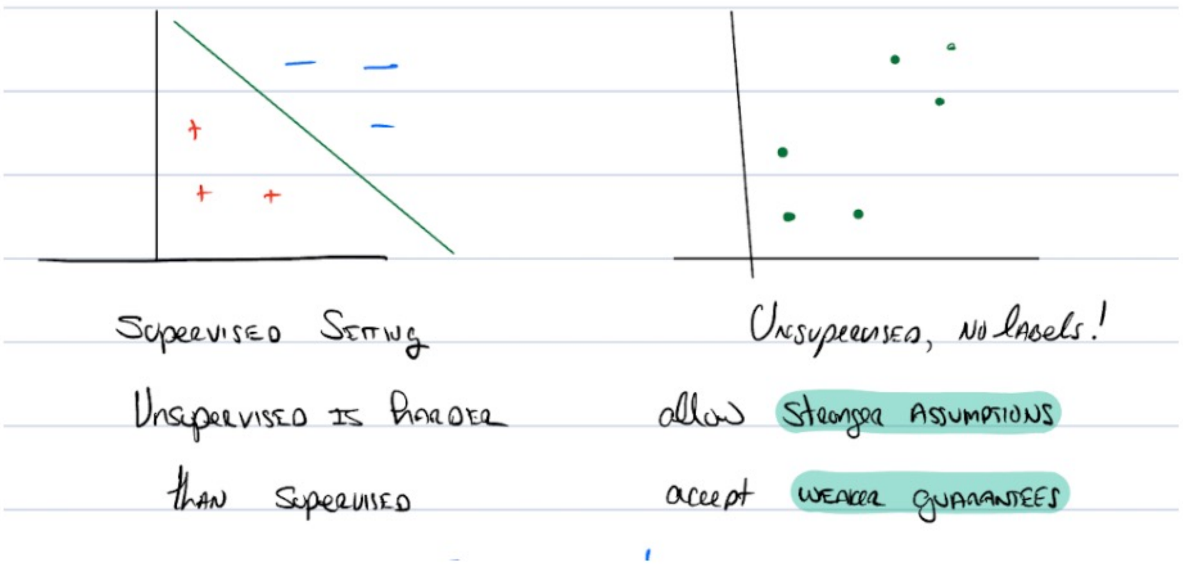
CMSC 478
Unsupervised Learning
K-means Clustering

KMA Solaiman

ksolaima@umbc.edu

Many slides courtesy Hamed Pirsiavash

Unsupervised Learning In Pictures



Unsupervised learning is "harder" than supervised, so we'll make *stronger* assumptions and accept *weaker guarantees*.

Outline

- Clustering basics
- K-means: basic algorithm & extensions
  - Cluster evaluation
  - Non-parametric mode finding: density estimation
- Graph & spectral clustering
- Hierarchical clustering
- K-Nearest Neighbor

Clustering

project where you need to predict the sales of a big mart:

Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales
Medium	Tier 1	Supermarket Type1	3735.1380
Medium	Tier 3	Supermarket Type2	443.4228
Medium	Tier 1	Supermarket Type1	2097.2700
NaN	Tier 3	Grocery Store	732.3800
High	Tier 3	Supermarket Type1	994.7052

Clustering

your task is to predict whether a loan will be approved or not:

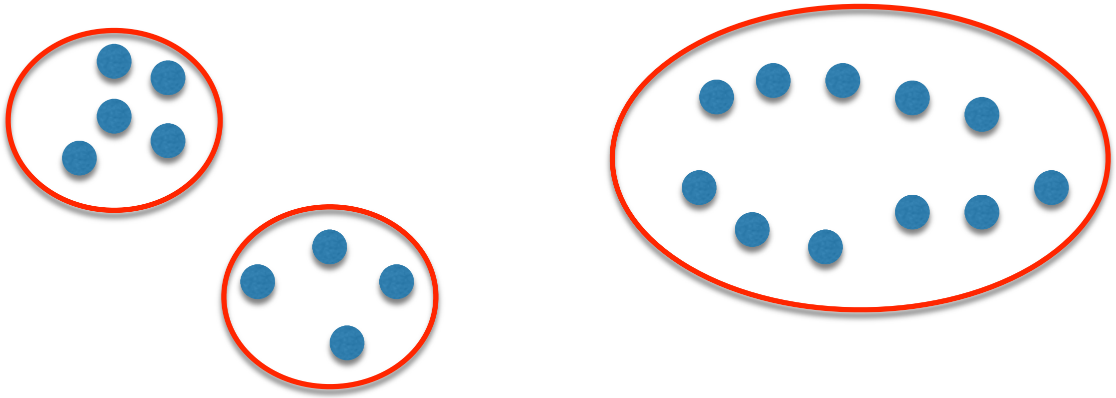
Loan_ID	Gender	Married	ApplicantIncome	LoanAmount	Loan_Status
LP001002	Male	No	5849	130.0	Y
LP001003	Male	Yes	4583	128.0	N
LP001005	Male	Yes	3000	66.0	Y
LP001006	Male	Yes	2583	120.0	Y
LP001008	Male	No	6000	141.0	Y

Clustering

Clustering

Basic idea: group together similar instances

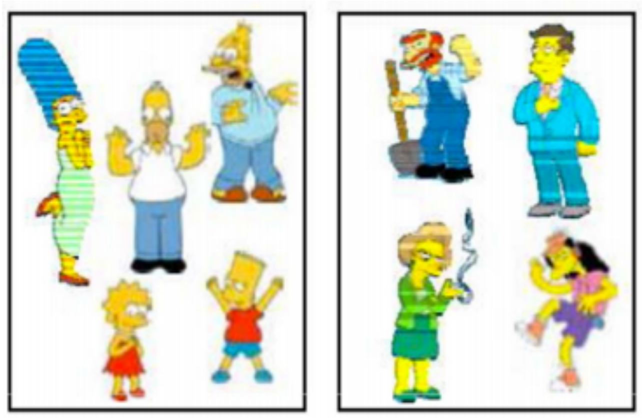
Example: 2D points



Clustering Algorithms

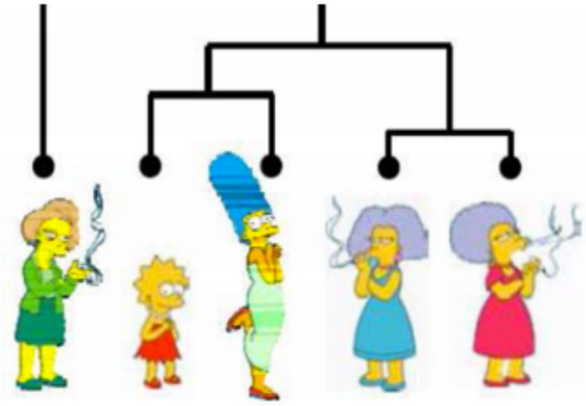
Simple clustering: organize elements into k groups

- K-means
- Mean shift
- Spectral clustering



Hierarchical clustering: organize elements into a hierarchy

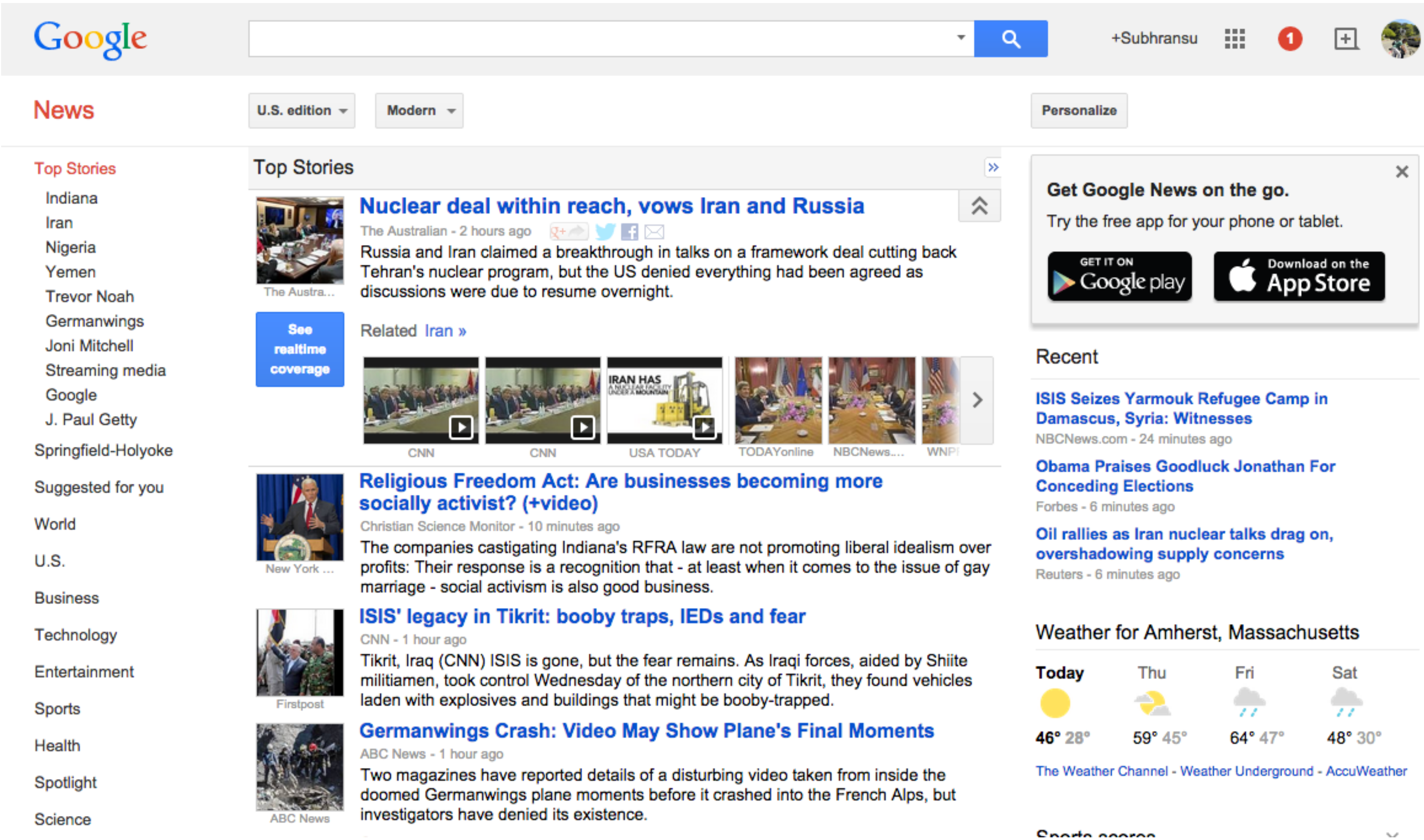
- Bottom up - agglomerative
- Top down - divisive



