Clustering

Huanle Xu

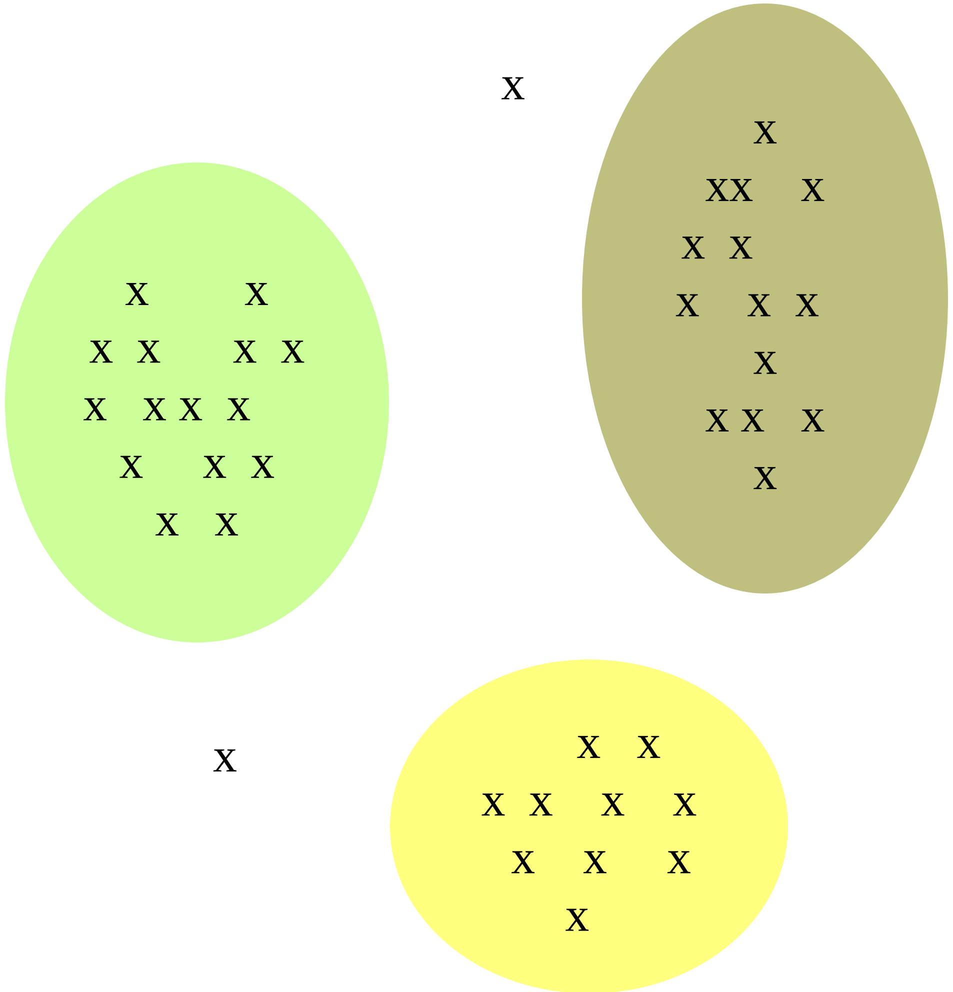
High Dimensional Data

- Given a cloud of data points we want to understand their structure

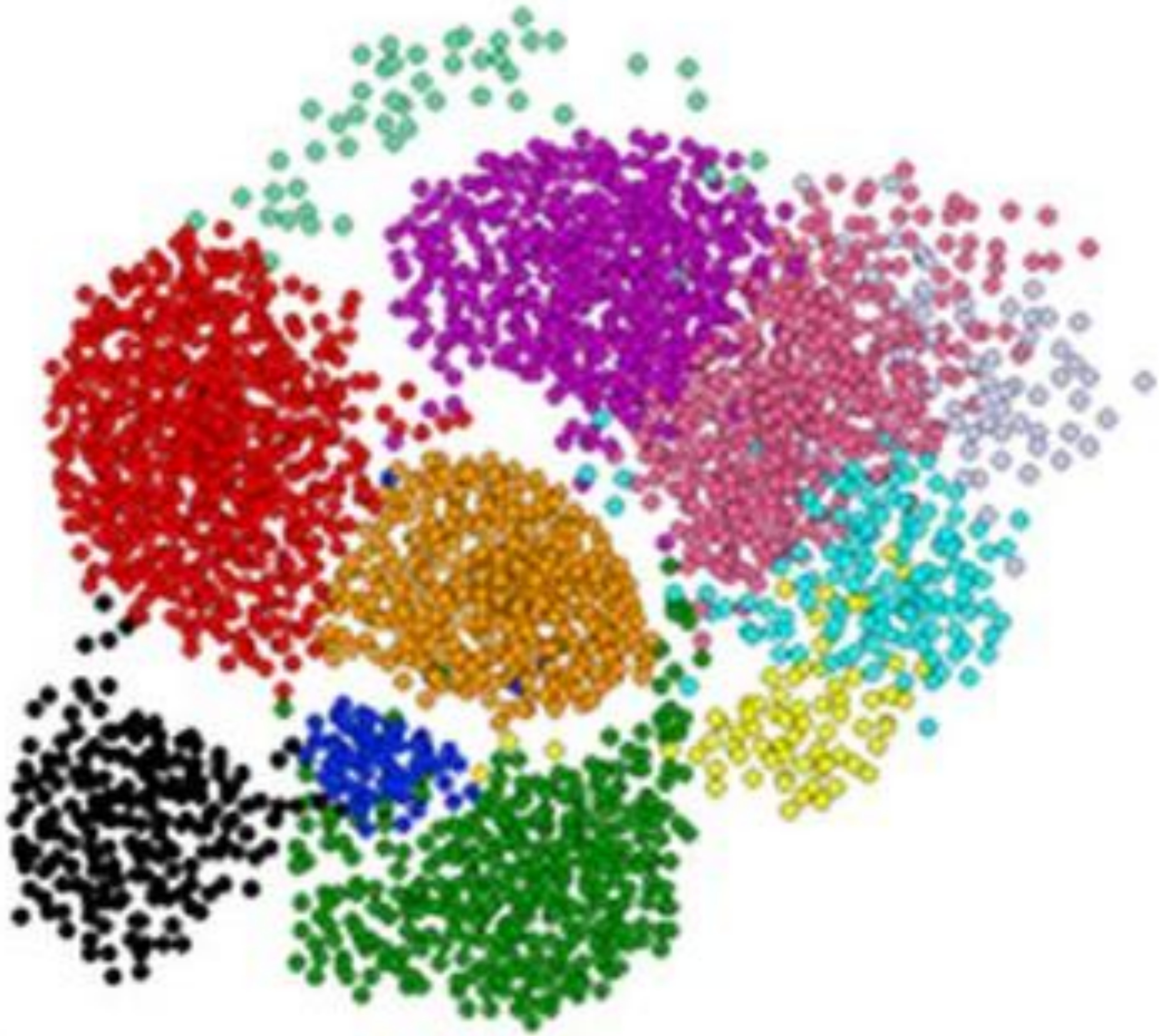
The Problem of Clustering

- Given a **set of points**, with a notion of **distance** between points, **group the points** into some number of ***clusters***, so that
  - Members of a cluster are close/similar to each other
  - Members of different clusters are dissimilar
- **Usually:**
  - Points are in a high-dimensional space
  - Similarity is defined using a distance measure
    - Euclidean, Cosine, Jaccard, edit distance, …

Example: Clusters



Clustering is a hard problem!



