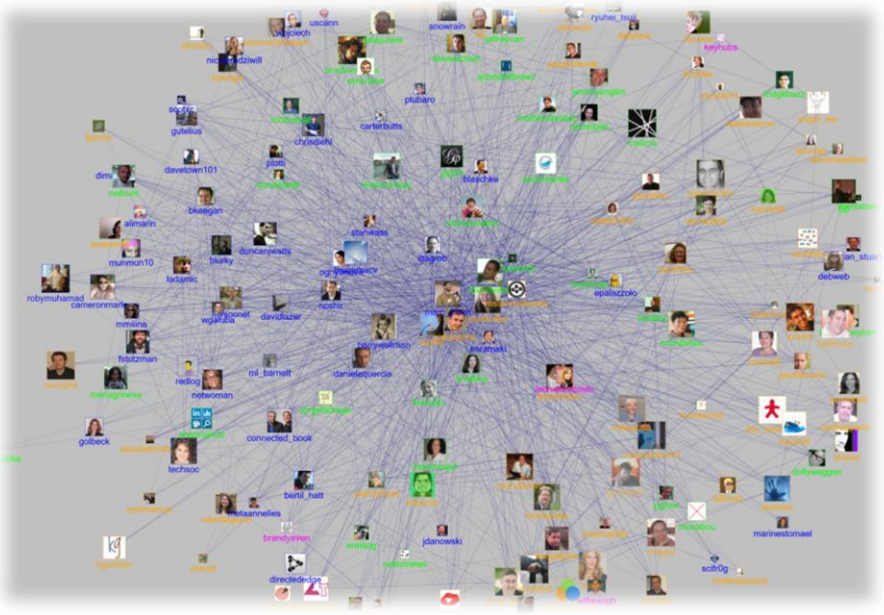
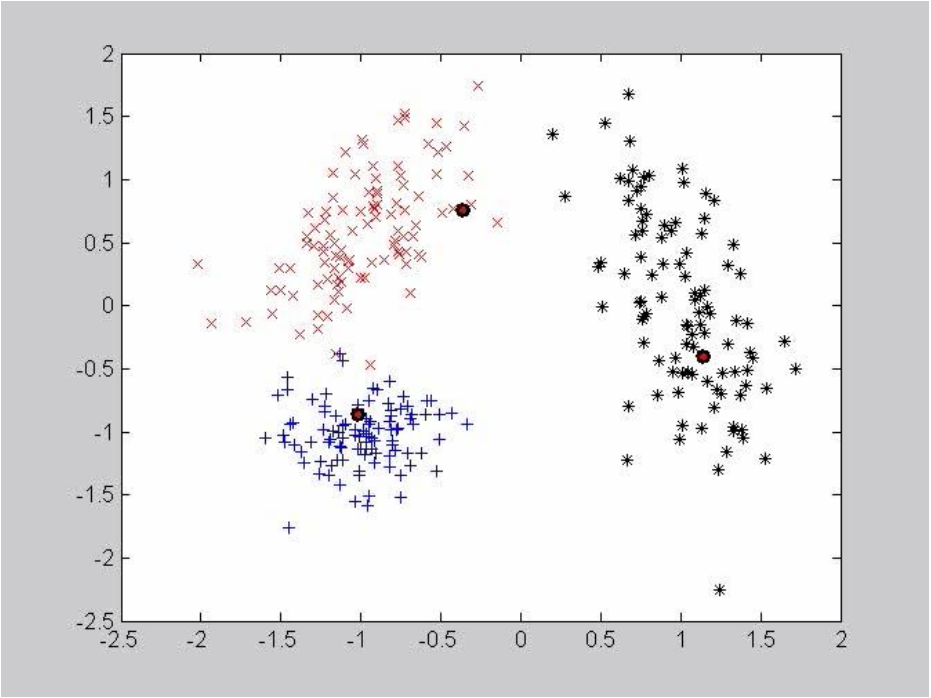
Scalable Clustering

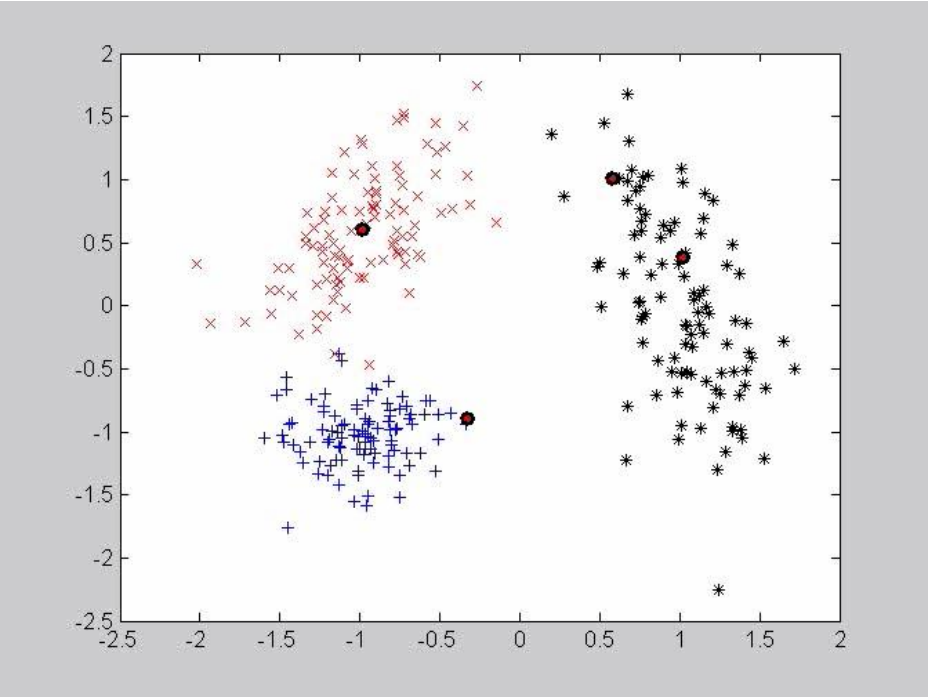
Motivation

- Large-scale and high dimensional data
  - Social networks
  - Images
  - Web documents
  - …
- Understand the structure of data



Motivation





- Traditional clustering methods (e.g., K-means)
  - Takes many iterations to converge
  - Very sensitive to initialization

Outline

- Introduction
- Hierarchical Clustering
- K-means
- Scalable K-means
  - K-means||

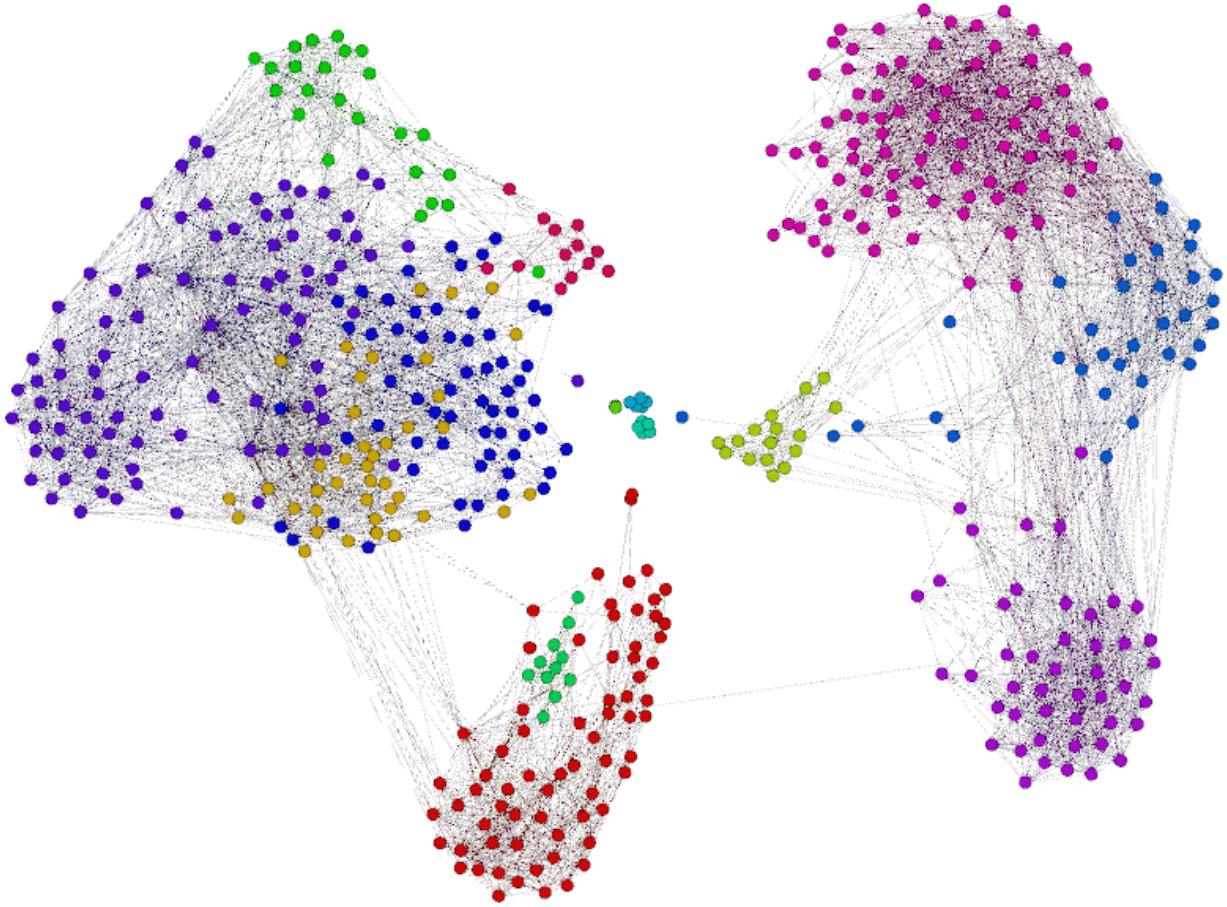
Outline

- Introduction
- Hierarchical Clustering
- K-means
- Scalable K-means
  - K-means||

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, …

Clustering is a Hard Problem!



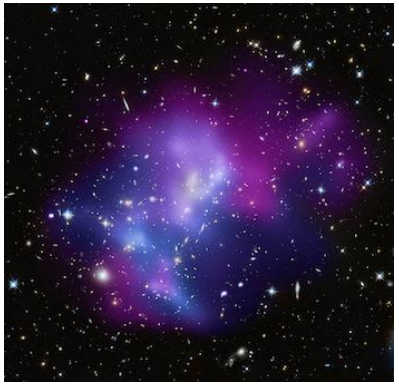
Why is it Hard?