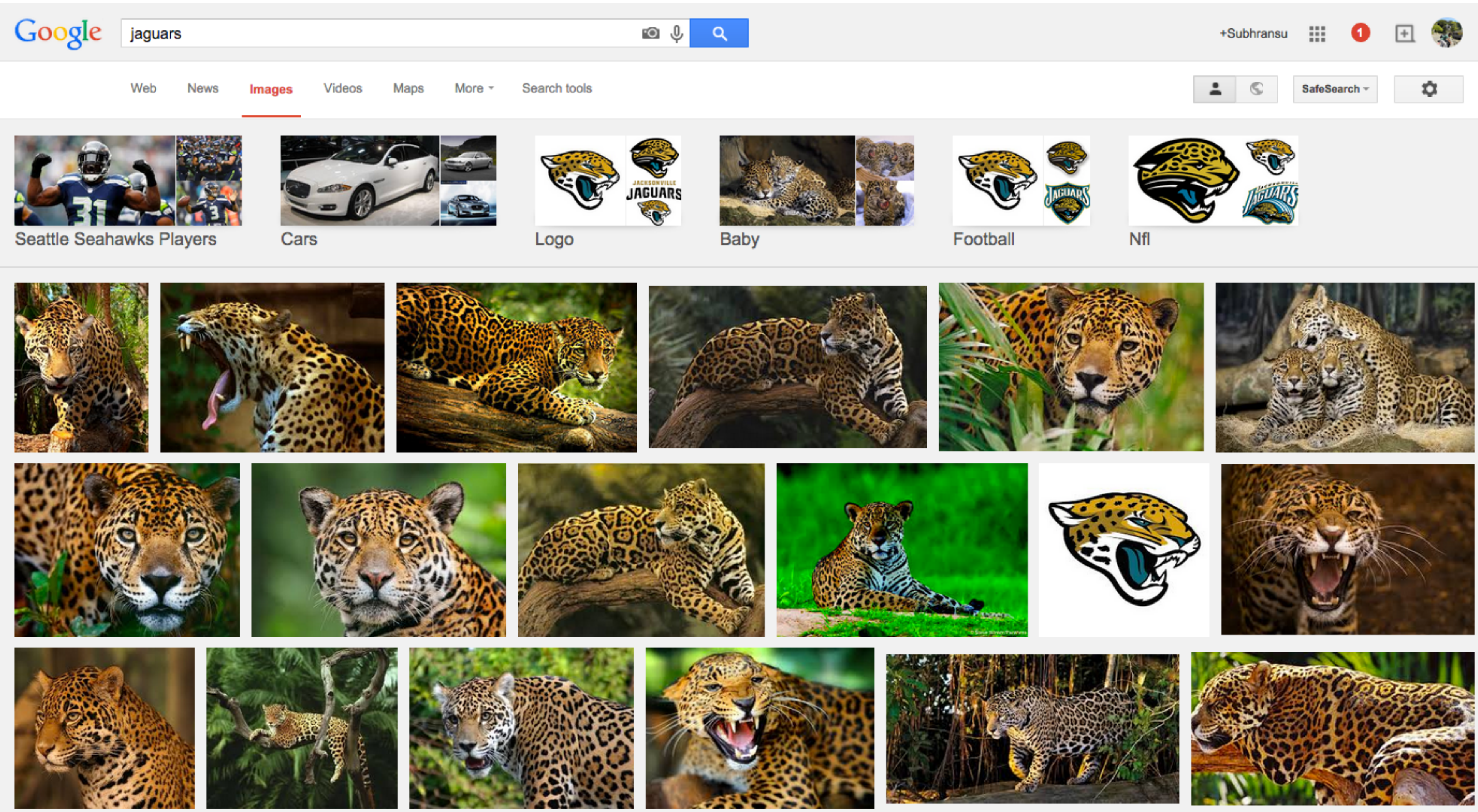
Clustering examples: Image Segmentation

image credit: Berkeley segmentation benchmark

Clustering examples: News Feed



Clustering examples: Image Search

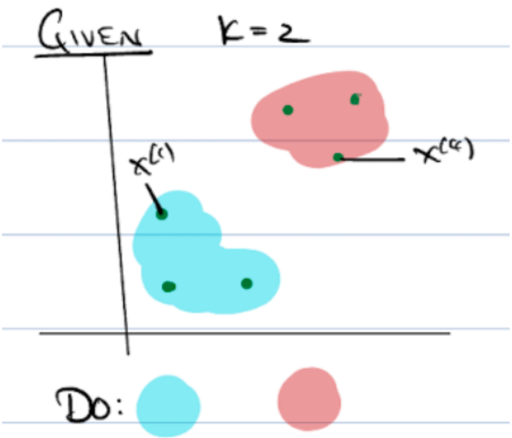


Outline

- Clustering basics
- K-means: basic algorithm & extensions
  - Cluster evaluation
  - Non-parametric mode finding: density estimation
- Graph & spectral clustering
- Hierarchical clustering
- K-Nearest Neighbor

K-Means

Given $k = 2$ and the following data find clusters.



- **Given** an integer k (the number of clusters) and $x^{(1)}, \ldots, x^{(n)} \in \mathbb{R}^d$.
- **Do** find an assignment of $x^{(i)}$ to one of the k clusters.

$$C^{(i)} = j \text{ means point } i \text{ in cluster } j$$

e.g., $C^{(2)} = 2$ and $C^{(4)} = 1$

How do we find these clusters? (Iterative Approach)

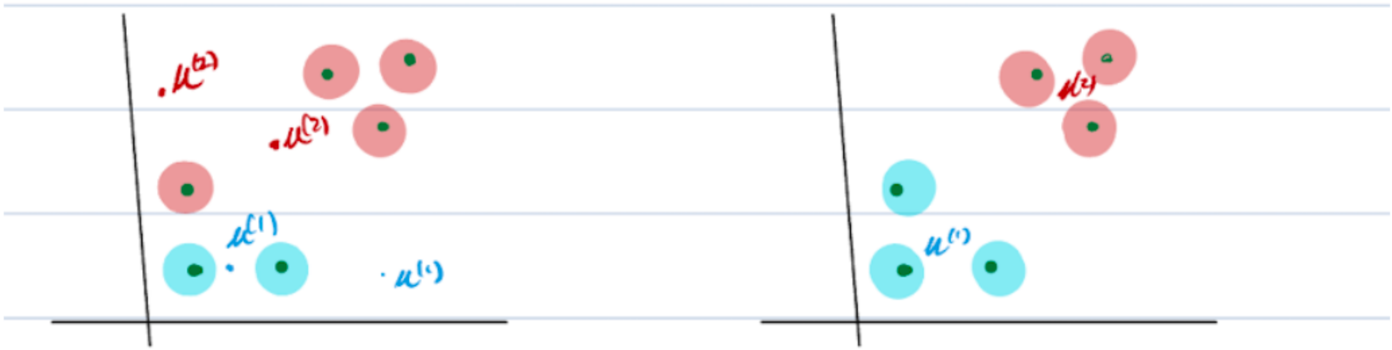
- (Randomly) Initialize Centers $\mu^{(1)}$ and $\mu^{(2)}$.

How do we find these clusters? (Iterative Approach)

- (Randomly) Initialize Centers $\mu^{(1)}$ and $\mu^{(2)}$.
- Assign each point, $x^{(i)}$, to closest cluster

$$C^{(i)} = \underset{j=1,\ldots,k}{\operatorname{argmin}} \|\mu^{(j)} - x^{(i)}\|^2 \text{ for } i = 1, \ldots, n$$

How do we find these clusters? (Iterative Approach)



- (Randomly) Initialize Centers $\mu^{(1)}$ and $\mu^{(2)}$.
- Assign each point, $x^{(i)}$, to closest cluster

$$C^{(i)} = \operatorname*{argmin}_{j=1,\ldots,k} \|\mu^{(j)} - x^{(i)}\|^2 \text{ for } i = 1, \ldots, n$$

- Compute new center of each cluster:

$$\mu^{(j)} = \frac{1}{|\Omega_j|} \sum_{i \in \Omega_j} x^{(i)} \text{ where } \Omega_j = \{i : C^{(i)} = j\}$$

How do we find these clusters? (Iterative Approach)

- (Randomly) Initialize Centers $\mu^{(1)}$ and $\mu^{(2)}$.
- Assign each point, $x^{(i)}$, to closest cluster

$$C^{(i)} = \underset{j=1,\ldots,k}{\operatorname{argmin}} \|\mu^{(j)} - x^{(i)}\|^2 \text{ for } i = 1, \ldots, n$$

- Compute new center of each cluster:

$$\mu^{(j)} = \frac{1}{|\Omega_j|} \sum_{i \in \Omega_j} x^{(i)} \text{ where } \Omega_j = \{i : C^{(i)} = j\}$$

Repeat until clusters stay the same!

Properties of the Lloyd's algorithm

Guaranteed to converge in a finite number of iterations

- objective decreases monotonically
- local minima if the partitions don't change.
- finitely many partitions → k-means algorithm must converge

Running time per iteration

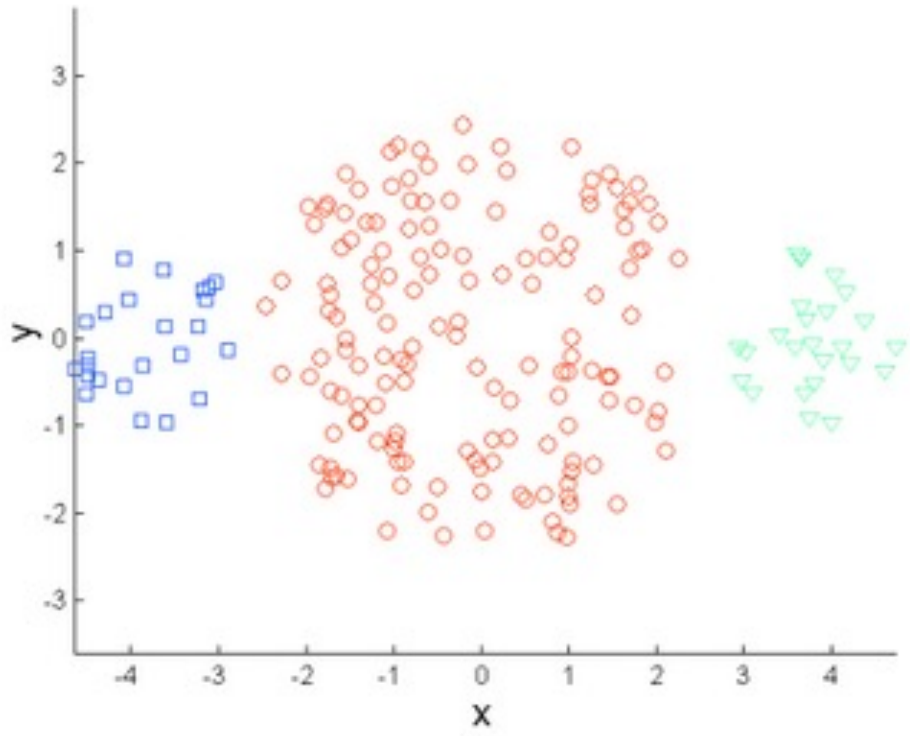
- Assignment step: O(NKD)
- Computing cluster mean: O(ND)

Issues with the algorithm:

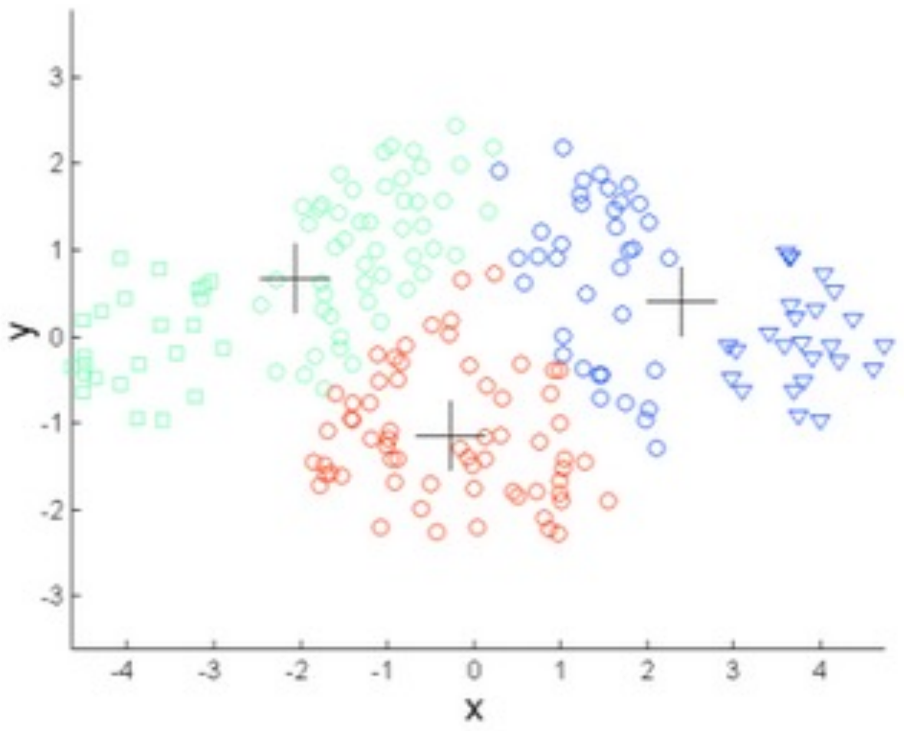
- Worst case running time is super-polynomial in input size
- No guarantees about global optimality
  - Optimal clustering even for 2 clusters is NP-hard [Aloise et al., 09]

- *N* is the number of *D*-dimensional vectors (to be clustered)
- *K* the number of clusters
- *i* the number of iterations needed until convergence.