Example: Clustering CD's

- **Intuitively: Music divides into categories, and customers prefer a few categories**
  - But what are categories really?
- Represent a CD by a set of customers who bought it
- Similar CDs have similar sets of customers, and vice-versa

Example: Clustering CDs

Space of all CDs:

- Think of a space with one dim. for each customer
  - Values in a dimension may be 0 or 1 only
  - A CD is a point in this space is $(x_1, x_2, \ldots, x_k)$, where $x_i = 1$ iff the i^{th} customer bought the CD
    - Compare with boolean matrix: rows = customers; cols. = CDs
- For Amazon, the dimension is tens of millions
- **Task:** Find clusters of similar CDs

- **An alternative:** Use Minhash/LSH to get Jaccard distance between "close" CDs
- Use that as input to clustering

Example: Clustering Documents

Finding topics:

- Represent a document by a vector $(x_1, x_2,\ldots, x_k)$, where $x_i = 1$ iff the i th word (in some order) appears in the document
  - It actually doesn't matter if k is infinite; i.e., we don't limit the set of words

- **Documents with similar sets of words may be about the same topic**

k-means clustering

k–means Algorithm(s)

- Assumes Euclidean space/distance
- Start by picking ***k***, the number of clusters
- Initialize clusters by picking one point per cluster
  - **Example:** Pick one point at random, then ***k*-1** other points, each as far away as possible from the previous points

Populating Clusters

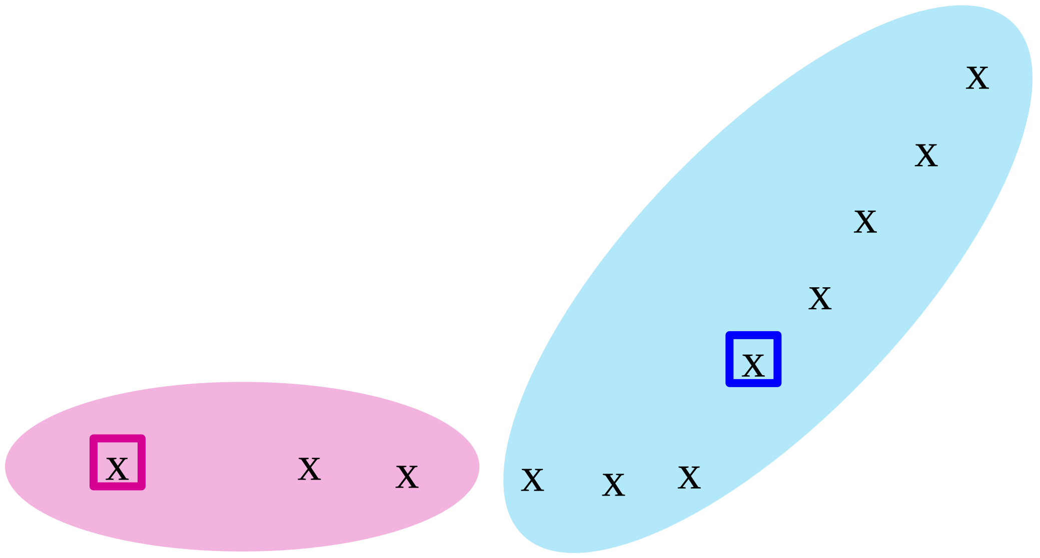
- **1)** For each point, place it in the cluster whose current centroid it is nearest

- **2)** After all points are assigned, update the locations of centroids of the ***k*** clusters

- **3)** Reassign all points to their closest centroid
  - Sometimes moves points between clusters

- **Repeat 2 and 3 until convergence**
  - **Convergence:** Points don't move between clusters and centroids stabilize

