# Clustering

Subhransu Maji

CMPSCI 689: Machine Learning

2 April 2015

7 April 2015

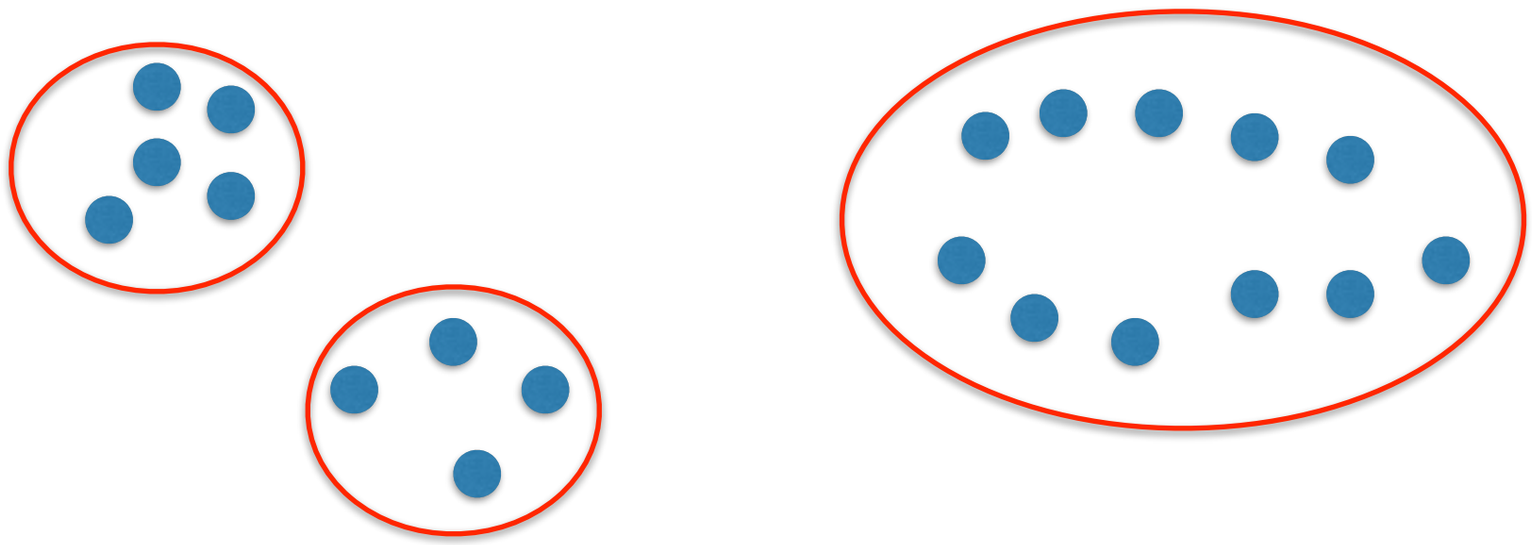
# So far in the course

- Supervised learning: learning with a teacher
  - You had training data which was (feature, label) pairs and the goal was to learn a mapping from features to labels

- Unsupervised learning: learning without a teacher
  - Only features and no labels
- Why is unsupervised learning useful?
  - Discover hidden structures in the data — clustering ← today
  - Visualization — dimensionality reduction
    - ➡ lower dimensional features might help learning

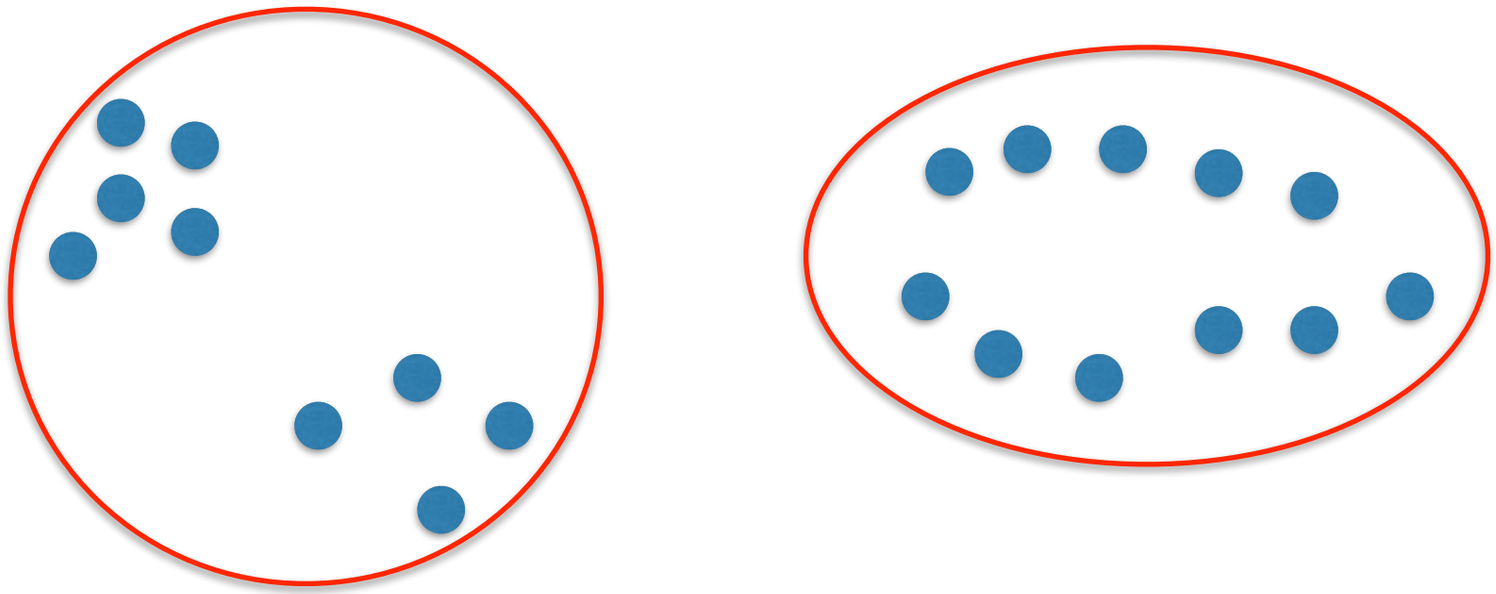
# Clustering

- Basic idea: group together similar instances
- Example: 2D points



# Clustering

- Basic idea: group together similar instances
- Example: 2D points



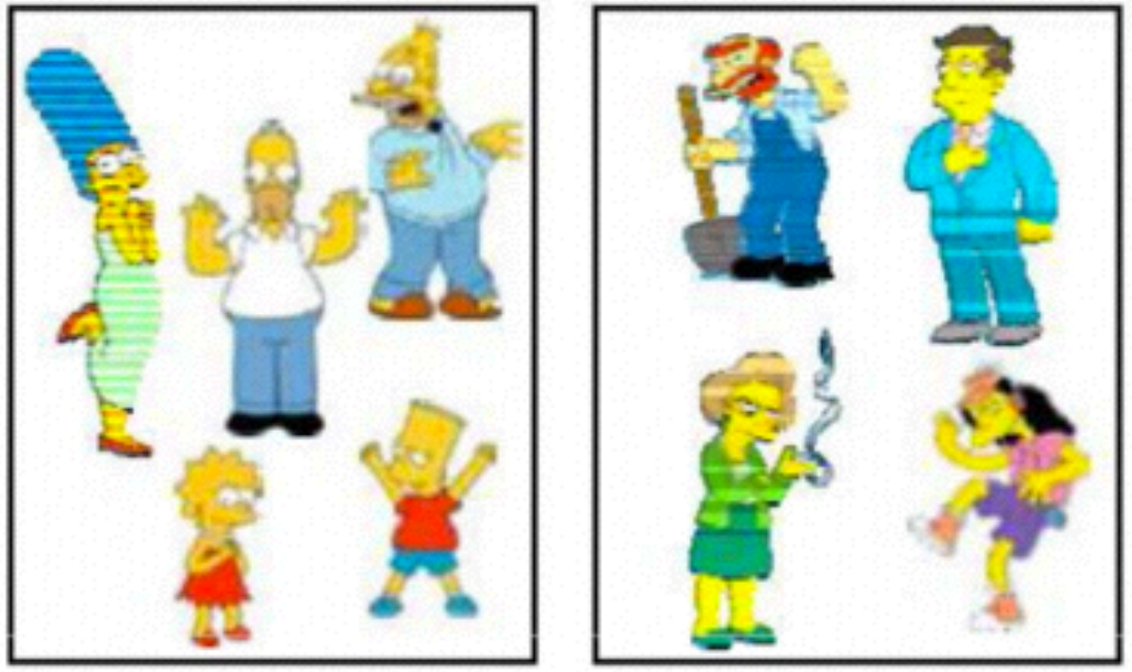
- What could similar mean?
  - One option: small Euclidean distance (squared)

$$\text{dist}(\mathbf{x}, \mathbf{y}) = ||\mathbf{x} - \mathbf{y}||_2^2$$

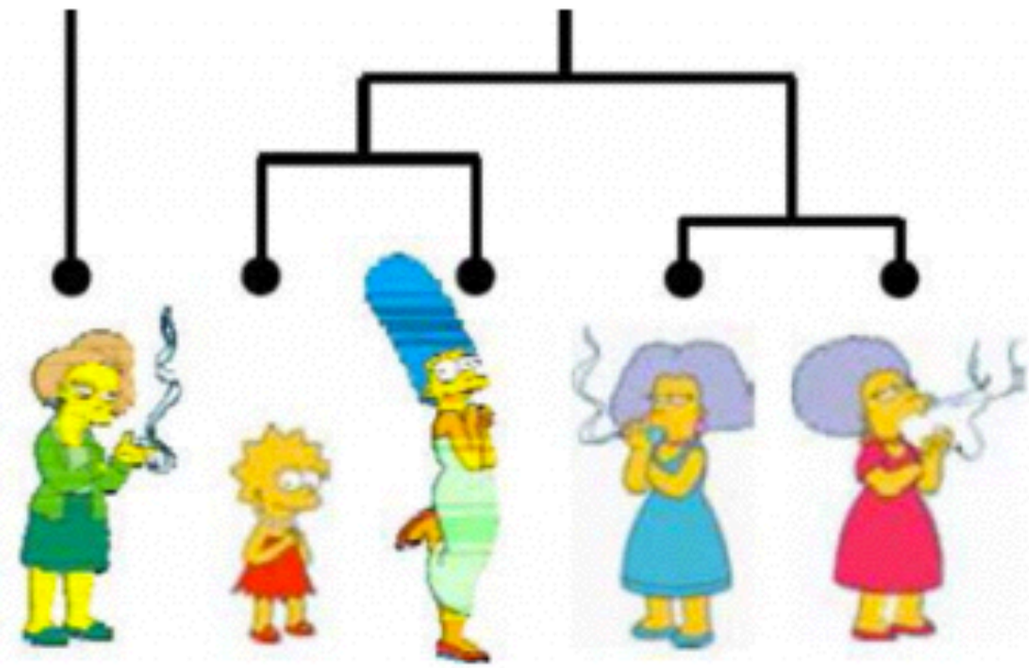
  - Clustering results are crucially dependent on the measure of similarity (or distance) between points to be clustered

