CS434a/541a: Pattern Recognition
Prof. Olga Veksler

Lecture 15

Today

- New Topic: *Unsupervised Learning*
  - Supervised vs. unsupervised learning
  - Unsupervised learning
    - Next Time: parametric unsupervised learning
    - Today: nonparametric unsupervised learning = clustering
      - Proximity Measures
      - Criterion Functions
      - Flat Clustering
        - k-means
      - Hierarchical Clustering
        - Divisive
        - Agglomerative

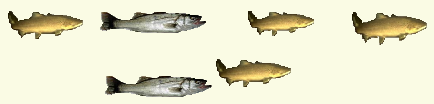
Supervised vs. Unsupervised Learning

- Up to now we considered ***supervised learning*** scenario, where we are given
  1. samples $\boldsymbol{x}_1,\ldots,\boldsymbol{x}_n$
  2. class labels for all samples $\boldsymbol{x}_1,\ldots,\boldsymbol{x}_n$
  - This is also called learning with teacher, since correct answer (the true class) is provided

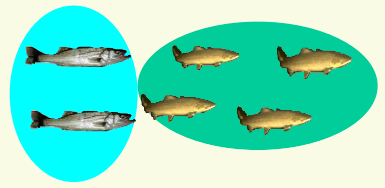
- In the next few lectures we consider ***unsupervised learning*** scenario, where we are only given
  1. samples $\boldsymbol{x}_1,\ldots,\boldsymbol{x}_n$
  - This is also called learning without teacher, since correct answer is not provided
  - do not split data into training and test sets

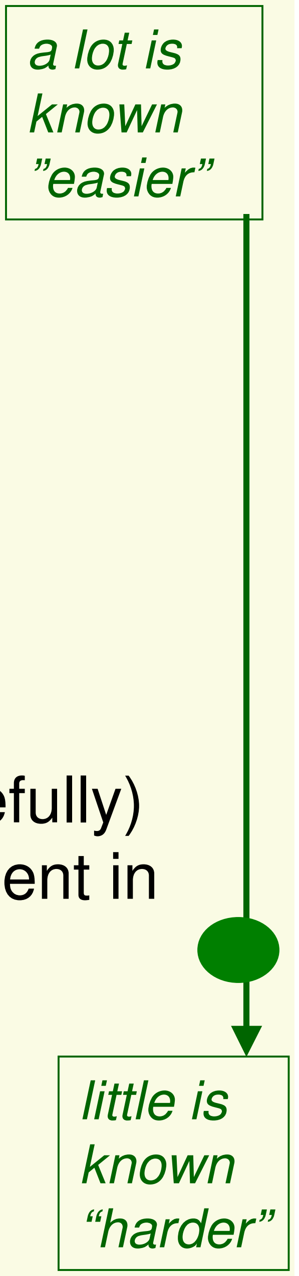
Unsupervised Learning

- Data is *not labeled*



1. Parametric Approach
   - assume parametric distribution of data
   - estimate parameters of this distribution
   - much “harder” than supervised case
- NonParametric Approach
  - group the data into ***clusters***, each cluster (hopefully) says something about categories (classes) present in the data



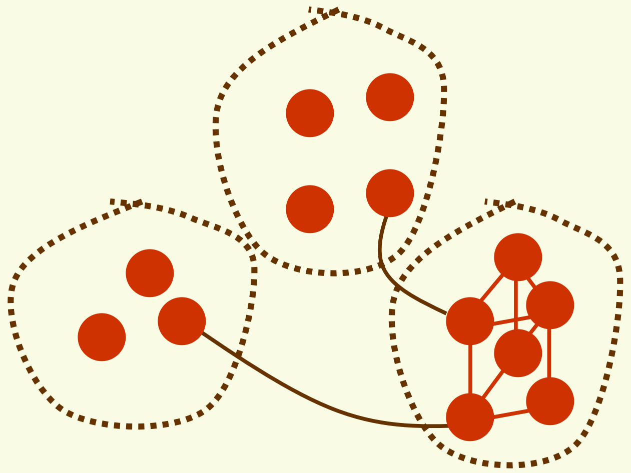


Why Unsupervised Learning?

- Unsupervised learning is harder
  - How do we know if results are meaningful? No answer labels are available.
    - Let the expert look at the results (external evaluation)
    - Define an objective function on clustering (internal evaluation)
- We nevertheless need it because
  1. Labeling large datasets is very costly (speech recognition)
     - sometimes can label only a few examples by hand
  2. May have no idea what/how many classes there are (data mining)
  3. May want to use clustering to gain some insight into the structure of the data before designing a classifier
     - Clustering as data description

Clustering

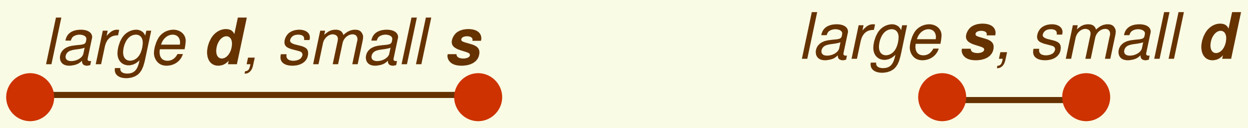
- Seek “natural” clusters in the data



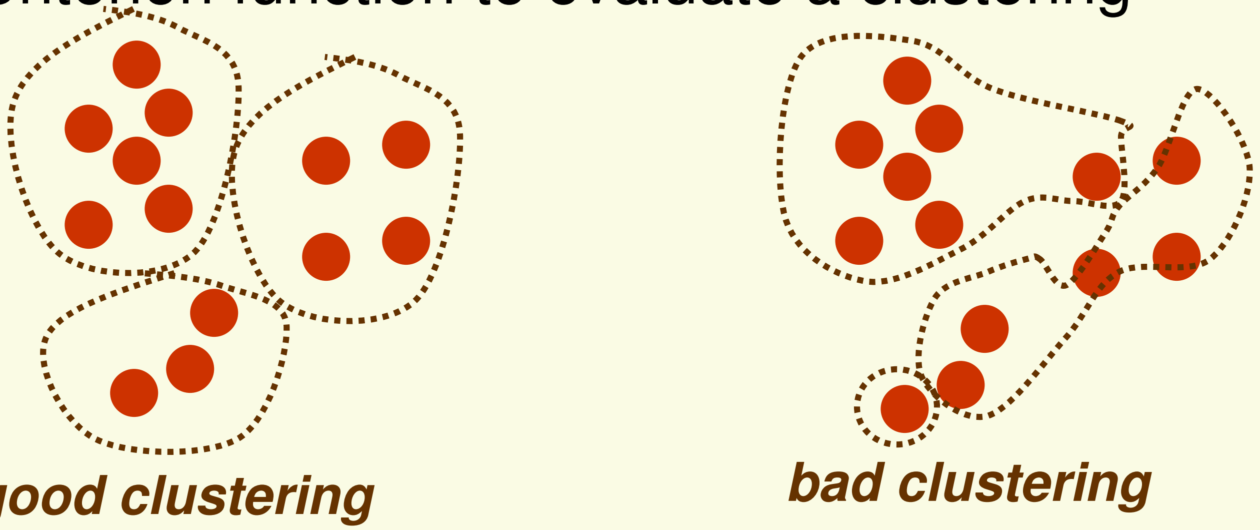
- What is a good clustering?
  - internal (within the cluster) distances should be small
  - external (intra-cluster) should be large
- Clustering is a way to discover new categories (classes)

What we Need for Clustering

1. Proximity measure, *either*
   - similarity measure $\boldsymbol{s}(\boldsymbol{x}_i,\boldsymbol{x}_k)$: large if $\boldsymbol{x}_i,\boldsymbol{x}_k$ are similar
   - dissimilarity(or distance) measure $\boldsymbol{d}(\boldsymbol{x}_i,\boldsymbol{x}_k)$: small if $\boldsymbol{x}_i,\boldsymbol{x}_k$ are similar