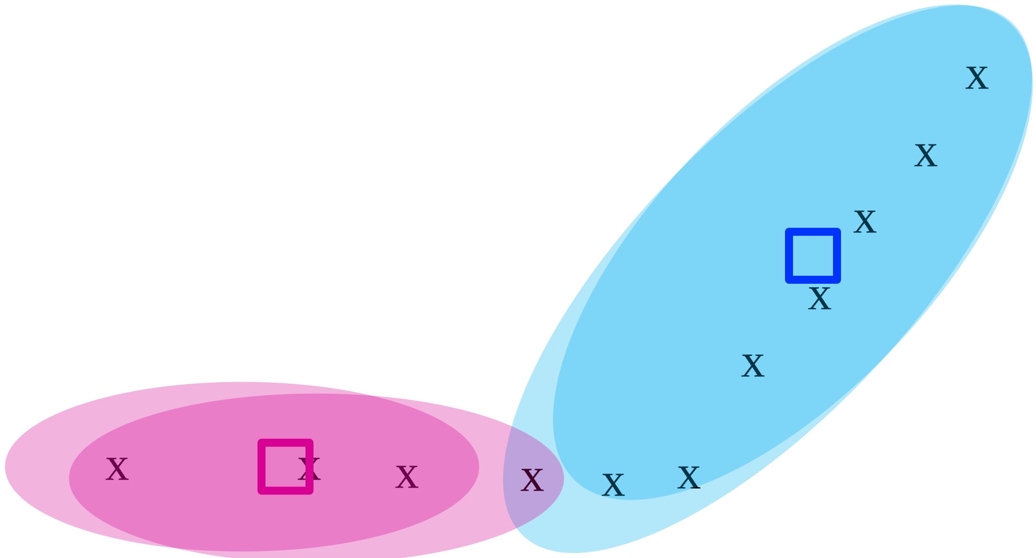
Example: Assigning Clusters



x … data point
… centroid

Clusters after round 1

Example: Assigning Clusters

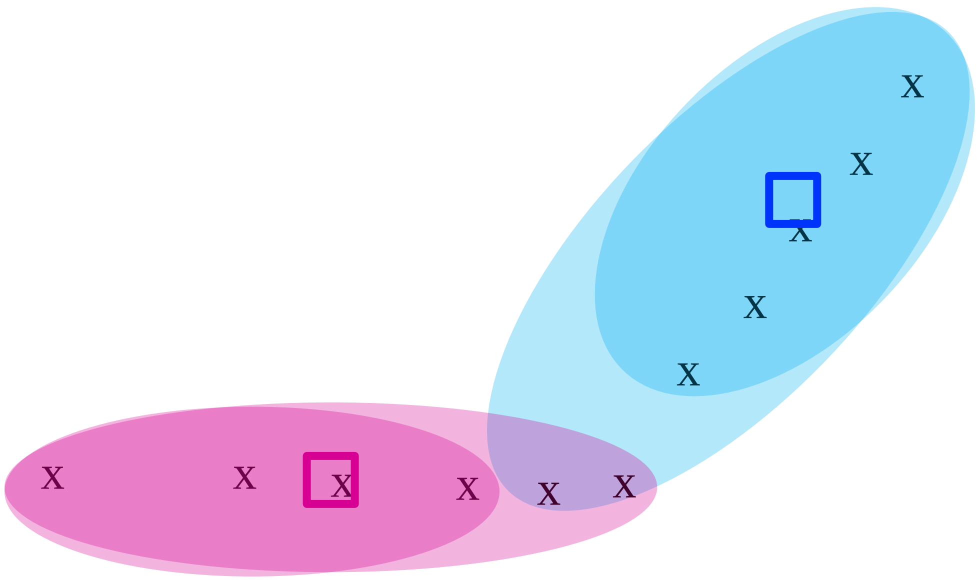


x … data point

… centroid