# 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



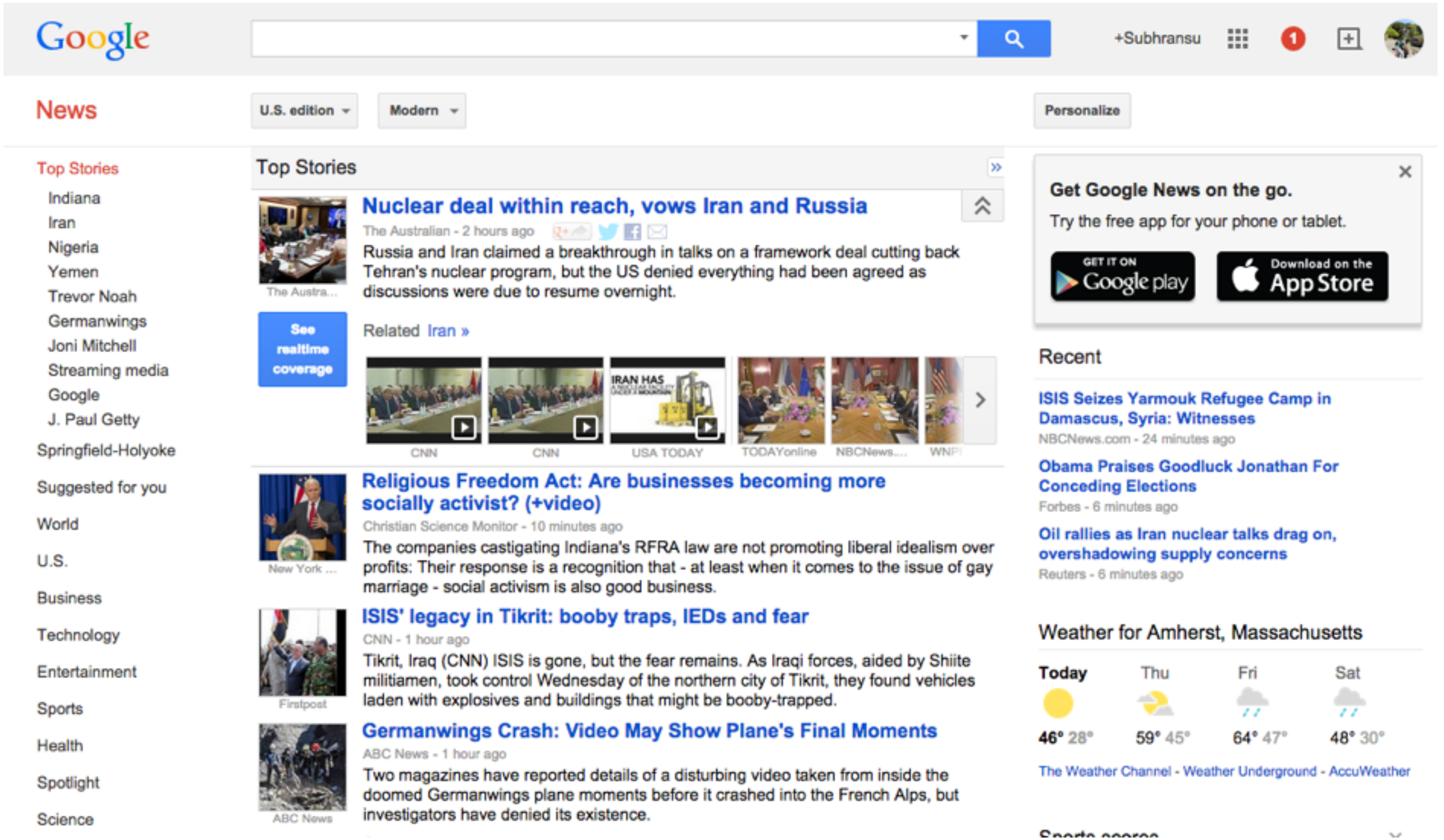
# Clustering examples

- Image segmentation: break up the image into similar regions

image credit: Berkeley segmentation benchmark

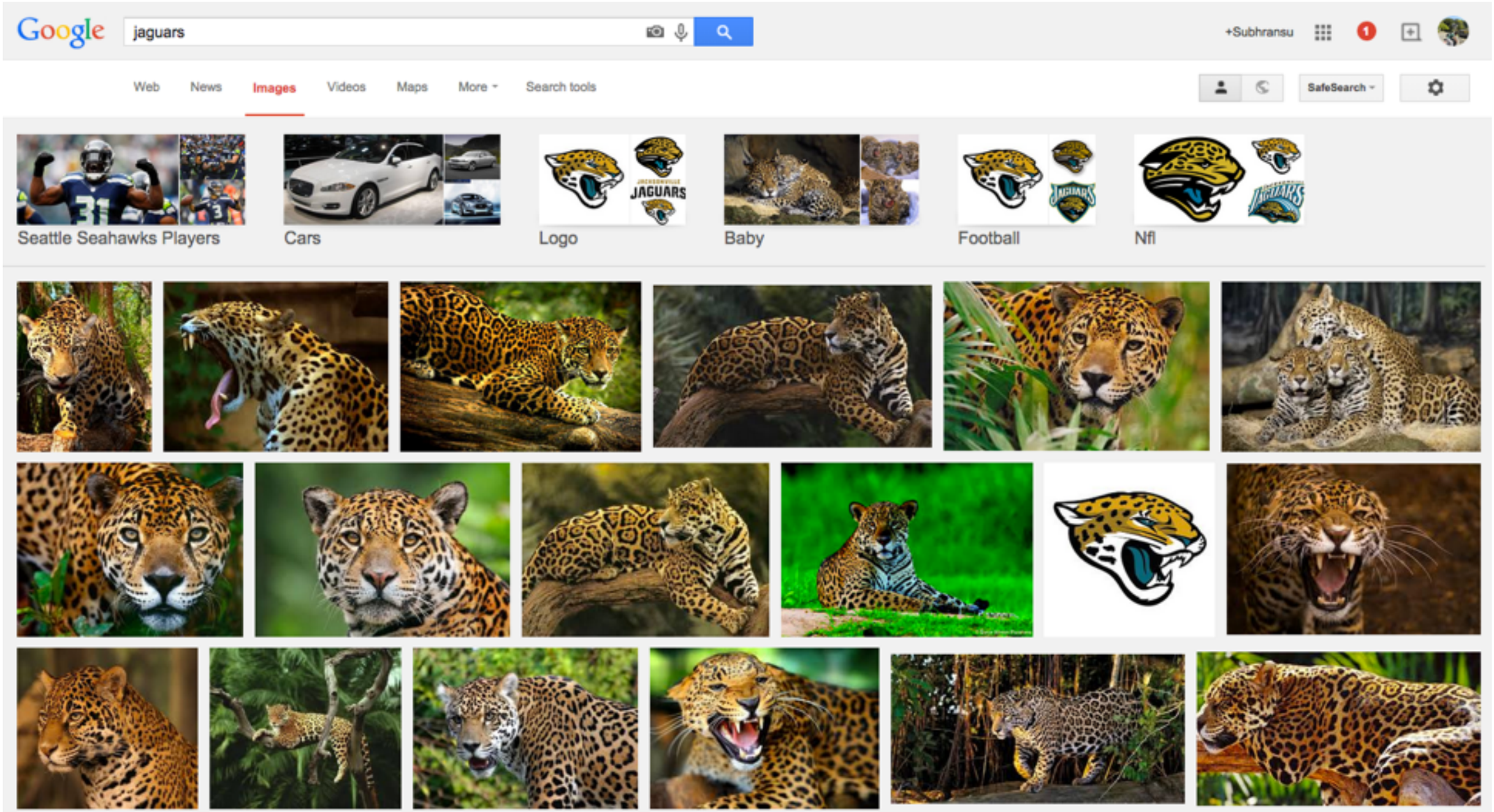
# Clustering examples

- Clustering news articles



# Clustering examples

- Clustering queries



# Clustering examples

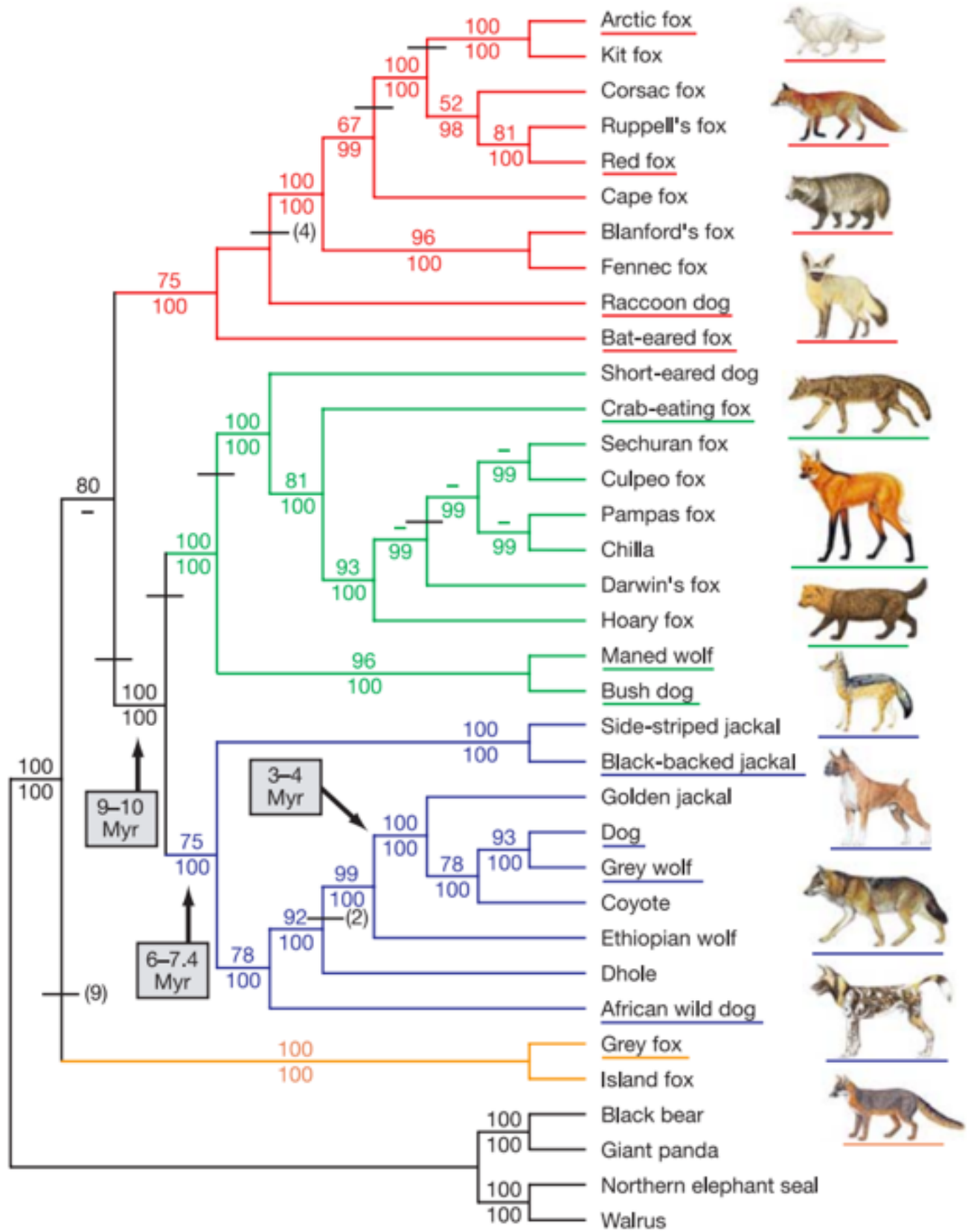
- Clustering people by space and time


image credit: Pilho Kim

# Clustering examples

- Clustering species (phylogeny)

phylogeny of canid species
(dogs, wolves, foxes, jackals, etc)




[K. Lindblad-Toh, Nature 2005]

# Clustering using k-means

- Given **($x_1$, $x_2$, …, $x_n$)** partition the **n** observations into **k (≤ n)** sets **S =** {$S_1$, $S_2$, …, $S_k$} so as to minimize the within-cluster sum of squared distances

- The objective is to minimize:

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} ||\mathbf{x} - \mu_i||^2$$

cluster center

# Lloyd's algorithm for k-means

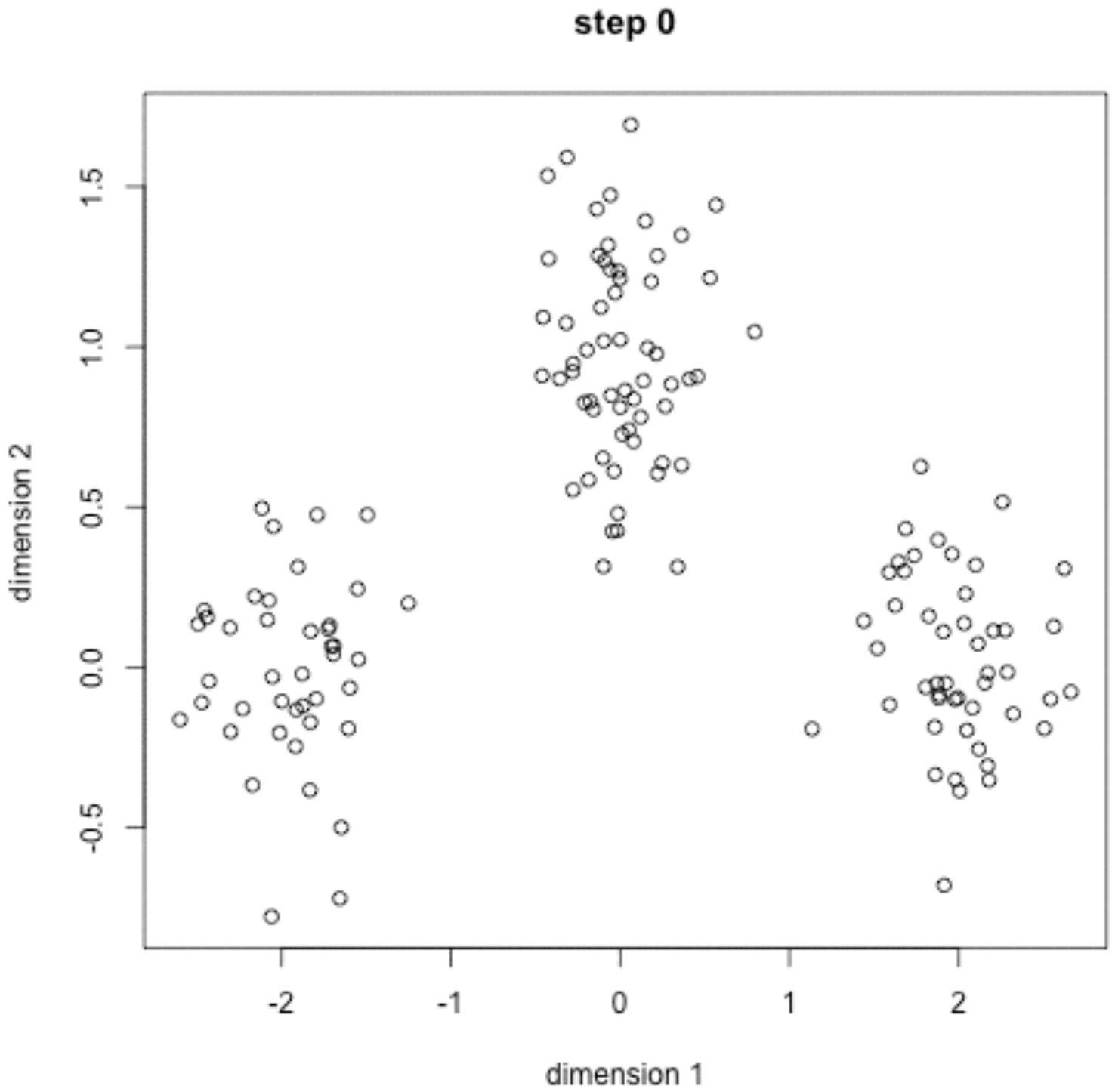
- Initialize k centers by picking k points randomly among all the points
- Repeat till convergence (or max iterations)
  - Assign each point to the nearest center (assignment step)

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} ||\mathbf{x} - \mu_i||^2$$

  - Estimate the mean of each group (update step)

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} ||\mathbf{x} - \mu_i||^2$$

# k-means in action




http://simplystatistics.org/2014/02/18/k-means-clustering-in-a-gif/

# Properties of the Lloyd's algorithm

- Guaranteed to converge in a finite number of iterations
  - The objective decreases monotonically over time
  - Local minima if the partitions don't change. Since there are 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]

k-means++

# k-means++ algorithm

- A way to pick the good initial centers
  - Intuition: spread out the k initial cluster centers
- The algorithm proceeds normally once the centers are initialized
- k-means++ algorithm for initialization:
  1. Chose one center uniformly at random among all the points
  2. For each point $\mathbf{x}$, compute $D(\mathbf{x})$, the distance between x and the nearest center that has already been chosen
  3. Chose one new data point at random as a new center, using a weighted probability distribution where a point $\mathbf{x}$ is chosen with a probability proportional to $D(\mathbf{x})^2$
  4. Repeat Steps 2 and 3 until k centers have been chosen

- [Arthur and Vassilvitskii'07] The approximation quality is O(log k) in expectation

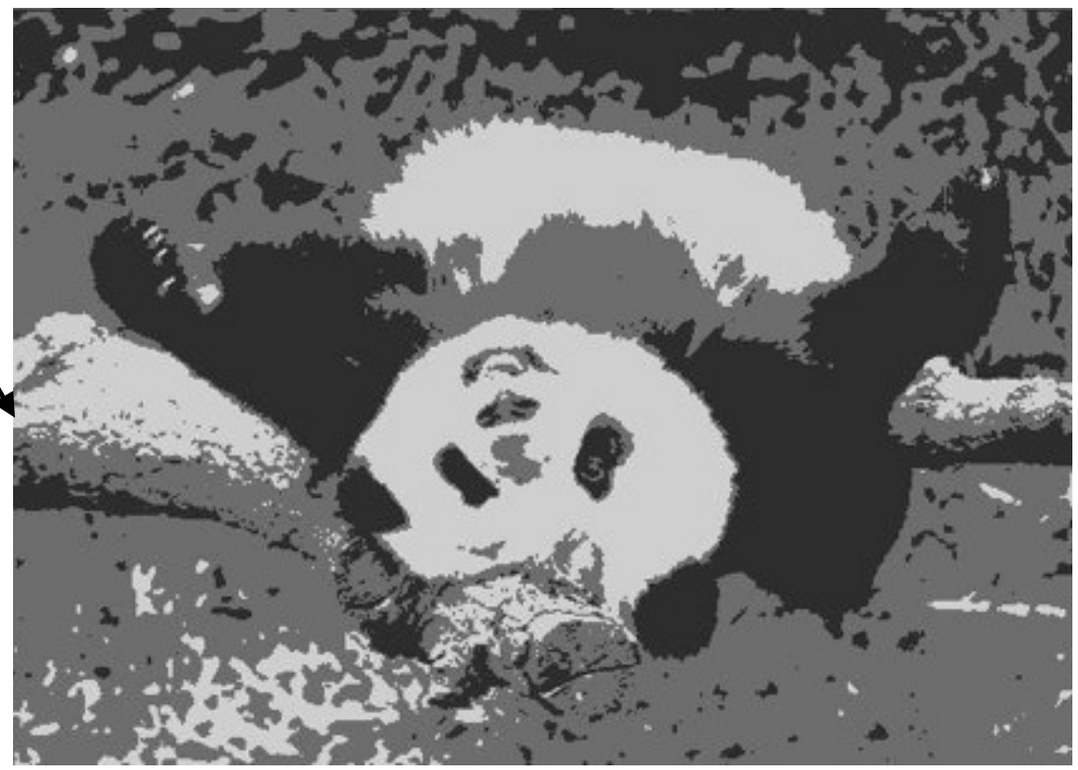
http://en.wikipedia.org/wiki/K-means%2B%2B