Different number of clusters

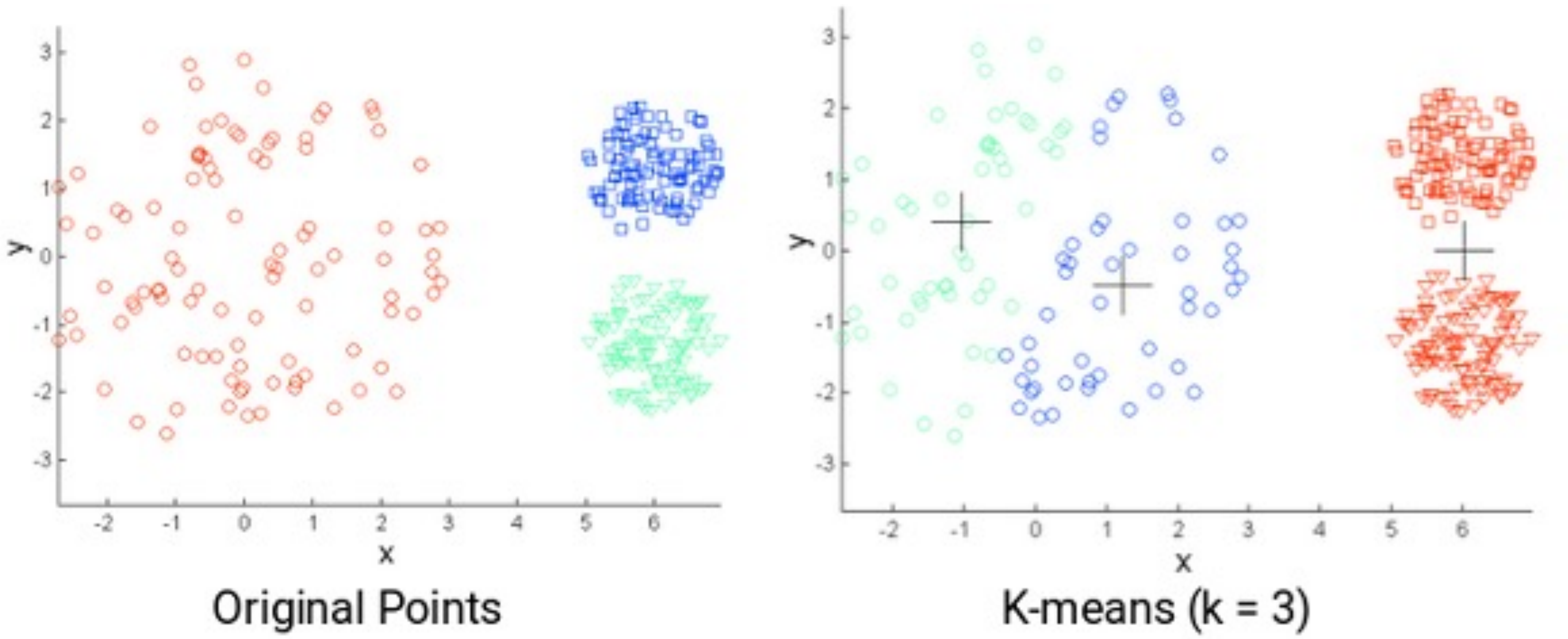


Original Points



K-means (k = 3)

Different Densities



Choosing K?

- # of clusters
- Cluster centers
  - K-means++
- Sensitivity to outliers
  - identify and handle outliers before applying k-means clustering
  - removing them, transforming them, or using a robust variant of k-means clustering that is less sensitive to the presence of outliers

K-means++

- Intuition: spread out the k initial cluster centers
- Compute Density Estimation
- Assign centroids based on that
- The algorithm proceeds normally once the centers are initialized

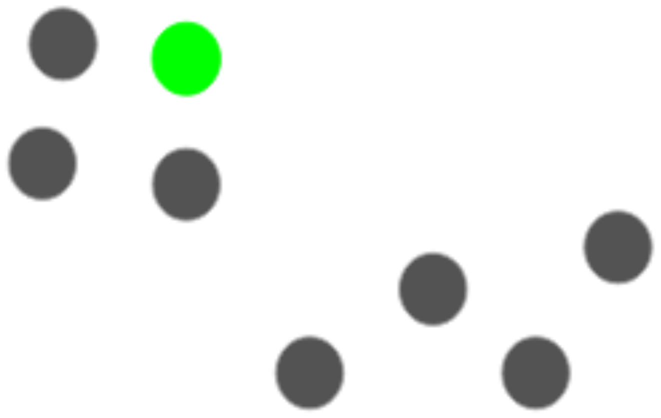
K-means++

- Compute Density Estimation
- Assign centroids based on that

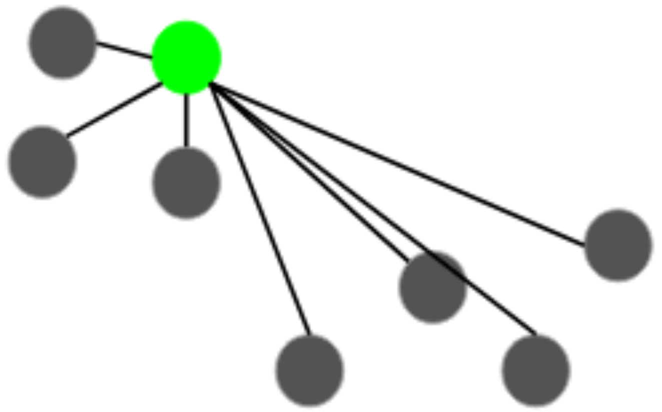
K-means++

- Compute Density Estimation
- Assign centroids based on that
- 3 clusters

K-means++



Random Pick



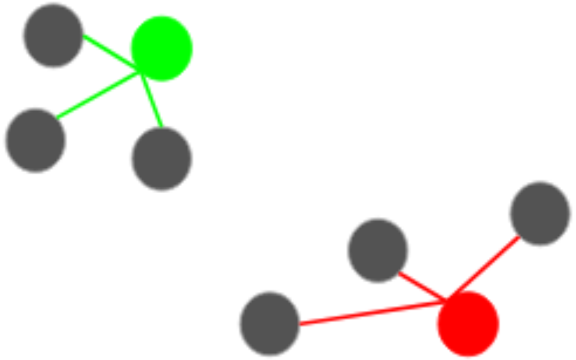
Calculate D(x)

K-means++

Largest $D(x)^2$

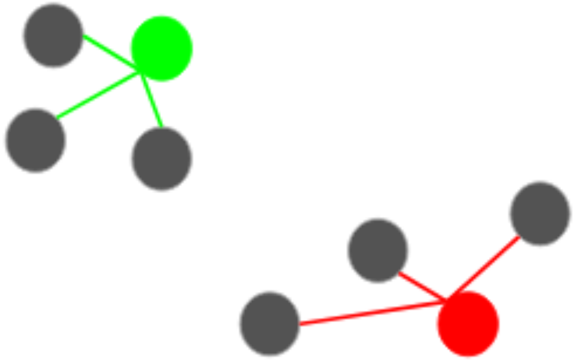
K-means++

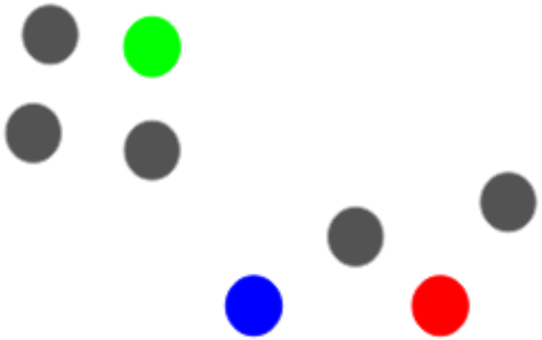
Largest $D(x)^2$



K-means++

Largest $D(x)^2$





Largest $D(x)^2$
from both center

K-means++

- Steps to Initialize the Centroids Using K-Means++

1. The first cluster is chosen uniformly at random from the data points we want to cluster. This is similar to what we do in K-Means, but instead of randomly picking all the centroids, we just pick one centroid here
2. Next, we compute the distance (D(x)) of each data point (x) from the cluster center that has already been chosen
3. Then, choose the new cluster center from the data points with the probability of x being proportional to $(D(x))^2$
4. We then repeat steps 2 and 3 until *k* clusters have been chosen

Fast kmeans

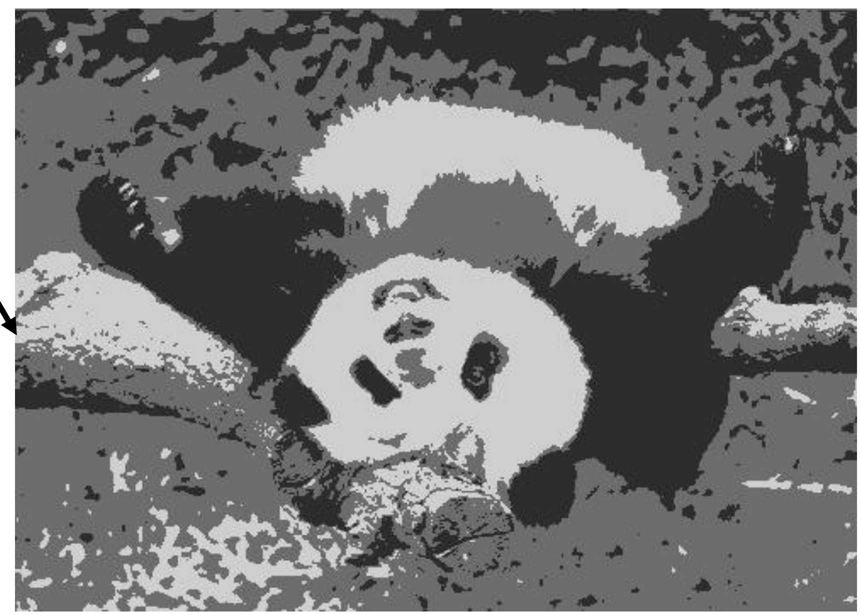
- Intuition: If a data point is close to center i and far from center j, and center j has not moved much since the last iteration, we don't need to recalculate the distance for center j.

- Use triangle inequality to prune the number of distances that you should recalculate.