Clusters after round 2

Example: Assigning Clusters



x … data point

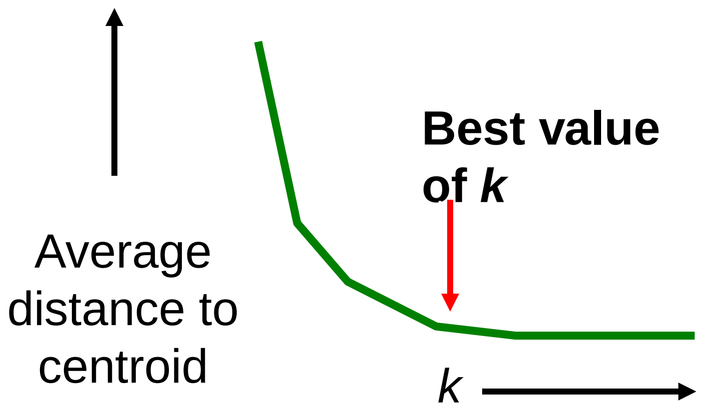
… centroid

Clusters at the end

Getting the *k* right

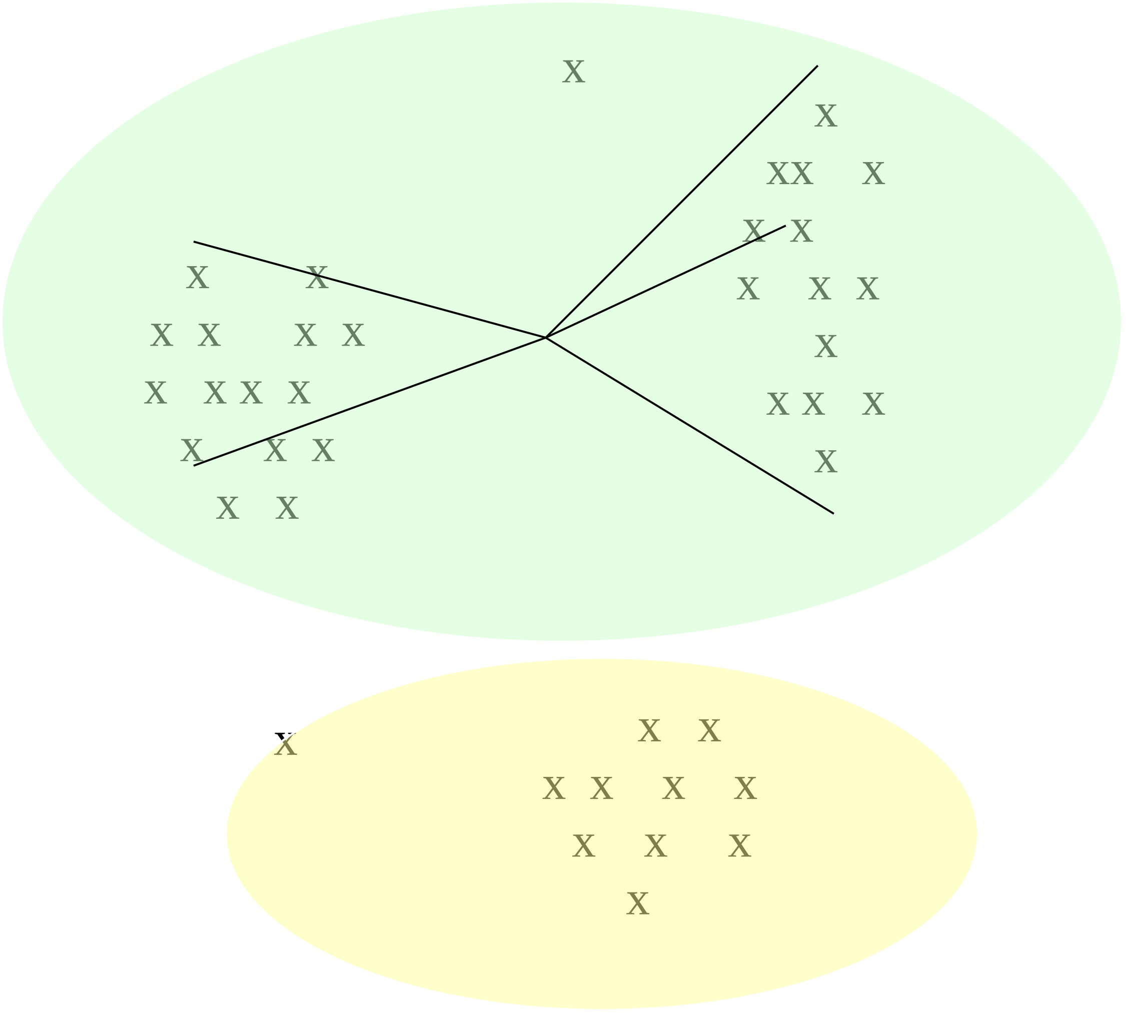
How to select *k*?