- Clustering in two dimensions looks easy
- Clustering small amounts of data looks easy
- And in most cases, looks are not deceiving

- Many applications involve not 2, but 10 or 10,000 dimensions
- High-dimensional spaces look different:
  - Almost all pairs of points are at about the same distance

Clustering Problem: SkyCat

- A catalog of 2 billion "sky objects" represents objects by their radiation in 7 dimensions (frequency bands)
- Problem: Cluster into similar objects, e.g., galaxies, nearby stars, quasars, etc.
- Sloan Digital Sky Survey is a newer, better version of this



Example: Clustering CDs

- 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,…, 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

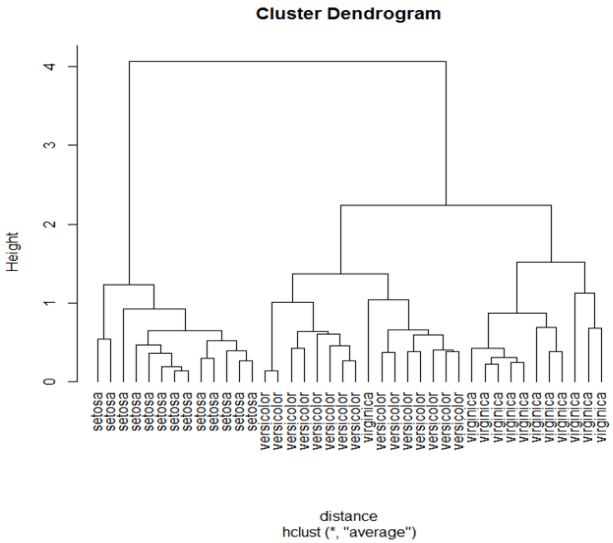
- Documents with similar sets of words may be about the same topic

Cosine, Jaccard, and Euclidean

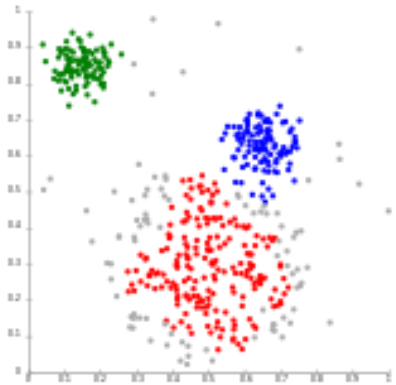
- As with CDs we have a choice when we think of documents as sets of words or shingles:
  - Sets as vectors: measure similarity by the cosine distance
  - Sets as sets: measure similarity by the Jaccard distance
  - Sets as points: measure similarity by Euclidean distance

Overview: Methods of Clustering

- Hierarchical:
  - Agglomerative (bottom up):
    - Initially, each point is a cluster
    - Repeatedly combine the two “nearest” clusters into one
  - Divisive (top down):
    - Start with one cluster and recursively split it



- Point assignment:
  - Maintain a set of clusters
  - Points belong to “nearest” cluster



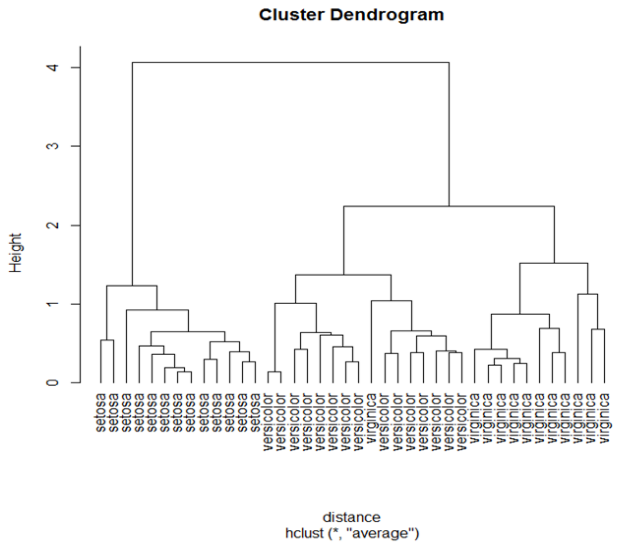
Outline

- Introduction
- Hierarchical Clustering
- K-means
- Scalable K-means
  - K-means||

Hierarchical Clustering

- Key operation:

  Repeatedly combine two nearest clusters



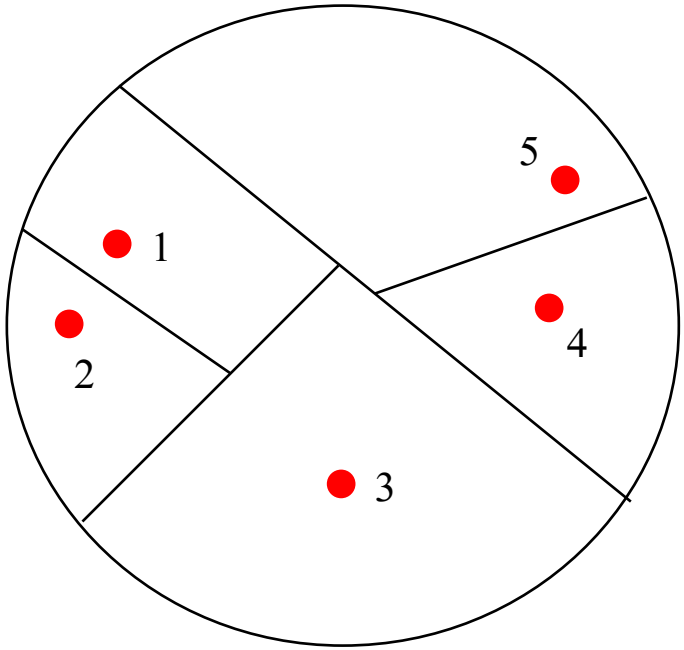
- Three important questions:
  a) How do you represent a cluster of more than one point?
  b) How do you determine the “nearness” of clusters?
  c) When to stop combining clusters?

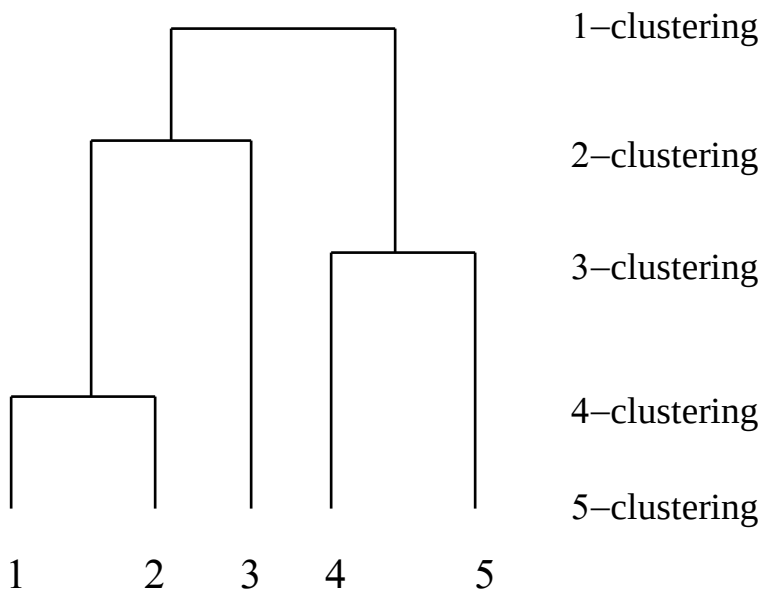
Hierarchical Clustering

- Key operation: Repeatedly combine two nearest clusters
- (1) How to represent a cluster of many points?
  - Key problem: As you build clusters, how do you represent the location of each cluster, to tell which pair of clusters is closest?
- Euclidean case: each cluster has a centroid = average of its (data)points
- (2) How to determine "nearness" of clusters?
  - Measure cluster distances by distances of centroids

Hierarchical Clustering

- Recursive partitioning of a data set





And in the Non-Euclidean Case?

What about the Non-Euclidean case?

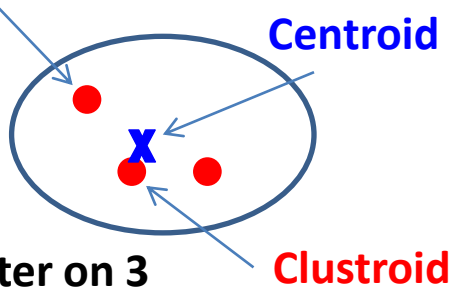
- The only "locations" we can talk about are the points themselves
  - i.e., there is no "average" of two points

- Approach 1:
  - (1) How to represent a cluster of many points?
    clustroid = (data)point "closest" to other points
  - (2) How do you determine the "nearness" of clusters?
    Treat clustroid as if it were centroid, when computing intercluster distances

"Closest" Point?

- (1) How to represent a cluster of many points?
  clustroid = point "closest" to other points
- Possible meanings of "closest":
  - Smallest maximum distance to other points
  - Smallest average distance to other points
  - Smallest sum of squares of distances to other points
    - For distance metric ***d*** clustroid ***c*** of cluster ***C*** is: $\min_c \sum_{x \in C} d(x,c)^2$

Data point



A cluster on 3 data points

Centroid is the avg. of all (data) points in the cluster. This means centroid is an "artificial" point.
Clustroid is an **existing** (data) point that is "closest" to all other points in the cluster.

Defining “Nearness” of Clusters

- (2) How do you determine the “nearness” of clusters?
  - Approach 2:

    Intercluster distance = minimum of the distances between any two points, one from each cluster
  - Approach 3:

    Pick a notion of “cohesion” of clusters, *e.g.*, maximum distance from the clustroid
    - Merge clusters whose union is most cohesive

Cohesion

- Approach 3.1: Use the diameter of the merged cluster = maximum distance between points in the cluster
- Approach 3.2: Use the average distance between points in the cluster
- Approach 3.3: Use a density-based approach
  - Take the diameter or avg. distance, e.g., and divide by the number of points in the cluster
  - Perhaps raise the number of points to a power first, e.g., square-root

Implementation

- Naïve implementation of hierarchical clustering:
  - At each step, compute pairwise distances between all pairs of clusters, then merge
  - $O(N^3)$
- Careful implementation using priority queue can reduce time to $O(N2 \log N)$
  - Still too expensive for really big datasets that do not fit in memory

Outline

- Introduction
- Hierarchical Clustering
- K-means
- Scalable K-means
  - K-means||

K-means Clustering

- Fundamental algorithm in data analysis and machine learning
- “By far the most popular clustering algorithm used in scientific and industrial applications” [Berkhin ’06]
- Identified as one of the top 10 algorithms in data mining [Wu et al ’07]

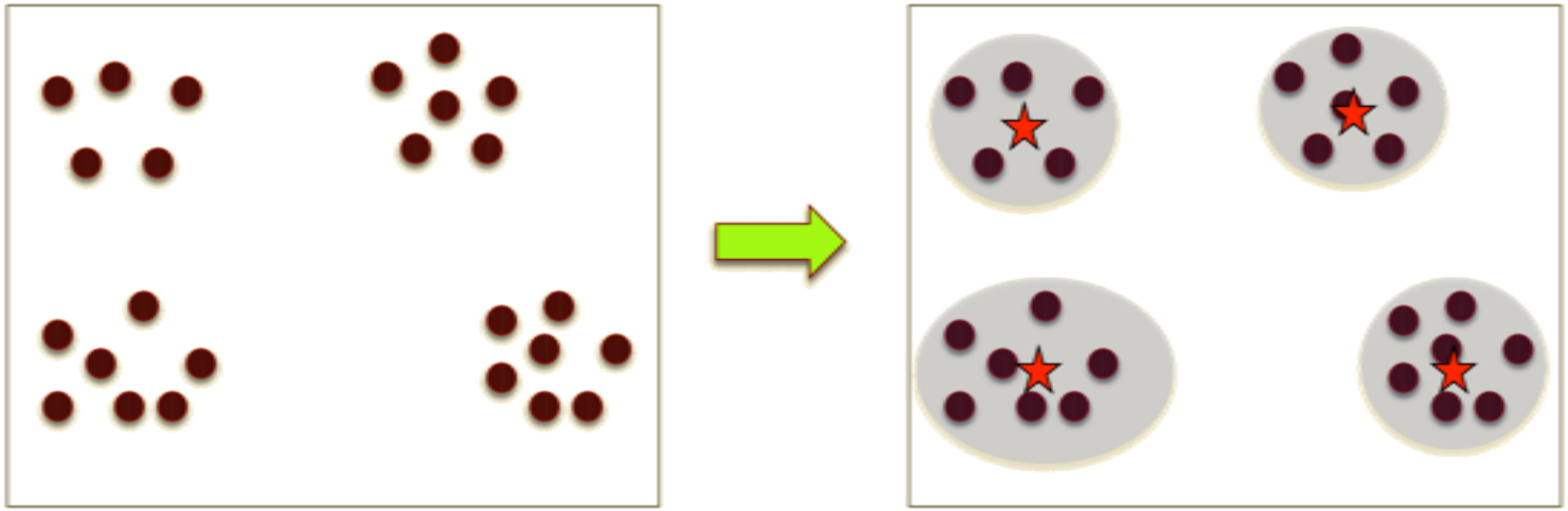
Problem Setting

- Input
  - A set X={x1, x2, …, xn} of n data points
  - Number of clusters k
- For a set C={c1, c2, …, ck} of cluster "centers" define:

$$\varphi_X(C) = \sum_{x \in X} d(x,C)^2$$

  where d(x,C) = distance from x to closest center in C
- Goal: To find a set C of centers that minimizes the objective function $\varphi_x(C)$

