Lecture 11

K-means and Mean Shift

Administrative

A3 is out

- Due Feb 21st

A4 will be out soon

Administrative

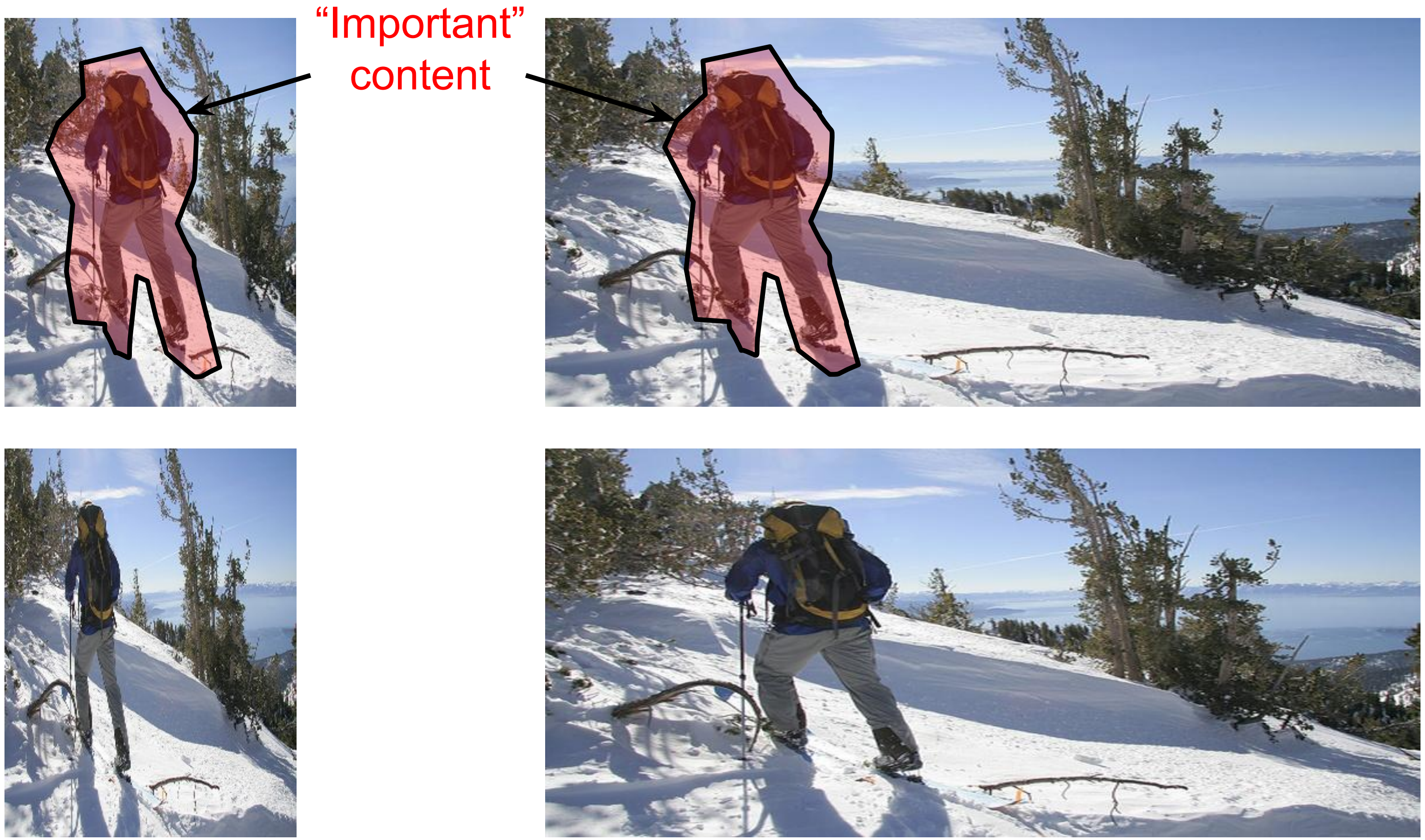
Recitation

- Multiview geometry

Content-aware Retargeting Operators

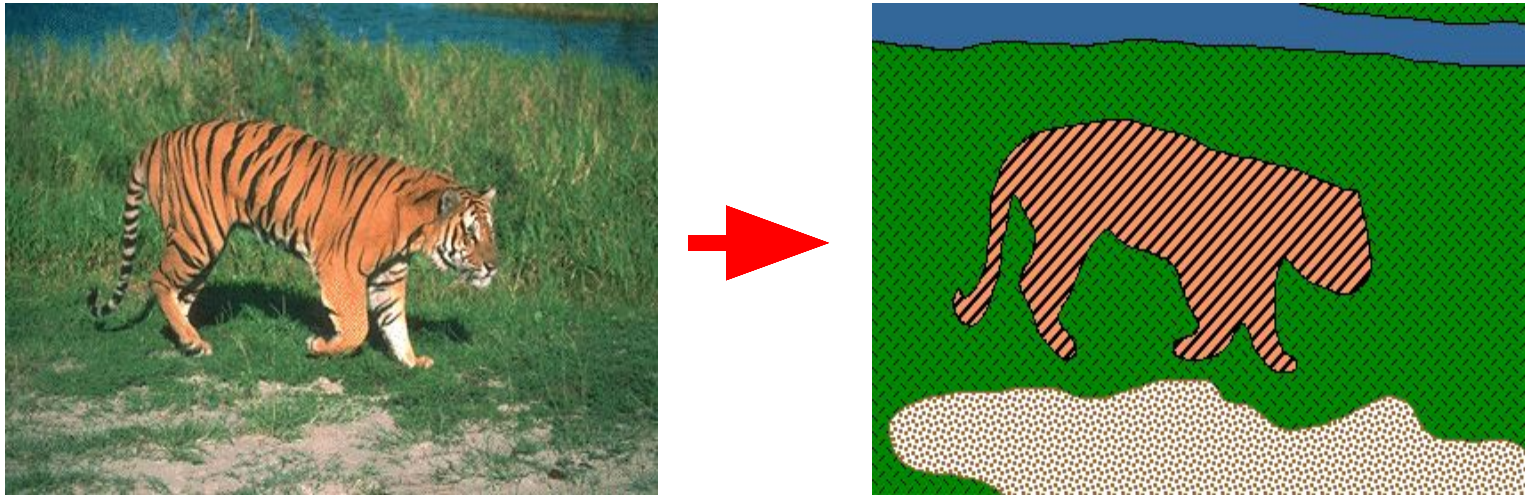
Content-*aware*

Content-*oblivious*