k-means for image segmentation

K=2

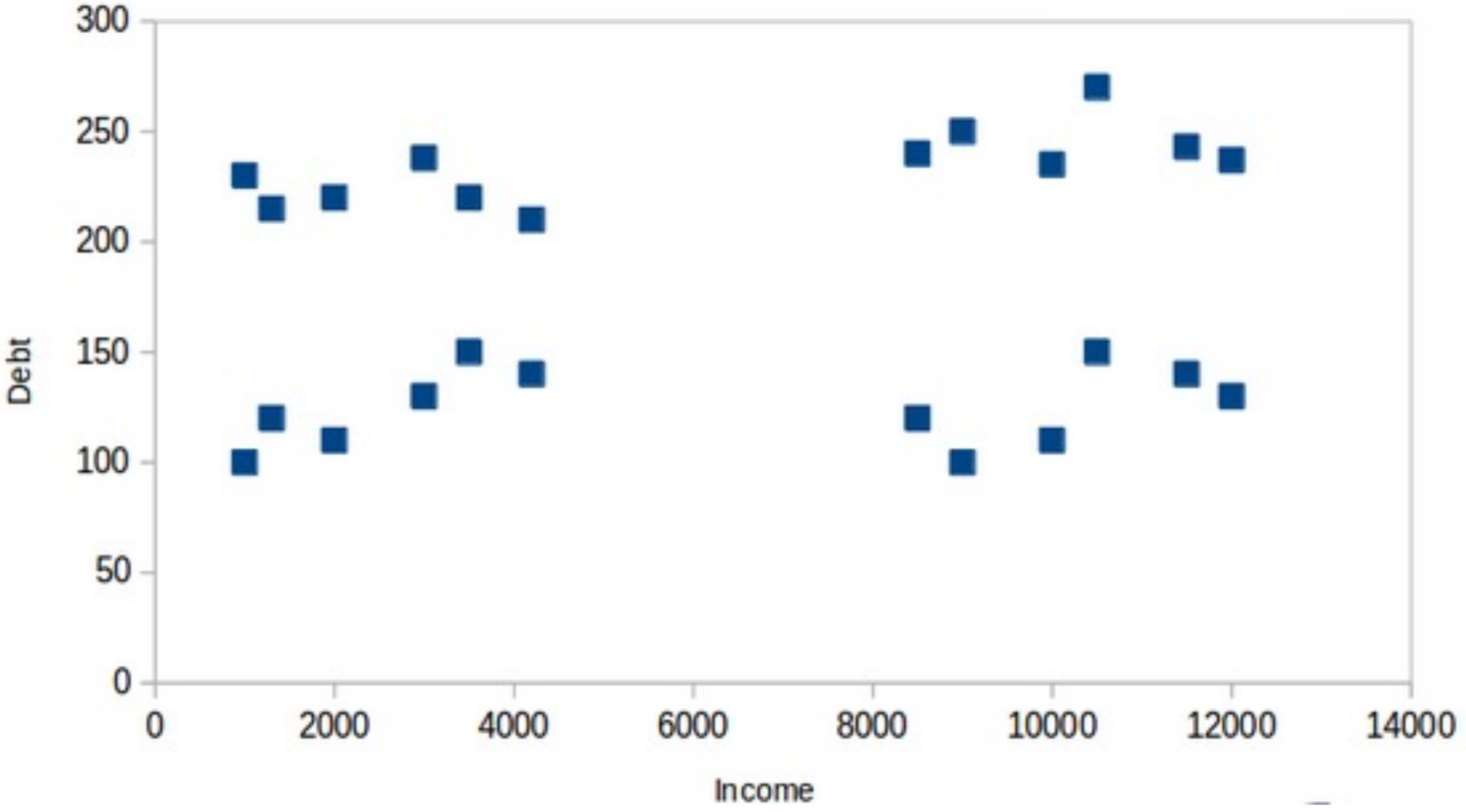
K=3



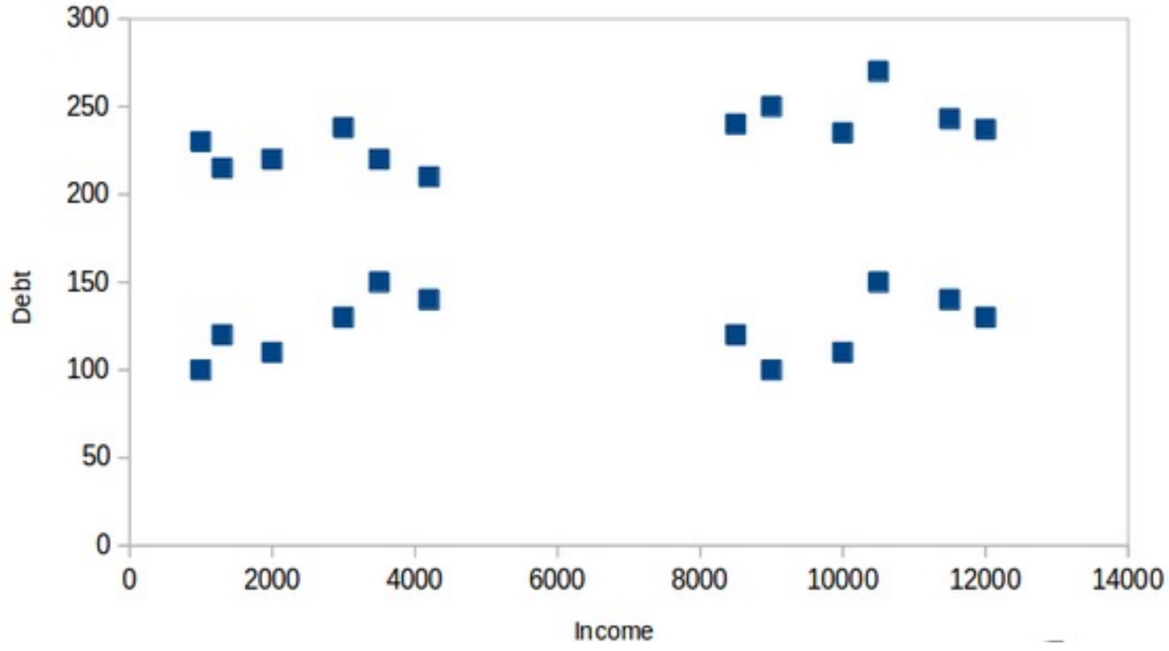
Grouping pixels based on intensity similarity

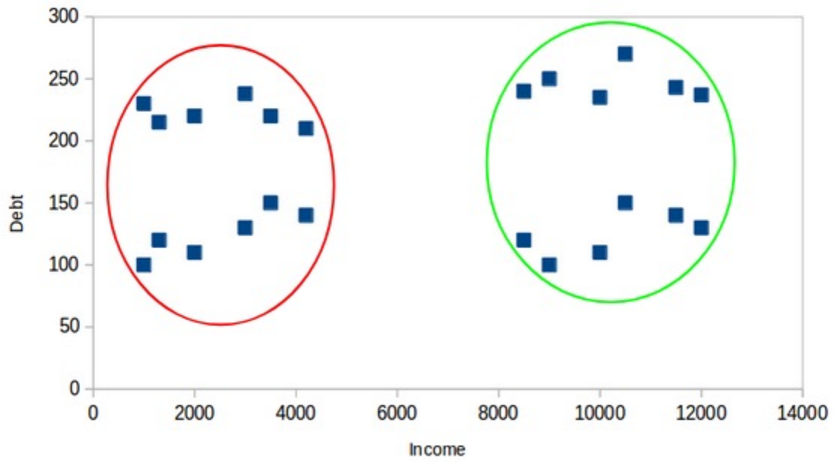
feature space: intensity value (1D)

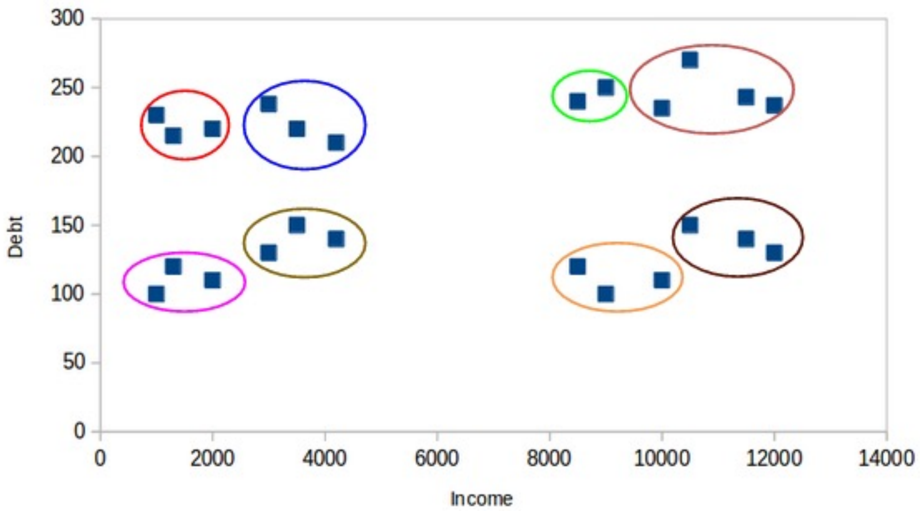
How to Choose the Right Number of Clusters?



How to Choose the Right Number of Clusters?







Outline

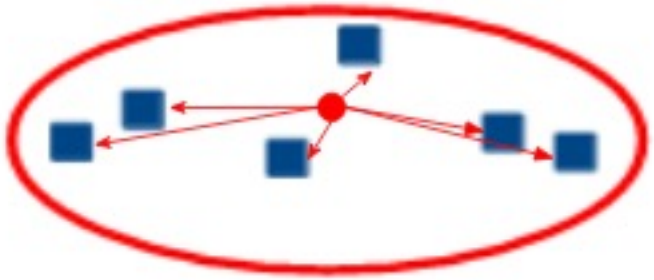
- Clustering basics
- K-means: basic algorithm & extensions
    - Cluster evaluation
    - Non-parametric mode finding: density estimation
- Graph & spectral clustering
- Hierarchical clustering
- K-Nearest Neighbor

Evaluation Metrics

- Inertia
  - sum of distances of all the points within a cluster from the centroid of that cluster.
  - lesser the inertia value, the better our clusters are.
- Silhouette Score
  - high silhouette score = clusters are well separated
  - 0 = overlapping clusters,
  - negative score suggests poor clustering solutions.
  - For each data,

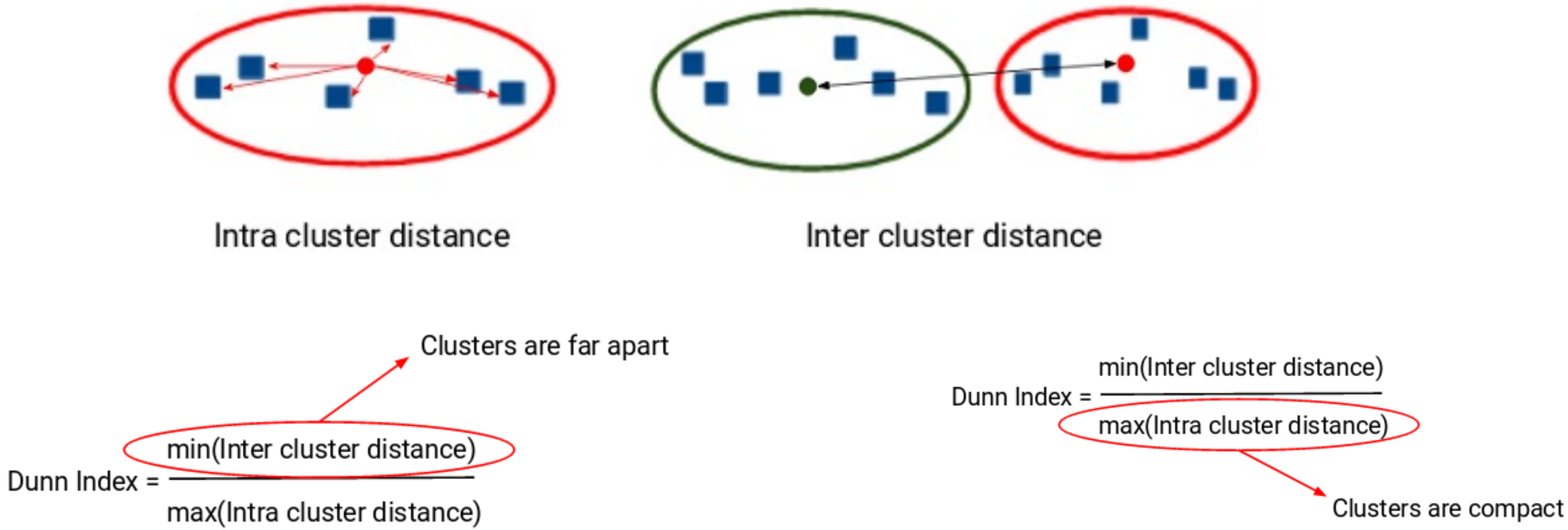
$$s = (b - a) / \max(a, b)$$

    - 'a' is the average distance within the cluster, 'b' is the average distance to the nearest cluster, and 'max(a, b)' is the maximum of 'a' and ‘b’
  - Mean for all points