- Try different *k*, looking at the change in the average distance to centroid, as *k* increases.
- Average falls rapidly until right *k*, then changes little



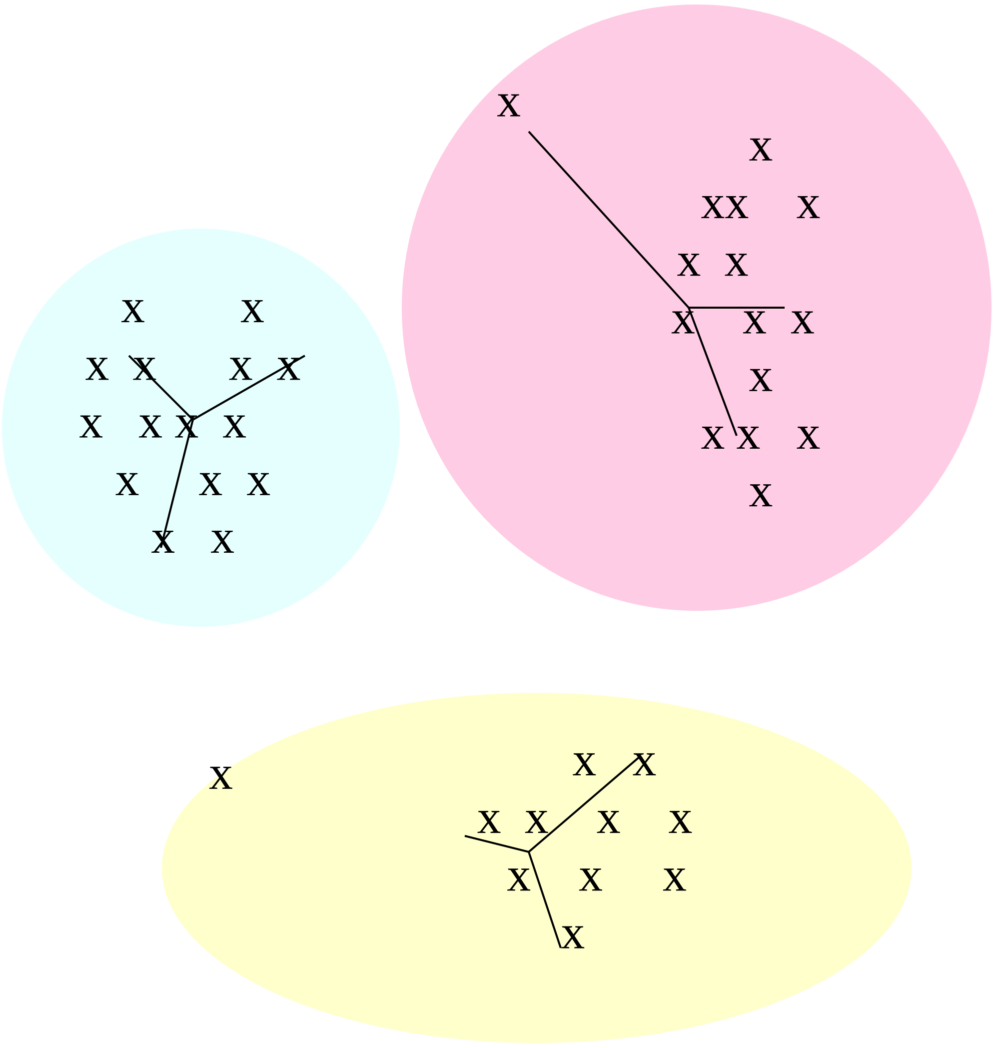
Example: Picking *k*

Too few;
many long
distances
to centroid.