# k-means for image segmentation

K=2

Grouping pixels based on **intensity** similarity

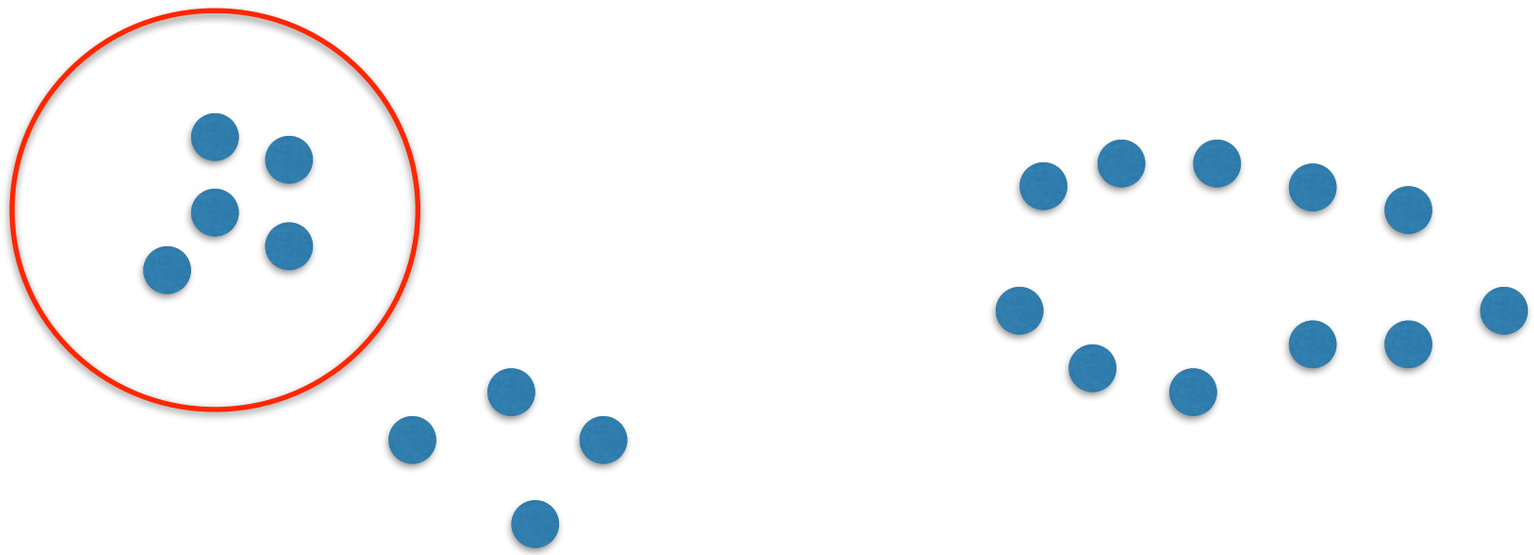
K=3



feature space: intensity value (1D)

# 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 — automatically determines the number of clusters



Kernel density estimator

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

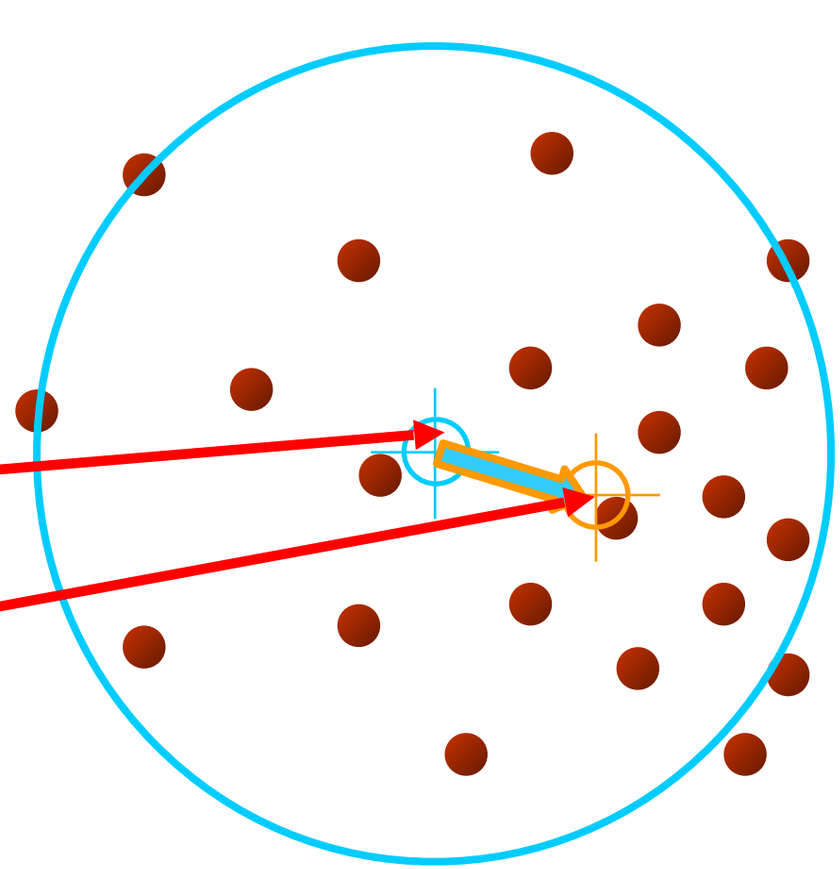
Small *h* implies more modes (bumpy distribution)

# Mean shift algorithm

- Mean shift procedure:
  - For each point, repeat till convergence
  - Compute mean shift vector
  - Translate the kernel window by **m(x)**
- Simply following the gradient of density

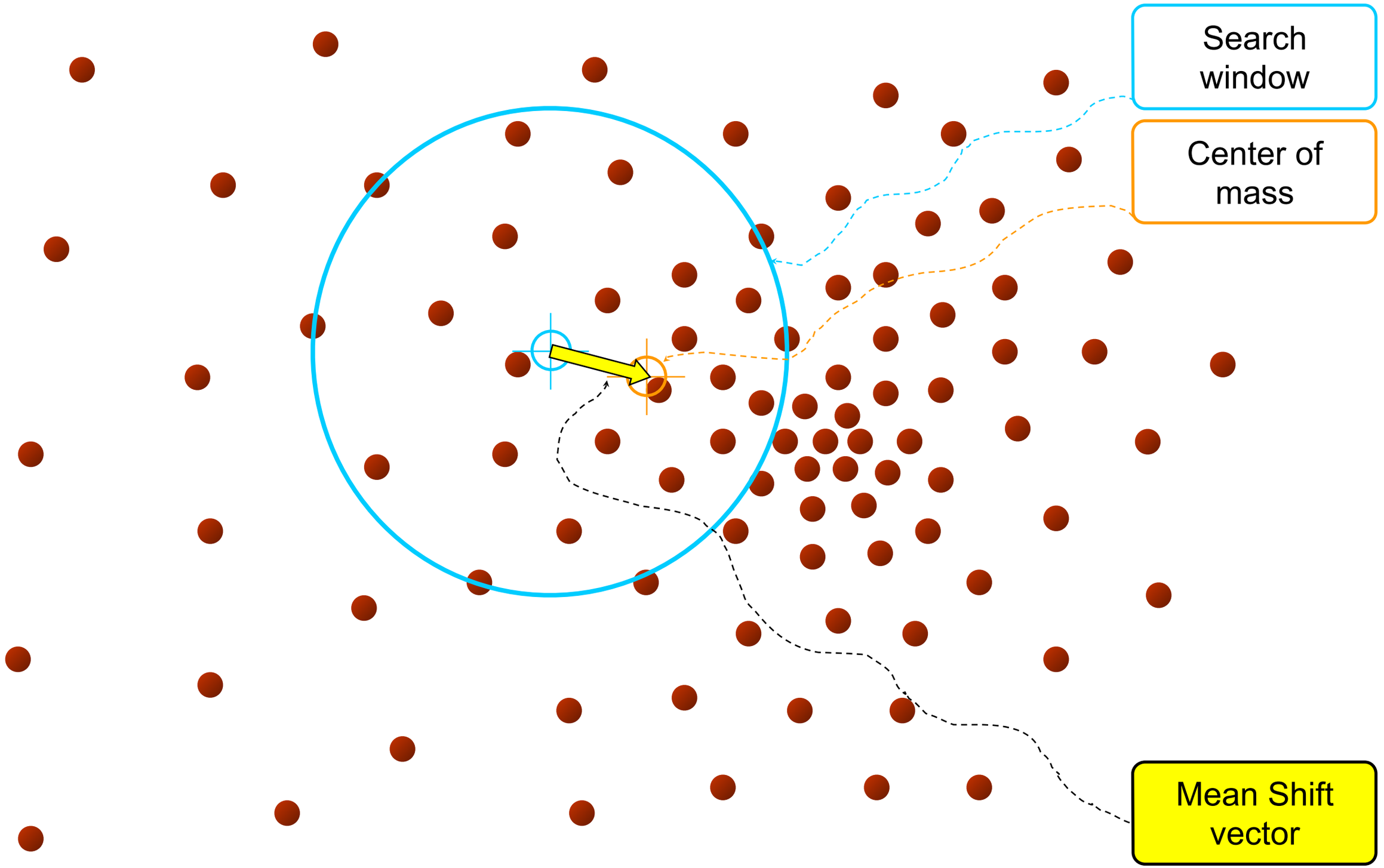
$$\exp\left(-\frac{||\mathbf{x}-\mathbf{x}_i||^2}{h}\right)$$

$$\mathbf{m}(\mathbf{x}) = \left[\frac{\sum_{i=1}^{n}\mathbf{x}_i g\left(\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{h}\right)}{\sum_{i=1}^{n} g\left(\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{h}\right)}\right] - \mathbf{x}$$