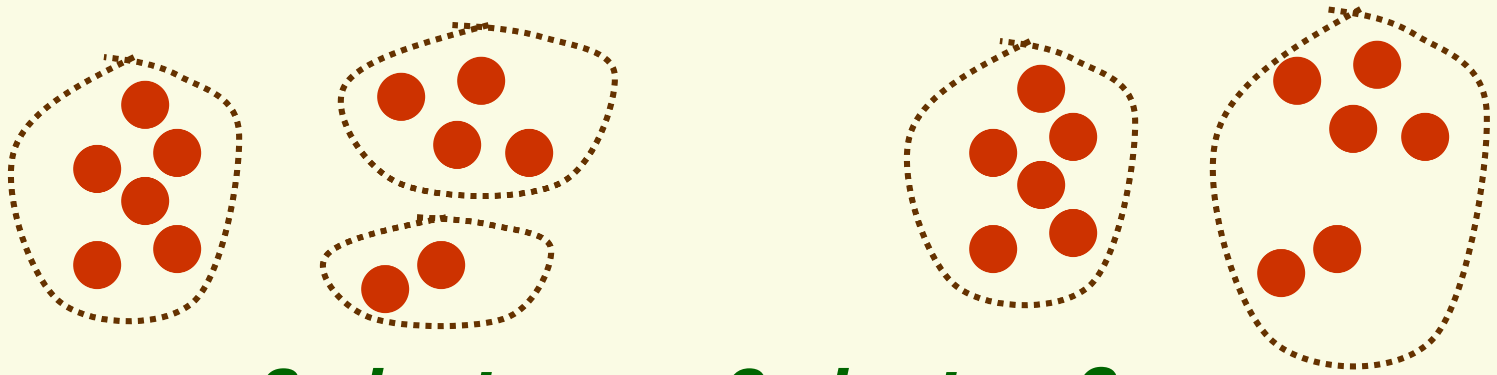


2. Criterion function to evaluate a clustering



3. Algorithm to compute clustering
   - For example, by optimizing the criterion function

How Many Clusters?

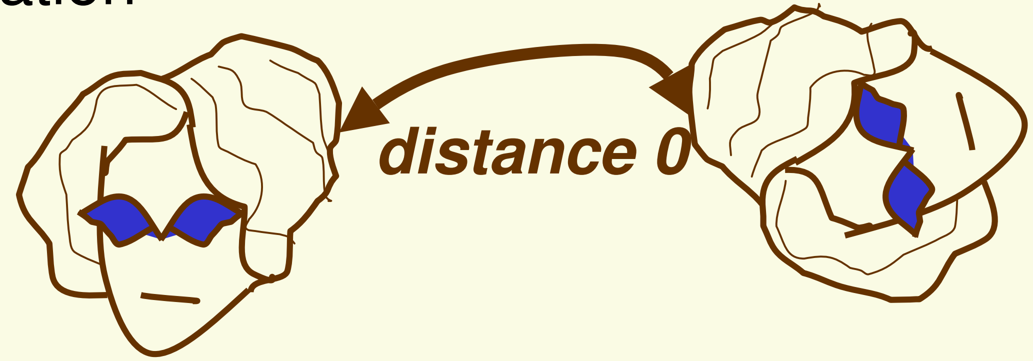


3 clusters or 2 clusters?

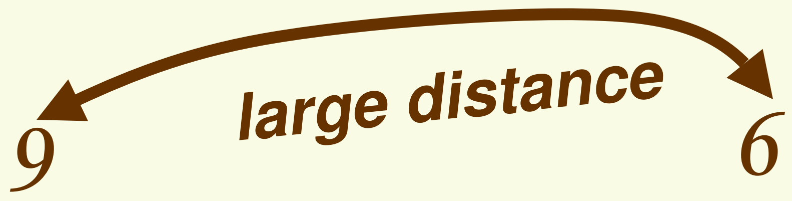
- Possible approaches
  1. fix the number of clusters to ***k***
  2. find the best clustering according to the criterion function (number of clusters may vary)

Proximity Measures

- good proximity measure is VERY application dependent
  - Clusters should be invariant under the transformations "natural" to the problem
  - For example for object recognition, should have invariance to rotation



  - For character recognition, no invariance to rotation

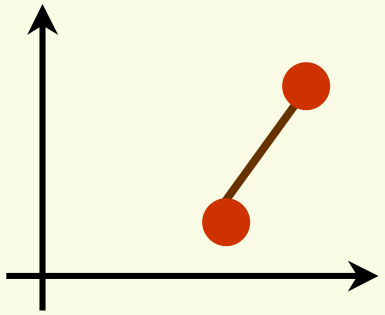


Distance (dissimilarity) Measures

- Euclidean distance

$$d(x_i, x_j) = \sqrt{\sum_{k=1}^{d} \left(x_i^{(k)} - x_j^{(k)}\right)^2}$$

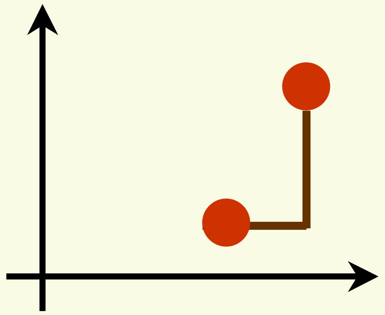
  - translation invariant



- Manhattan (city block) distance

$$d(x_i, x_j) = \sum_{k=1}^{d} \left| x_i^{(k)} - x_j^{(k)} \right|$$

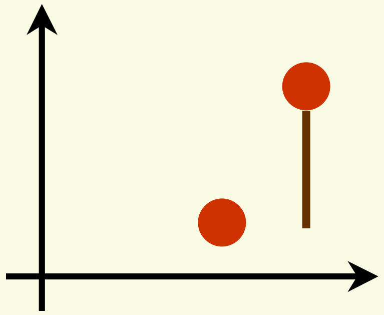
  - approximation to Euclidean distance, cheaper to compute



- Chebyshev distance

$$d(x_i, x_j) = \max_{1 \le k \le d} | x_i^{(k)} - x_j^{(k)} |$$

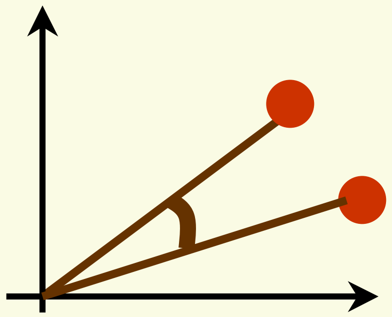
  - approximation to Euclidean distance, cheapest to compute



Similarity Measures

- Cosine similarity:

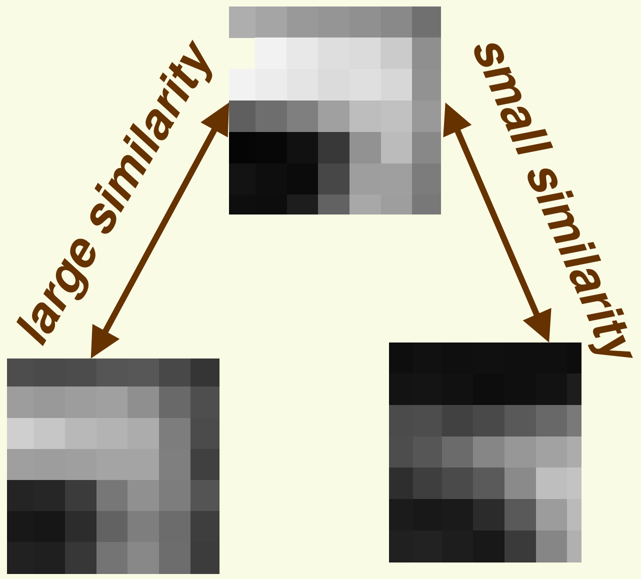
$$s(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{\boldsymbol{x}_i^T \boldsymbol{x}_j}{\|\boldsymbol{x}_i\| \, \|\boldsymbol{x}_j\|}$$



  - the smaller the angle, the larger the similarity
  - scale invariant measure
  - popular in text retrieval

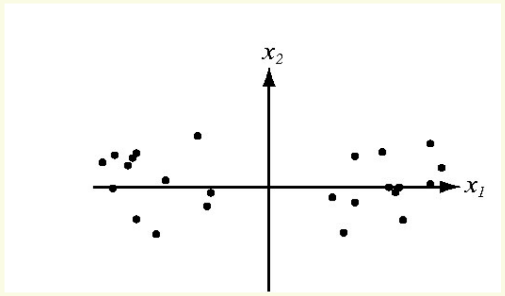
- Correlation coefficient
  - popular in image processing

$$s(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{\sum_{k=1}^{d} \left(x_i^{(k)} - \overline{x}_i\right)\left(x_i^{(k)} - \overline{x}_i\right)}{\left[\sum_{k=1}^{d} \left(x_i^{(k)} - \overline{x}_i\right)^2 \sum_{k=1}^{d} \left(x_j^{(k)} - \overline{x}_j\right)^2\right]^{1/2}}$$