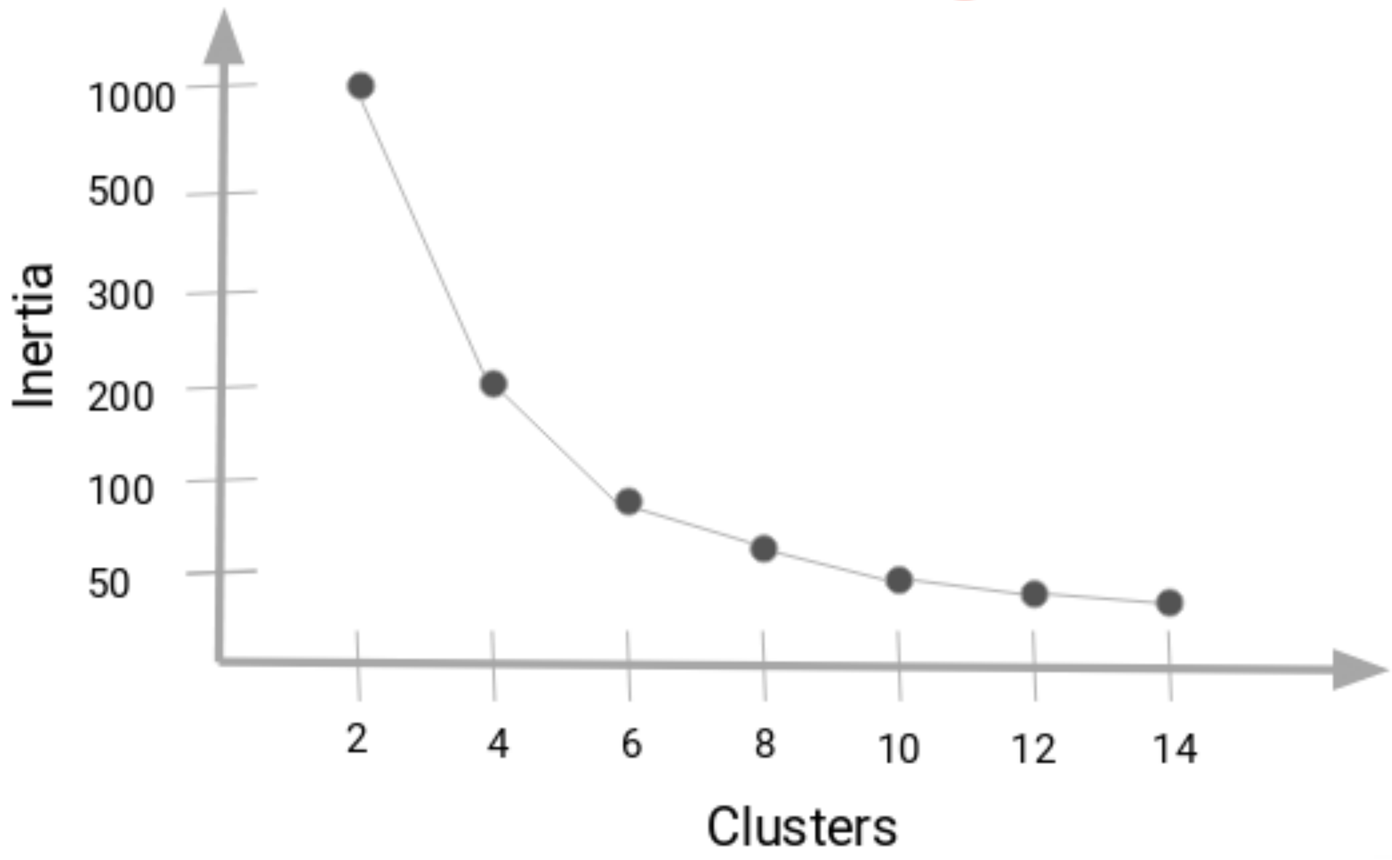


Intra cluster distance

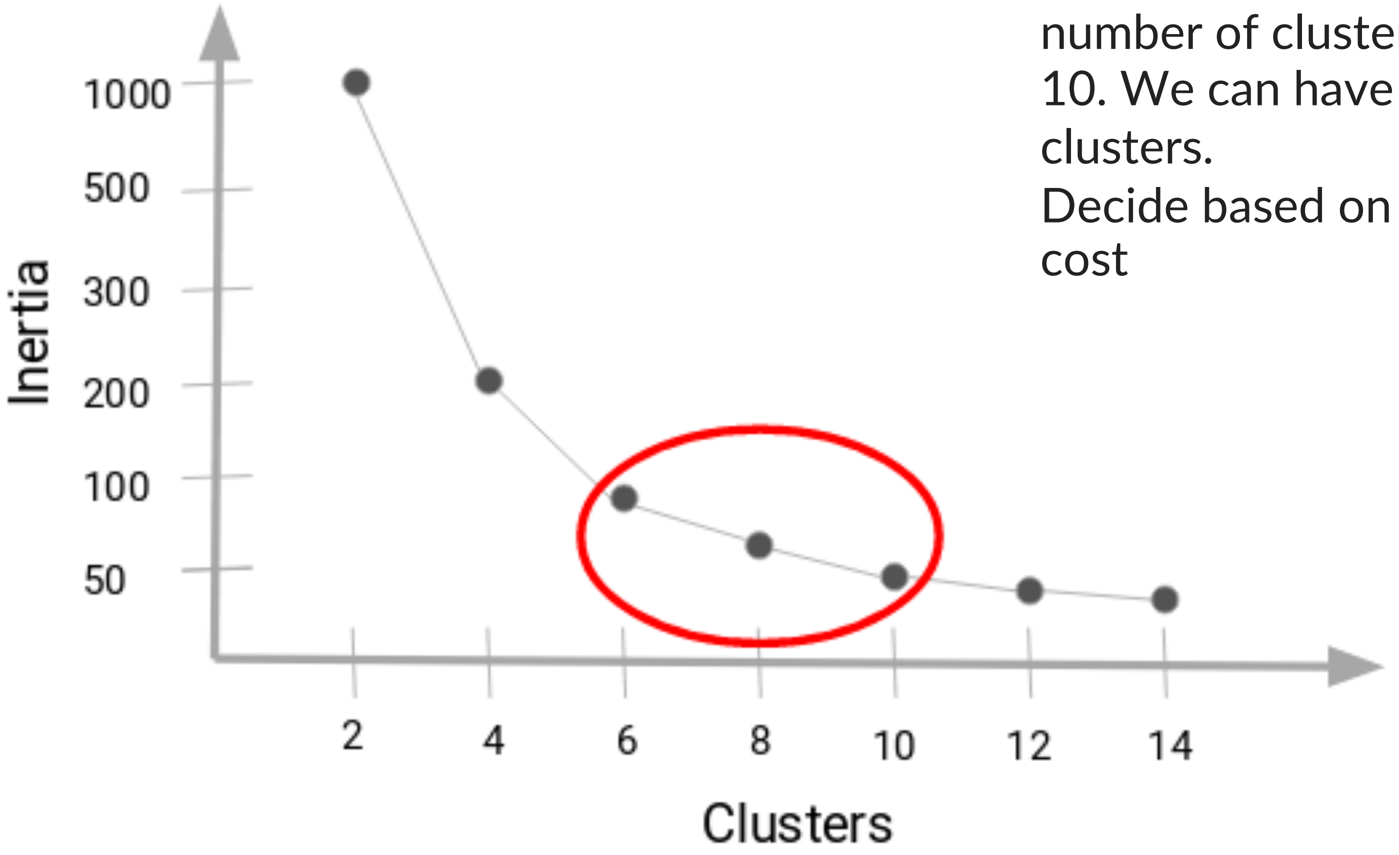
- Dunn index


Intra cluster distance
Inter cluster distance
Clusters are far apart
min(Inter cluster distance)
Dunn Index =
max(Intra cluster distance)
min(Inter cluster distance)
Dunn Index =
max(Intra cluster distance)
Clusters are compact

Empirical Choice of K



Empirical Choice of K



number of clusters between 6 and 10. We can have 7, 8, or even 9 clusters.
Decide based on computational cost

Outline

Clustering basics

K-means: basic algorithm & extensions

Cluster evaluation

Non-parametric mode finding: density estimation

Graph & spectral clustering

Hierarchical clustering

K-Nearest Neighbor

Clustering using density estimation

One issue with k-means is that it is sometimes hard to pick k

The mean shift algorithm seeks modes or local maxima of density in the feature space

Mean shift automatically determines the number of clusters

$$K(\mathbf{x}) = \frac{1}{Z} \sum_i \exp\left(-\frac{||\mathbf{x} - \mathbf{x}_i||^2}{h}\right)$$

Kernel density estimator

Small *h* implies more modes (bumpy distribution)

Mean shift algorithm

For each point x_i:

find m_i, the amount to shift each point x_i to its centroid

return $\{m_i\}$

Mean shift algorithm

For each point x_i:

set $m_i = x_i$

while not converged:

compute *weighted average of neighboring point*

return $\{m_i\}$

Mean shift algorithm

For each point x_i:

  set $m_i = x_i$

  while not converged:

    compute $m_i = \frac{\sum_{x_j \in N(x_i)} x_j K(m_i, x_j)}{\sum_{x_j \in N(x_i)} K(m_i, x_j)}$

return $\{m_i\}$

Neighbors of x_i

weighted average

self-clustering to based on kernel (similarity to other points)

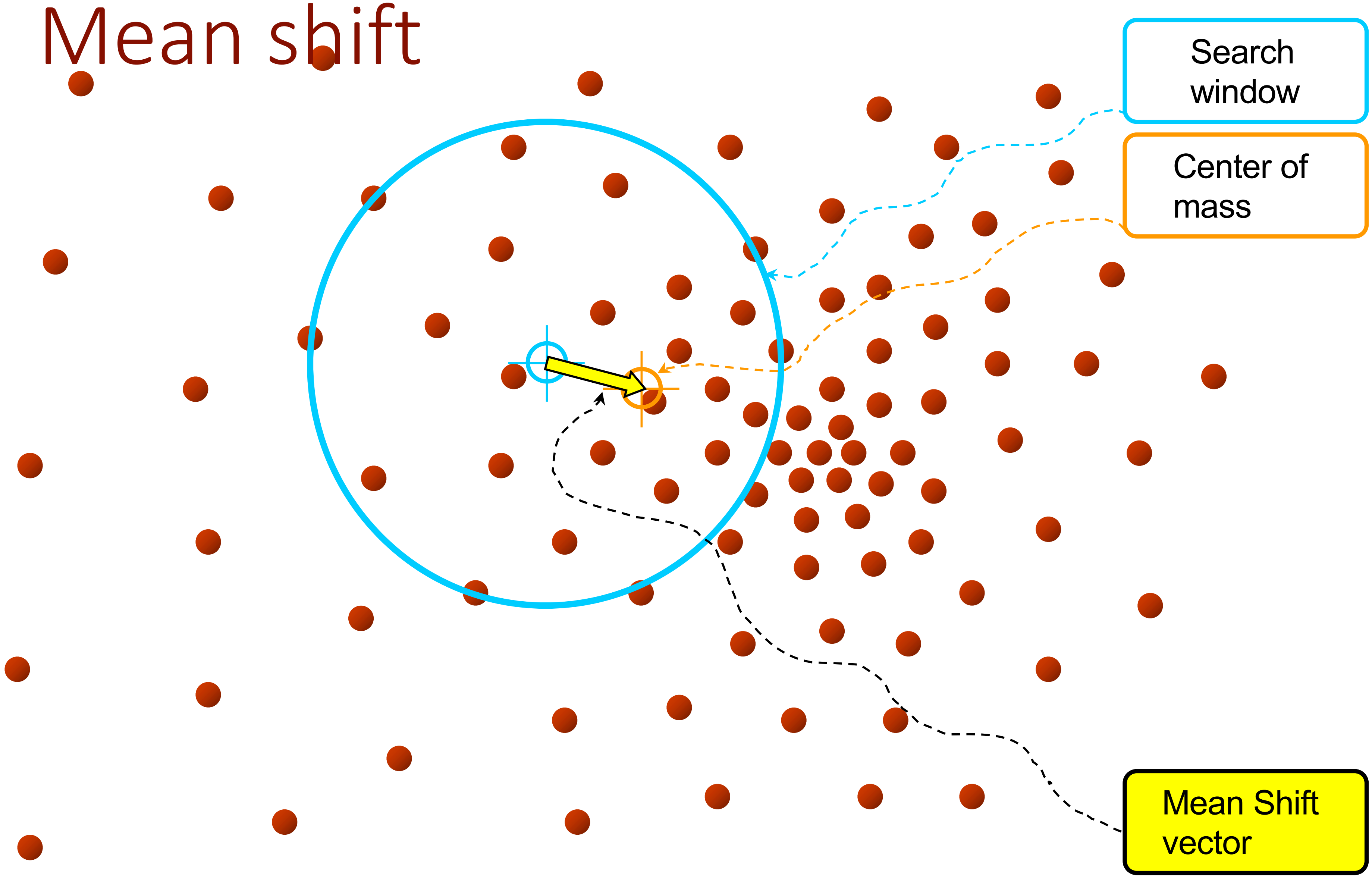
Pros:

- Does not assume shape on clusters
- Generic technique
- Finds multiple modes
- Parallelizable

Cons:

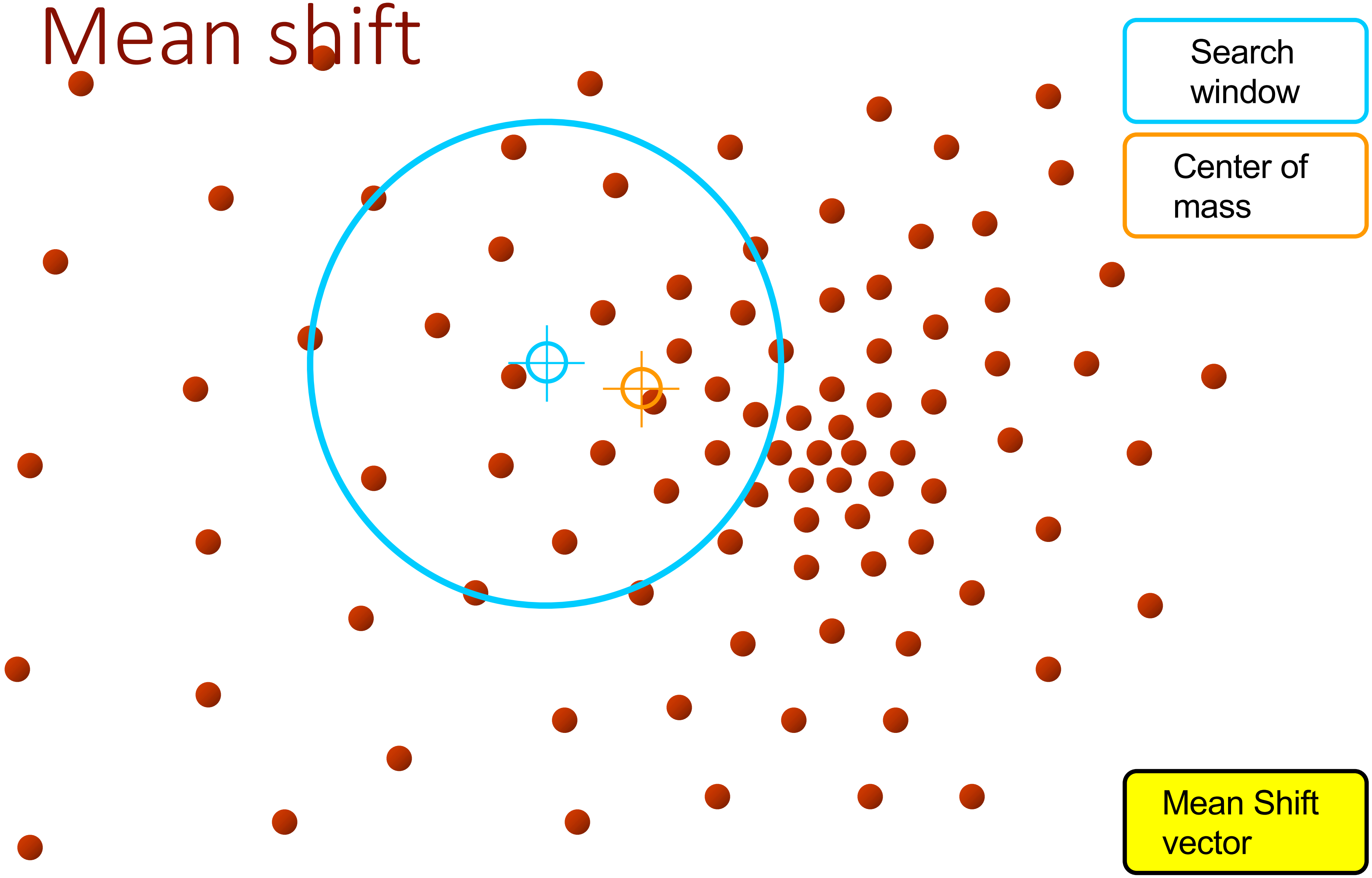
- Slow: $O(DN^2)$ per iteration
- Does not work well for high-dimensional features