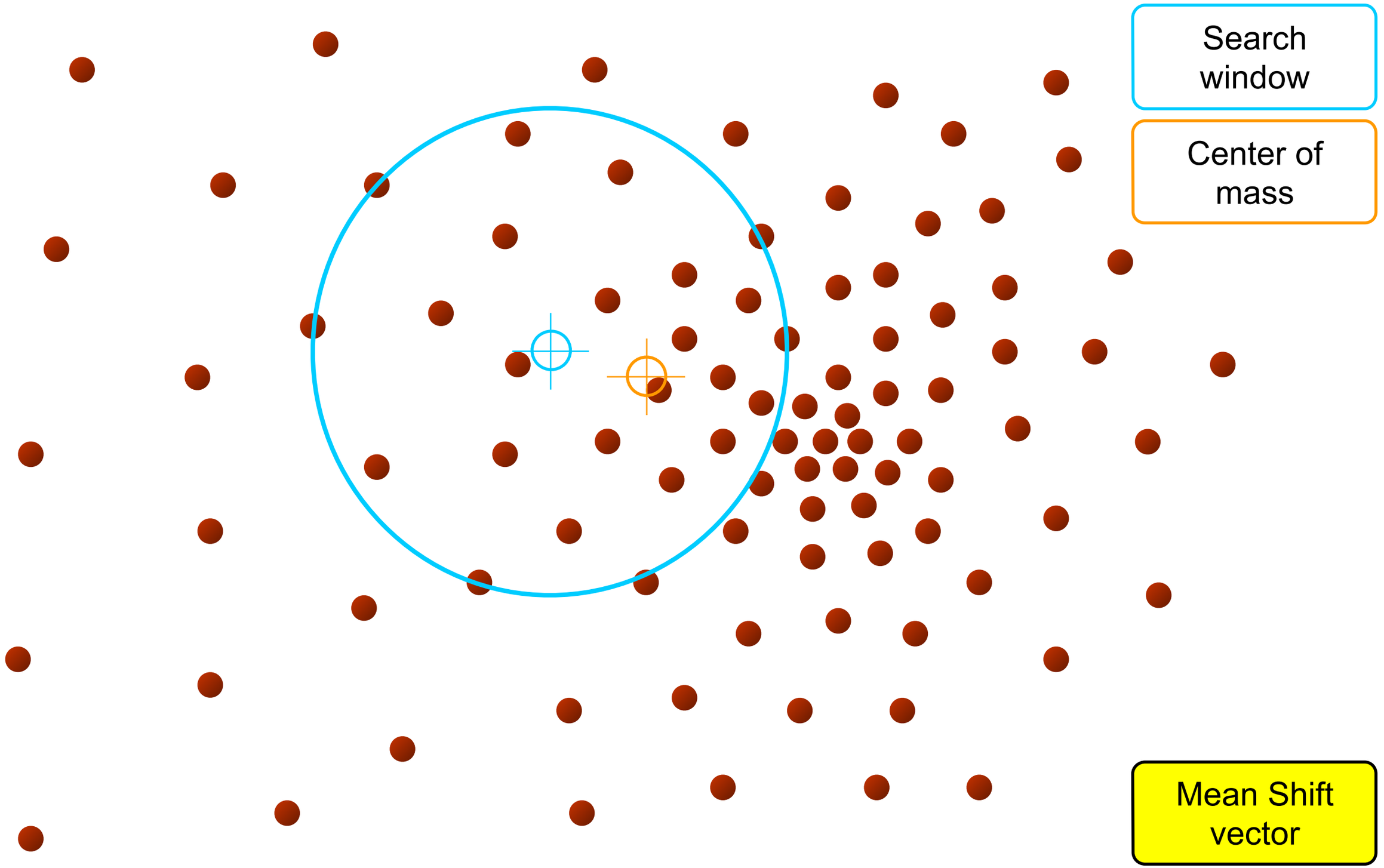
Slide by Y. Ukrainitz & B. Sarel

# Mean shift



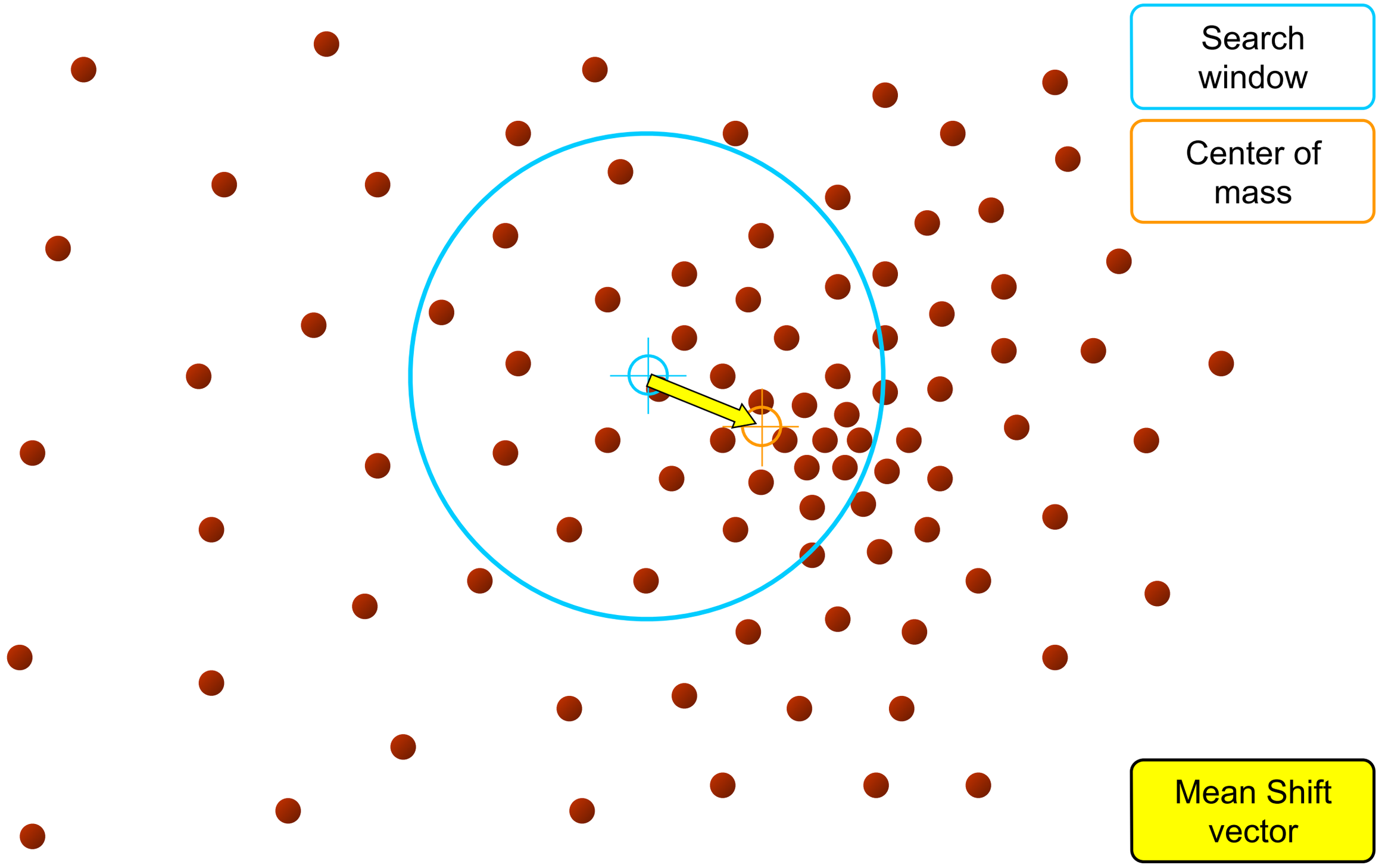

Slide by Y. Ukrainitz & B. Sarel

# Mean shift



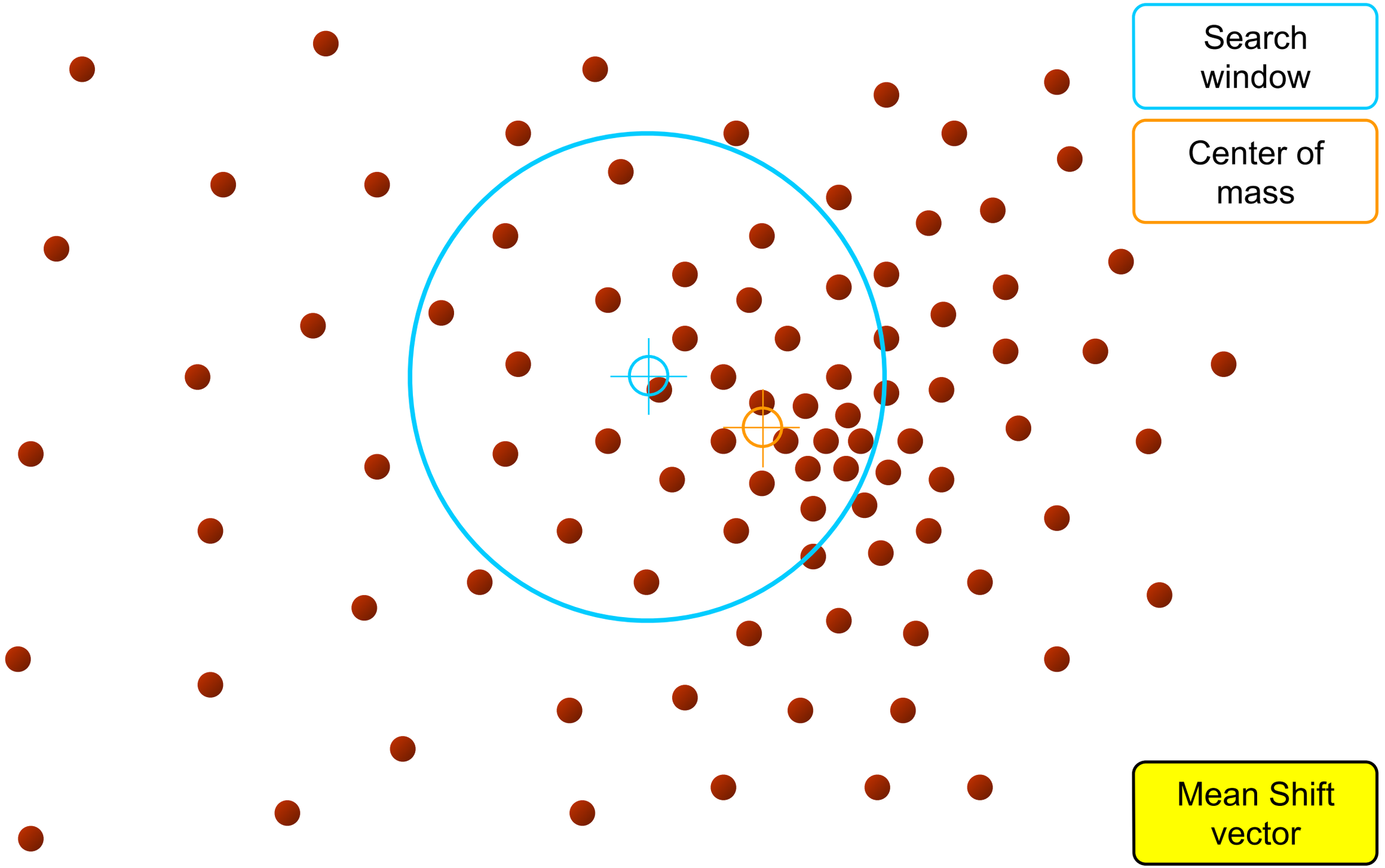

Slide by Y. Ukrainitz & B. Sarel

# Mean shift



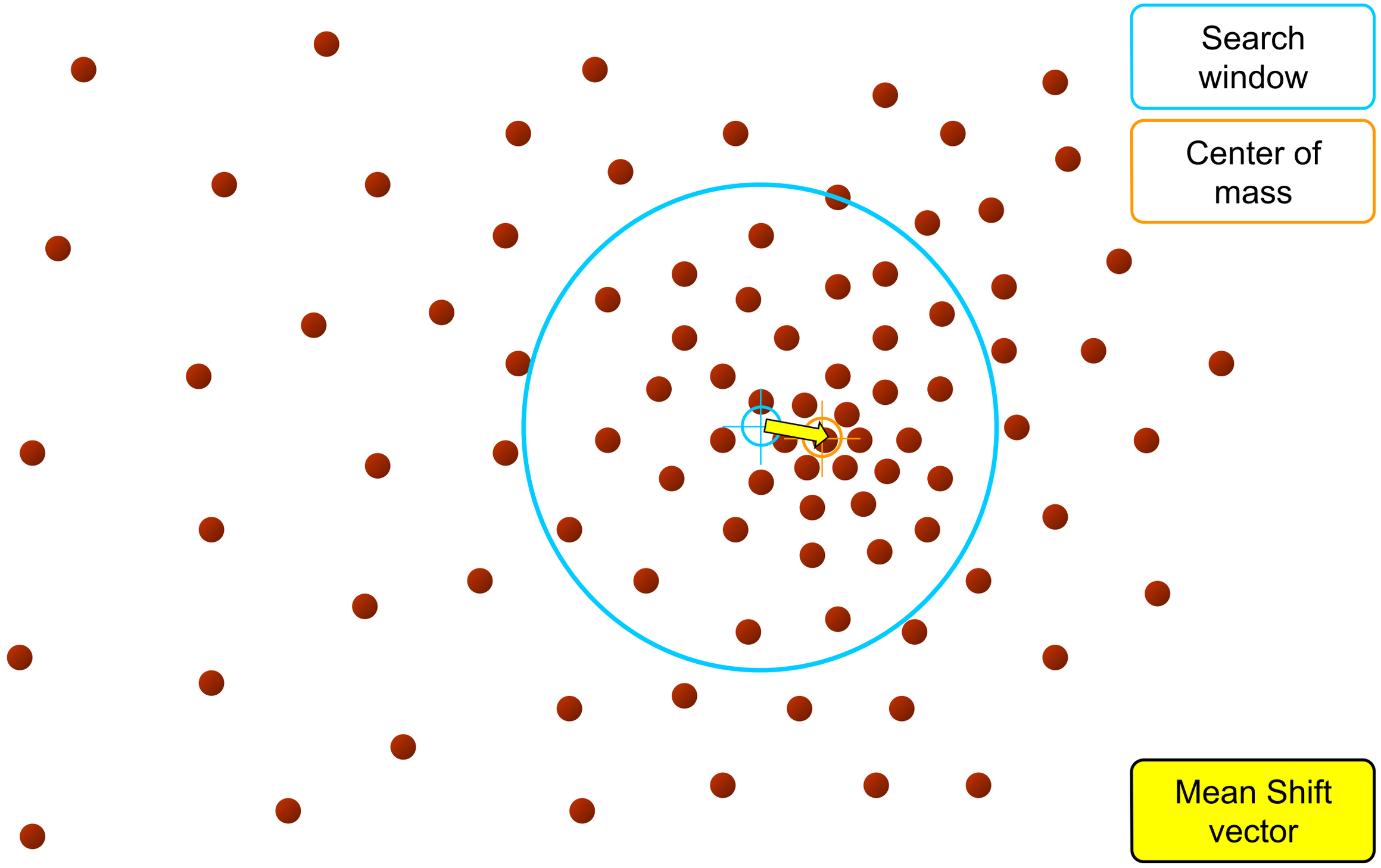

Slide by Y. Ukrainitz & B. Sarel

# Mean shift



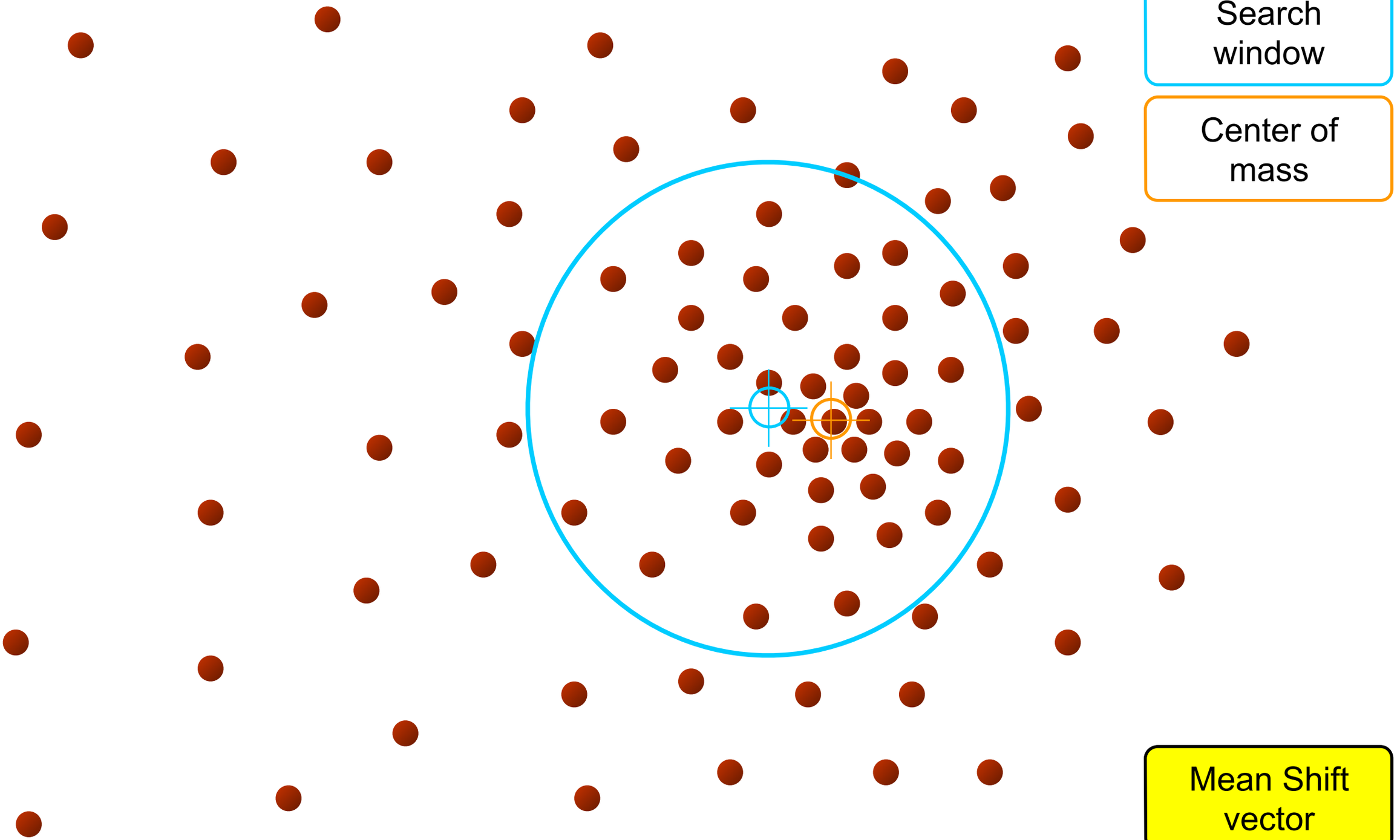

Slide by Y. Ukrainitz & B. Sarel

# Mean shift



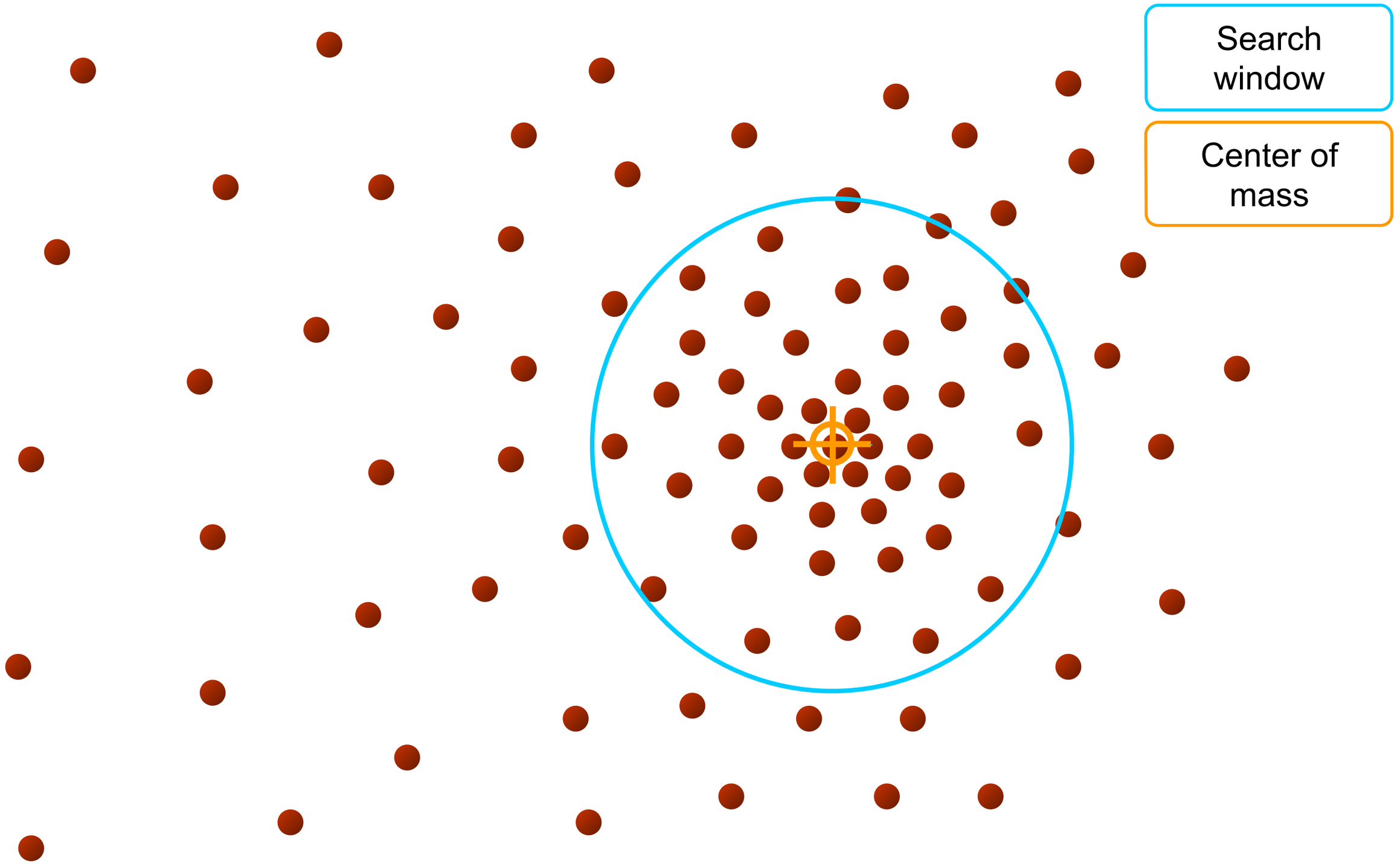

Slide by Y. Ukrainitz & B. Sarel

# Mean shift




Slide by Y. Ukrainitz & B. Sarel

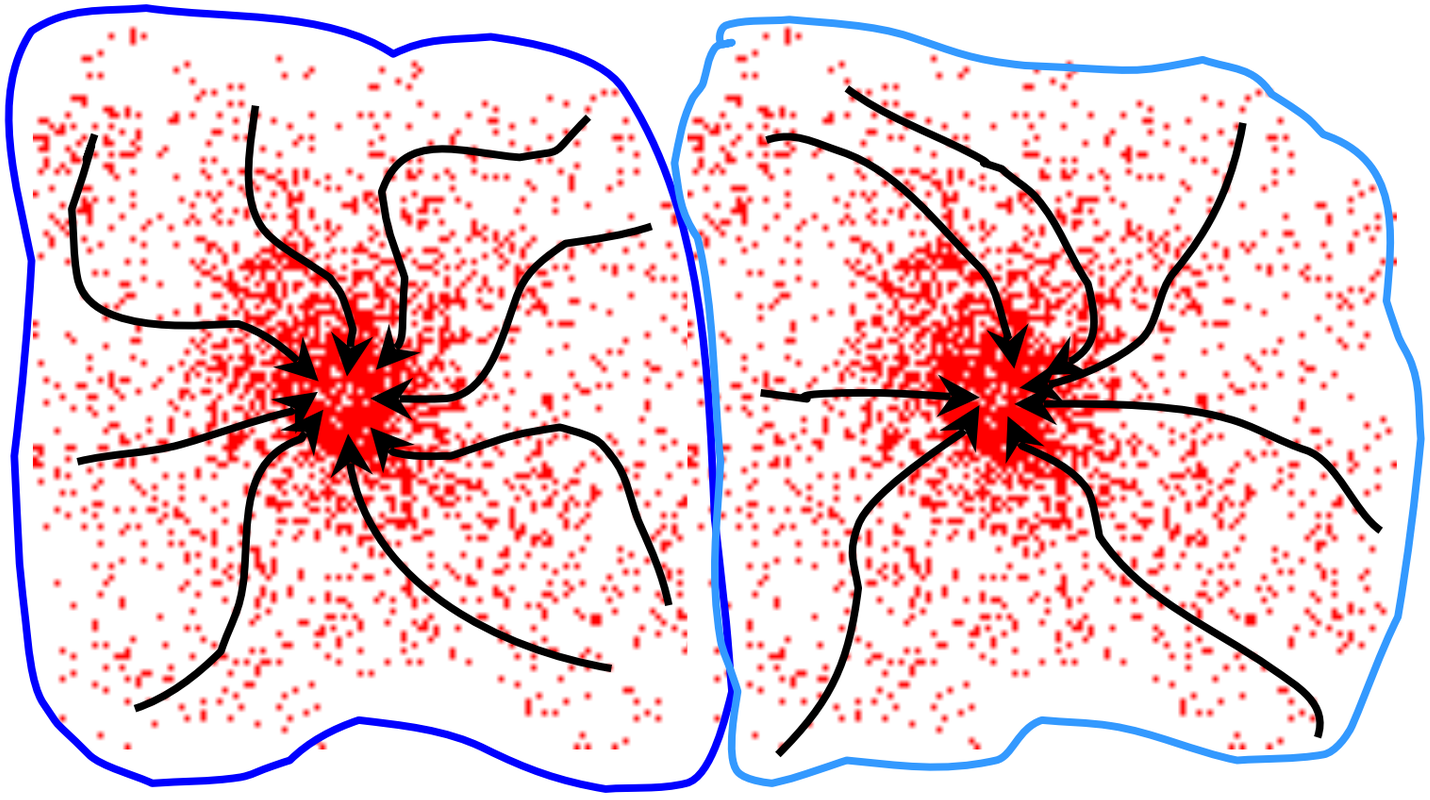
# Mean shift




Slide by Y. Ukrainitz & B. Sarel

# Mean shift clustering

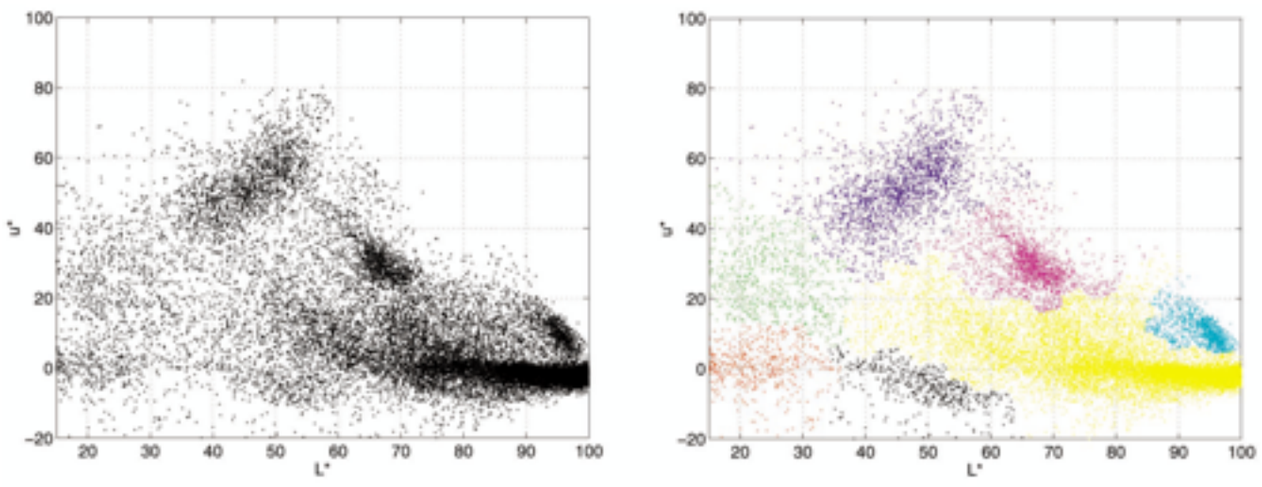