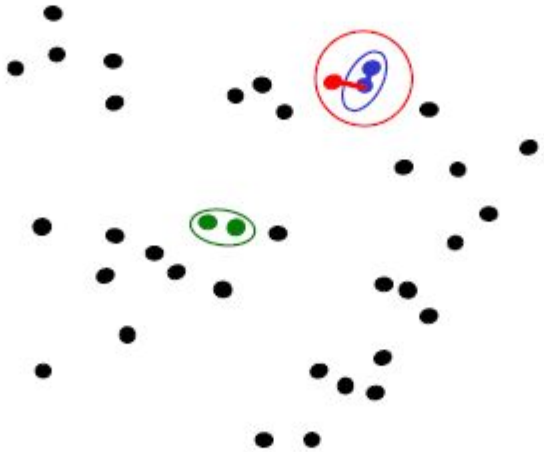


So far: Segmentation and clustering

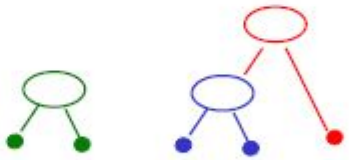
- Goal: identify groups of pixels that go together



So far: Agglomerative clustering



1. Say "Every point is its own cluster"
2. Find "most similar" pair of clusters
3. Merge it into a parent cluster
4. Repeat



