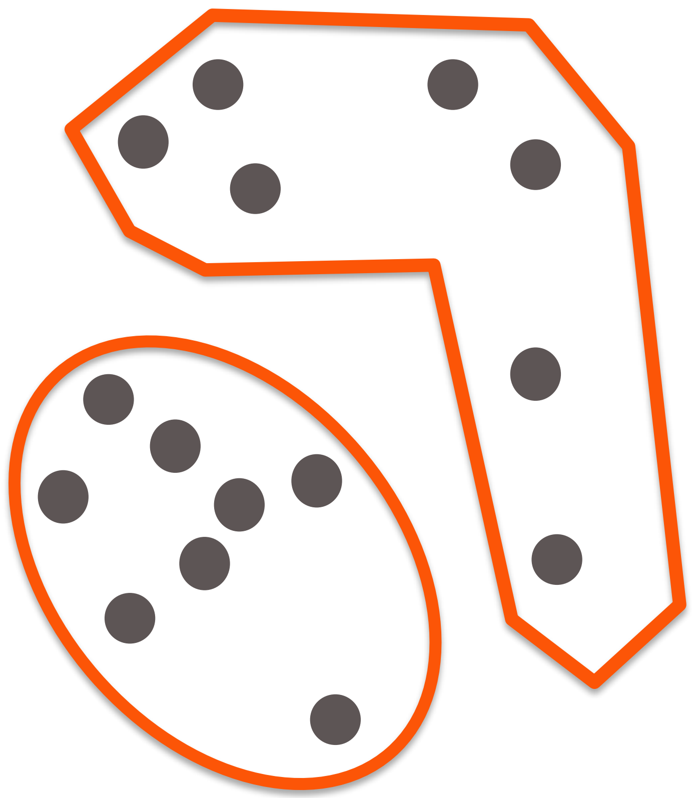
Dendrograms tell us a lot of useful information

- Cluster C1 has an **outlier**,
  who is at different distance from other in the cluster
- We can detect such outlier, from Dendrogram
  by looking at branches whose distance to the next branch
  is unusually long, compared to the others in the cluster
- Dendrograms created via agglomerative clustering
  gives us the power to have an algorithm
  that can detect such clusters with undesired properties
  and deal with them accordingly.