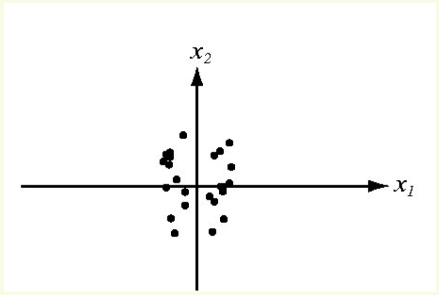


Feature Scale

- old problem: how to choose appropriate relative scale for features?
  - [length (in meters or cms?), weight(in in grams or kgs?)]
  - In supervised learning, can normalize to zero mean unit variance with no problems
  - in clustering this is more problematic, ***if variance in data is due to cluster presence, then normalizing features is not a good thing***



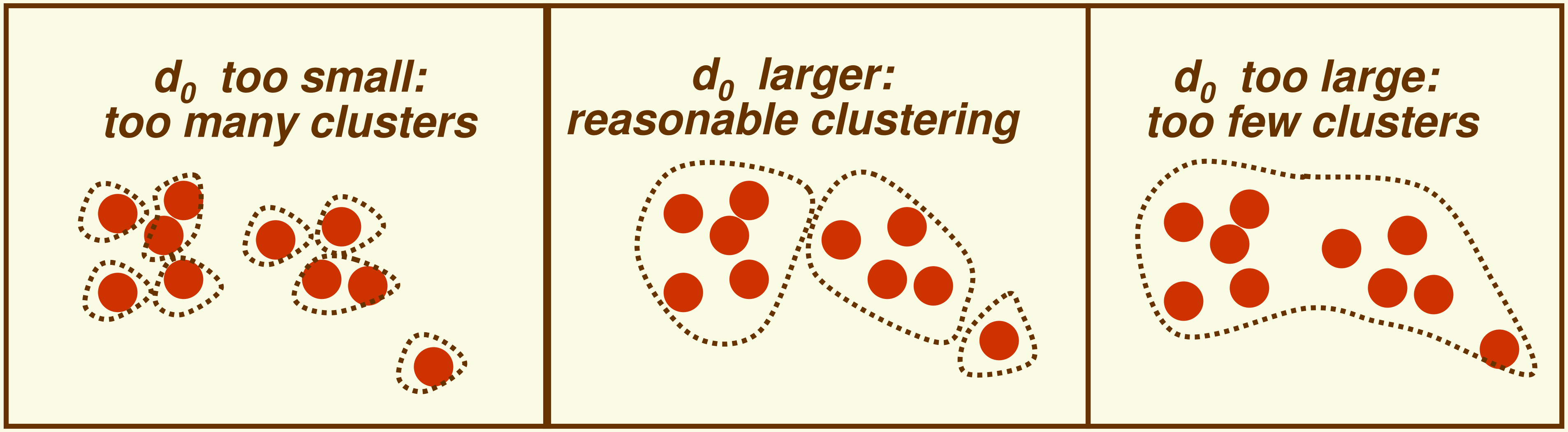
before normalization



after normalization

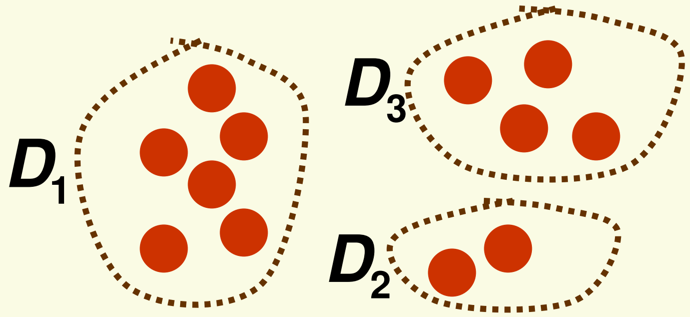
Simplest Clustering Algorithm

- Having defined a proximity function, can develop a simple clustering algorithm
  - go over all sample pairs, and put them in the same cluster if the distance between them is less then some threshold distance $\boldsymbol{d_0}$ (or if similarity is larger than $\boldsymbol{s_0}$)
  - Pros: simple to understand and implement
  - Cons: very dependent on $\boldsymbol{d_0}$ (or $\boldsymbol{s_0}$), automatic choice of $\boldsymbol{d_0}$ (or $\boldsymbol{s_0}$)is not an easily solved issue



Criterion Functions for Clustering

- Have samples $\mathbf{x}_1,\ldots,\mathbf{x}_n$
- Suppose partitioned samples into c subsets $D_1,\ldots,D_c$



- There are approximately $c^n/c!$ distinct partitions
- Can define a criterion function $J(D_1,\ldots,D_c)$ which measures the quality of a partitioning $D_1,\ldots,D_c$
- Then the clustering problem is a well defined problem
  - the optimal clustering is the partition which optimizes the criterion function

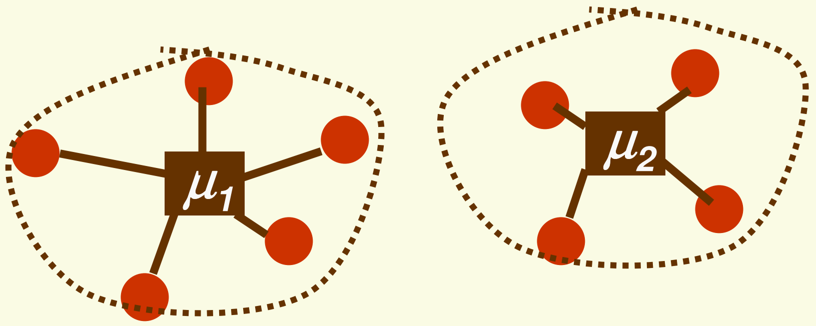
SSE Criterion Function

- Let n_i be the number of samples in D_i, and define the mean of samples in in D_i

$$\mu_i = \frac{1}{n_i} \sum_{x \in D_i} x$$

- Then the sum-of-squared errors criterion function (to minimize) is:

$$J_{SSE} = \sum_{i=1}^{c} \sum_{x \in D_i} \| x - \mu_i \|^2$$

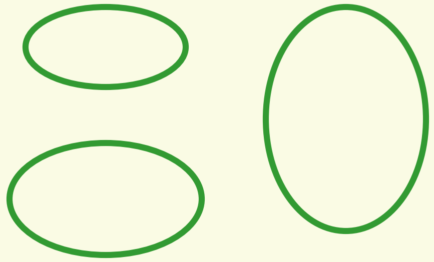


- Note that the number of clusters, c, is fixed

SSE Criterion Function

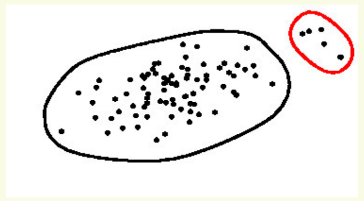
$$J_{SSE} = \sum_{i=1}^{c} \sum_{x \in D_i} \| \boldsymbol{x} - \mu_i \|^2$$

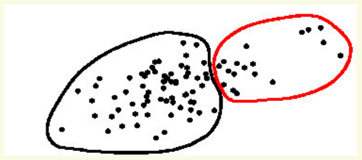
- SSE criterion appropriate when data forms compact clouds that are relatively well separated



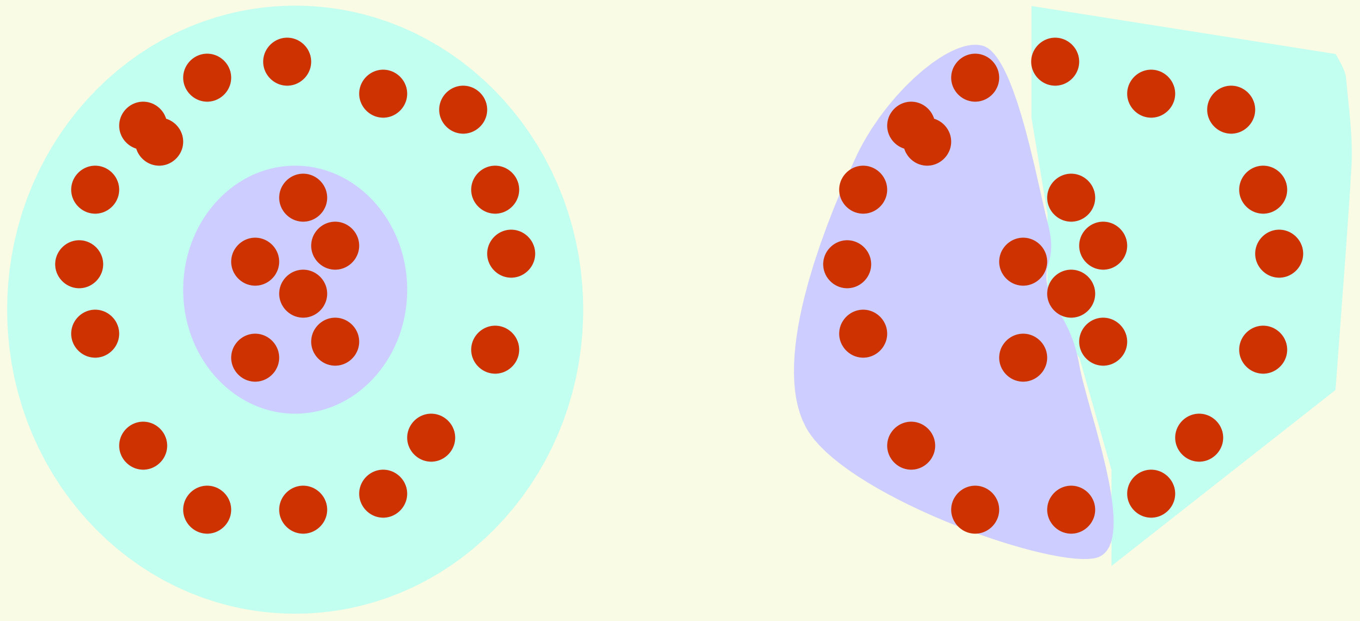
- SSE criterion favors equally sized clusters, and may not be appropriate when "natural" groupings have very different sizes