Mean shift



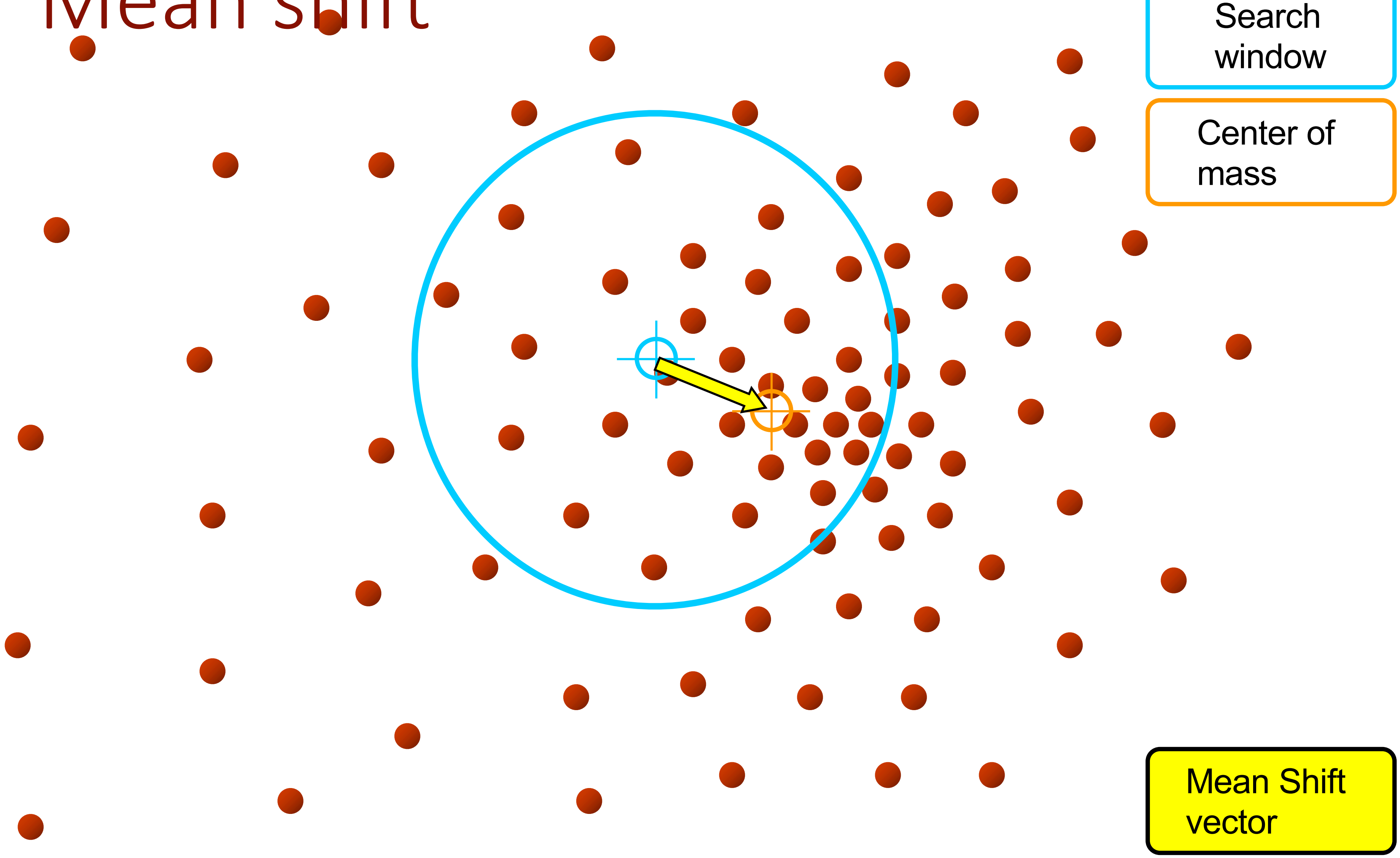
Slide by Y. Ukrainitz & B. Sar

Mean shift

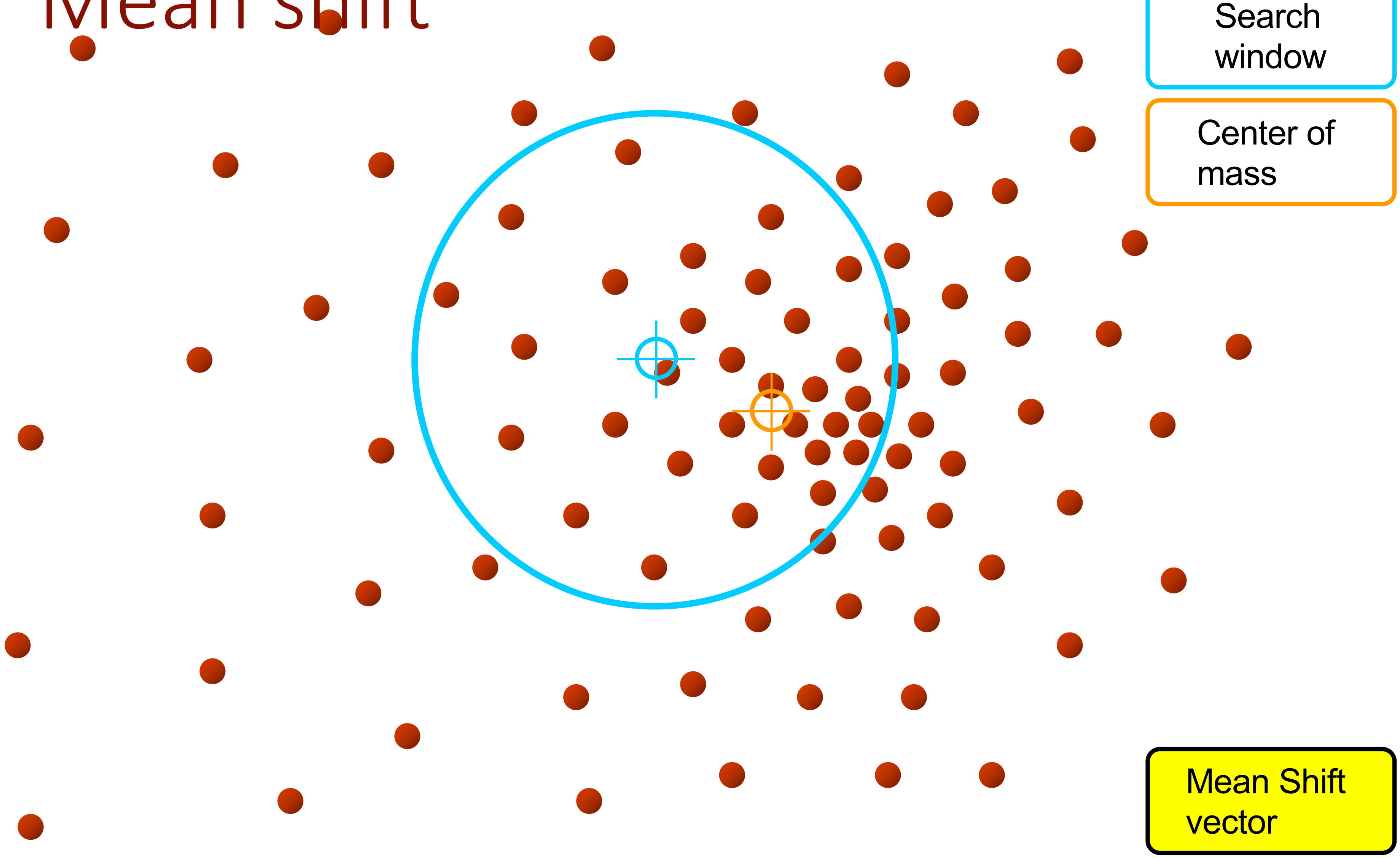


Slide by Y. Ukrainitz & B. Sare

Mean shift

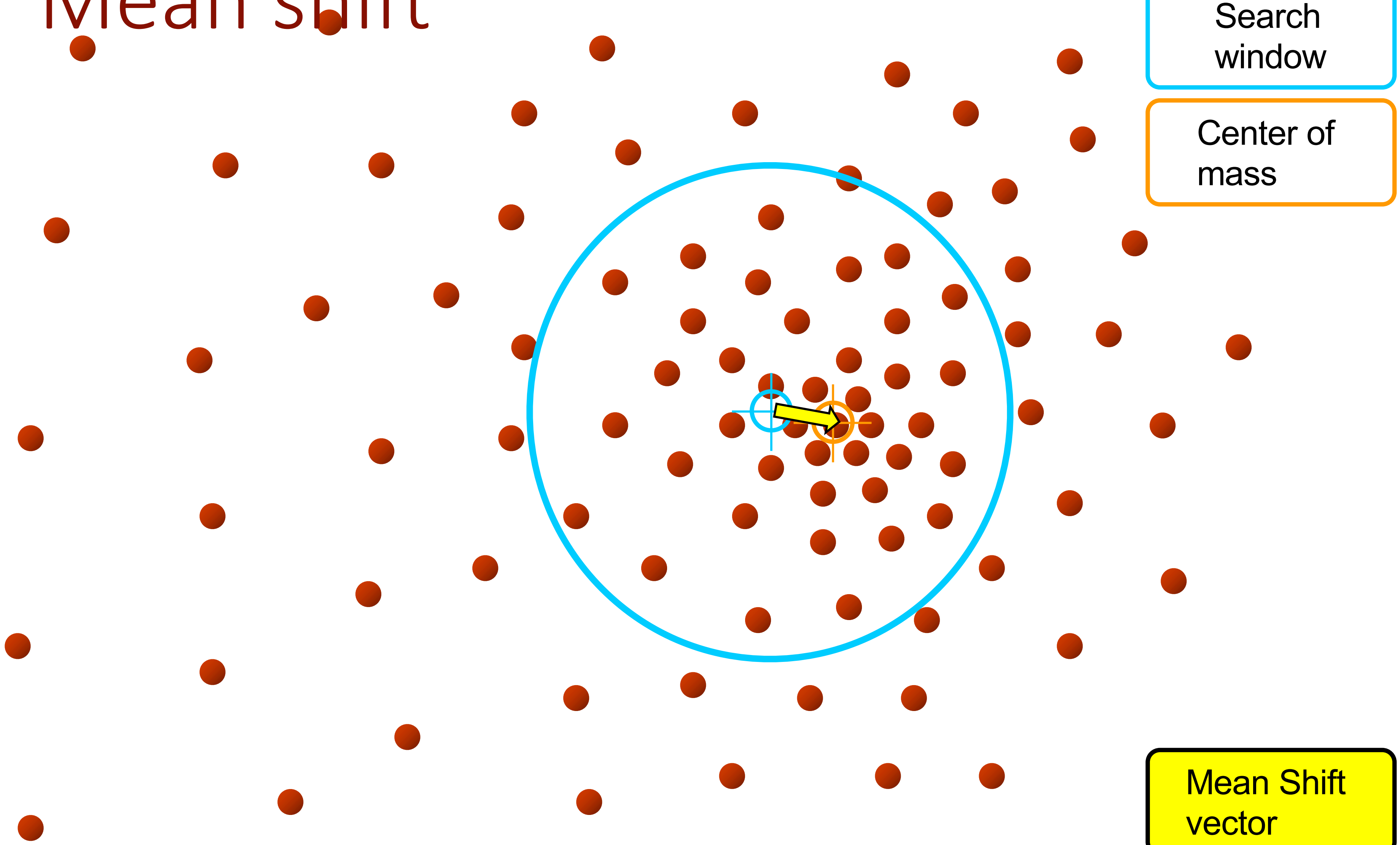


Mean shift



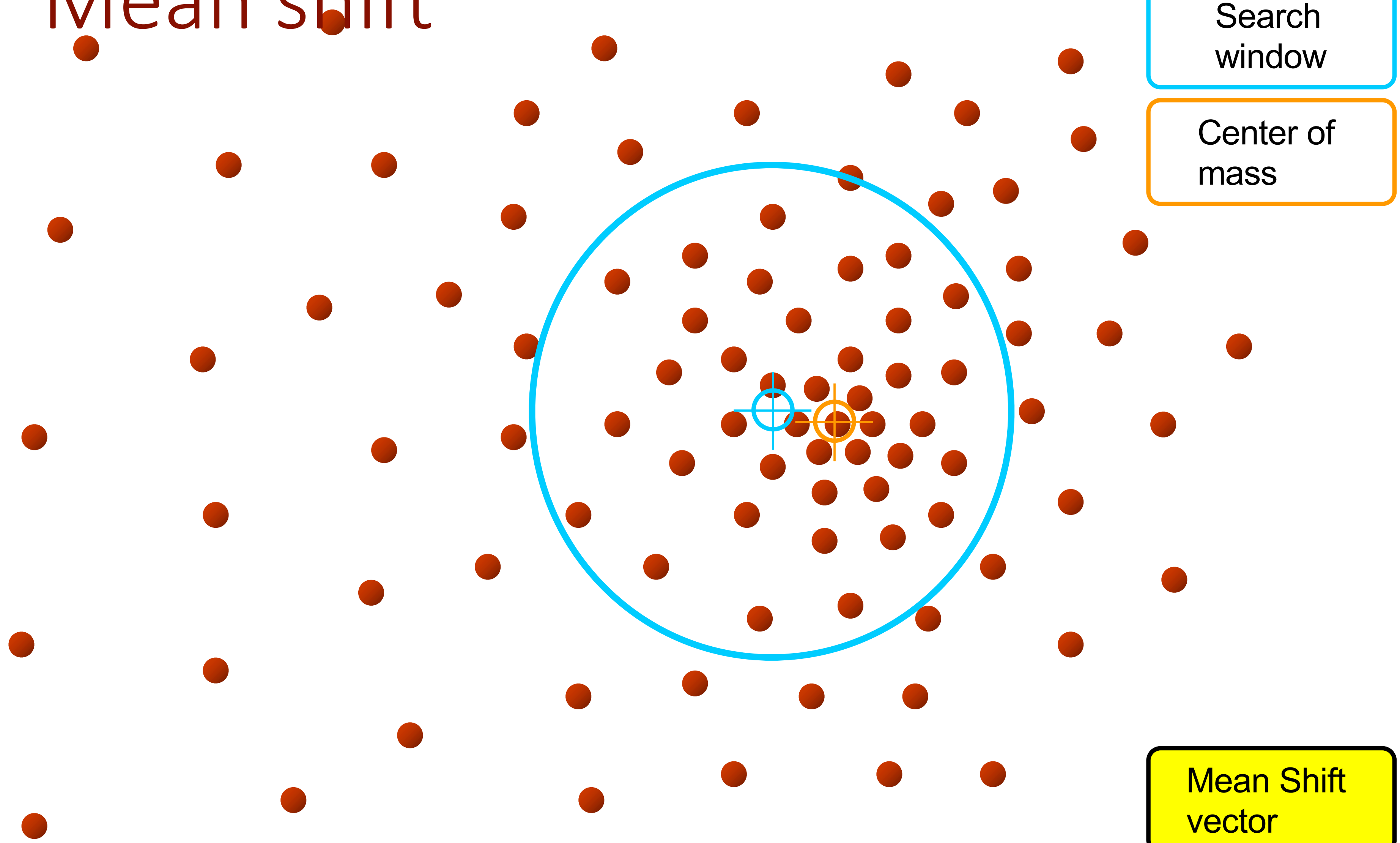
Slide by Y. Ukrainitz & B. Sar

Mean shift



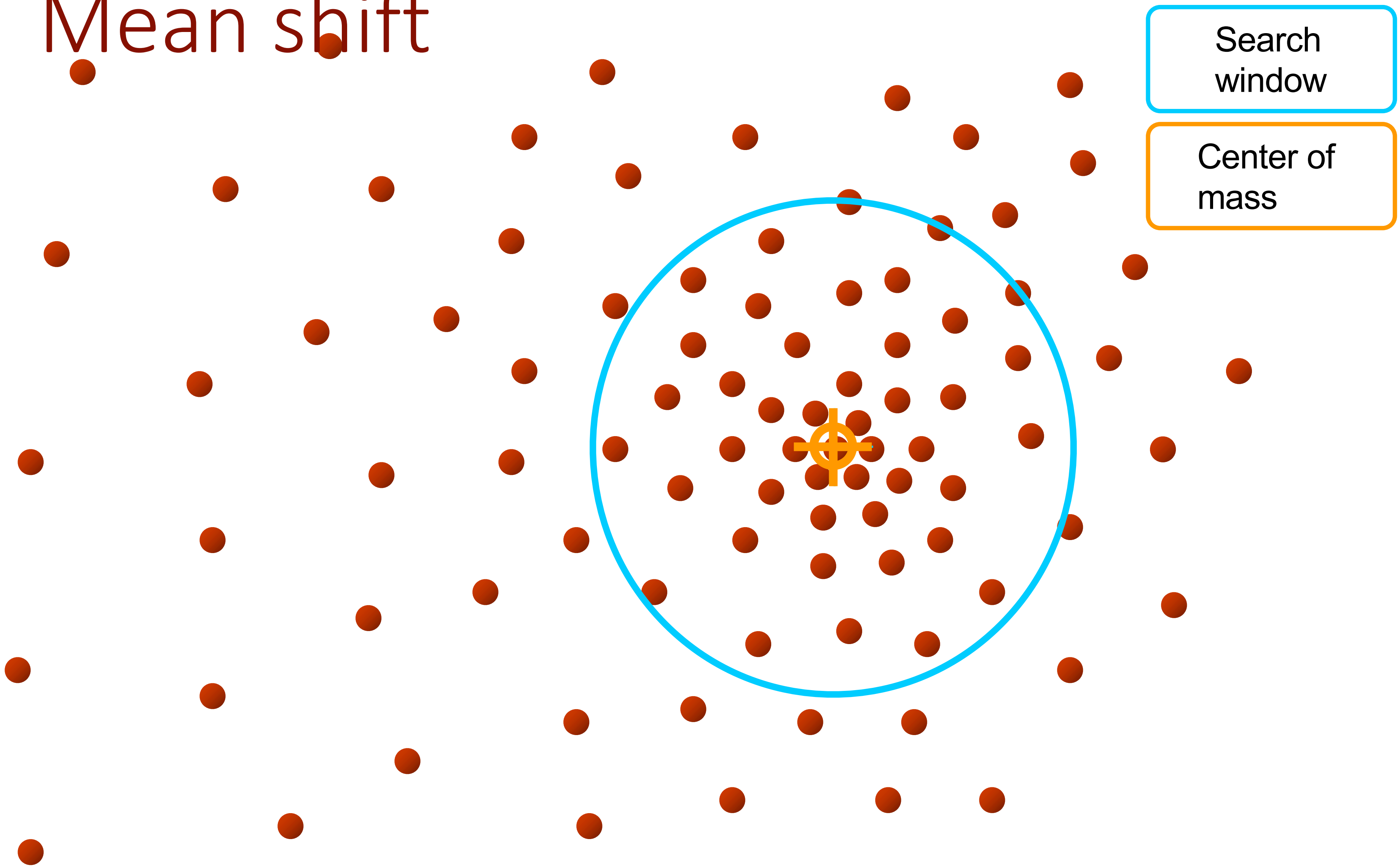
Slide by Y. Ukrainitz & B. Sar

Mean shift



Slide by Y. Ukrainitz & B. Sar

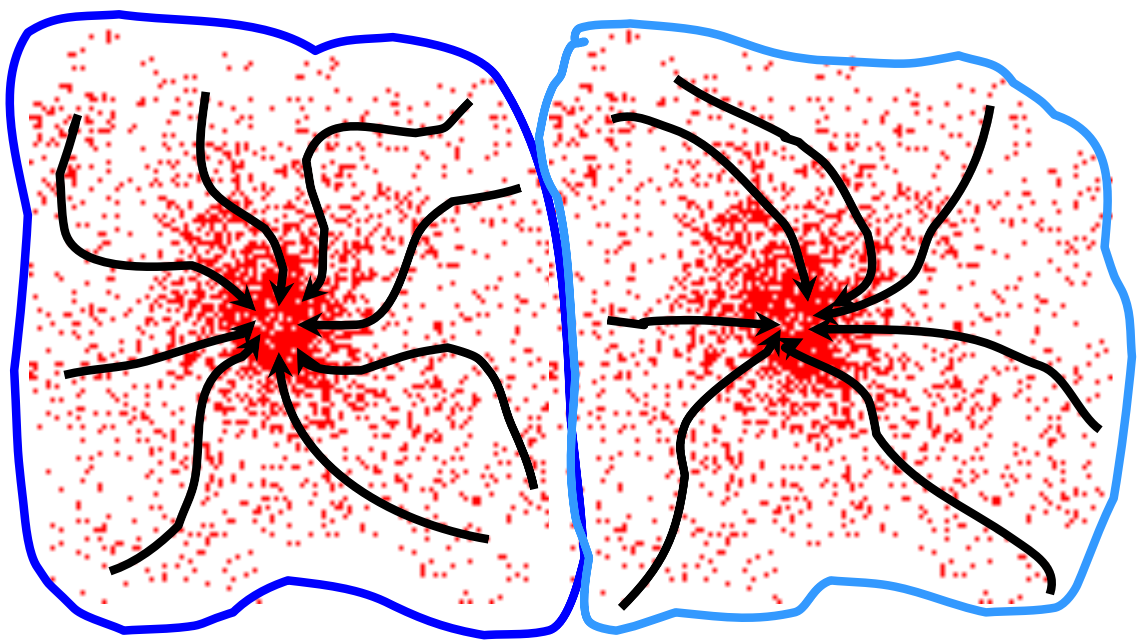
Mean shift



Slide by Y. Ukrainitz & B. Sar

Mean shift clustering

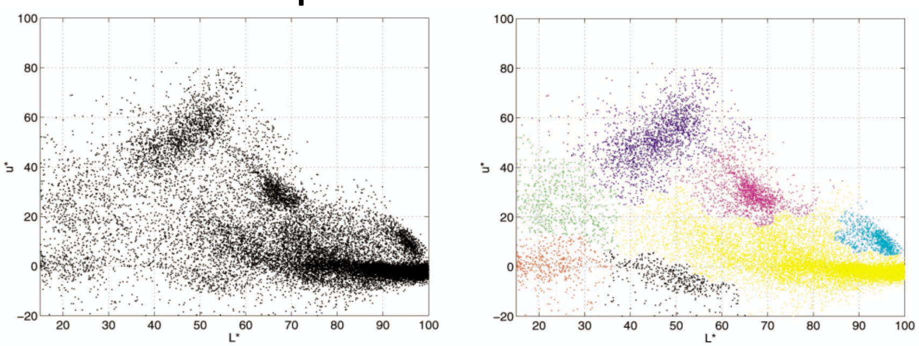
- Cluster all data points in the attraction basin of a mode
- Attraction basin is the region for which all trajectories lead to the same mode — correspond to clusters



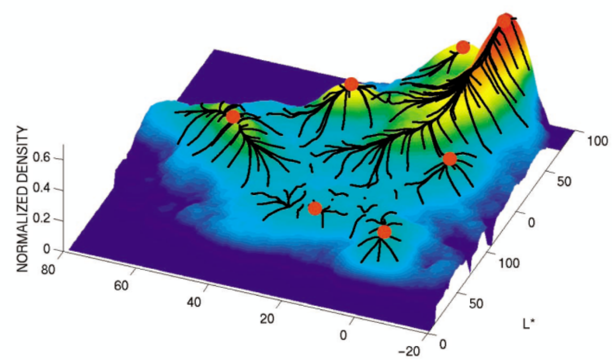
Slide by Y. Ukrainitz & B. Sar

Mean shift for image segmentation

- Feature: L*u*v* color values
- Initialize windows at individual feature points
- Perform mean shift for each window until convergence
- Merge windows that end up near the same “peak” or mode



(a) (b)



Mean shift clustering results

http://www.caip.rutgers.edu/~comanici/MSPAMI/msPamiResults.html

Outline

Clustering basics

K-means: basic algorithm & extensions

Cluster evaluation

Non-parametric mode finding: density estimation

Graph & spectral clustering

Hierarchical clustering

K-Nearest Neighbor

Hierarchical clustering

Agglomerative: a "bottom up" approach where elements start as individual clusters and clusters are merged as one moves up the hierarchy

Divisive: a "top down" approach where elements start as a single cluster and clusters are split as one moves down the hierarchy

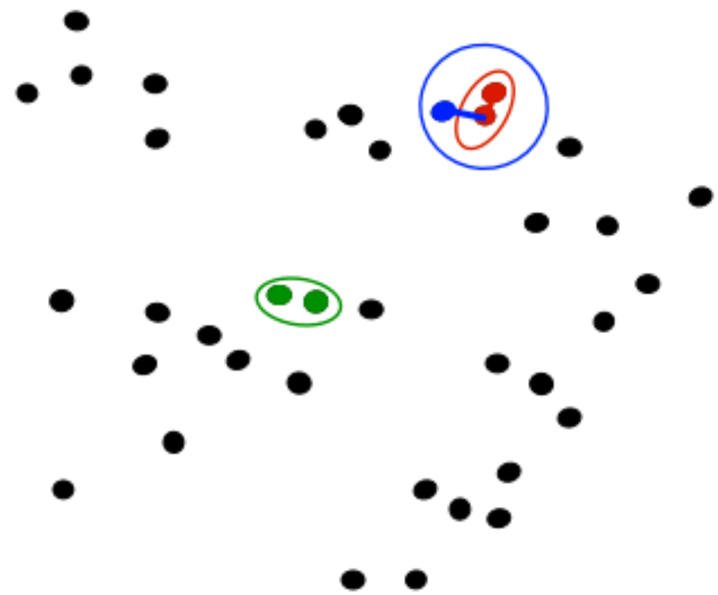
Agglomerative clustering

Agglomerative clustering:

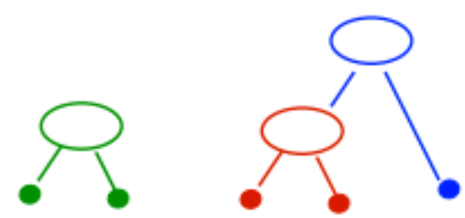
- First merge very similar instances
- Incrementally build larger clusters out of smaller clusters

Algorithm:

- Maintain a set of clusters
- Initially, each instance in its own cluster
- Repeat:
  - Pick the two “closest” clusters
  - Merge them into a new cluster
  - Stop when there’s only one cluster left



Produces not one clustering, but a family of clusterings represented by a dendrogram