Today's agenda

- K-means clustering
- Mean-shift clustering
- Normalized cuts

Reading:
Szeliski, 2nd edition, Chapter 7.5

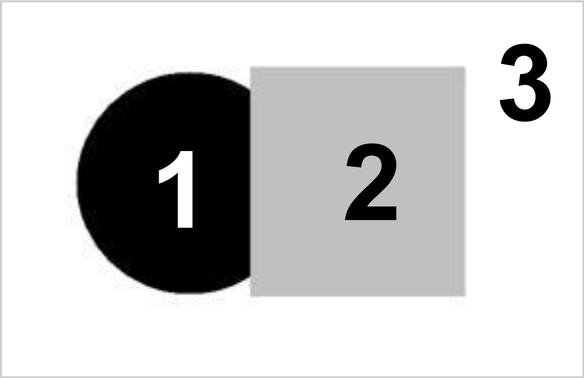
Today’s agenda

- K-means clustering
- Mean-shift clustering
- Normalized cuts

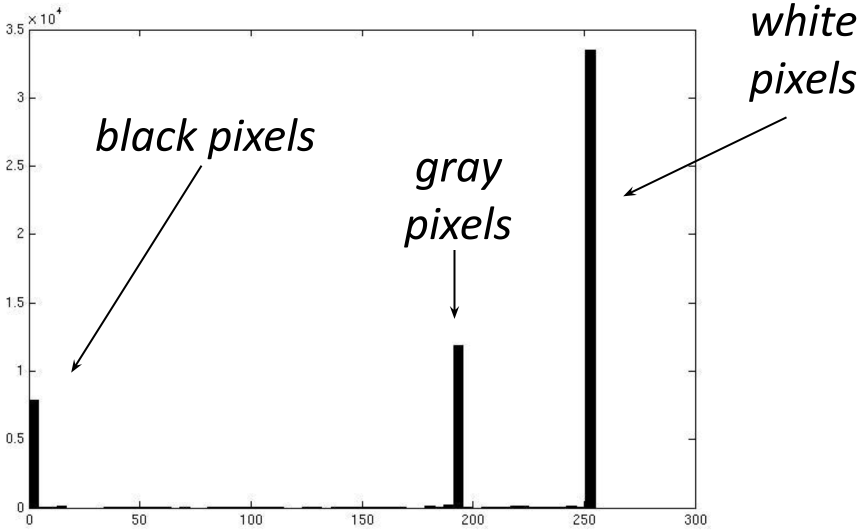
Reading: Szeliski Chapters: 5.2.2, 7.5.2