large J_{SSE} ***small J_{SSE}***





Failure Example for J_{SSE}



larger J_{SSE}

smaller J_{SSE}

- The problem is that one of the “natural” clusters is not compact (the outer ring)

Other Minimum Variance Criterion Functions

- We can rewrite the SSE as

$$J_{SSE} = \sum_{i=1}^{c} \sum_{x \in D_i} || \mathbf{x} - \mu_i ||^2 \quad = \frac{1}{2} \sum_{i=1}^{c} n_i \boxed{\left[\frac{1}{n_i^2} \sum_{y \in D_i} \sum_{x \in D_i} || \mathbf{x} - \mathbf{y} ||^2 \right]}$$

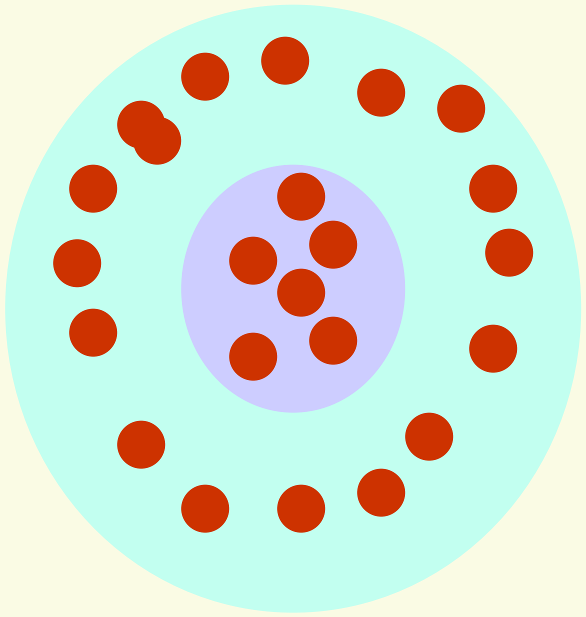
$\mathbf{d}_i$ = average Euclidian distance between all pairs of samples in $\mathbf{D}_i$

- Can obtain other criterion functions by replacing $||\mathbf{x} - \mathbf{y}||^2$ by any other measure of distance between points in $\mathbf{D}_i$
- Alternatively can replace $\mathbf{d}_i$ by the median, maximum, etc. instead of the average distance

Maximum Distance Criterion

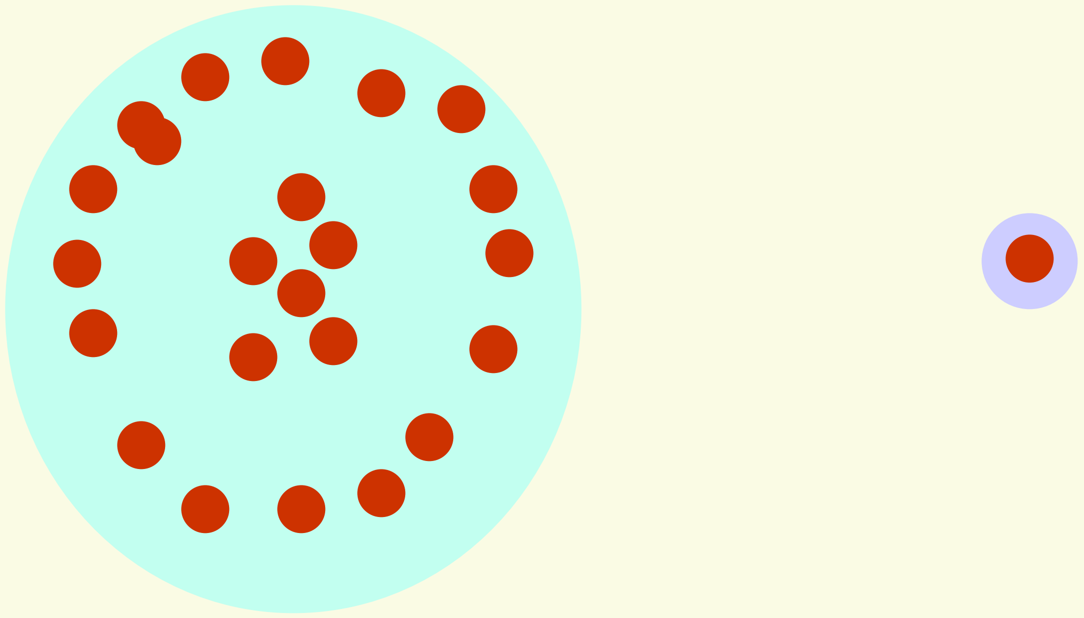
- Consider $J_{\max} = \sum_{i=1}^{c} n_i \left[\max_{y \in D_i, x \in D_i} \| x - y \|^2 \right]$

- Solves previous case



smallest J_{max}

- However J_{max} is not robust to outliers



smallest J_{max}

Other Criterion Functions

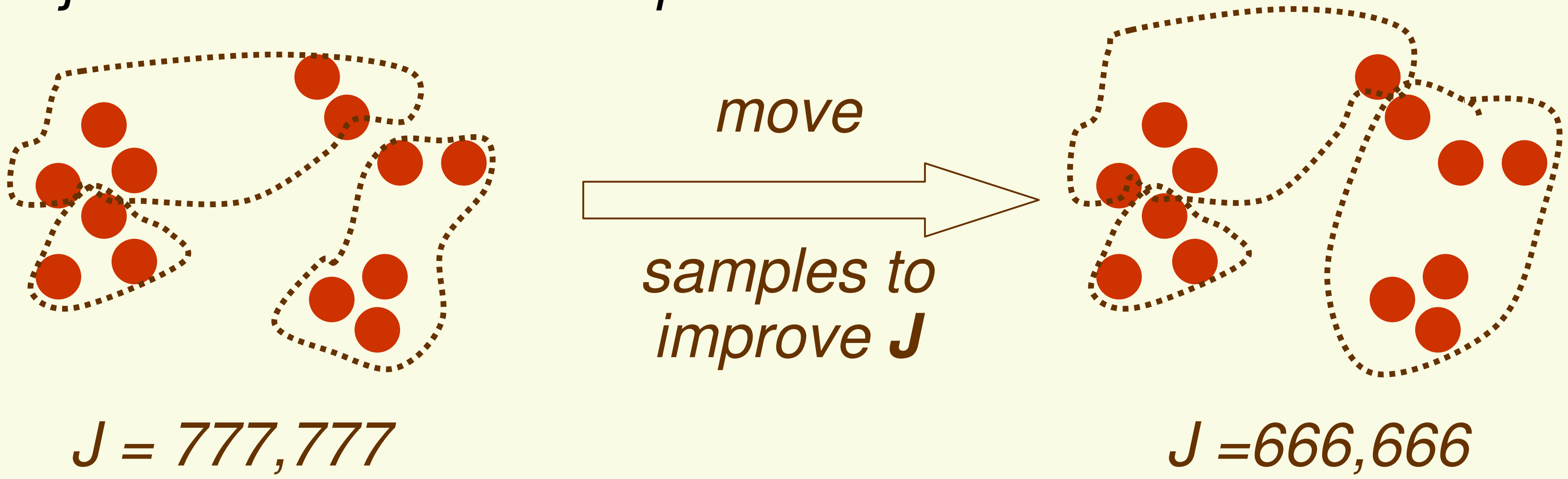
- Recall definition of scatter matrices
  - scatter matrix for ith cluster $\boldsymbol{S}_i = \sum_{\boldsymbol{x} \in D_i} (\boldsymbol{x} - \mu_i)(\boldsymbol{x} - \mu_i)^t$
  - within the cluster scatter matrix $\boldsymbol{S}_W = \sum_{i=1}^{c} \boldsymbol{S}_i$
- Determinant of $\boldsymbol{S}_W$ roughly measures the square of the volume
- Assuming $\boldsymbol{S}_W$ is nonsingular, define determinant criterion function:

$$\boldsymbol{J}_d = |\boldsymbol{S}_W| = \left| \sum_{i=1}^{c} \boldsymbol{S}_i \right|$$