K-means Clustering: Example



K=4

K-means Algorithm

- Start with k arbitrary centers {c1, c2, …, ck} (typically chosen uniformly at random from data points)
- Performs an EM-type local search till convergence

Expectation Maximization (EM)

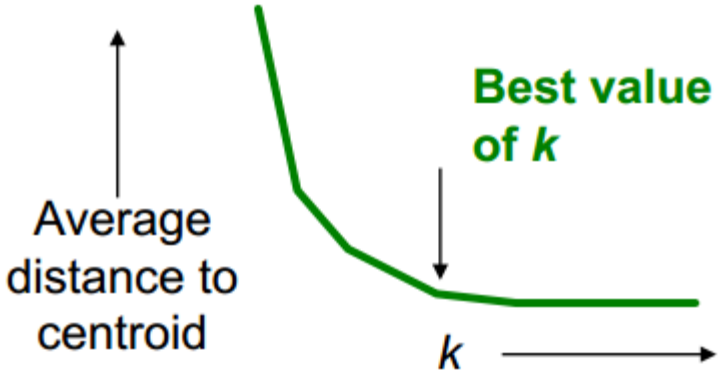
- An EM algorithm is an iterative method for finding maximum likelihood or maximum a posteriori estimates of parameters in statistical models
  - Expectation (E) step
    - Create a function for the expectation of the log-likelihood evaluated using the current estimate for the parameters
  - Maximization (M) step
    - Compute parameters maximizing the expected log-likelihood found on the *E* step

EM in K-means

- E step
  - Keep centers of clusters unchanged, assign points to closest clusters to minimizes the objective function $\varphi_{\mathrm{X}}(C)$
- M step
  - Keep the assignments of points unchanged, re-estimate cluster centers to minimizes the objective function $\varphi_{\mathrm{X}}(C)$