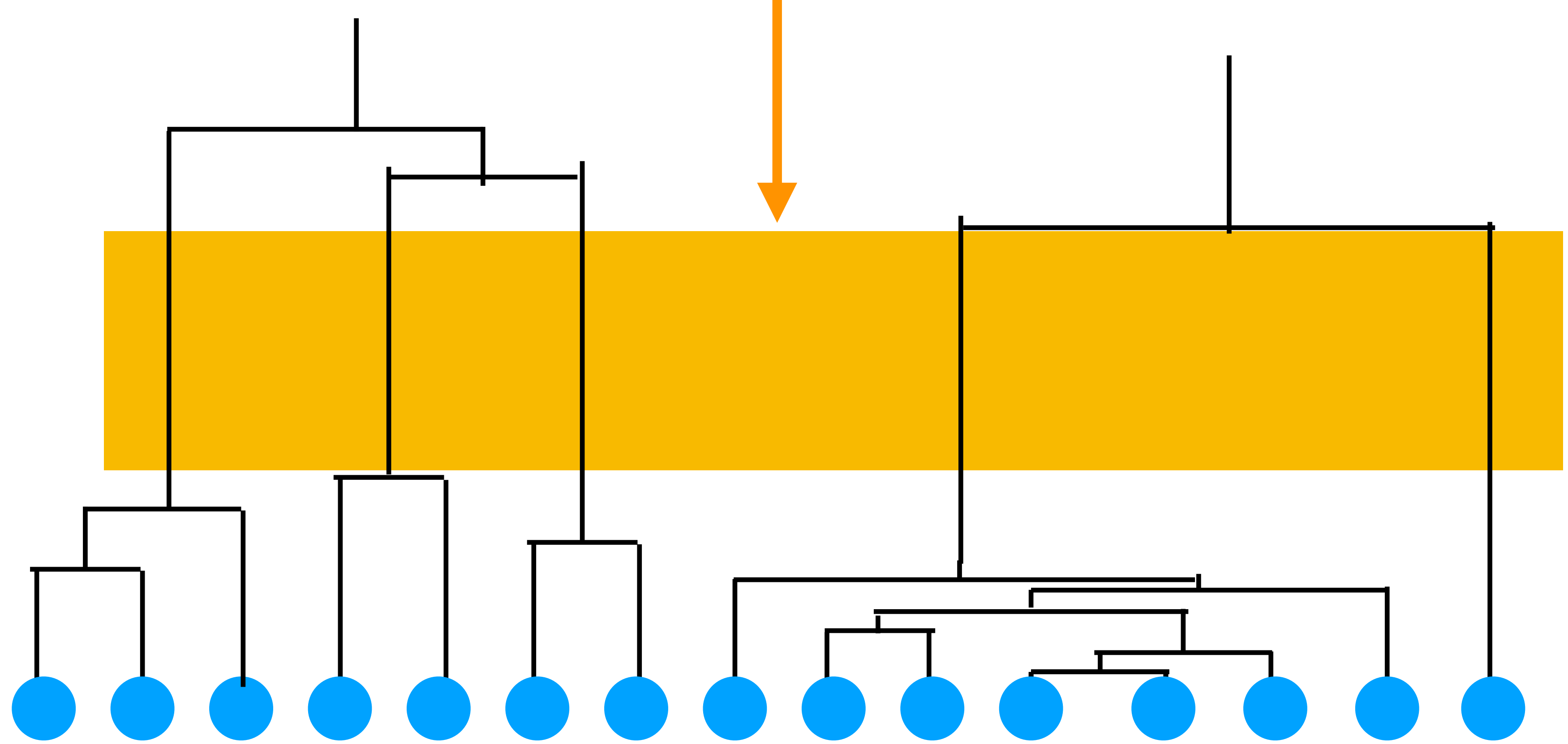


Agglomerative: Single linkage

4. Repeat step 3 until all points are in one cluster

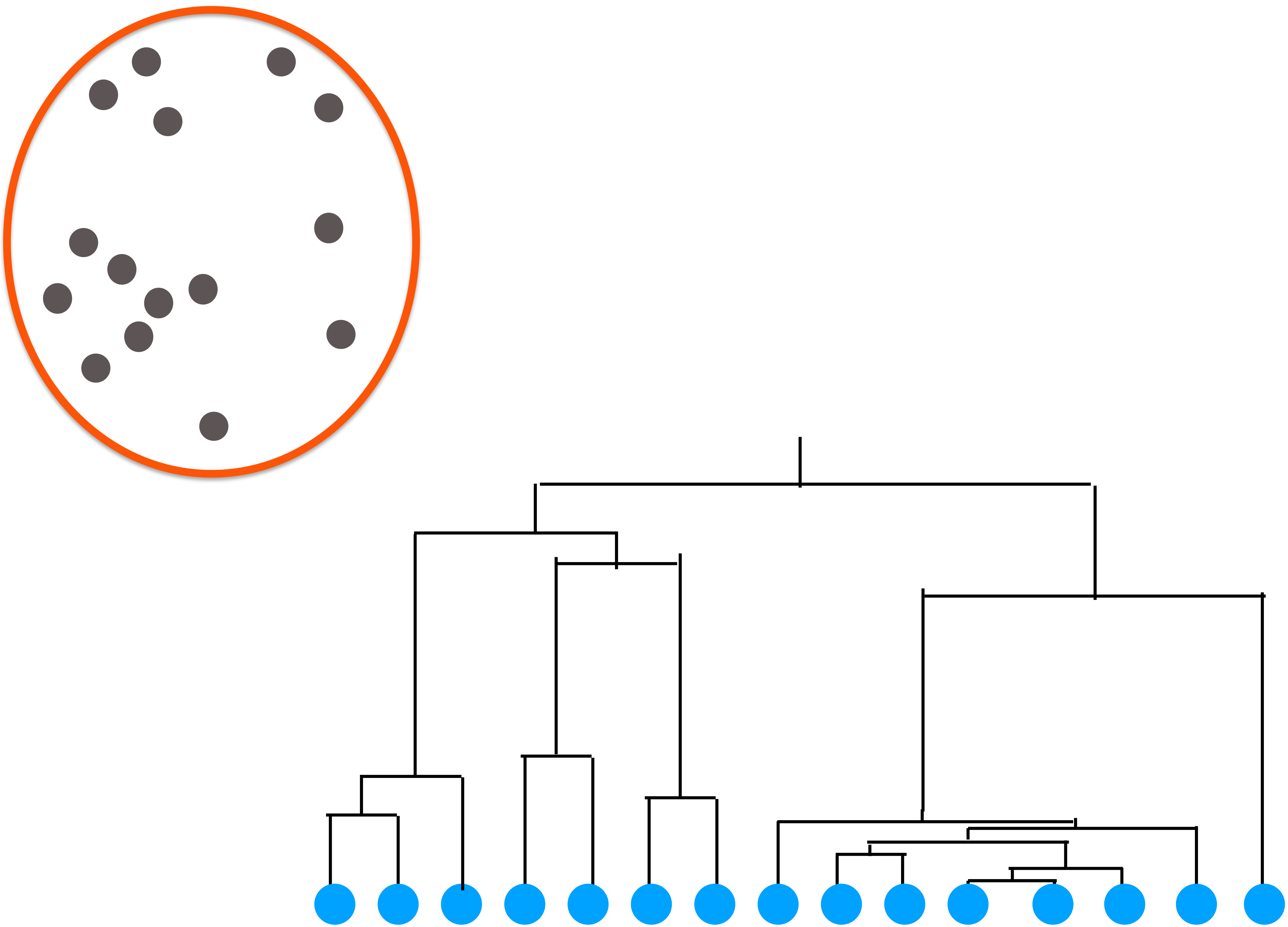


- We know we merged clusters that are far apart in the orange regime, as the height is relatively larger than the other mergers