- Cluster all data points in the attraction basin of a mode
- Attraction basin is the region for which all trajectories lead to the same mode — correspond to clusters



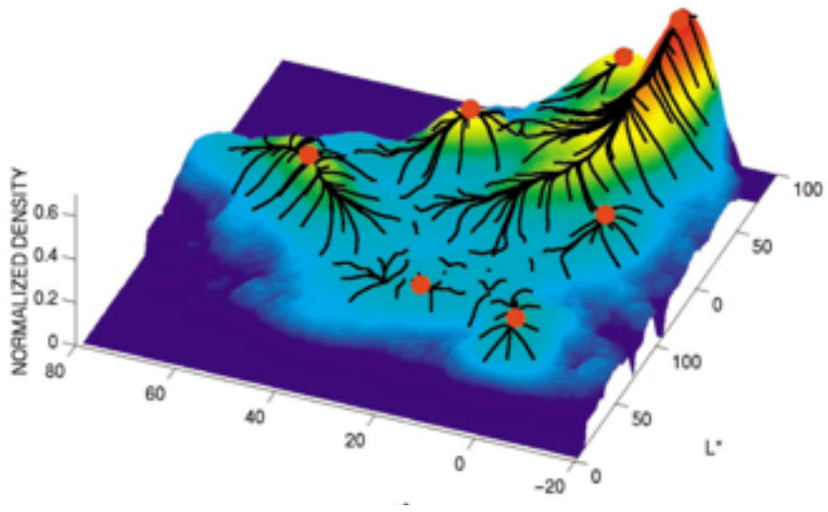
Slide by Y. Ukrainitz & B. Sarel

# 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

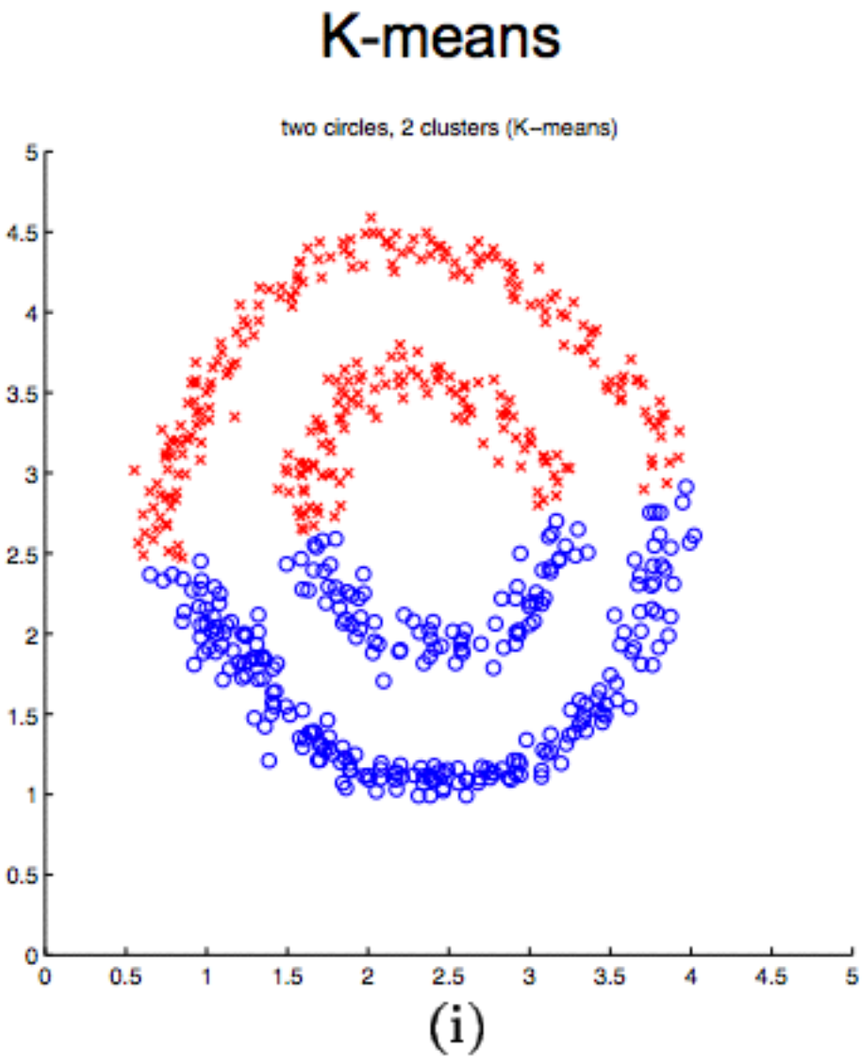


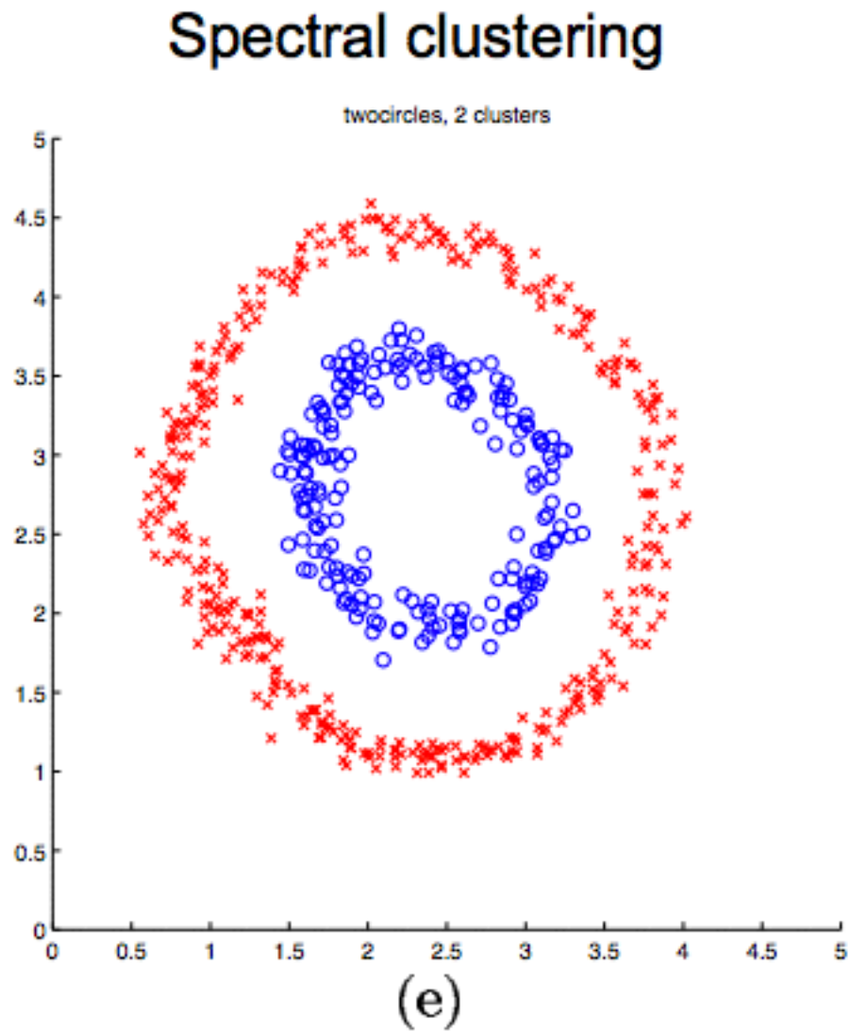

http://www.caip.rutgers.edu/~comanici/MSPAMI/msPamiResults.html

# Mean shift discussion

- Pros:
  - Does not assume shape on clusters
  - One parameter choice (window size)
  - Generic technique
  - Finds multiple modes
- Cons:
  - Selection of window size
  - Is rather expensive: $O(DN^2)$ per iteration
  - Does not work well for high-dimensional features