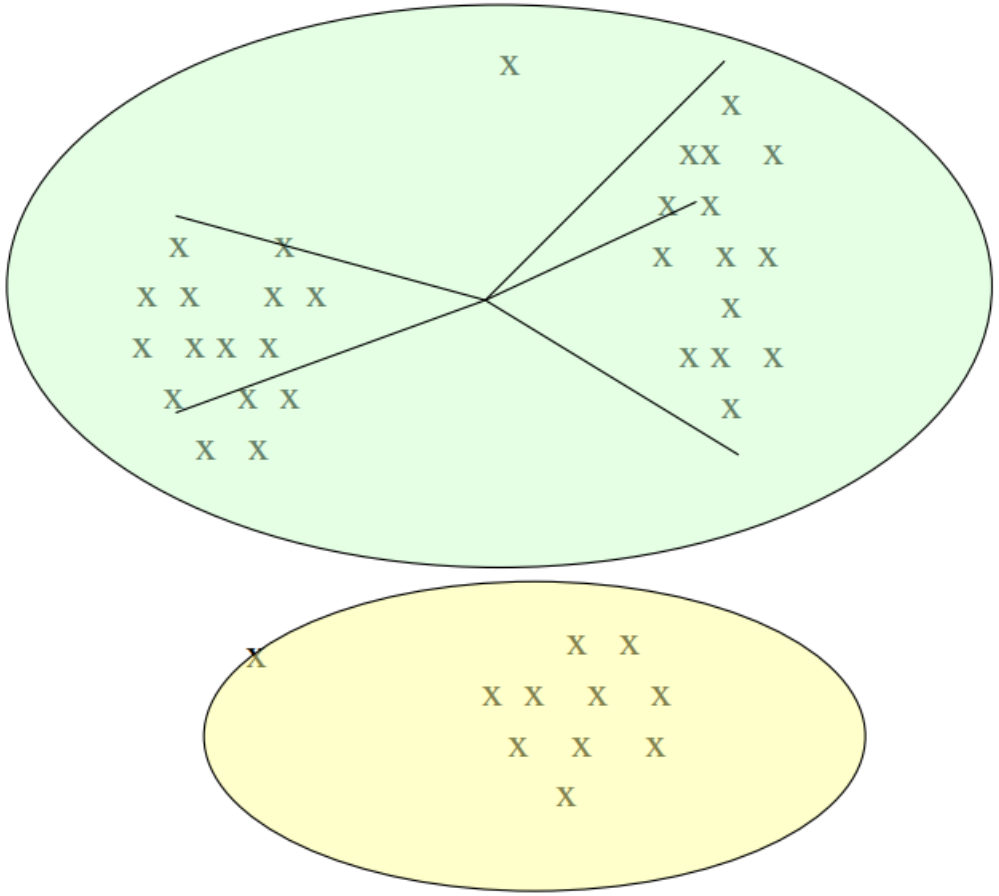
Getting the *k* Right

- 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



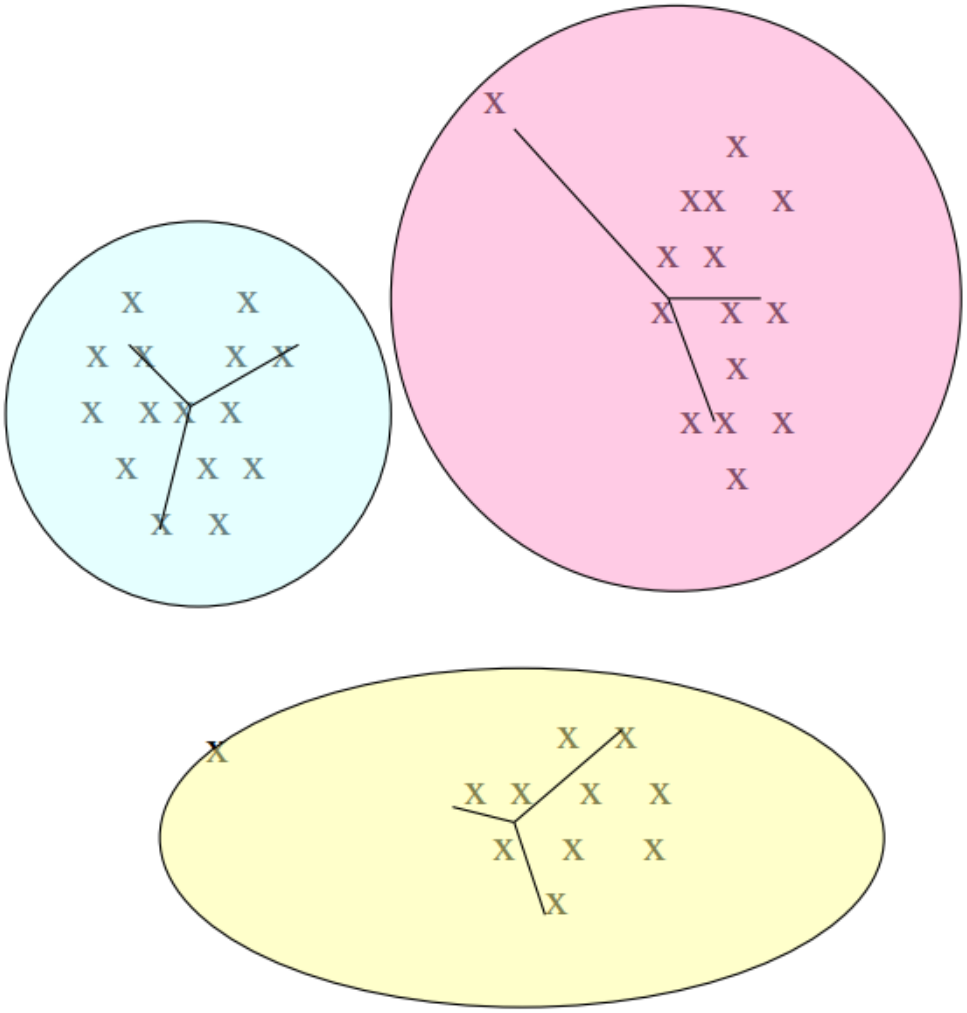
Example: Picking *k*

Too few:
many long
distances
to centroid



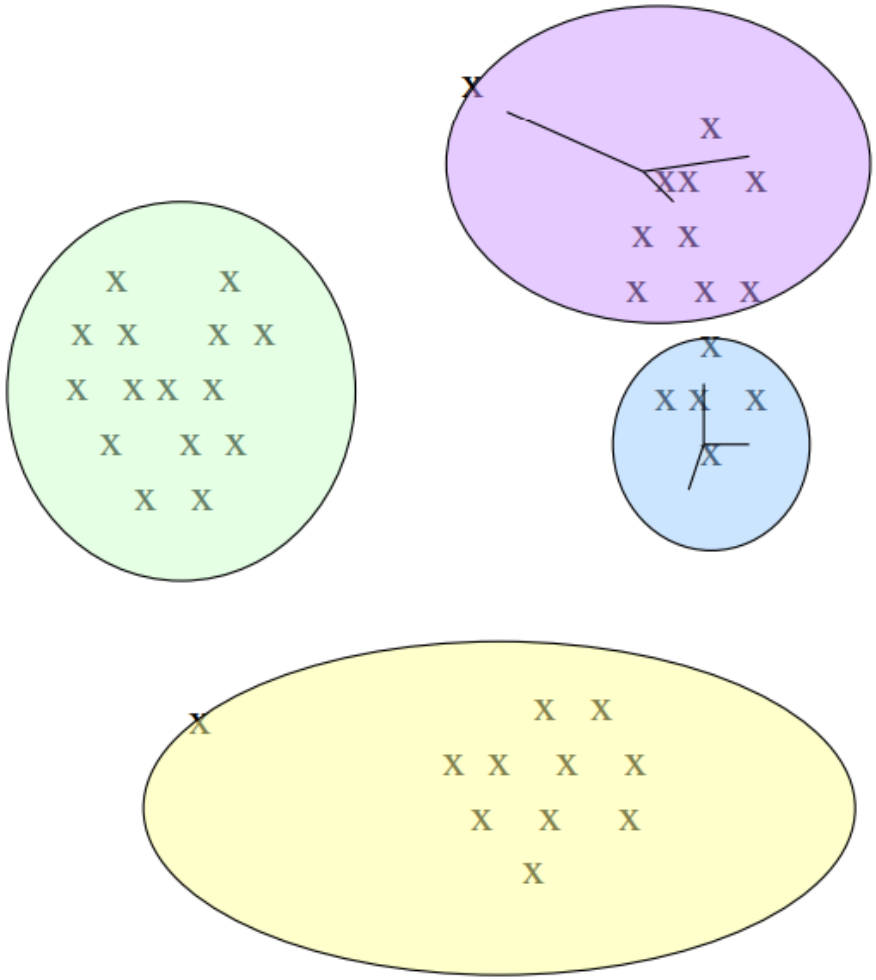
Example: Picking *k*

Just right:
distances
rather short



Example: Picking *k*

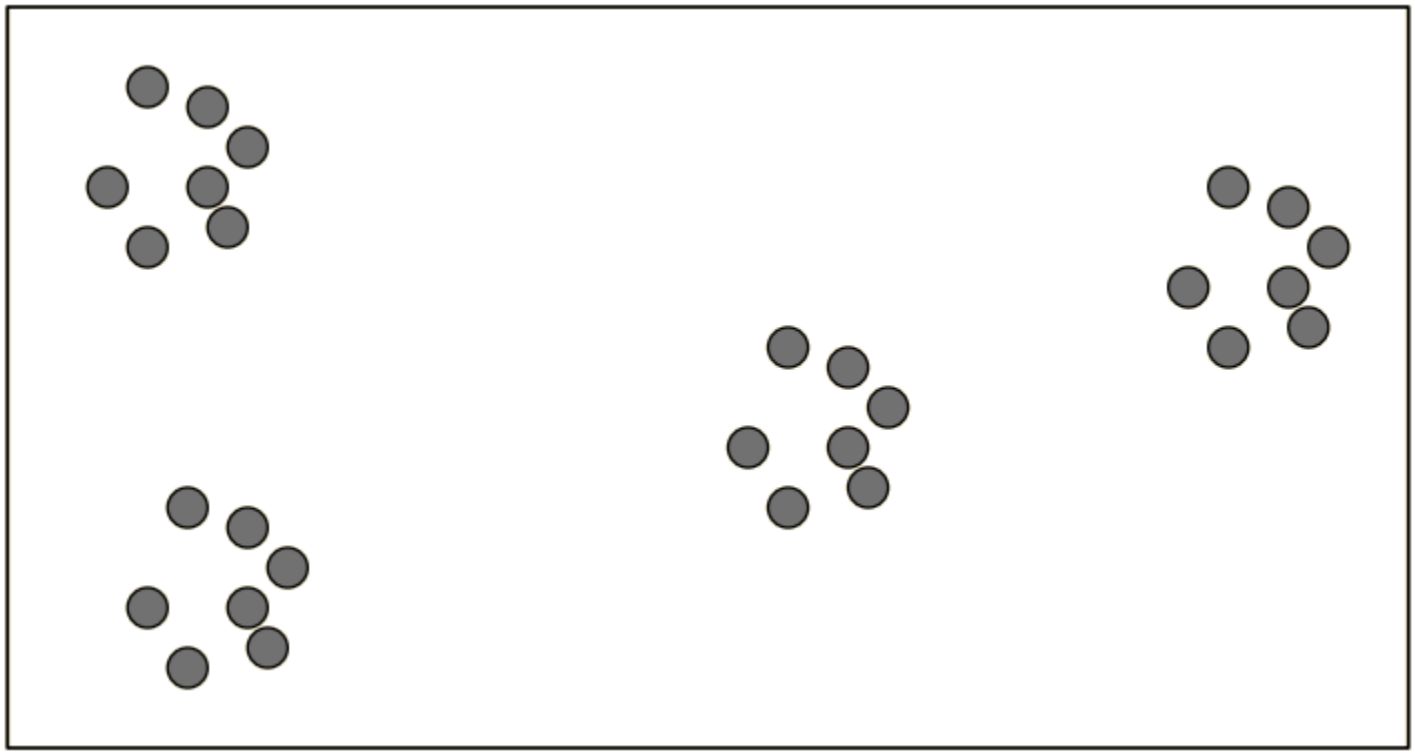
Too many:
little improvement in average distance.



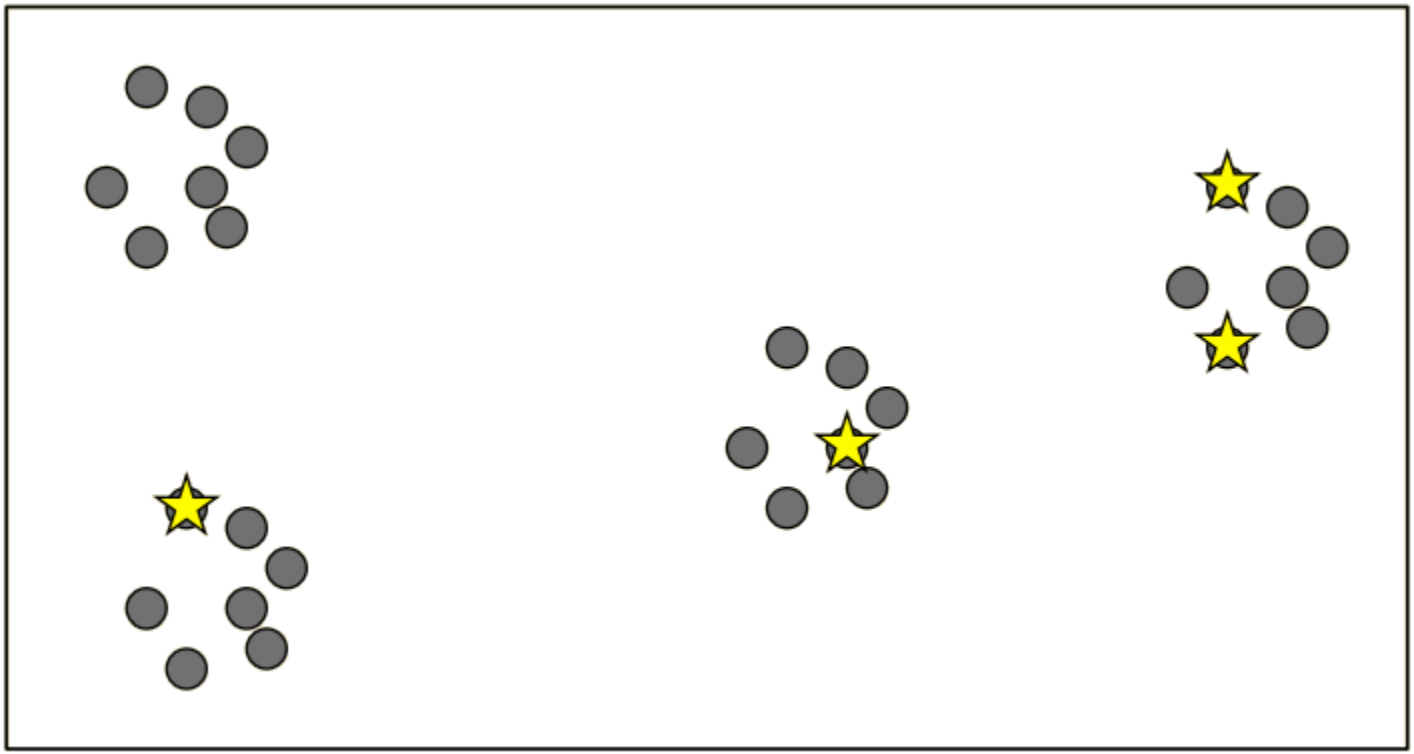
Pros & Cons

- Advantages
  - Simplicity
  - Scalability
- Disadvantages
  - Takes many iterations to converge
  - Very sensitive to initialization
    - Random initialization can easily get two centers in the same cluster
  - Gets stuck in a local optimum