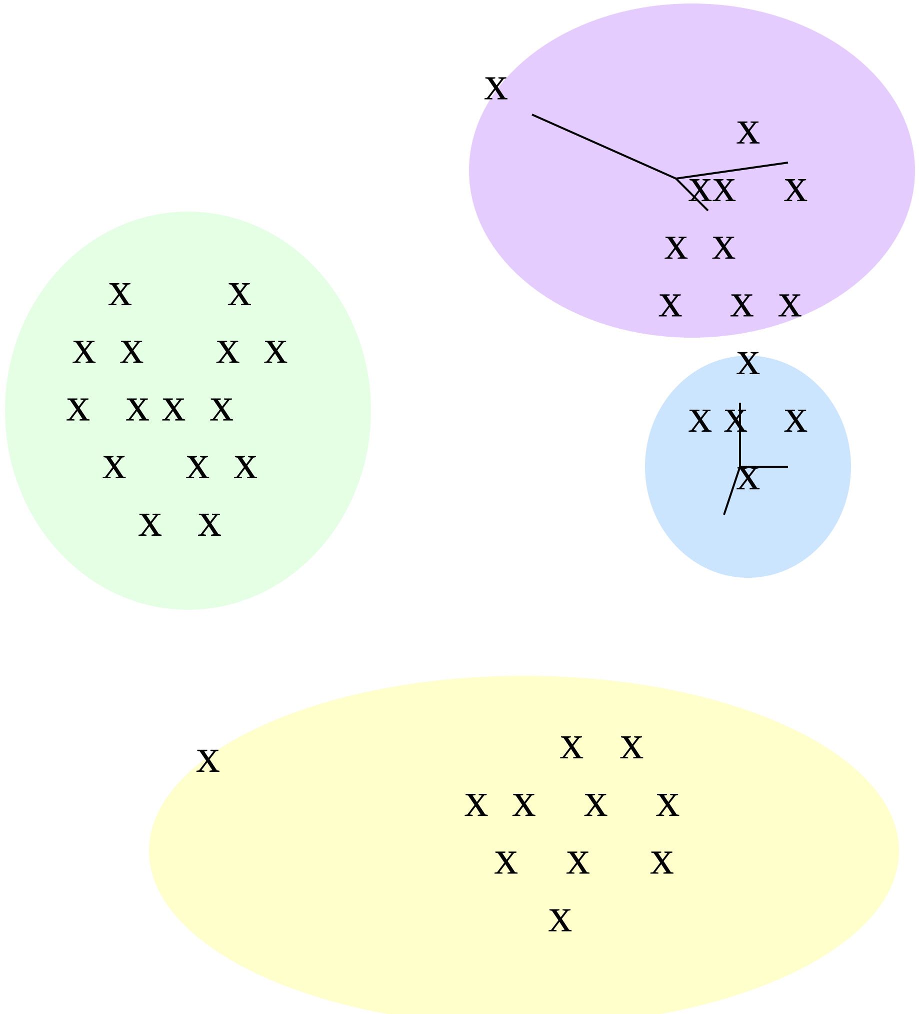
Example: Picking *k*

Just right;
distances
rather short.