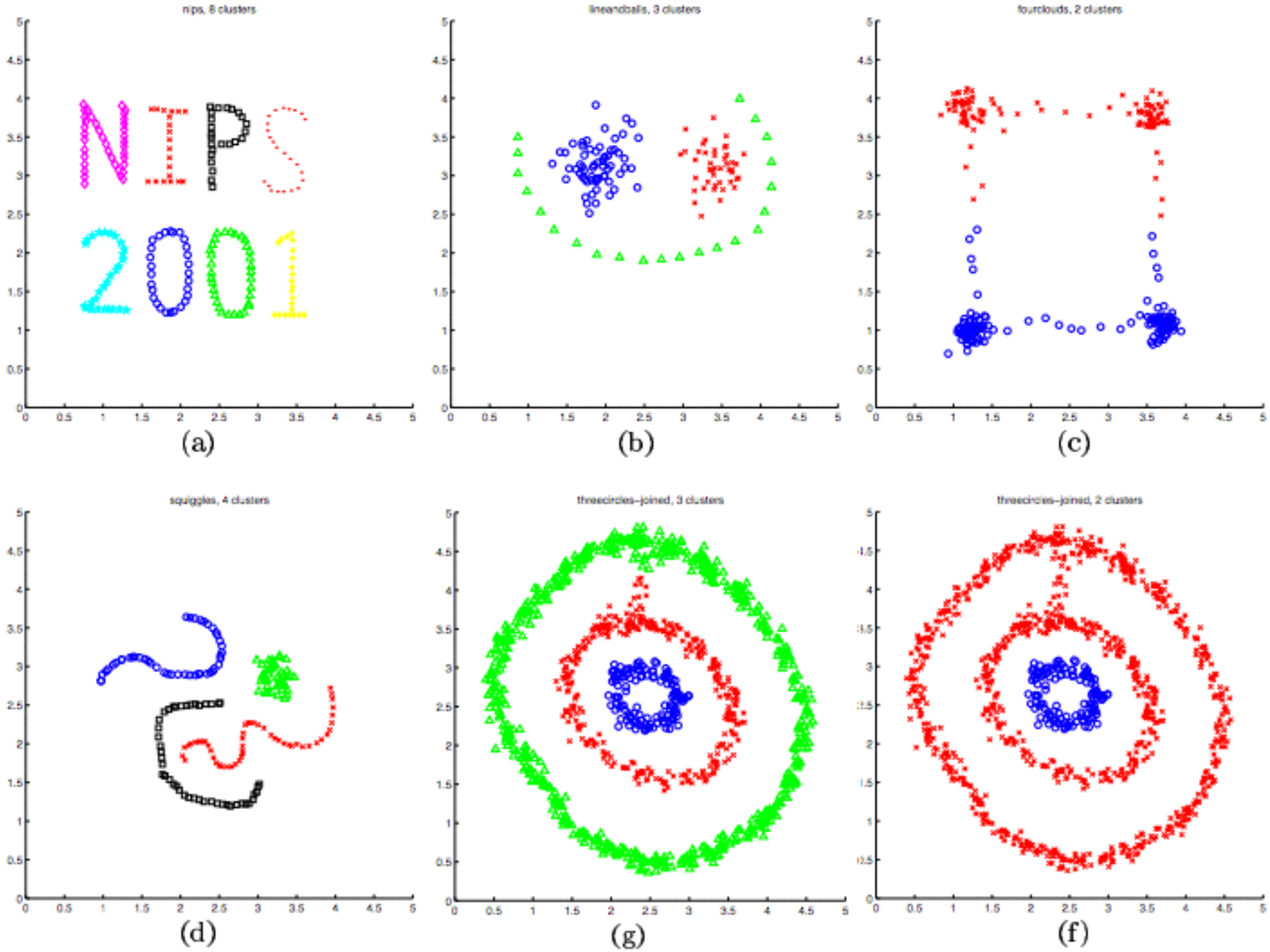
# Spectral clustering




[Shi & Malik ‘00; Ng, Jordan, Weiss NIPS ‘01]

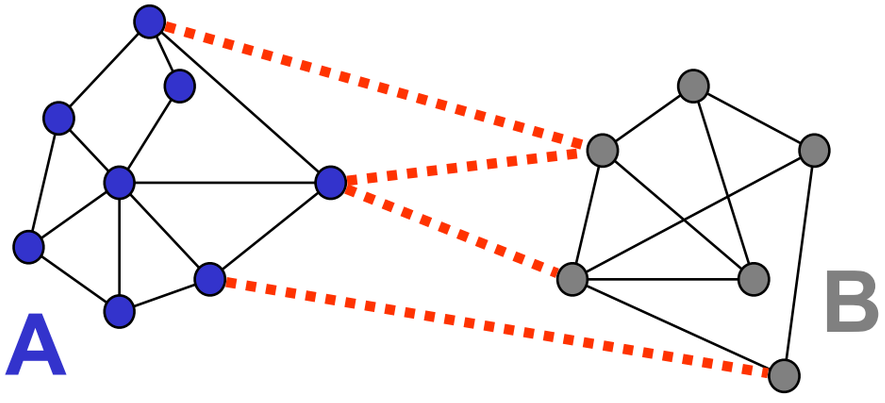
# Spectral clustering




[Figures from Ng, Jordan, Weiss NIPS ‘01]

# Spectral clustering

- Group points based on the links in a graph




- How do we create the graph?
  - Weights on the edges based on similarity between the points
  - A common choice is the Gaussian kernel

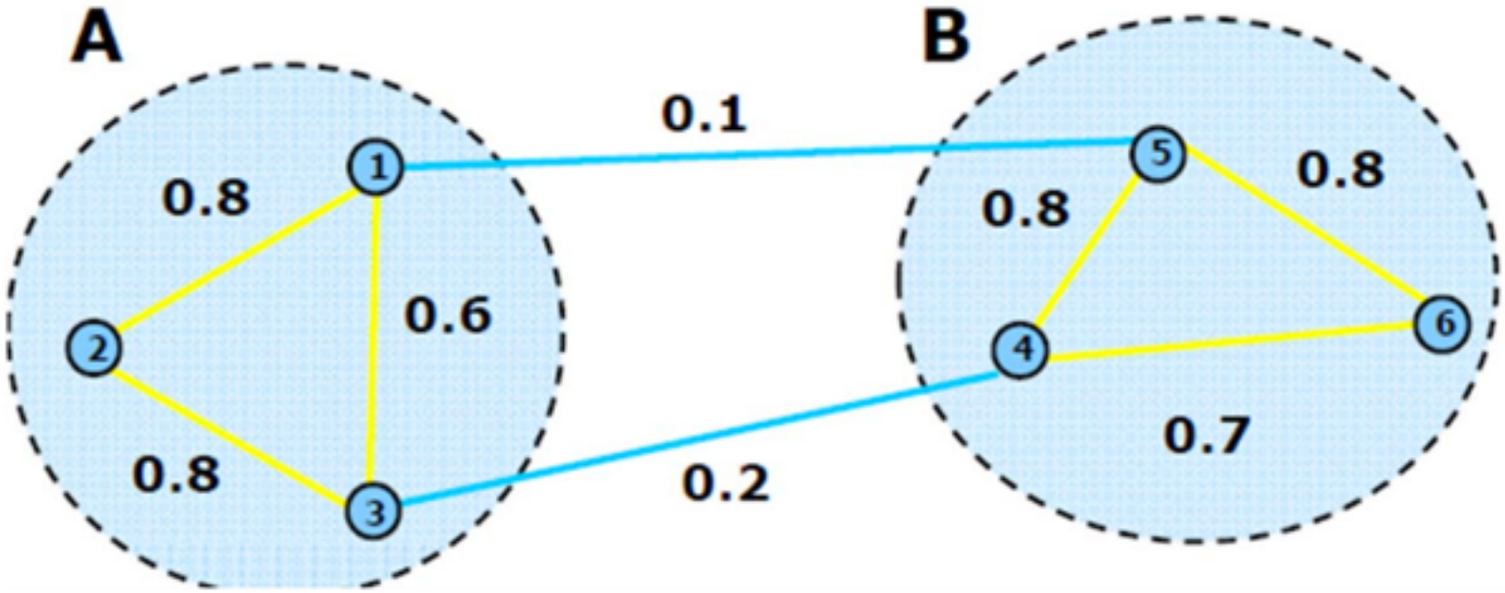
$$W(i,j) = \exp\left(-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma^2}\right)$$

- One could create
  - A fully connected graph
  - k-nearest graph (each node is connected only to its k-nearest neighbors)

slide credit: Alan Fern

# Graph cut

- Consider a partition of the graph into two parts A and B




- Cut(A, B) is the weight of all edges that connect the two groups

$$\mathrm{Cut}(A, B) = \sum_{i \in A, j \in B} W(i, j) = 0.3$$

- An intuitive goal is to find a partition that minimizes the cut
  - min-cuts in graphs can be computed in polynomial time

# Problem with min-cut

- The weight of a cut is proportional to number of edges in the cut; tends to produce small, isolated components.

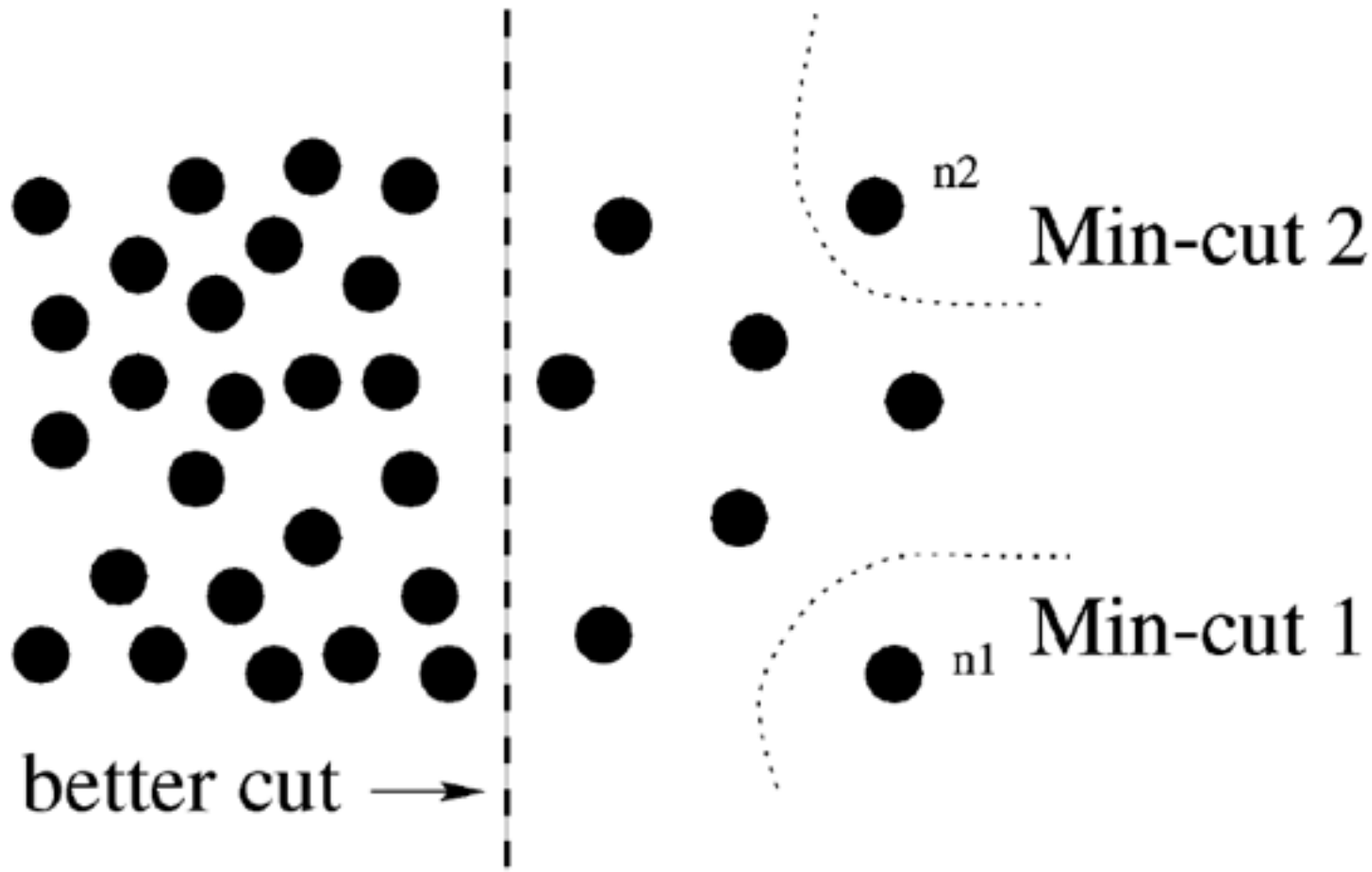



Fig. 1. A case where minimum cut gives a bad partition.