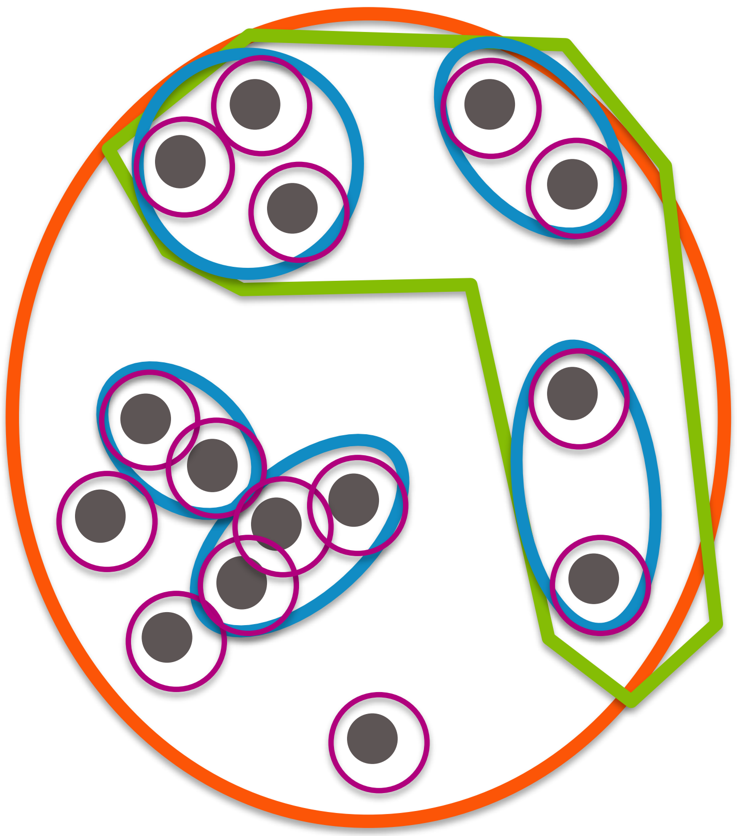
Agglomerative: Single linkage

4. Repeat step 3 until all points are in one cluster



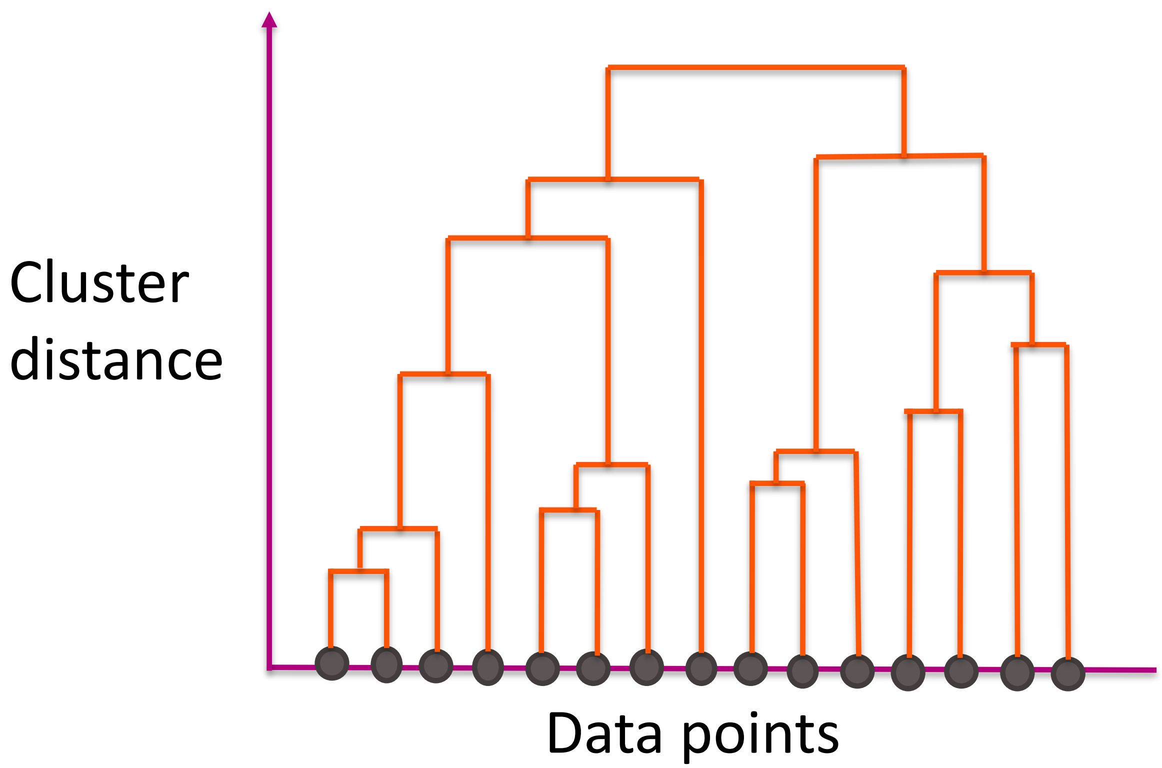
Clusters of clusters

Just like our picture for divisive clustering…



The dendrogram