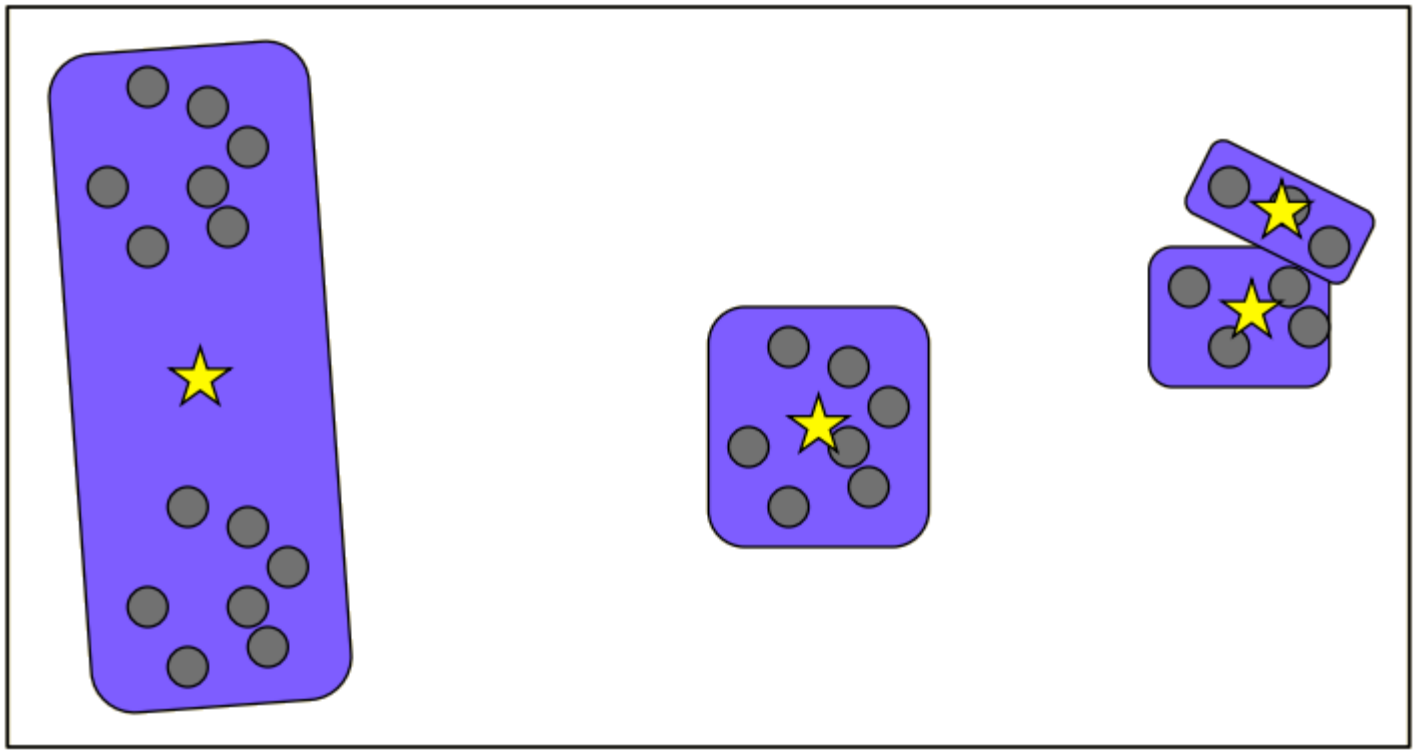
K-means: Initialization



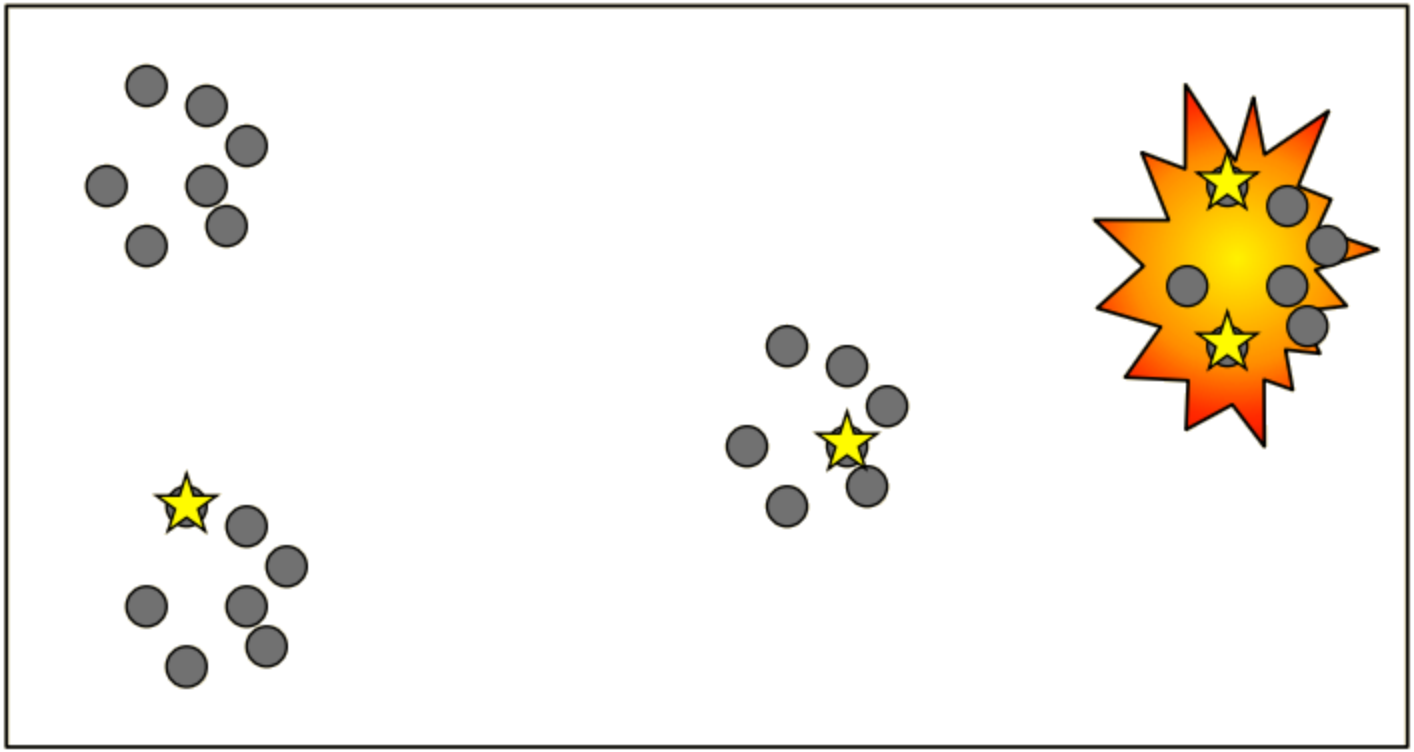
K-means: Initialization



K-means: Initialization



K-means: Initialization



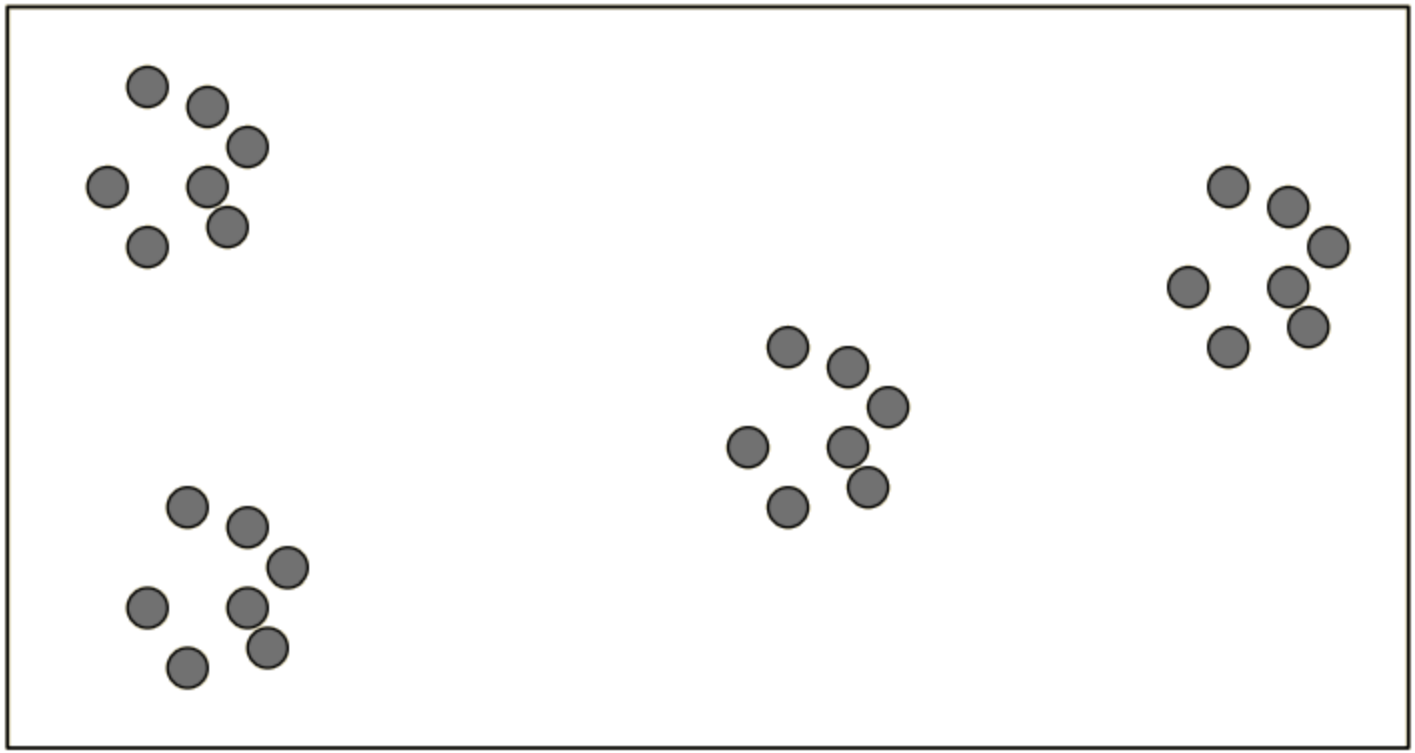
K-means++ [Arthur et al. ’07]

- Spreads out the centers
- Choose first center, c1, uniformly at random from the data set
- Repeat for $2 \leq i \leq k$:
  - Choose ci to be equal to a data point x0 sampled from the distribution:

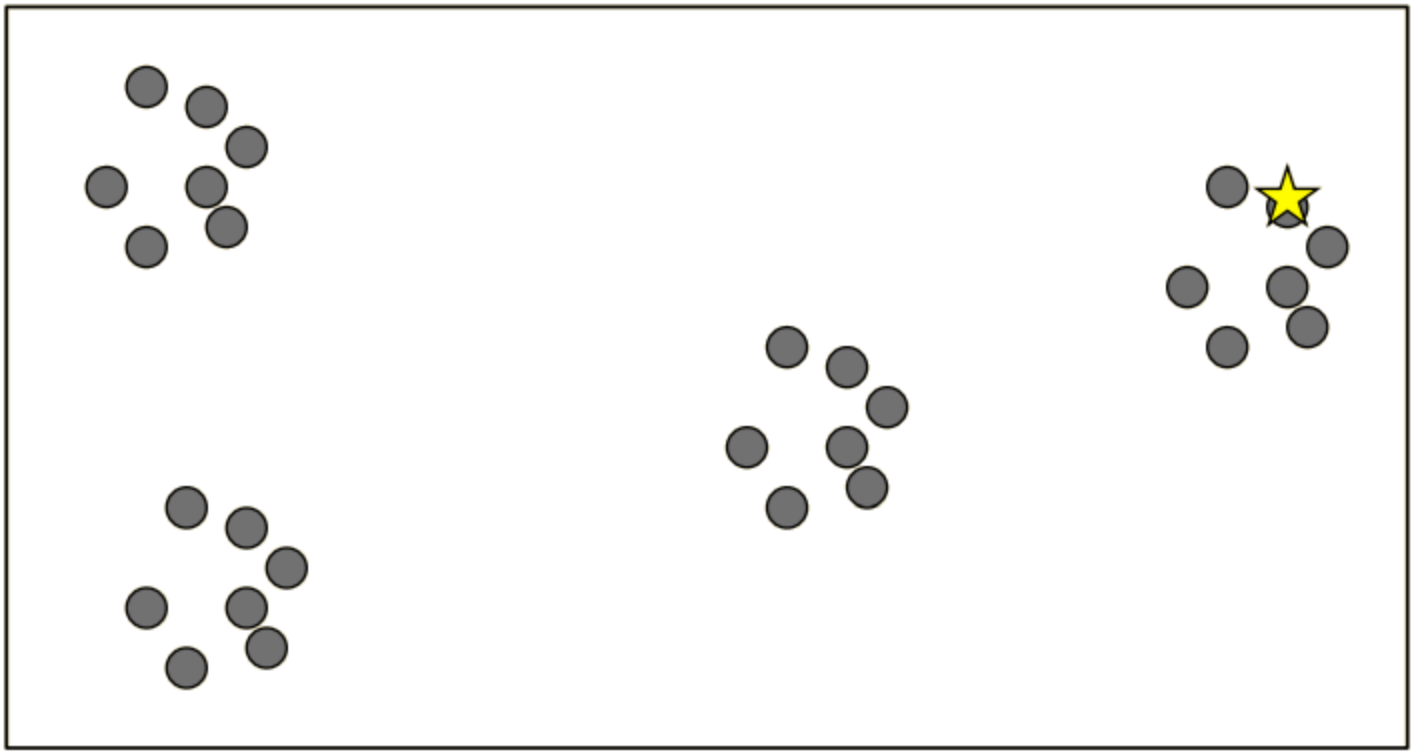
$$\frac{d(x_0, C)^2}{\varphi_X(C)} \propto d(x_0, C)^2$$