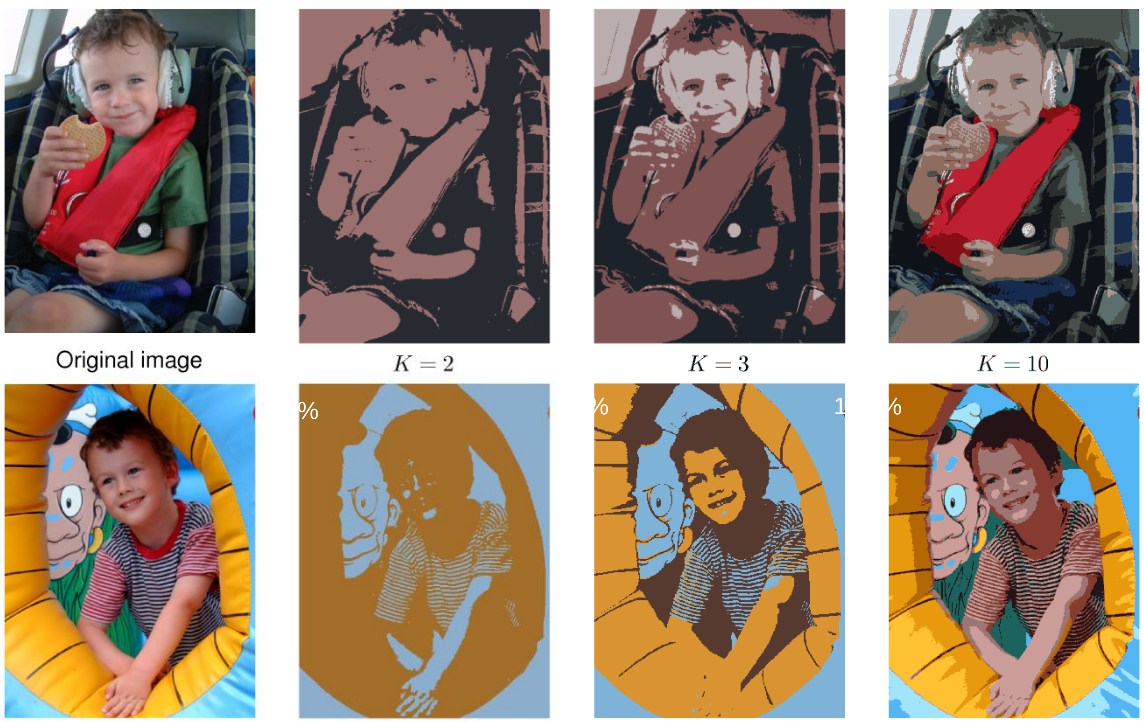
Example: Picking *k*

Too many;
little improvement
in average
distance.



Examples of k-means clustering

- Clustering RGB vectors of pixels in images
- Compression of image file: N x 24 bits
  - Store RGB values of cluster centers: K x 24 bits
  - Store cluster index of each pixel: N x log K bits



Limitations of K-means

- Need to determine "K" via domain knowledge or heuristics (as stated before)
- Only converge to local optimal
  - Need to try multiple starting points
- "Hard" assignment of each data point to a single cluster:
  - Each data point can only be assigned to 1 cluster (class)
  - What about points that lie in between groups ? e.g. Jazz + Classical
- Overall results can be affected by a few Outliners

Can we do better ?

Comparing to the K-means algorithm

1. Initialize means μ_k
   2. E Step: Assign each point to a cluster

   3. M Step: Given clusters, refine mean μ_k of each cluster k
4. Stop when change in means is small

Application: Using GMM for Image Segmentation

Source:
https://kittipatkampa.wordpress.com/2011/02/17/image-segmentation-using-gaussian-mixture-models/

Original Image

Segmentation results using GMM with 3 components

Input Features:

x-y pixel locations & pixel lightness/color in L*a*b color space

Output Results:

Each color represents a class ; The brightness represents the posterior probability – darker pixels represent high uncertainty of the posterior distribution.