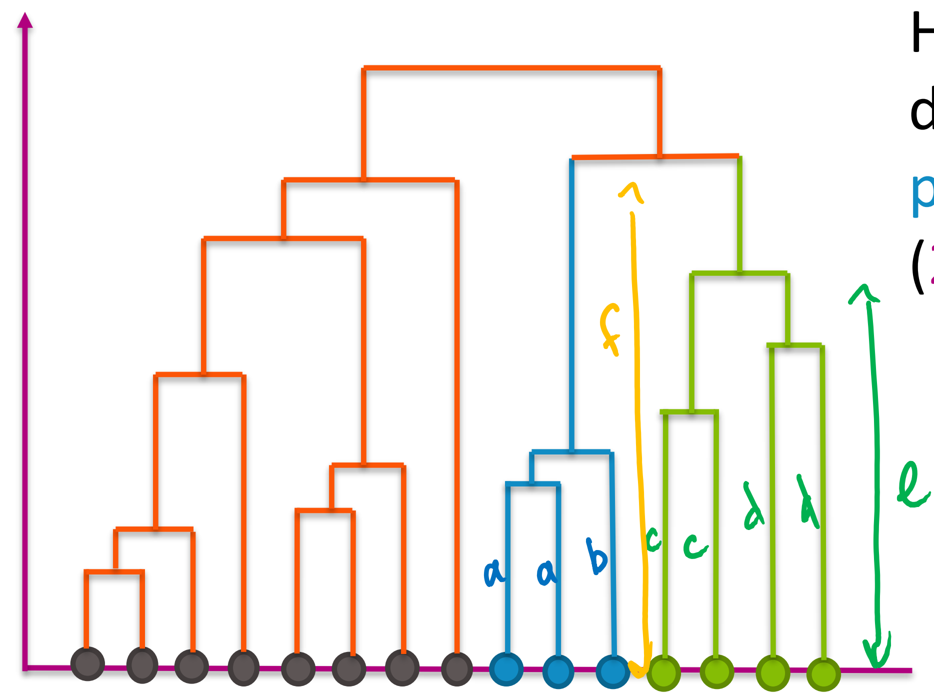
- x axis shows data points (carefully ordered)
- y-axis shows distance between pair of clusters



The dendrogram

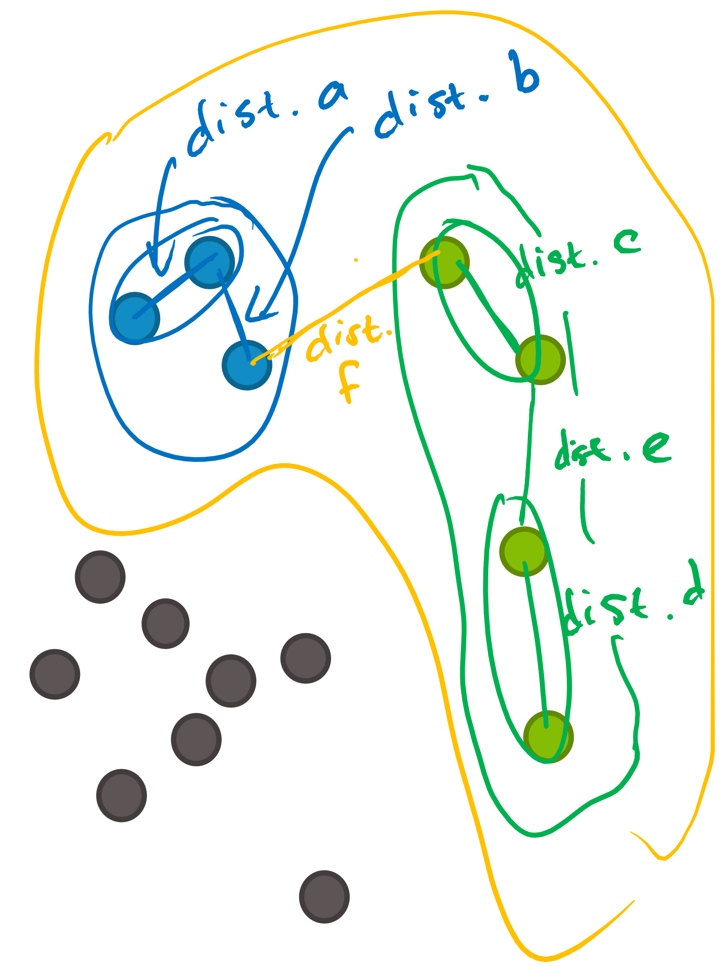
- x axis shows data points (carefully ordered)
- y-axis shows distance between pair of clusters