  - $\boldsymbol{J}_d$ is invariant to scaling of the axis, and is useful if there are unknown irrelevant linear transformations of the data

Iterative Optimization Algorithms

- Now have both proximity measure and criterion function, need algorithm to find the optimal clustering
- Exhaustive search is impossible, since there are approximately ***$c^n/c!$*** possible partitions
- Usually some iterative algorithm is used
  1. Find a reasonable initial partition
  2. Repeat: *move samples from one group to another s.t. the objective function **J** is improved*



Iterative Optimization Algorithms

- Iterative optimization algorithms are similar to gradient descent
  - move in the direction of descent (ascent), but not in the steepest descent direction since have no derivative of the objective function
  - solution depends on the initial point
  - cannot find global minimum
- Main Issue
  - How to move from current partitioning to the one which improves the objective function

K-means Clustering

- We now consider an example of iterative optimization algorithm for the special case of $\boldsymbol{J_{SSE}}$ objective function

$$J_{SSE} = \sum_{i=1}^{k} \sum_{x \in D_i} || \boldsymbol{x} - \mu_i ||^2$$

  - for a different objective function, we need a different optimization algorithm, of course

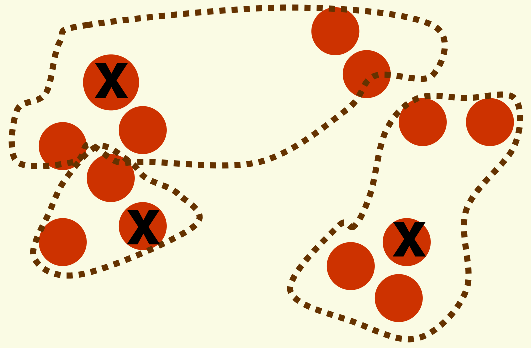
- Fix number of clusters to $\boldsymbol{k}$ ($\boldsymbol{c} = \boldsymbol{k}$)

- ***k***-means is probably the most famous clustering algorithm
  - it has a smart way of moving from current partitioning to the next one

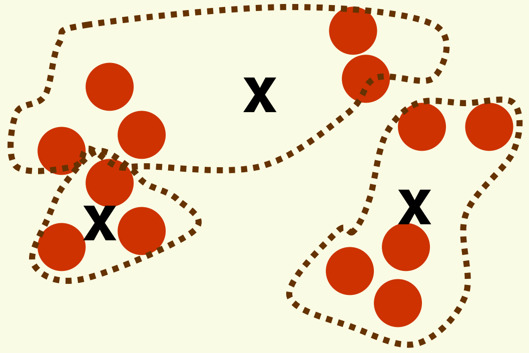
K-means Clustering

k = 3

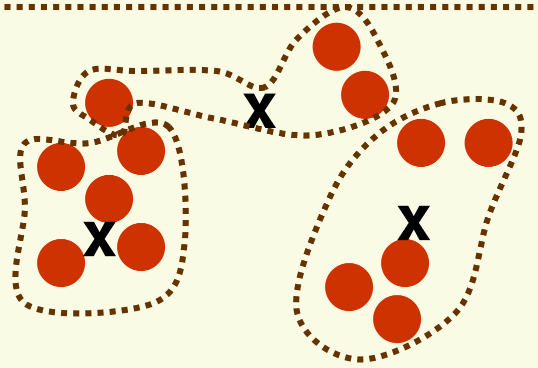
1. Initialize
   - pick ***k*** cluster centers arbitrary
   - assign each example to closest center



2. compute sample means for each cluster



3. reassign all samples to the closest mean

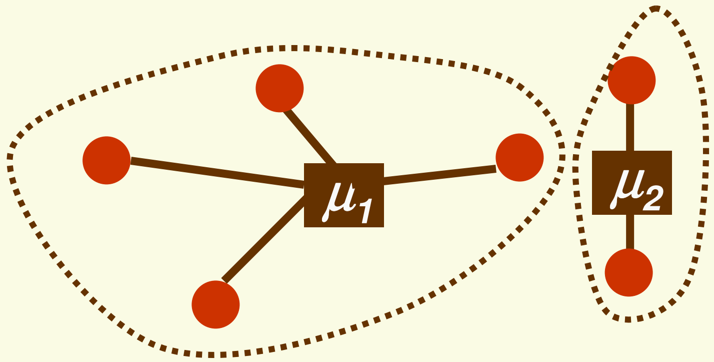


4. if clusters changed at step 3, go to step 2

K-means Clustering

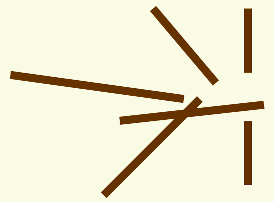
- Consider steps *2* and *3* of the algorithm

2. compute sample means for each cluster

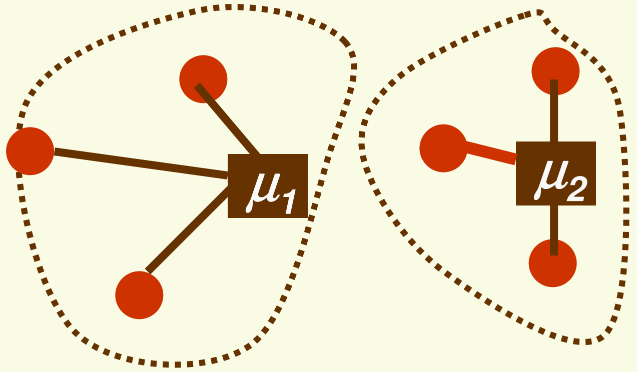


$$J_{SSE} = \sum_{i=1}^{k} \sum_{\boldsymbol{x} \in D_i} \| \boldsymbol{x} - \mu_i \|^2$$

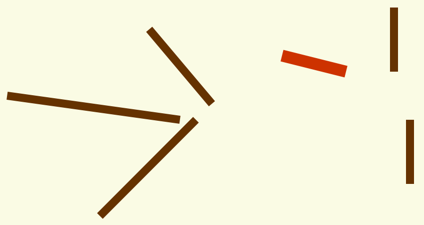
= sum of



3. reassign all samples to the closest mean

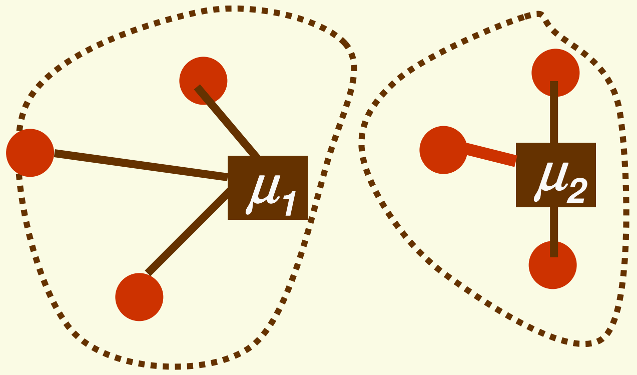


If we represent clusters by their old means, the error has gotten smaller

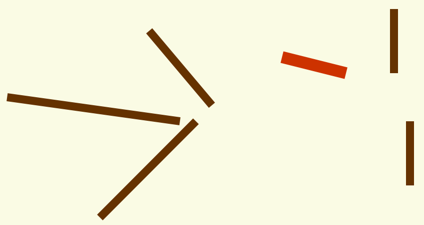


K-means Clustering

3. reassign all samples to the closest mean



If we represent clusters by their old means, the error has gotten smaller



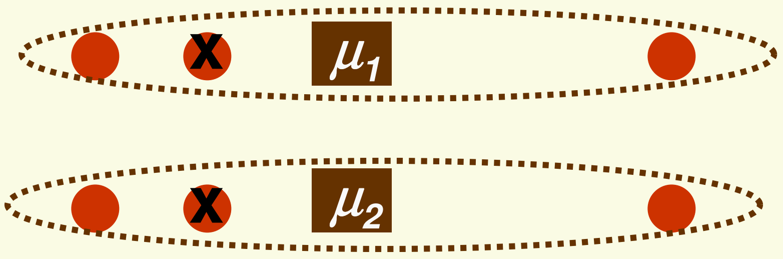
- However we represent clusters by their new means, and mean is always the smallest representation of a cluster

$$\frac{\partial}{\partial \boldsymbol{z}} \sum_{\boldsymbol{x} \in D_i} \frac{1}{2} \| \boldsymbol{x} - \boldsymbol{z} \|^2 = \frac{\partial}{\partial \boldsymbol{z}} \sum_{\boldsymbol{x} \in D_i} \frac{1}{2} \left(\| \boldsymbol{x} \|^2 - 2\boldsymbol{x}^t \boldsymbol{z} + \| \boldsymbol{z} \|^2 \right) = \sum_{\boldsymbol{x} \in D_i} \left(- \boldsymbol{x} + \boldsymbol{z} \right) = 0$$