- Theorem: O(log k)-approximation to optimum, right after initialization

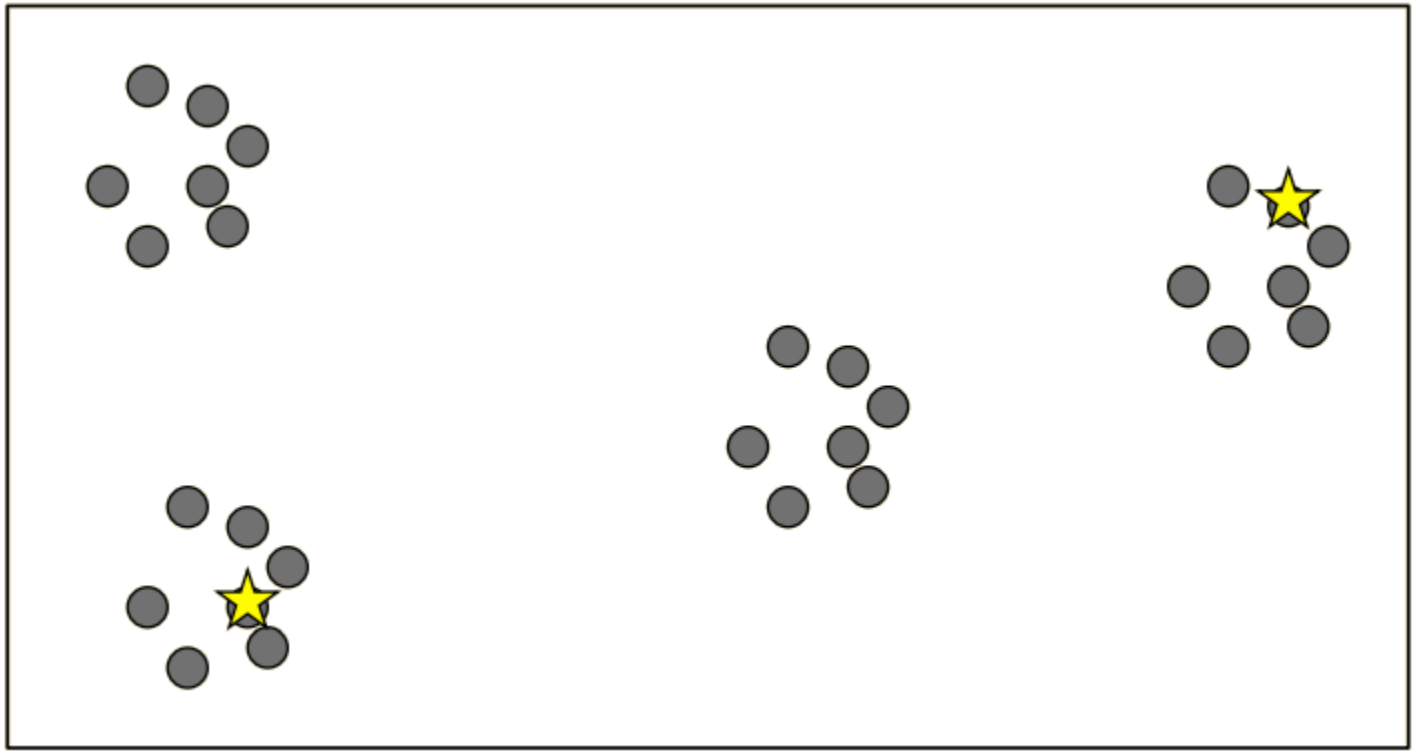
K-means++ Initialization



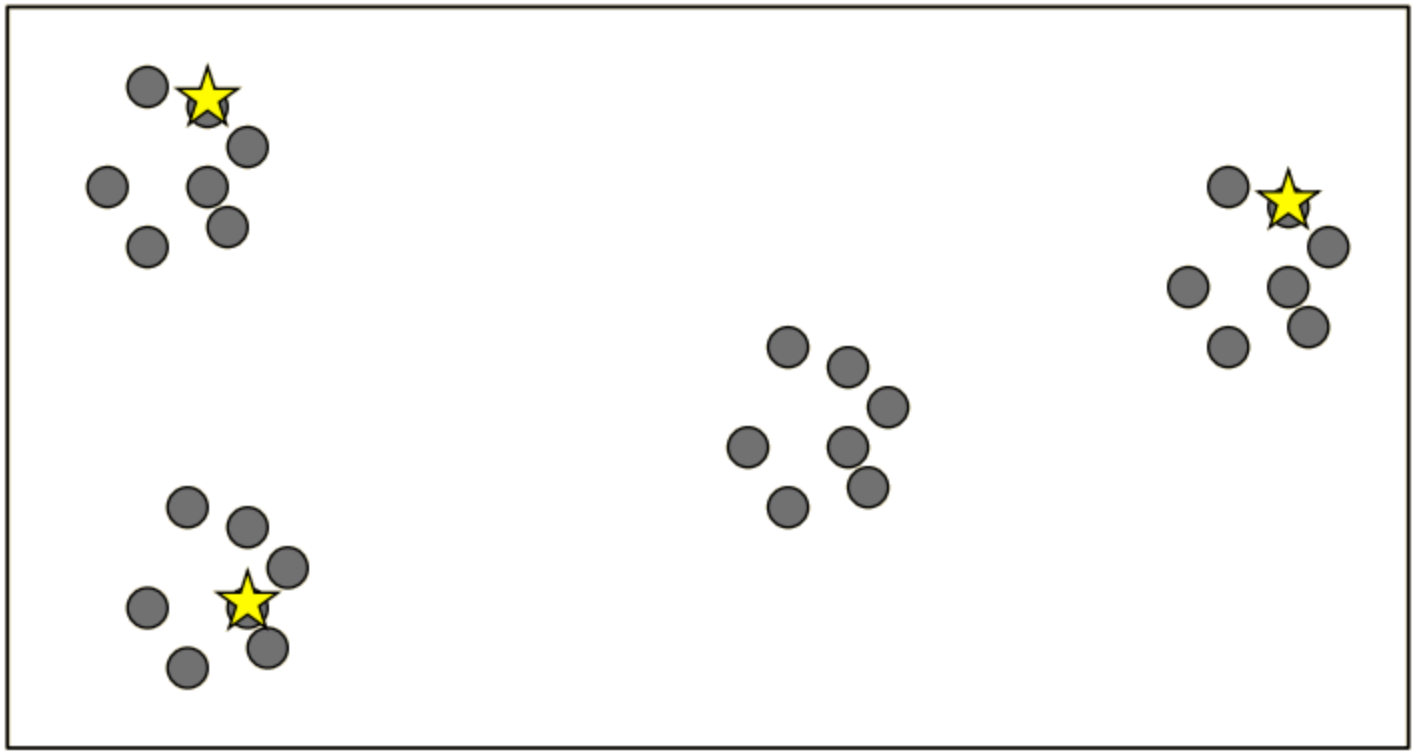
K-means++ Initialization



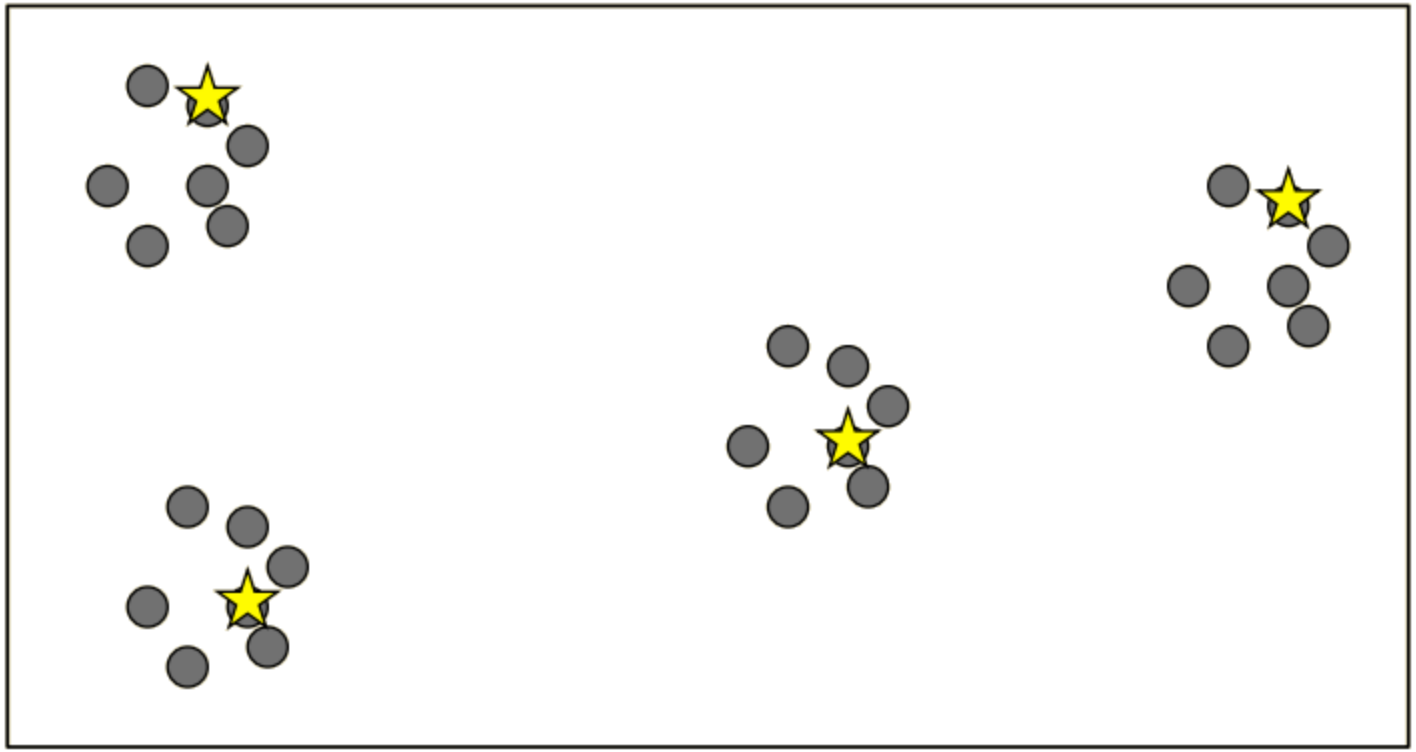
K-means++ Initialization



K-means++ Initialization



K-means++ Initialization



What's Wrong with K-means++?

- Needs K passes over the data
- In large data applications, not only the data is massive, but also K is typically large (e.g., easily 1000).
- Does not scale!

Outline

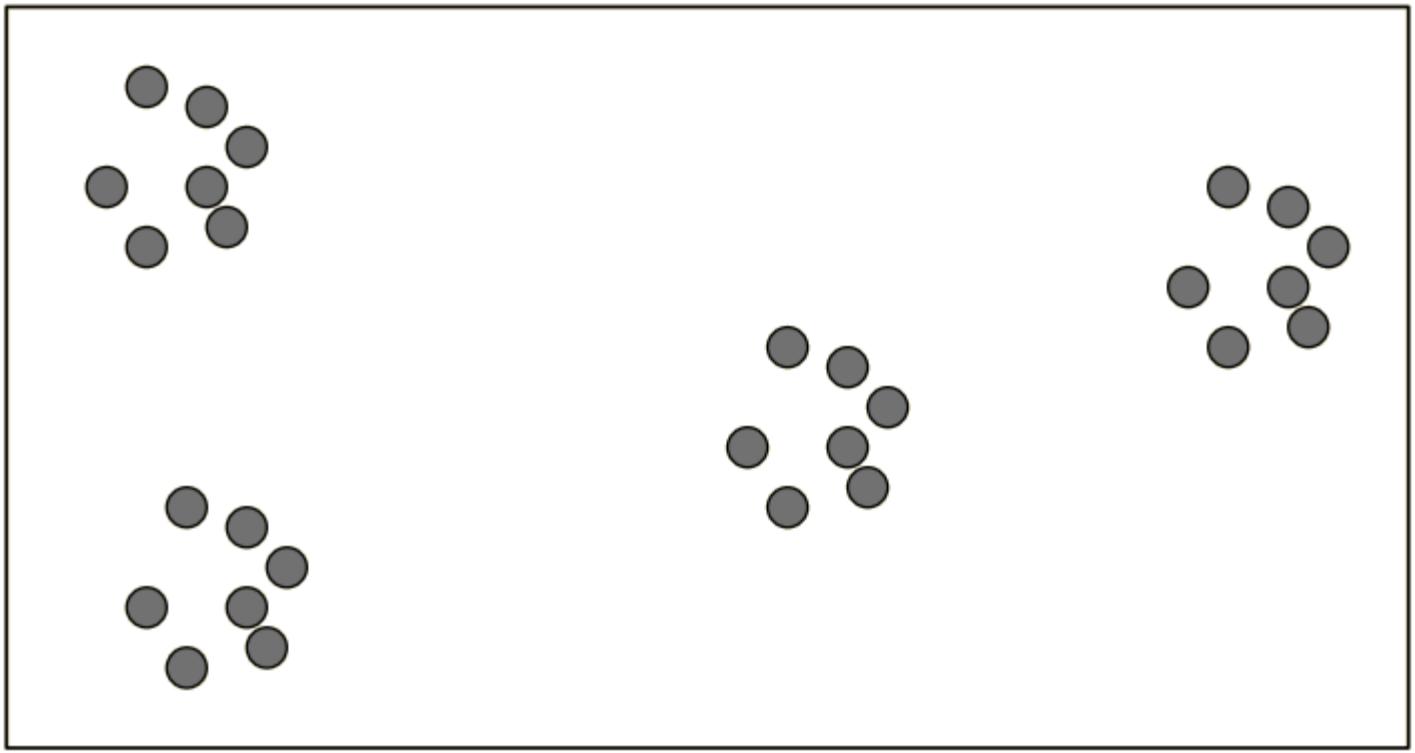
- Introduction
- Hierarchical Clustering
- K-means
- Scalable K-means
  - K-means||

Intuition of K-means||

- K-means++ samples one point per iteration and updates its distribution
- What if we **oversample** by sampling each point independently with a larger probability?
- Intuitively equivalent to updating the distribution much less frequently
  - Coarser sampling
- Turns out to be sufficient: K-means||