Cluster distance



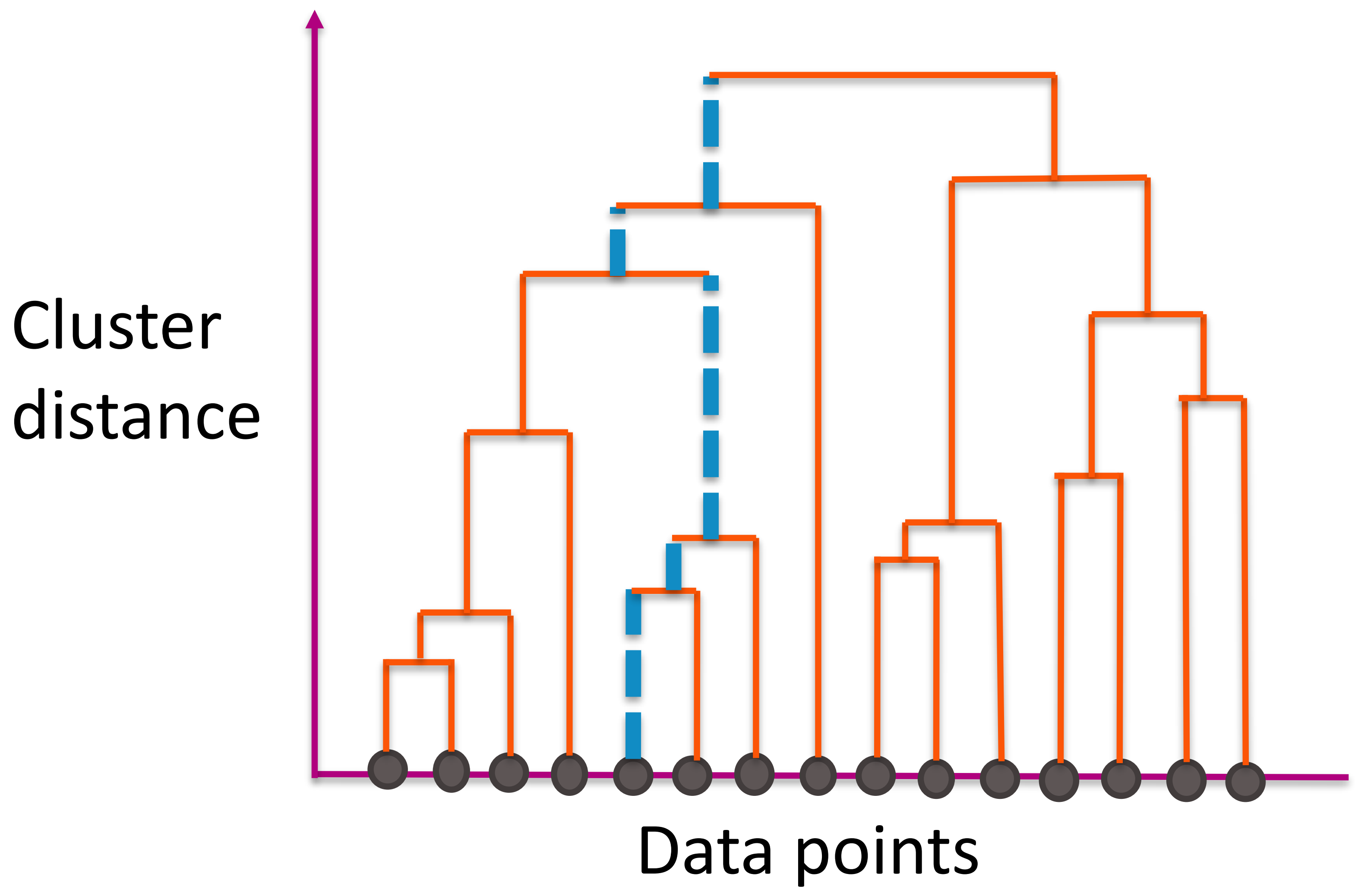
Data points

Height here indicates min distance between blue pts and green pts (2 clusters)