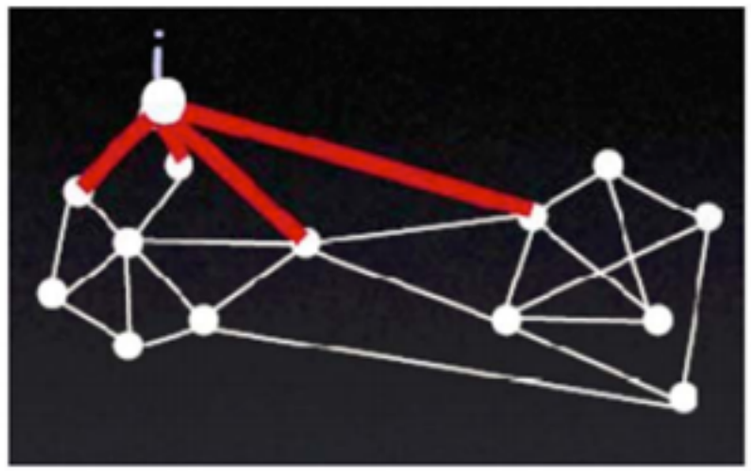
We would like a balanced cut

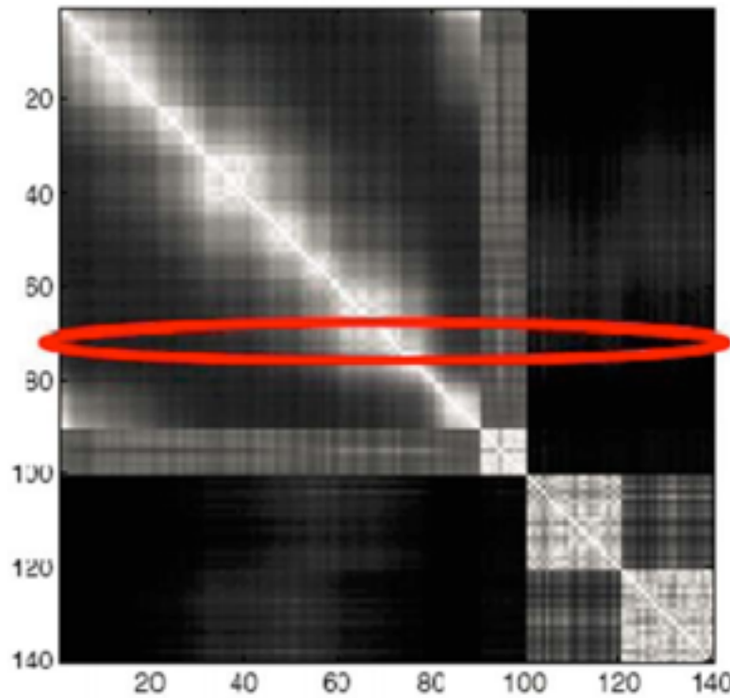
[Shi & Malik, 2000 PAMI]

# Graphs as matrices

- Let *W(i, j)* denote the matrix of the edge weights
- The degree of node in the graph is:

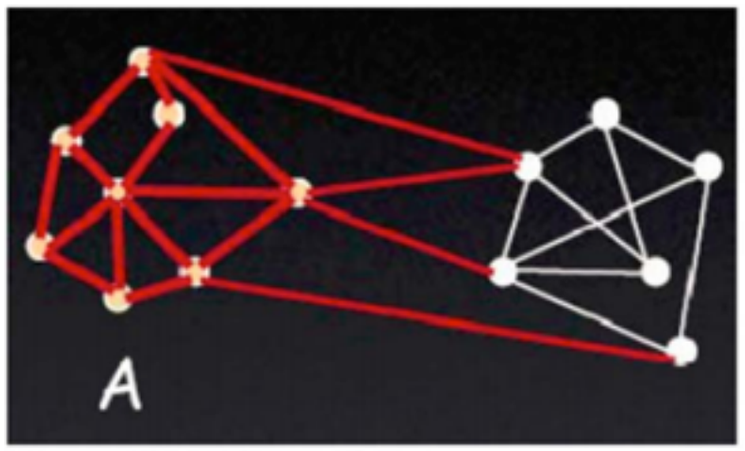
$$d(i) = \sum_j W(i,j)$$

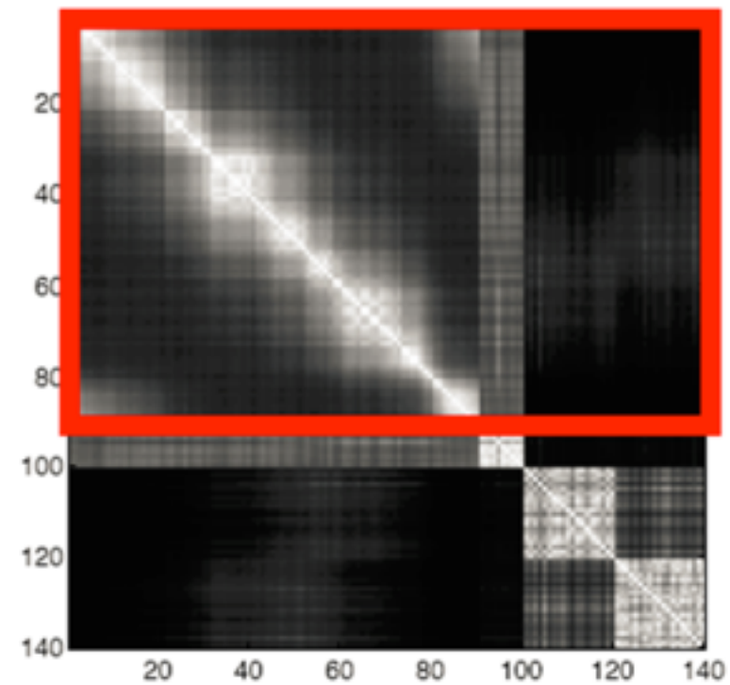



- The volume of a set A is defined as:

$$\mathrm{Vol}(A) = \sum_{i \in A} d(i)$$



# Normalized cut

- Intuition: consider the connectivity between the groups relative to the volume of each group:

$$\mathrm{NCut}(A,B) = \frac{\mathrm{Cut}(A,B)}{\mathrm{Vol}(A)} + \frac{\mathrm{Cut}(A,B)}{\mathrm{Vol}(B)}$$

$$\mathrm{NCut}(A,B) = \mathrm{Cut}(A,B)\left(\frac{\mathrm{Vol}(A)+\mathrm{Vol}(B)}{\mathrm{Vol}(A)\mathrm{Vol}(B)}\right)$$

minimized when Vol(A) = Vol(B)
encouraging a balanced cut

- Unfortunately minimizing normalized cut is NP-Hard even for planar graphs [Shi & Malik, 00]

# Solving normalized cuts

- We will formulate an optimization problem
  - Let $W$ be the similarity matrix
  - Let $D$ be a diagonal matrix with $D(i,i) = d(i)$ — the degree of node i
  - Let $\mathbf{x}$ be a vector $\{1, -1\}^N$ , $x(i) = 1 \leftrightarrow i \in A$
  - The matrix $(D-W)$ is called the Laplacian of the graph

- With some simplification we can show that the problem of minimizing normalized cuts can be written as:

$$\min_{\mathbf{x}} \mathrm{NCut}(\mathbf{x}) = \min_{\mathbf{y}} \frac{\mathbf{y}^T(D-W)\mathbf{y}}{\mathbf{y}^T D\mathbf{y}}$$

$$\text{subject to: } \mathbf{y}^T D\mathbf{1} = 0$$

$$\mathbf{y}(i) \in \{1, -b\}$$

# Solving normalized cuts

- Normalized cuts objective:

$$\min_{\mathbf{x}} \mathrm{NCut}(\mathbf{x}) = \min_{\mathbf{y}} \frac{\mathbf{y}^T(D-W)\mathbf{y}}{\mathbf{y}^T D\mathbf{y}}$$

$$\text{subject to: } \mathbf{y}^T D\mathbf{1} = 0$$

$$\mathbf{y}(i) \in \{1, -b\}$$

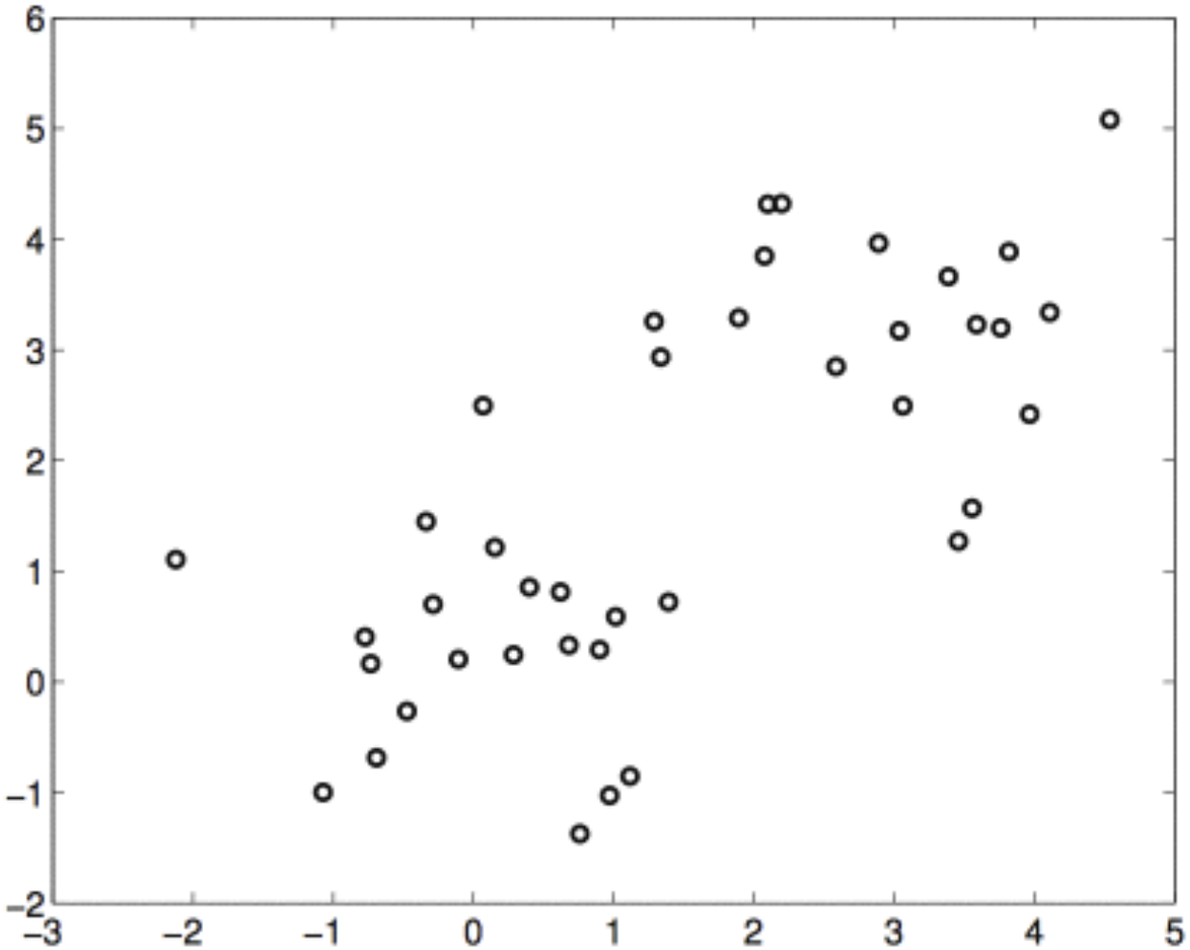
- Relax the integer constraint on $\mathbf{y}$:

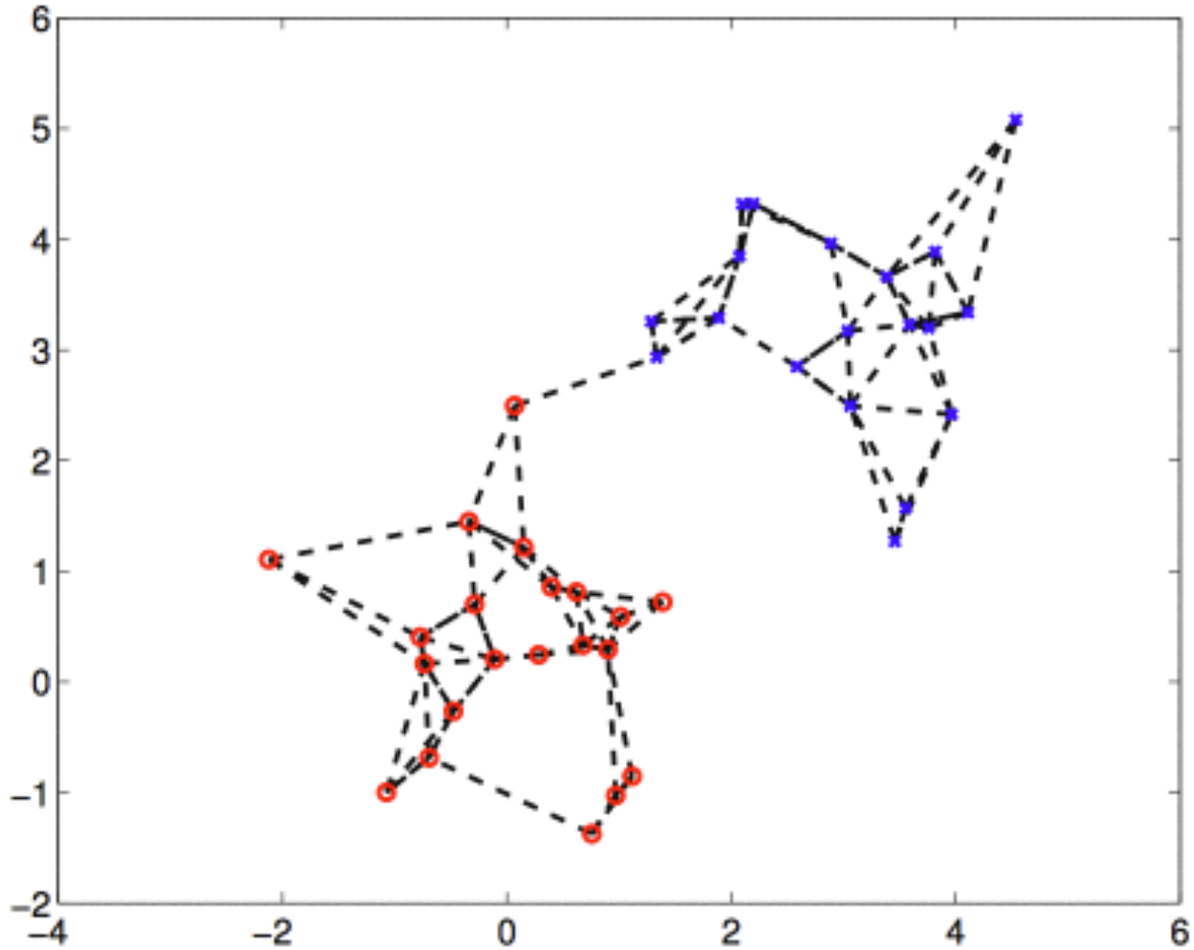
$$\min_{\mathbf{y}} \mathbf{y}^T(D-W)\mathbf{y}; \text{ subject to: } \mathbf{y}^T D\mathbf{y} = 1, \mathbf{y}^T D\mathbf{1} = 0$$

- Same as: $(D-W)\mathbf{y} = \lambda D\mathbf{y}$ (Generalized eigenvalue problem)
- Note that $(D-W)\mathbf{1} = 0$, so the first eigenvector is $\mathbf{y}_1 = \mathbf{1}$, with the corresponding eigenvalue of 0
- The eigenvector corresponding to the second smallest eigenvalue is the solution to the relaxed problem