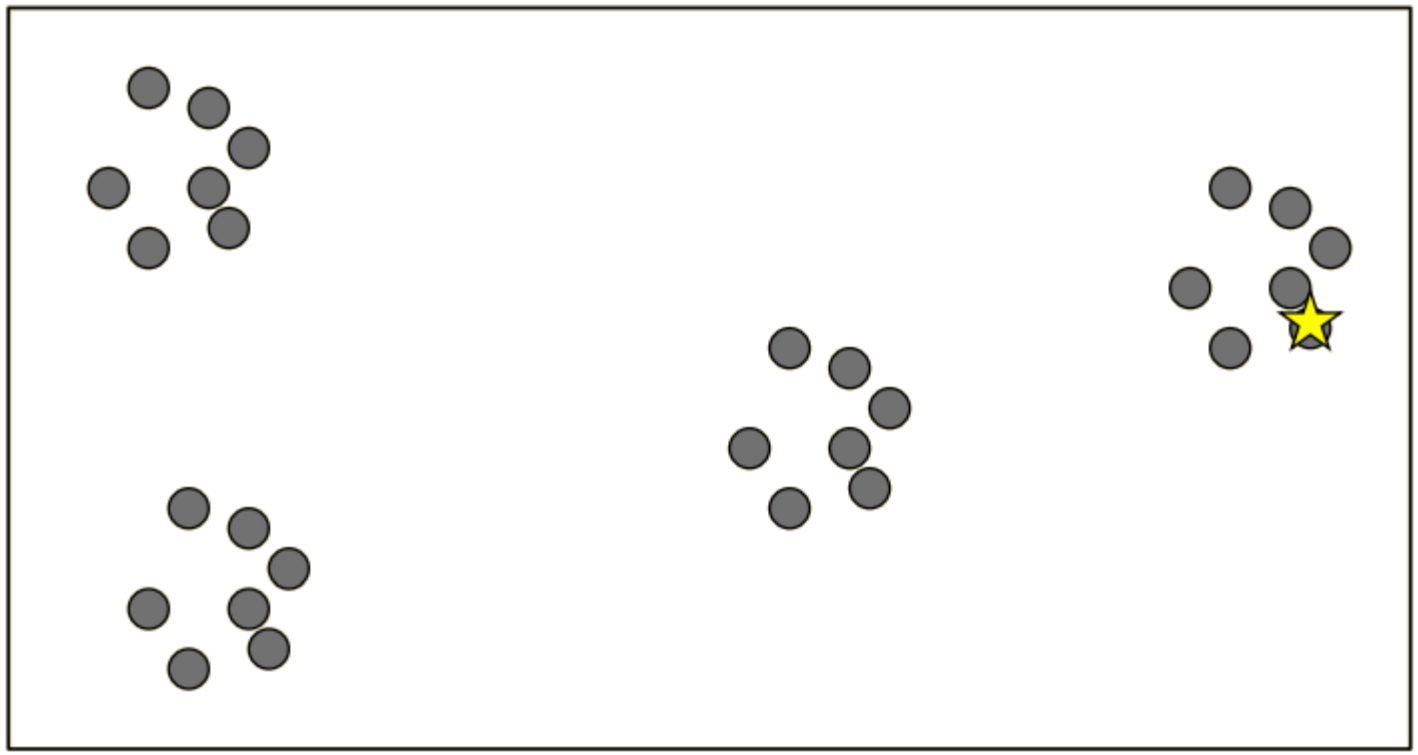
K-means|| Initialization

K=4,
Oversampling factor =3



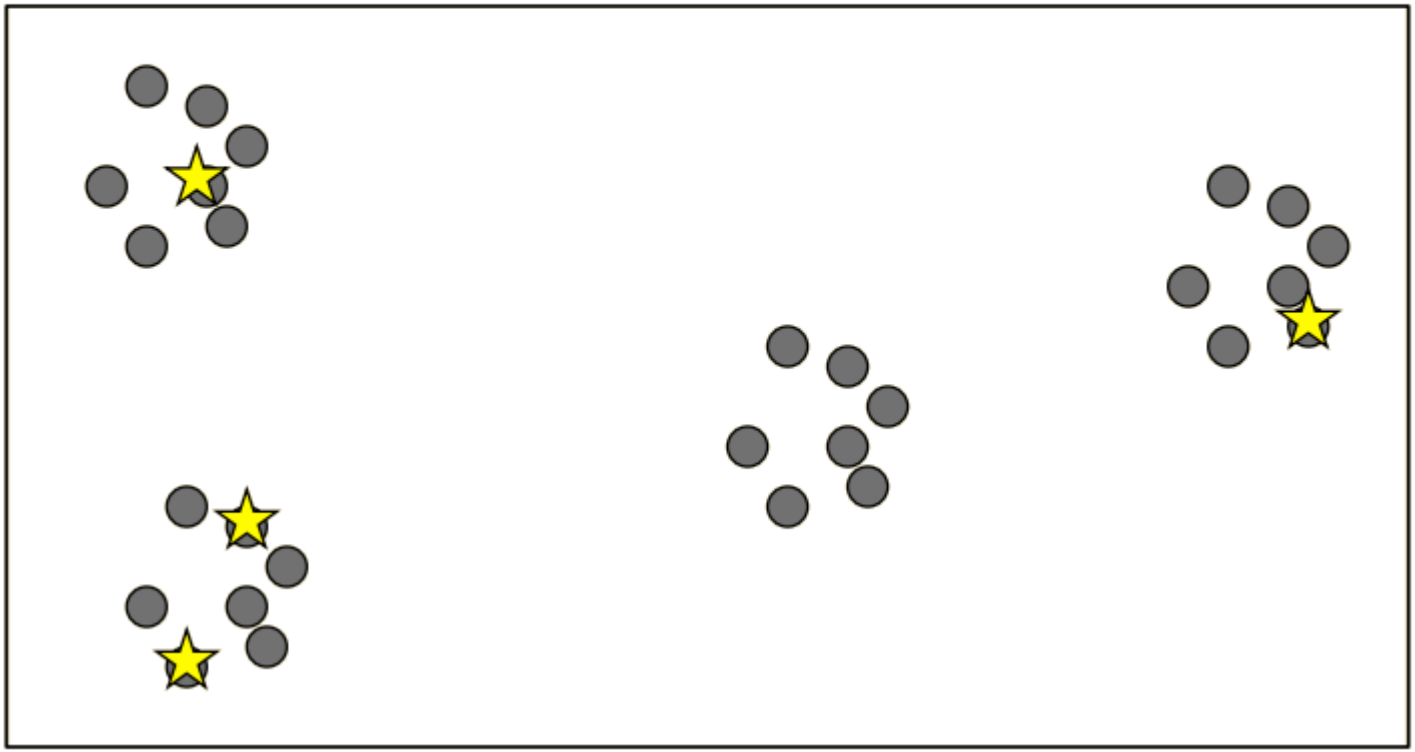
K-means|| Initialization

K=4,
Oversampling factor =3



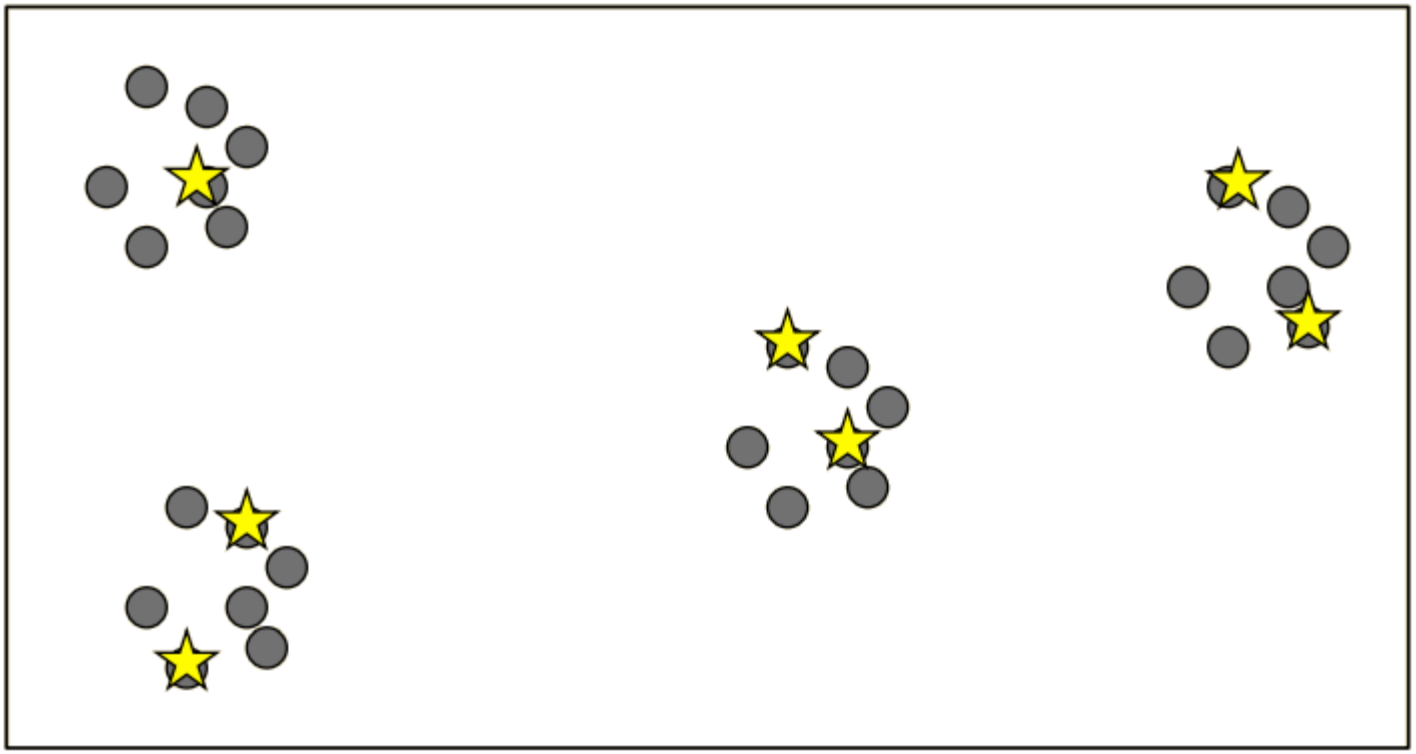
K-means|| Initialization

K=4,
Oversampling factor =3



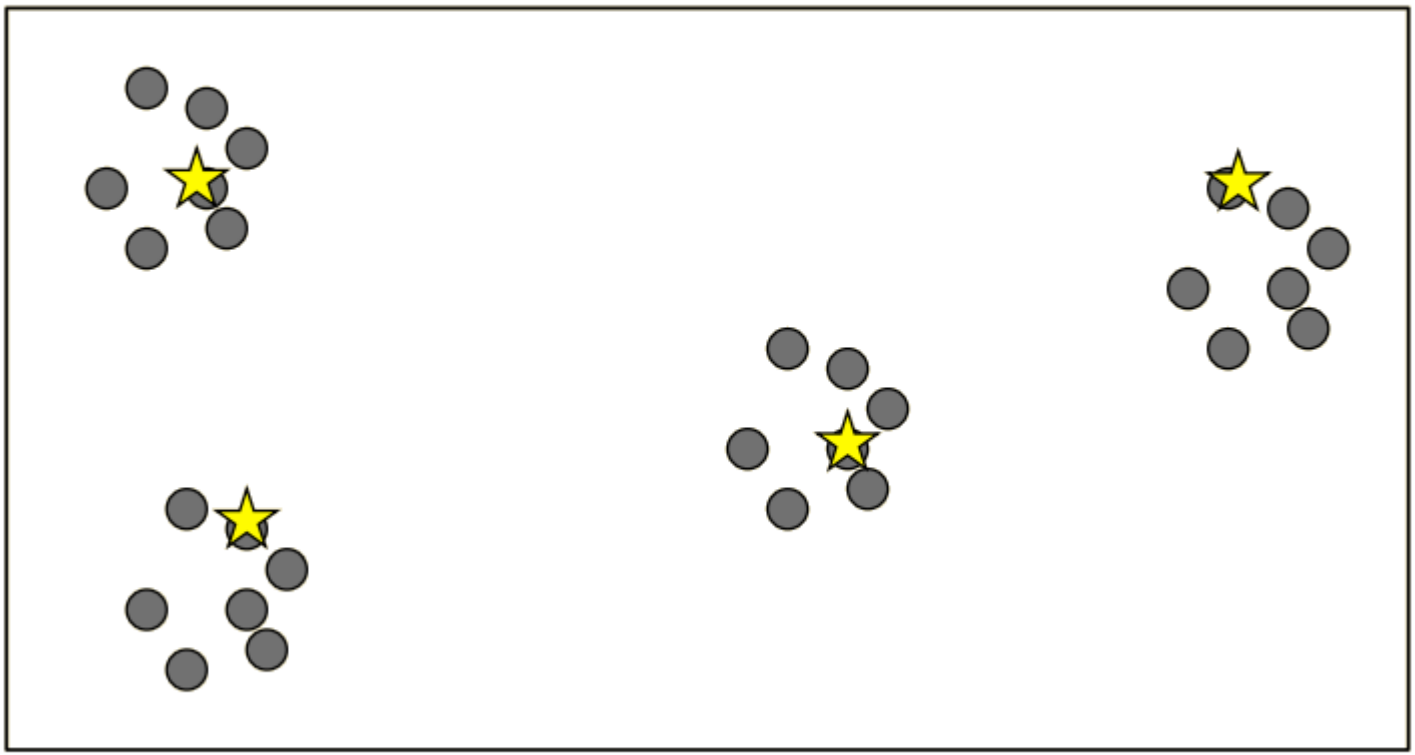
K-means|| Initialization

K=4,
Oversampling factor =3



K-means|| Initialization

K=4,
Oversampling factor =3



Cluster the intermediate centers

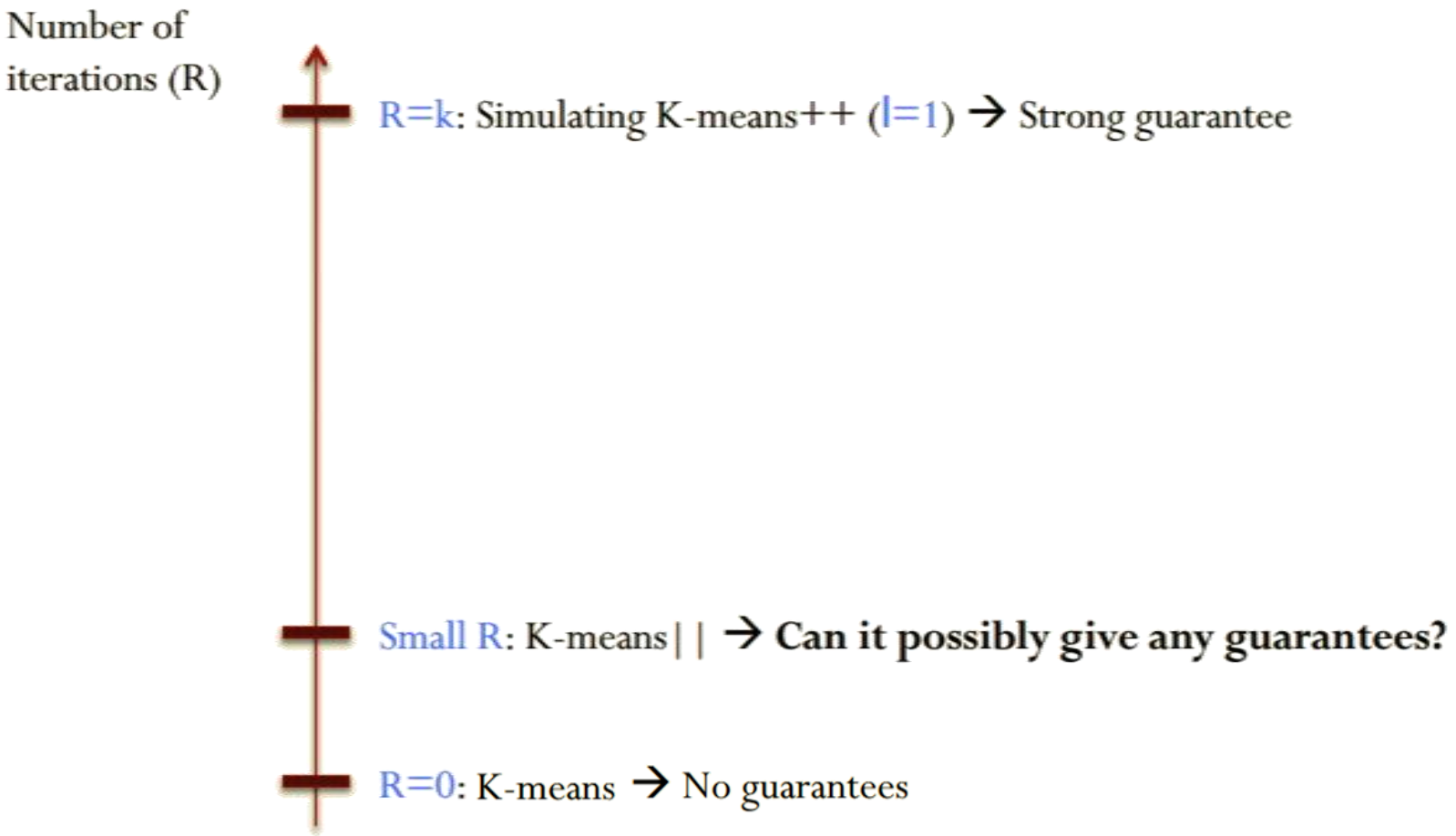
K-means|| [Bahmani et al. '12]

- Choose l>1 [Think l=Θ(k)]
- Initialize C to an arbitrary set of points
- For R iterations do:
  - Sample each point x in X independently with probability

$$p_x = l\,d^2(x,C)/\varphi_X(C).$$

  - Add all the sampled points to C
- Cluster the (weighted) points in C to find the final k centers

K-means, K-means++, and K-means||



Theorem

- Theorem: If φ and φ' are the costs of the clustering at the beginning and end of an iteration, and OPT is the cost of the optimum clustering:

$$E[\varphi'] \leq O(OPT) + \frac{k}{el}\varphi$$

- Corollary:
  - Let ψ= cost of initial clustering
  - K-means|| produces a constant-factor approximation to OPT, using only O(log (ψ/OPT)) iterations
  - Using K-means++ for clustering the intermediate centers, the overall approximation factor = O(log k)

Experimental Results: Quality

	Clustering Cost Right After Initialization	Clustering Cost After Convergence
Random	NA	22,000
K-means++	430	65
K-means||	16	14

GAUSSMIXTURE: 10,000 points in 15 dimensions
K=50
Costs scaled down by 10^4

- K-means|| much harder than K-means++ to get confused with noisy outliers

Experimental Results: Convergence

- K-means|| reduces number of iterations even more than K-means++

	Clustering Cost Right After Initialization
Random	167
K-means++	42
K-means||	28

SPAM: 4,601 points in 58 dimensions
K=50

Experimental Results

- K-means|| needs a small number of intermediate centers
- Better than K-means++ as soon as ~K centers chosen