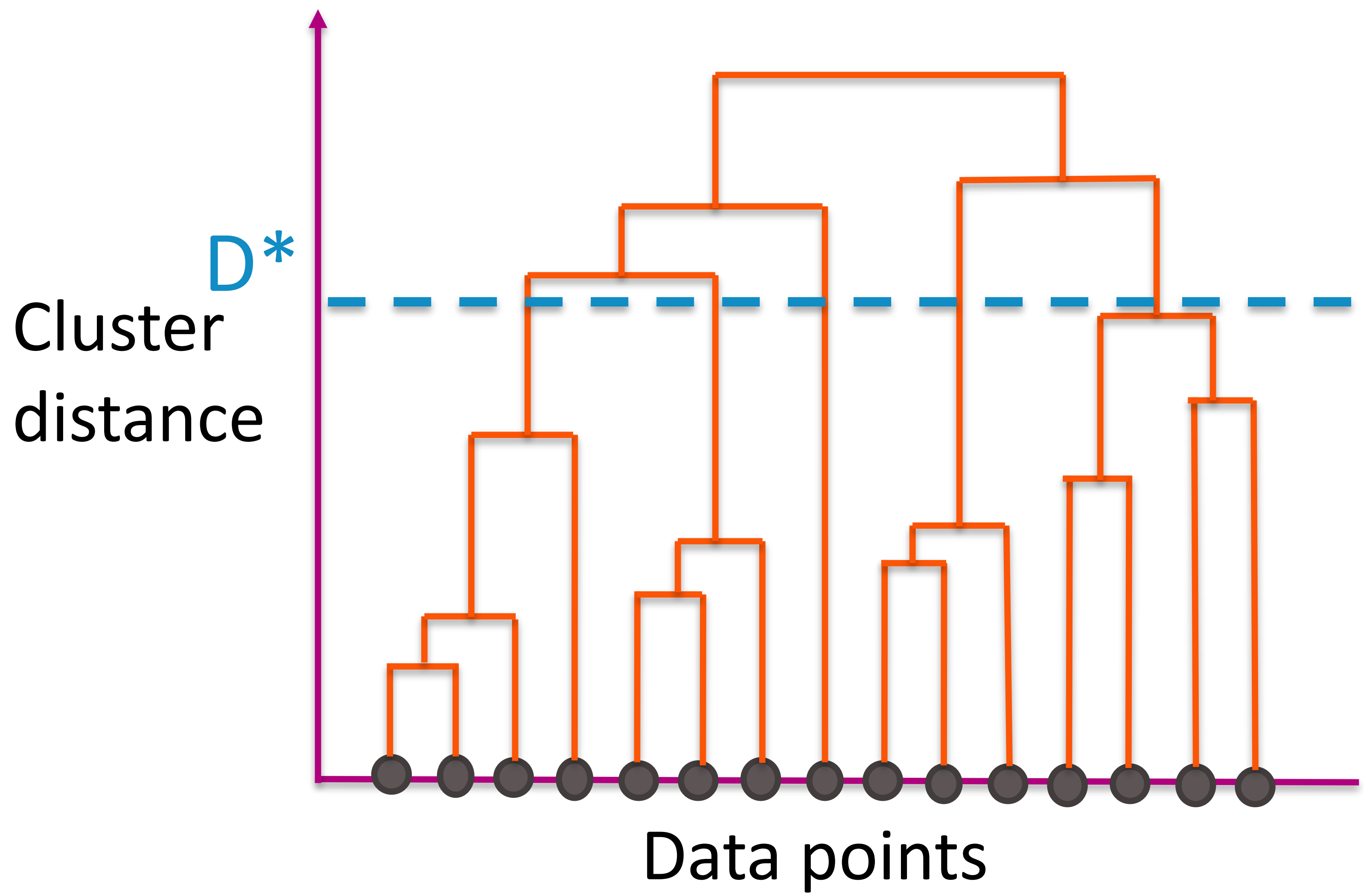
The dendrogram

Path shows all clusters to which a point belongs and the order in which clusters merge