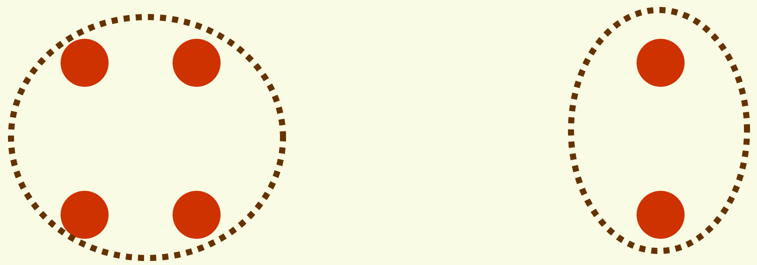
$$\Rightarrow \boldsymbol{z} = \frac{1}{n_i} \sum_{\boldsymbol{x} \in D_i} \boldsymbol{x}$$

K-means Clustering

- We just proved that by doing steps *2* and *3*, the objective function goes down
  - in two step, we found a “smart “ move which decreases the objective function
- Thus the algorithm converges after a finite number of iterations of steps *2* and *3*
- However the algorithm is not guaranteed to find a global minimum



2-means gets stuck here



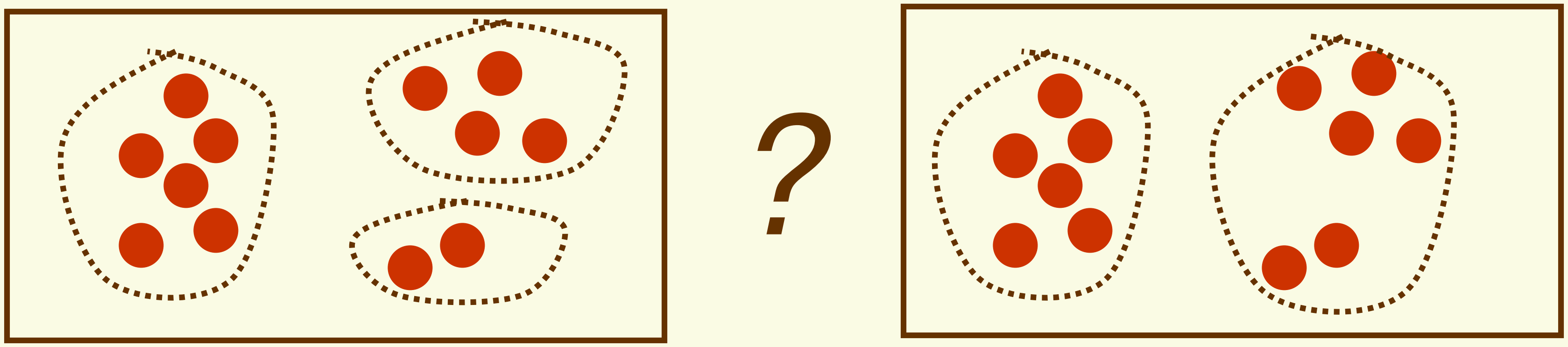
global minimum of J_{SSE}

K-means Clustering

- Finding the optimum of $\boldsymbol{J_{SSE}}$ is NP-hard
- In practice, **k**-means clustering performs usually well
- It is very efficient
- Its solution can be used as a starting point for other clustering algorithms
- Still 100’s of papers on variants and improvements of **k**-means clustering every year

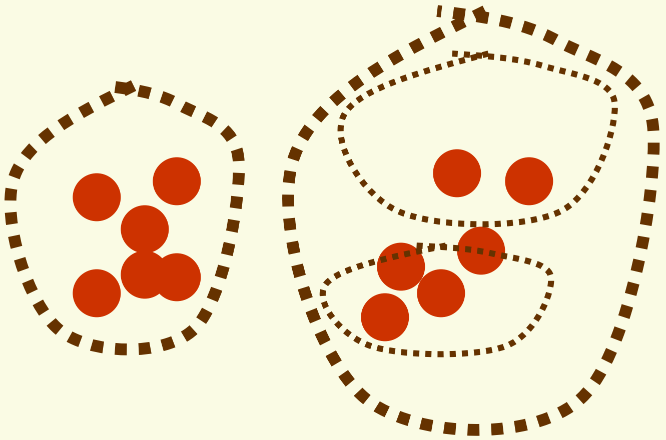
Hierarchical Clustering

- Up to now, considered “flat” clustering

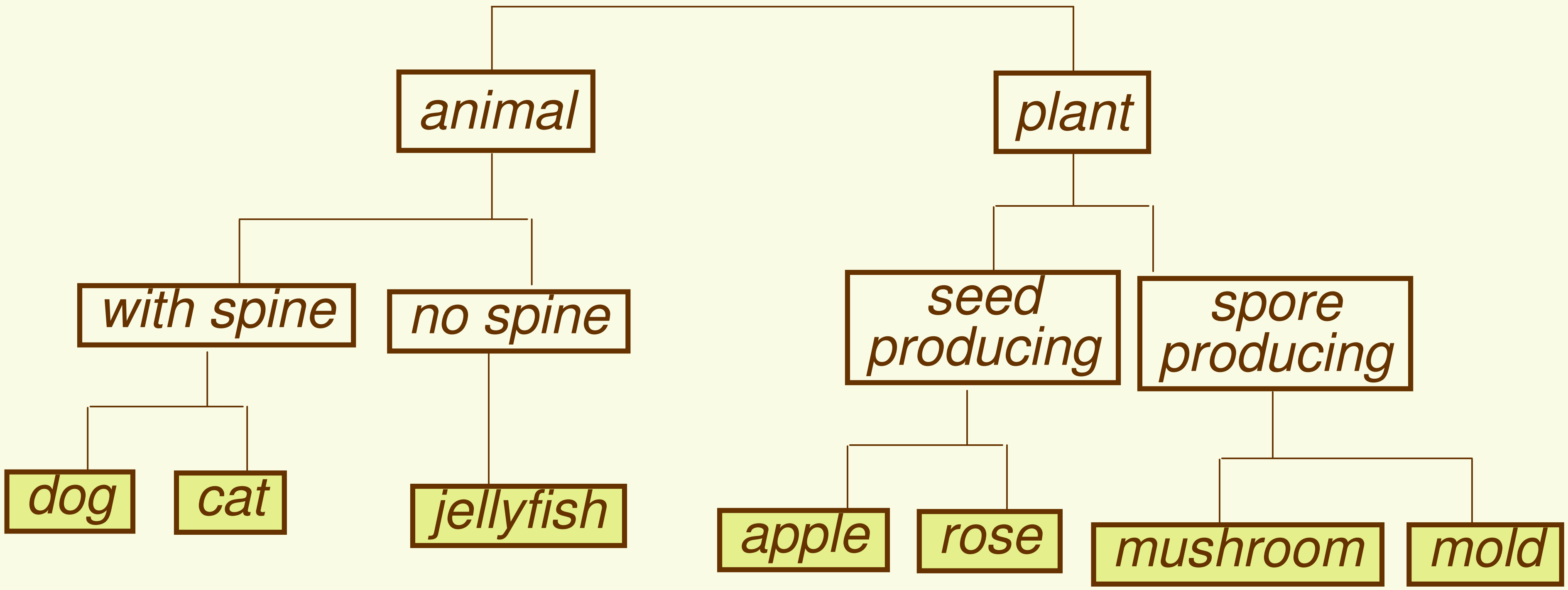


- For some data, hierarchical clustering is more appropriate than “flat” clustering