# Spectral clustering example

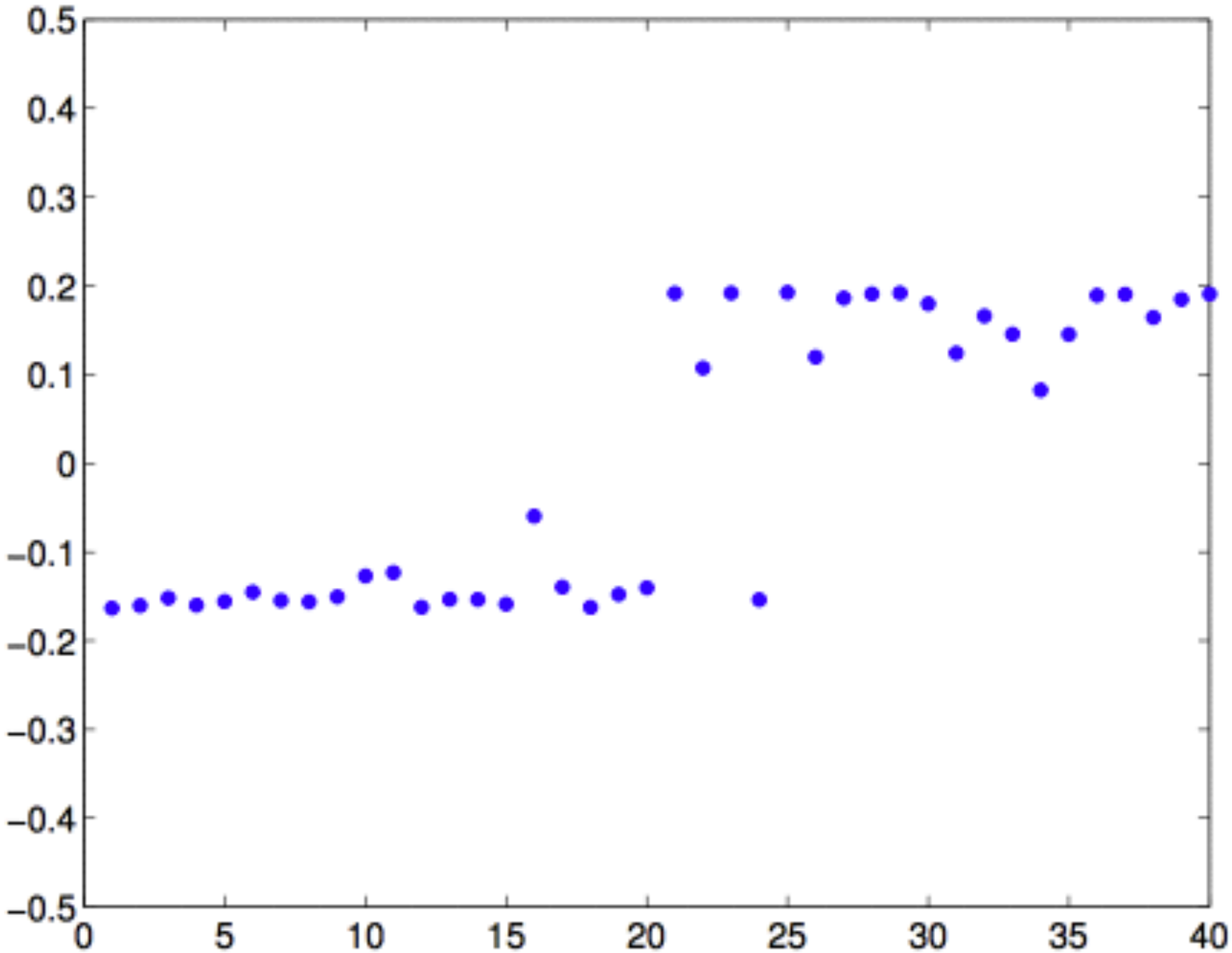
- **Data:** Gaussian weighted edges connected to 3 nearest neighbors



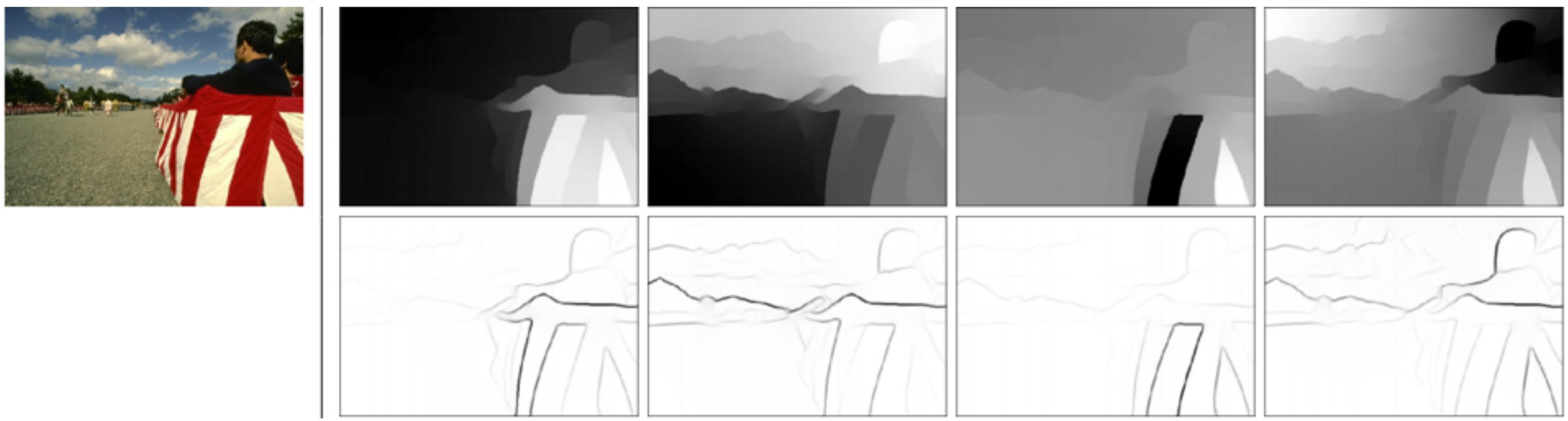
# Spectral clustering example

- Components of the eigenvector corresponding to the second smallest eigenvalue



# Creating partitions from eigenvalues

- The eigenvalue is real valued, so to obtain a split you may threshold it
- How to pick the threshold?
  - Pick the median value
  - Choose a threshold that minimizes the normalized cut objective
- How to create multiple partitions?
  - Recursively split each partition
  - Compute multiple eigenvalues and cluster them using k-means
    - ➡ Example: multiple eigenvalues of an image and their gradients



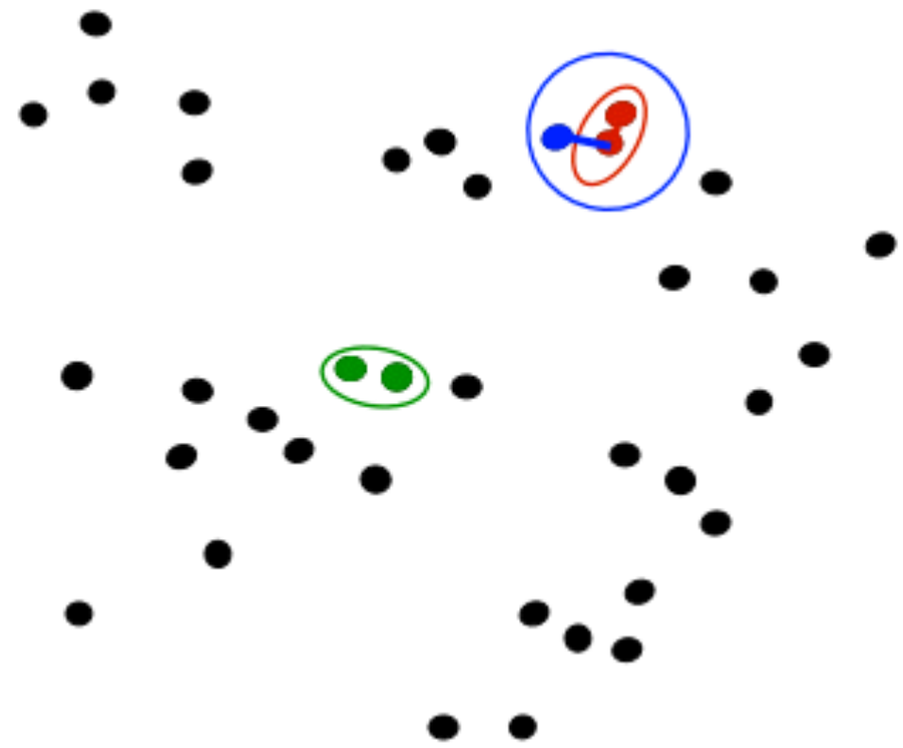
http://ttic.uchicago.edu/~mmaire/papers/pdf/amfm_tpami2011.pdf

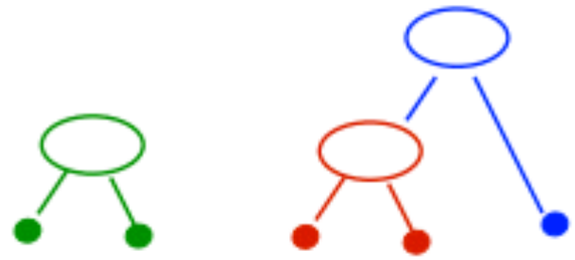
# Hierarchical clustering

- Organize elements into a hierarchy
- Two kinds of methods:
  - 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

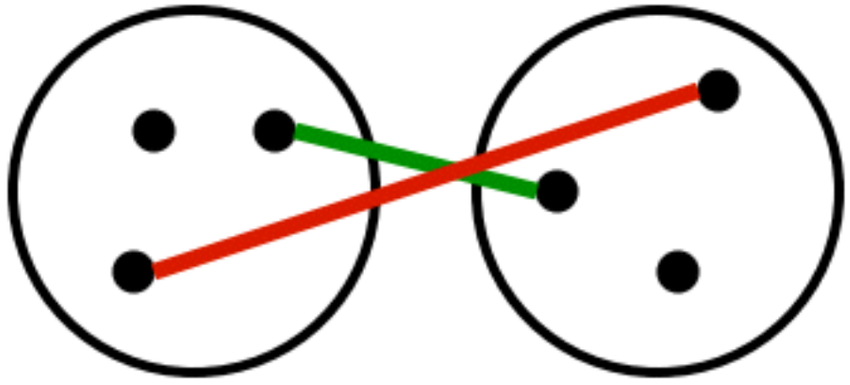


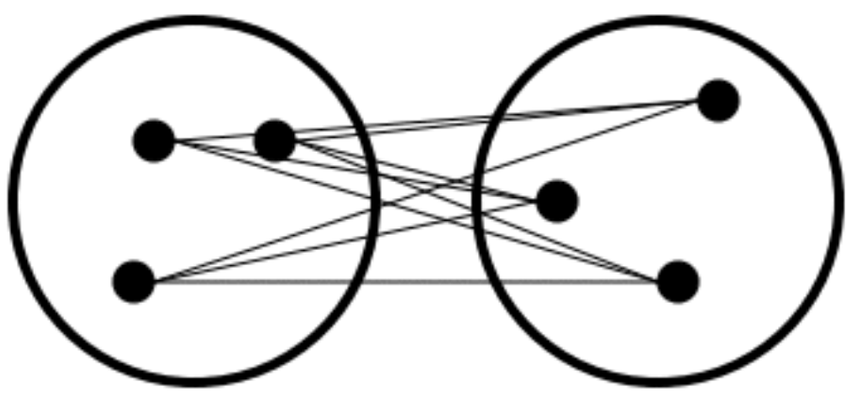


# Agglomerative clustering

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

- Many options:
  - Closest pair: single-link clustering
  - Farthest pair: complete-link clustering
  - Average of all pairs

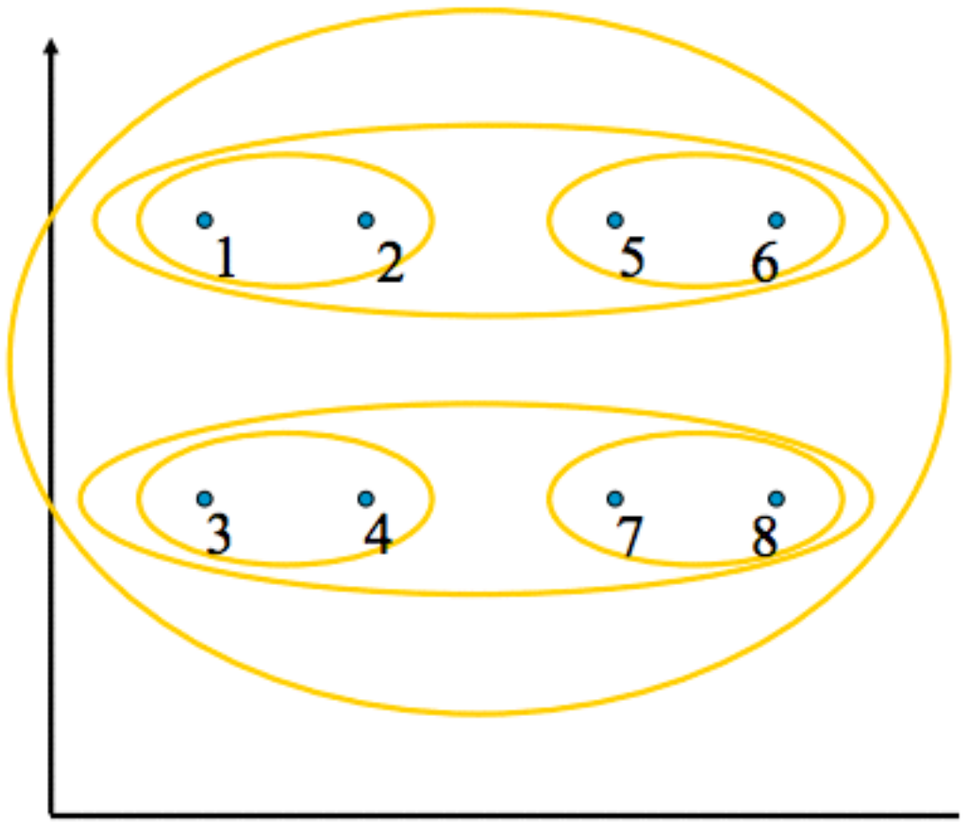




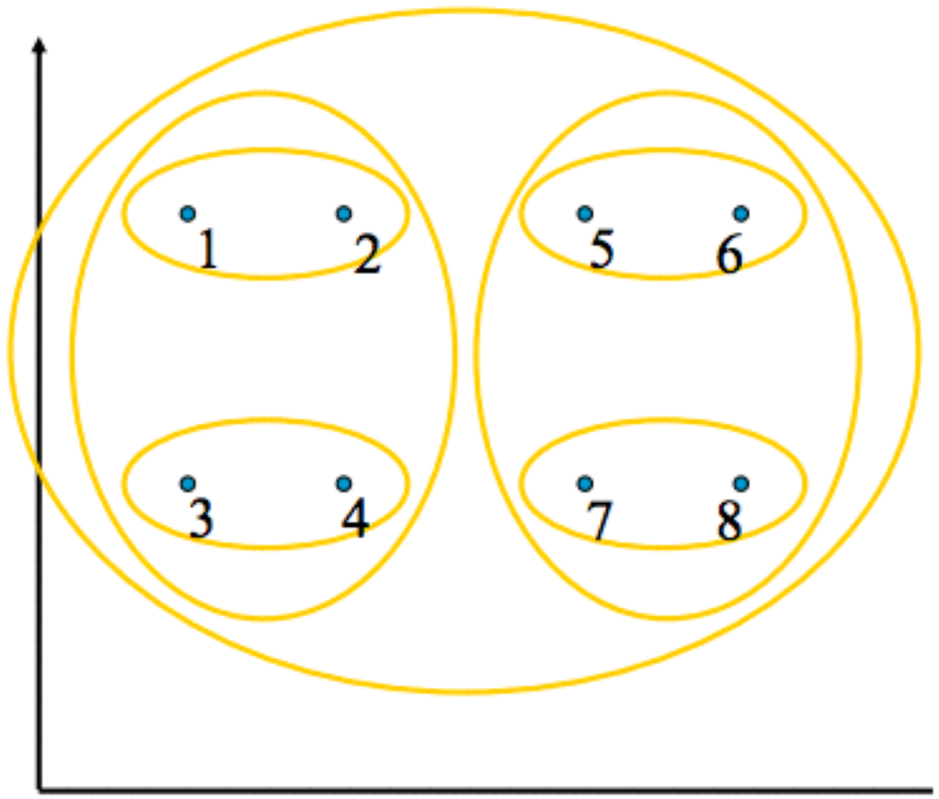
# Agglomerative clustering

- Different choices create different clustering behaviors

Closest pair
(single-link clustering)




Farthest pair
(complete-link clustering)




[Pictures from Thorsten Joachims]

# 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 adapted from David Sontag, Luke Zettlemoyer, Vibhav Gogate, Carlos Guestrin, Andrew Moore, Dan Klein, James Hays, Alan Fern, and Tommi Jaakkola

- 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