Extracting a partition

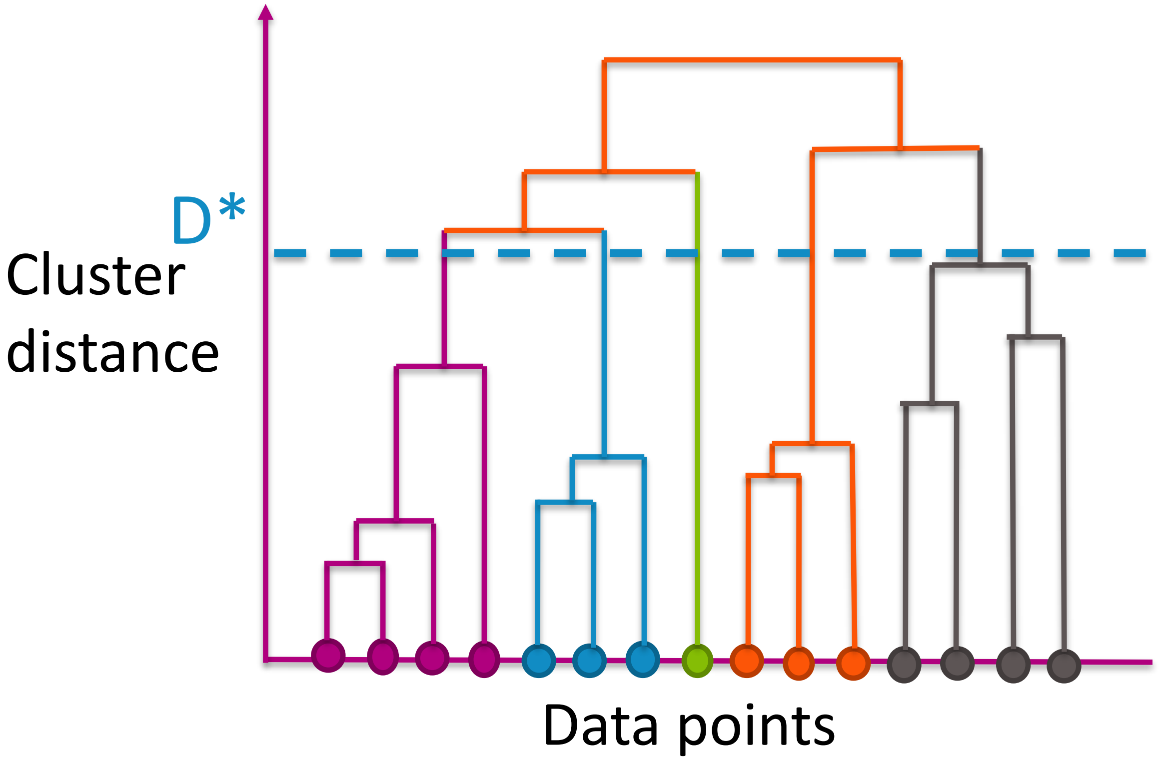
Choose a distance D* at which to cut dendogram

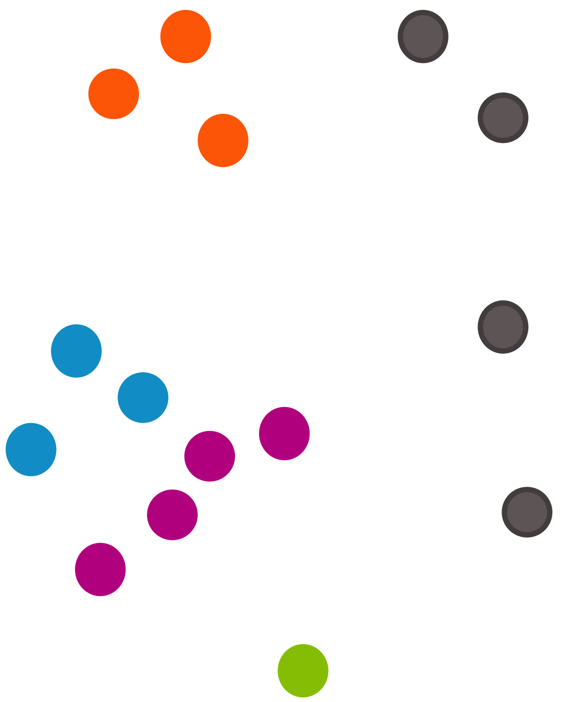


How many clusters do we get, with threshold D*?