- Hierarchical clustering

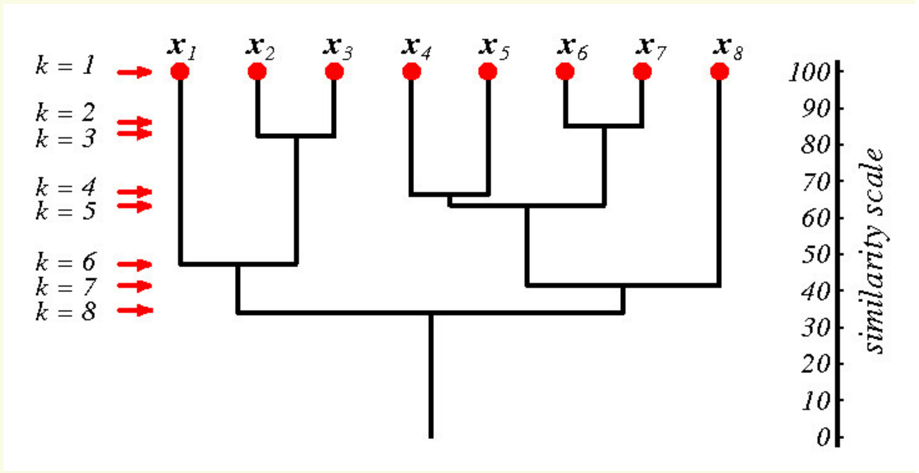


Hierarchical Clustering: Biological Taxonomy



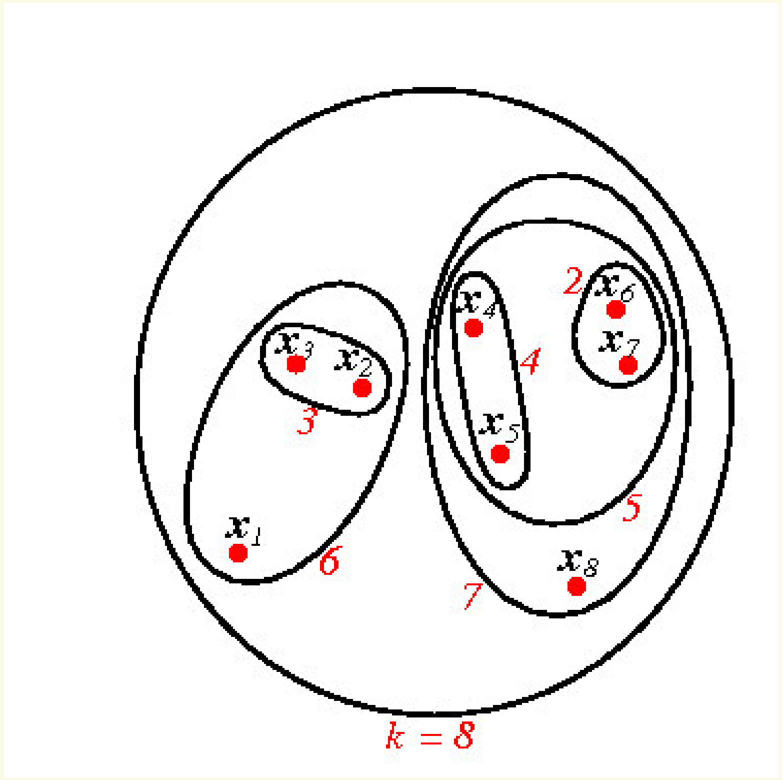
Hierarchical Clustering: Dendogram

- prefered way to represent a hierarchical clustering is a dendogram
  - Binary tree
  - Level ***k*** corresponds to partitioning with ***n-k+1*** clusters
  - if need ***k*** clusters, take clustering from level ***n-k+1***
  - If samples are in the same cluster at level ***k***, they stay in the same cluster at higher levels
  - dendogram typically shows the similarity of grouped clusters



Hierarchical Clustering: Venn Diagram

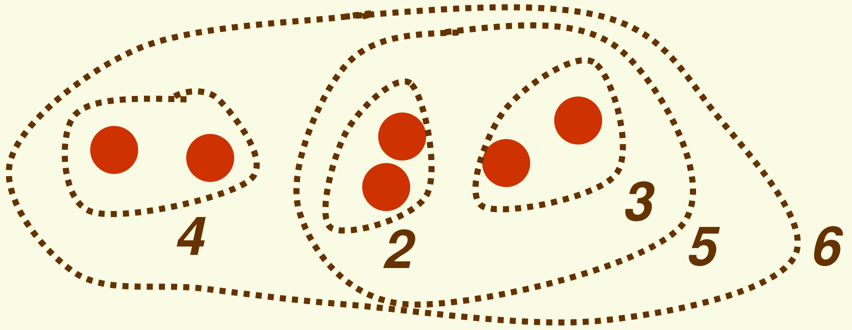
- Can also use Venn diagram to show hierarchical clustering, but similarity is not represented quantitatively



Hierarchical Clustering

- Algorithms for hierarchical clustering can be divided into two types:

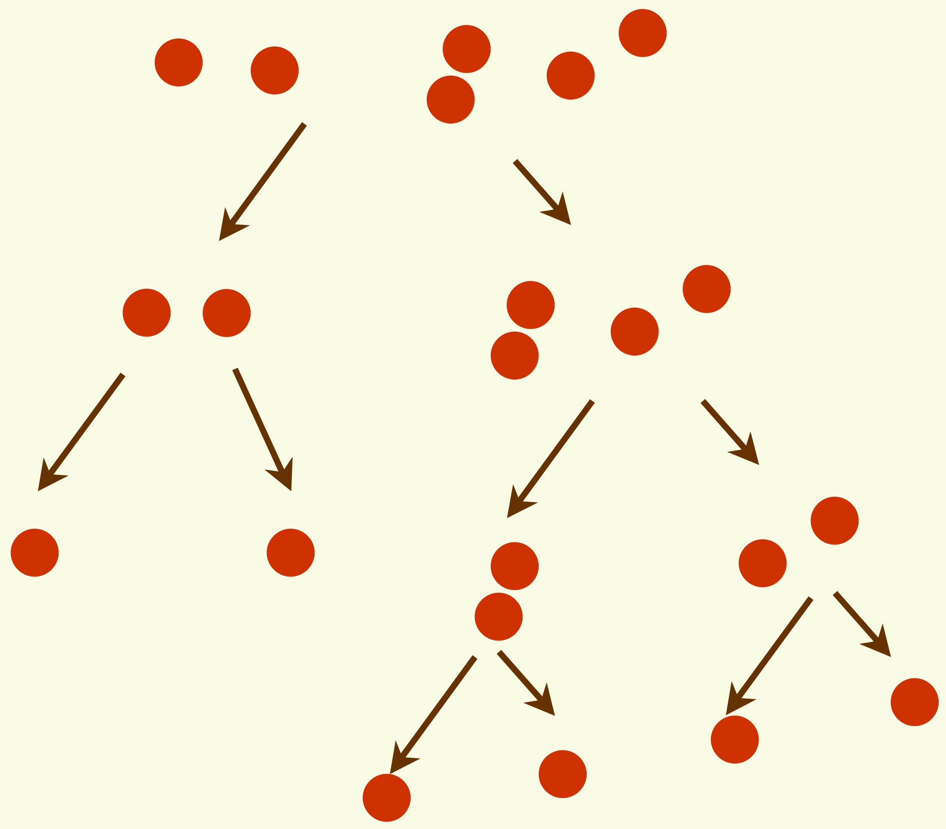
1. Agglomerative (bottom up) procedures
   - Start with ***n*** singleton clusters
   - Form hierarchy by merging most similar clusters



2. Divisive (top bottom) procedures
   - Start with all samples in one cluster
   - Form hierarchy by splitting the “worst” clusters

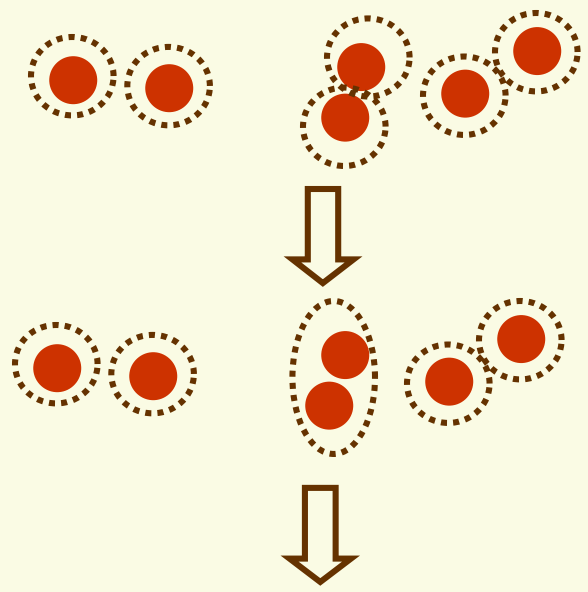
Divisive Hierarchical Clustering

- Any "flat" algorithm which produces a fixed number of clusters can be used
  - set ***c*** = ***2***