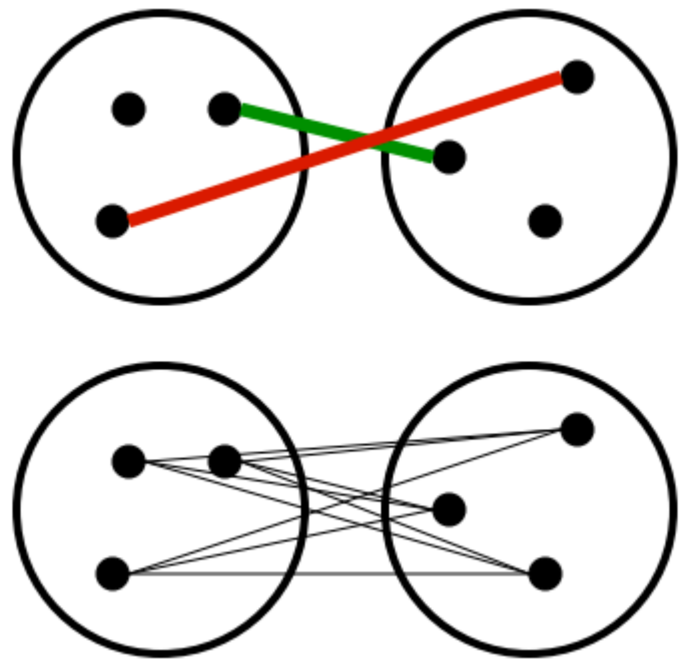
Agglomerative clustering

How should we define "closest" for clusters with multiple elements?

Closest pair: single-link clustering

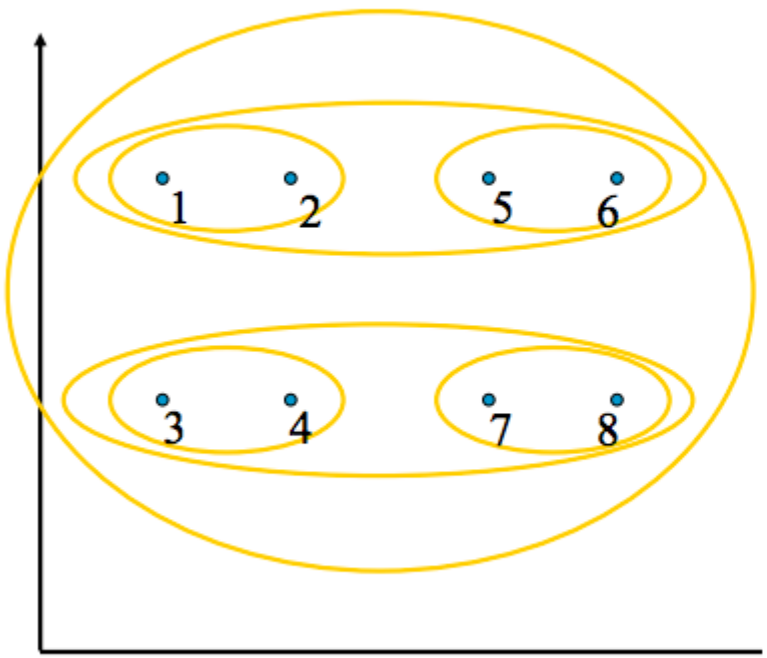
Farthest pair: complete-link clustering

Average of all pairs

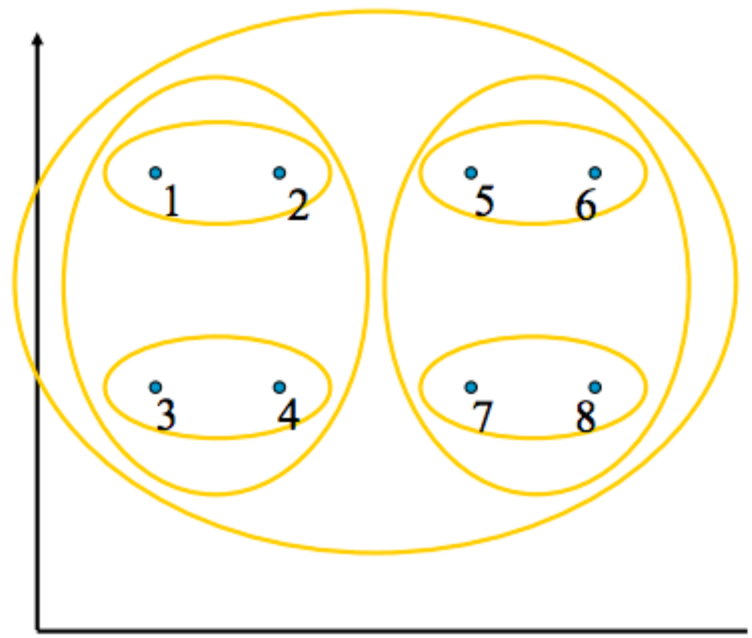


Agglomerative clustering

Closest pair
(single-link clustering)



Farthest pair
(complete-link clustering)



[Pictures from Thorsten Joachims]

Slides credit

Slides are closely following and adapted from Hal Daume's book and Subranshu Maji's course.

The fruit classification dataset is from Iain Murray at University of Edinburgh

http://homepages.inf.ed.ac.uk/imurray2/teaching/oranges_and_lemons/.

The slides on texture synthesis are from Efros and Leung's ICCV 2009 presentation.

Many images are from the Berkeley segmentation benchmark

http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds

Normalized cuts image segmentation:

http://www.timotheecour.com/research.html

Summary

- Clustering is an example of unsupervised learning
- Partitions or hierarchy
- Several partitioning algorithms:
  - k-means: simple, efficient and often works in practice
    - k-means++ for better initialization
  - mean shift: modes of density
    - slow but suited for problems with unknown number of clusters with varying shapes and sizes
  - spectral clustering: clustering as graph partitions
    - solve $(\mathbf{D} - \mathbf{W})x = \lambda \mathbf{D}x$ followed by k-means
- Hierarchical clustering methods:
  - Agglomerative or divisive
    - single-link, complete-link and average-link

Slides credit

Slides are closely following and adapted from Hal Daume's book and Subranshu Maji's course.

The fruit classification dataset is from Iain Murray at University of Edinburgh

http://homepages.inf.ed.ac.uk/imurray2/teaching/oranges_and_lemons/.

The slides on texture synthesis are from Efros and Leung's ICCV 2009 presentation.

Many images are from the Berkeley segmentation benchmark

http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds

Normalized cuts image segmentation:

http://www.timotheecour.com/research.html