Extracting a partition

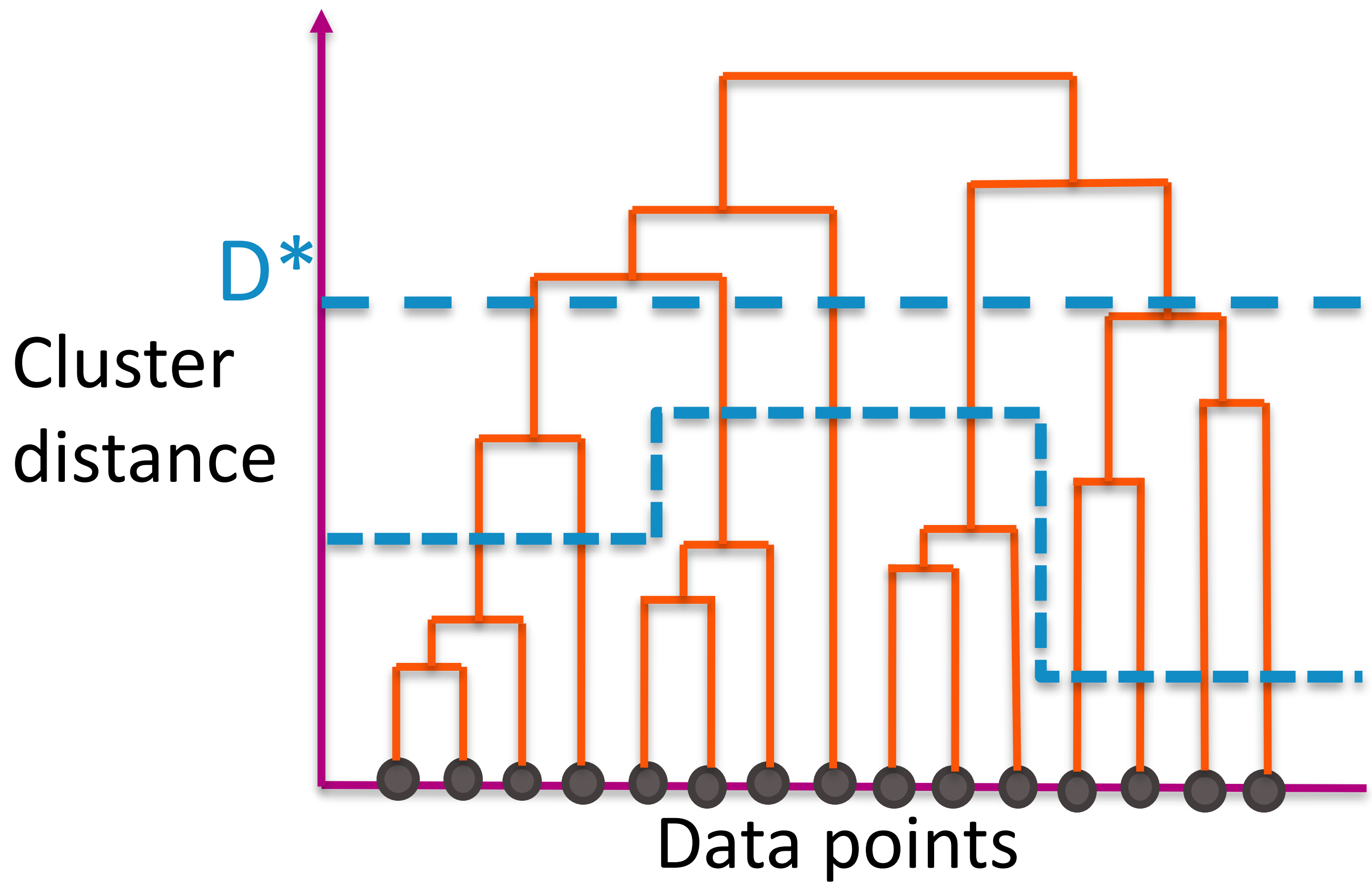
Every branch that crosses D* becomes a separate cluster





Agglomerative choices to be made

- Distance metric: $d(x_i, x_j)$
- Linkage function: e.g., $\min_{x_i \text{ in } C_1,\ x_j \text{ in } C_2} d(x_i, x_j)$
- Where and how to cut dendrogram



More on cutting dendrogram

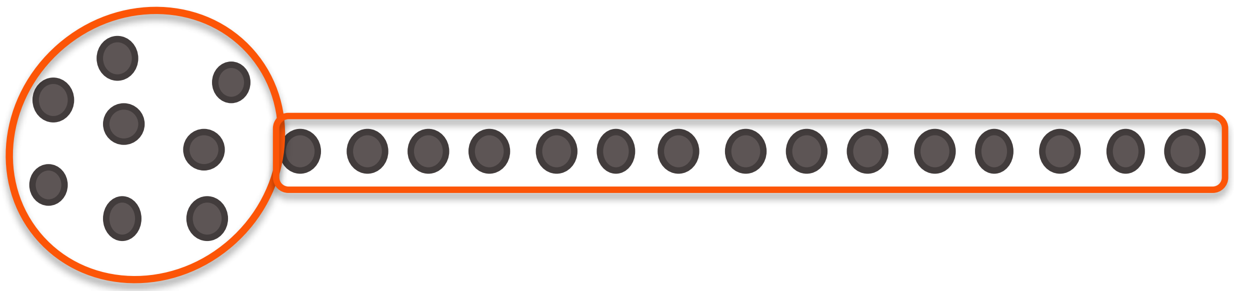
- For visualization, smaller # clusters is preferable
- For tasks like outlier detection, cut based on:
  - Distance threshold
  - Inconsistency coefficient
    - Compare height of merge to average merge heights below
    - If top merge is substantially higher, then it is joining two subsets that are relatively far apart compared to the members of each subset internally
    - Still have to choose a threshold to cut at, but now in terms of "inconsistency" rather than distance
- No cutting method is "incorrect", some are just more useful than others

Computational considerations

- Computing all pairs of distances is expensive
  - Brute force algorithm is $O(N^2 \log(N))$
- Smart implementations use triangle inequality to rule out candidate pairs
-
- Best known algorithm is $O(N^2)$

Statistical issues

Chaining: Distant points clustered together if there is a chain of pairwise close points between



Other linkage functions can be more robust, but restrict the shapes of clusters that can be found

- Complete linkage:
  max pairwise distance between clusters
- Ward criterion:
  min increase in within-cluster variance at each merge