Agglomerative Hierarchical Clustering

```
initialize with each example in
singleton cluster
while there is more than 1 cluster
    1. find 2 nearest clusters
    2. merge them
```



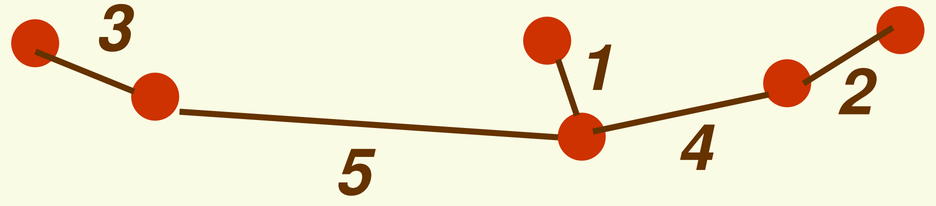
- Four common ways to measure cluster distance

1. minimum distance $d_{\min}(D_i, D_j) = \min_{x \in D_i, y \in D_j} \| x - y \|$
2. maximum distance $d_{\max}(D_i, D_j) = \max_{x \in D_i, y \in D_j} \| x - y \|$
3. average distance $d_{avg}(D_i, D_j) = \frac{1}{n_i n_j} \sum_{x \in D_i} \sum_{y \in D_j} \| x - y \|$
4. mean distance $d_{mean}(D_i, D_j) = \| \mu_i - \mu_j \|$

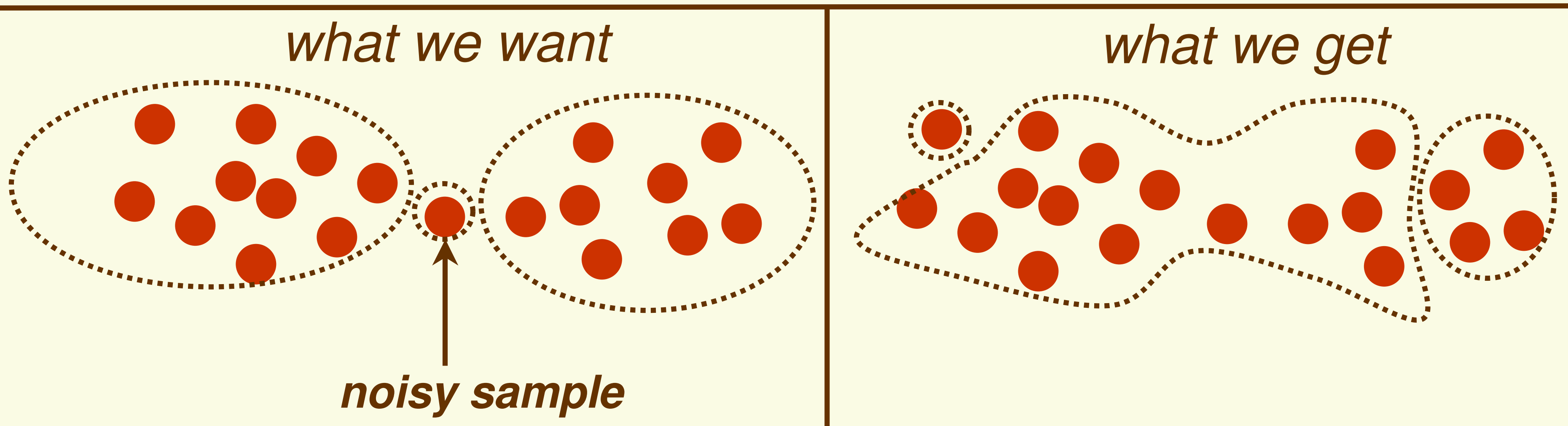
Single Linkage or Nearest Neighbor

- Agglomerative clustering with minimum distance

$$d_{\min}(D_i, D_j) = \min_{x \in D_i, y \in D_j} \| x - y \|$$



- generates minimum spanning tree
- encourages growth of elongated clusters
- disadvantage: very sensitive to noise

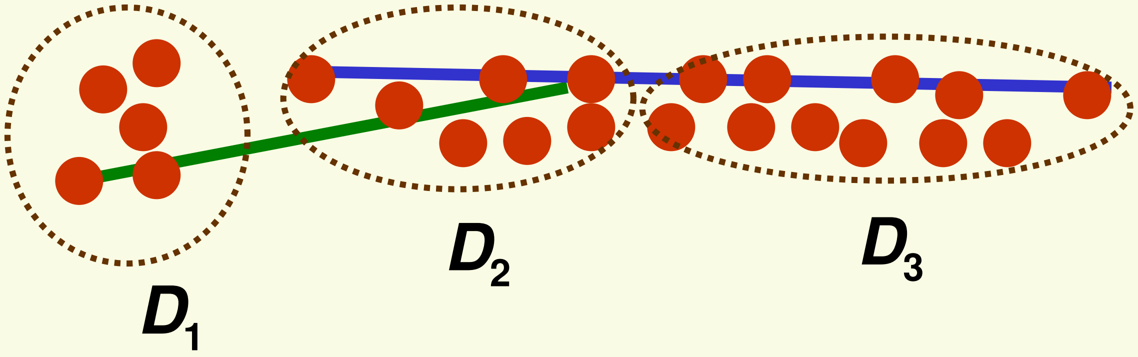


Complete Linkage or Farthest Neighbor

- Agglomerative clustering with maximum distance

$$d_{\max}(D_i, D_j) = \max_{x \in D_i, y \in D_j} \| x - y \|$$

- Encourages compact clusters
- Does not work well if elongated clusters present



  - $d_{\max}(D_1, D_2) < d_{\max}(D_2, D_3)$
  - thus D_1 and D_2 are merged instead of D_2 and D_3

Average and Mean Agglomerative Clustering

- Agglomerative clustering is more robust under the average or the mean cluster distance

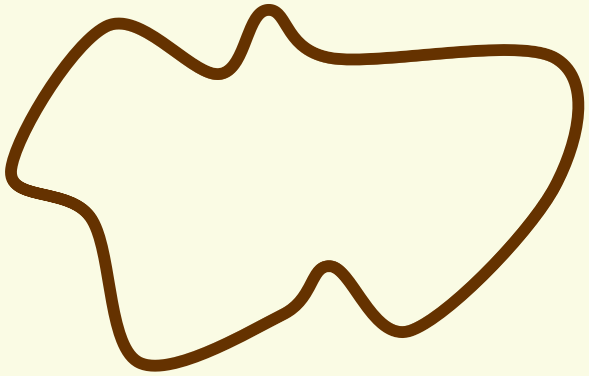
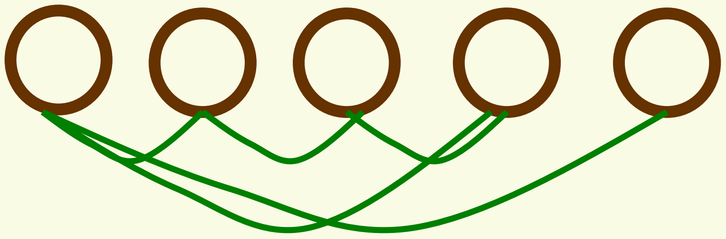
$$d_{avg}(D_i, D_j) = \frac{1}{n_i n_j} \sum_{x \in D_i} \sum_{y \in D_j} || x - y ||$$

$$d_{mean}(D_i, D_j) = || \mu_i - \mu_j ||$$

- mean distance is cheaper to compute than the average distance
- unfortunately, there is not much to say about agglomerative clustering theoretically, but it does work reasonably well in practice

Agglomerative vs. Divisive

- Agglomerative is faster to compute, in general
- Divisive may be less “blind” to the global structure of the data

Divisive	***Agglomerative***
when taking the first step (split), have access to all the data; can find the best possible split in 2 parts	when taking the first step merging, do not consider the global structure of the data, only look at pairwise structure
	

Summary

- Clustering (nonparametric learning) is useful for discovering inherent structure in data
- Clustering is immensely useful in different fields
- Clustering comes naturally to humans (in up to 3 dimensions), but not so to computers
- It is very easy to design a clustering algorithm, but it is very hard to say if it does anything good
- General purpose clustering is unlikely to exist, for best results, clustering should be tuned to application at hand