	Clustering Cost (Scaled down by 10^{10})	Number of intermediate centers	Time (In Minutes)
Random	6.4×10^7	NA	489
K-means++	1.9	1.47×10^6	1022
K-means||	1.5	3604	87

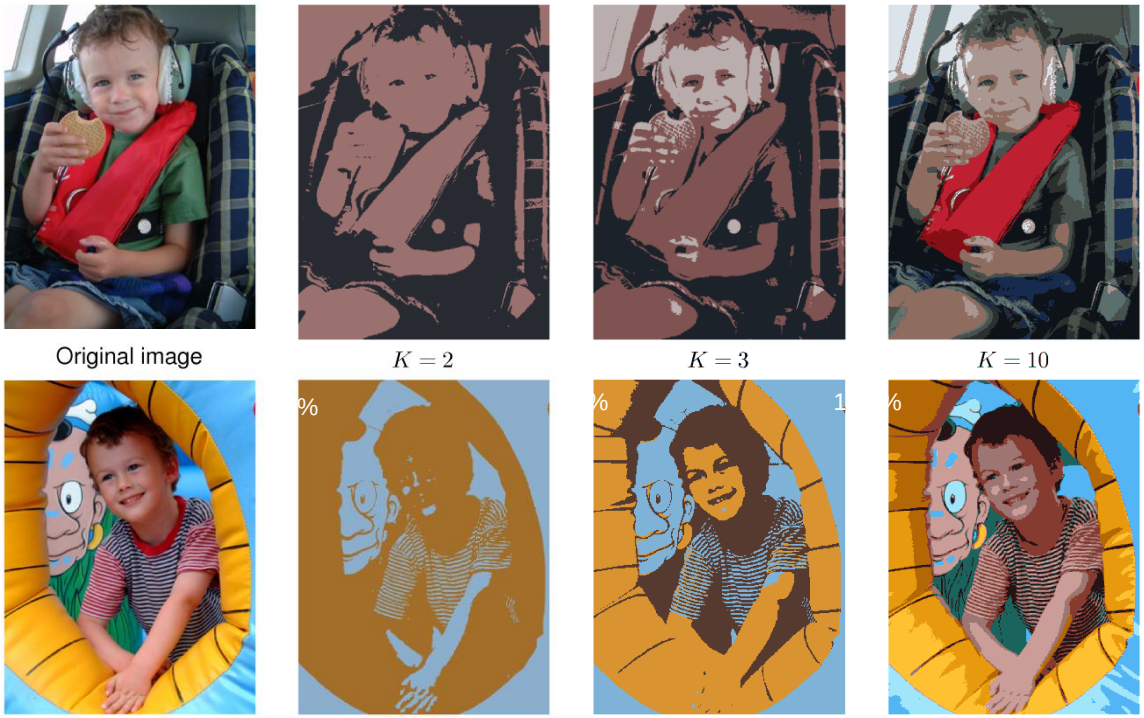
KDDCUP1999: 4.8M points in 42 dimensions
K=1000

In-class Practice

- Go to Practice
- Go to Solution

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.

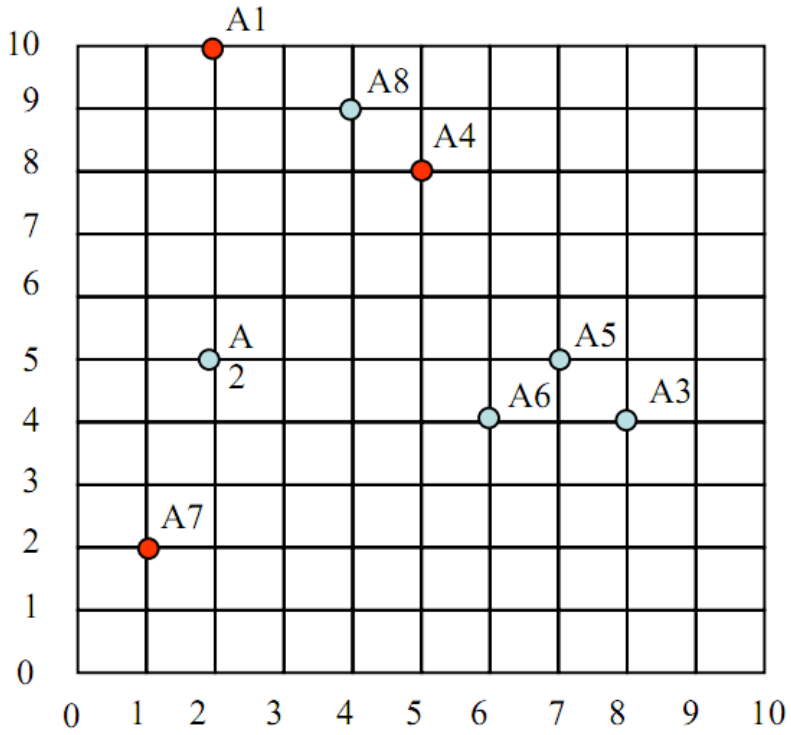
References

- Bahmani, Bahman, et al. "Scalable k means++." *Proceedings of the VLDB Endowment* 5.7 (2012): 622-633.
- Arthur, David, and Sergei Vassilvitskii. "k-means++: The advantages of careful seeding." *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 2007.
- Wu, Xindong, et al. "Top 10 algorithms in data mining." *Knowledge and Information Systems* 14.1 (2008): 1-37.
- Berkhin, Pavel. "A survey of clustering data mining techniques." *Grouping multidimensional data*. Springer Berlin Heidelberg, 2006. 25-71.

References

- G. Scott and H. Longuet-Higgins. An algorithm for associating the features of two patterns. In Proc. Royal Society London, volume B244, pages 21-26, 1991.
- A. Ng, M. Jordan, Y. Weiss. On Spectral clustering: analysis and algorithm. In Advances in Neural Information Processing Systems (2001), pp. 849-856
- Pietro Perona and William Freeman. A factorization approach to grouping. In ECCV'98, pp. 655-670.
- Jianbo Shi and Jitendra Malik. 2000. Normalized Cuts and Image Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* 22, 8 (August 2000), 888-905.
- Wing Cheong Lau, Web-scale Information Analytics, 2016.

In-class Practice



- **Given 8 points in the left 2D space, suppose that the initial seeds (centers of each cluster) are A1, A4 and A7. Run the k-means algorithm.**
  1. **Using Euclidean distance show the clusters after the first epoch and the new centroids.**
  2. **How many more iterations are needed to converge? Draw the result for each epoch.**
  3. **If we apply K-means++ to the data and choose A4 as the first seed, which point will be chosen as the second seed?**

Go Back