D. Comaniciu and P. Meer, Mean Shift: A Robust Approach toward Feature Space Analysis, PAMI 2002.

Image Segmentation: Binary image Example

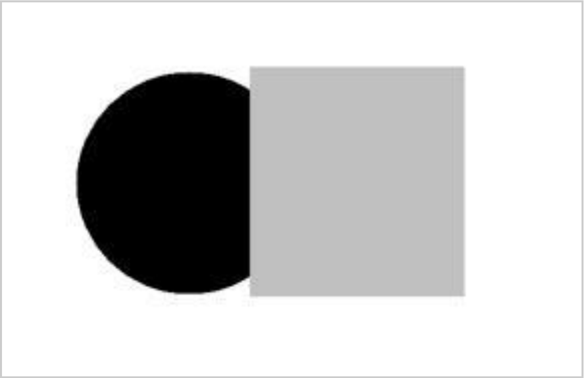


input image

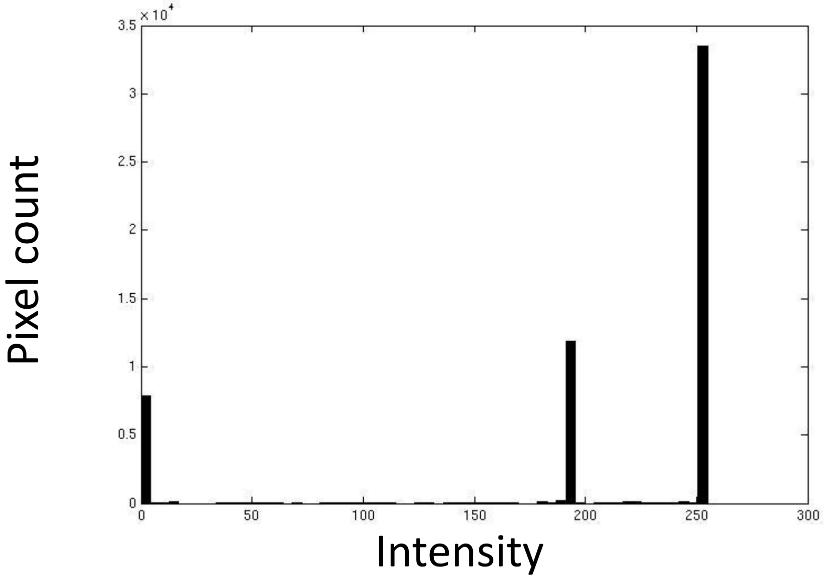


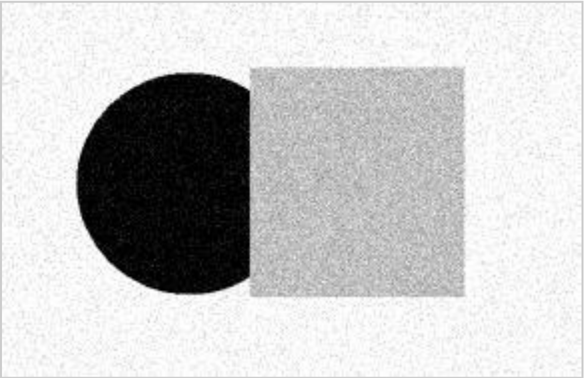
intensity

- These pixel values show that there are three things in the image.
- We could label every pixel in the image according to which of these primary intensities it is.
  - i.e., segment the image based on the intensity feature.
- What if the image isn't quite so simple?

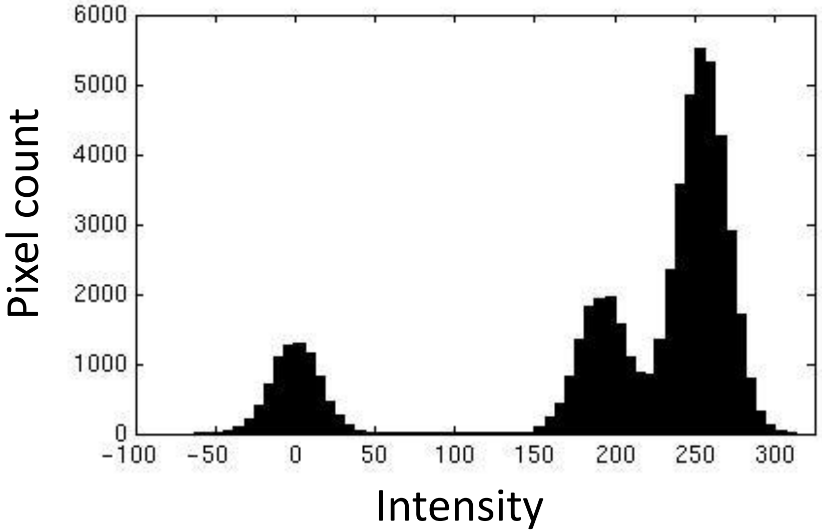


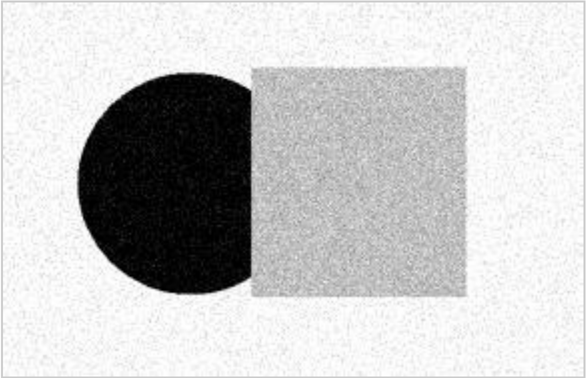
Input image



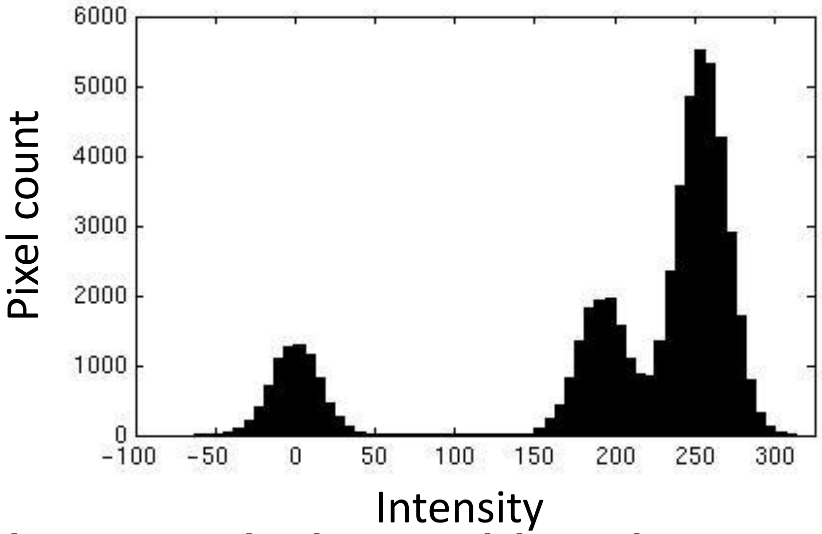


Input image

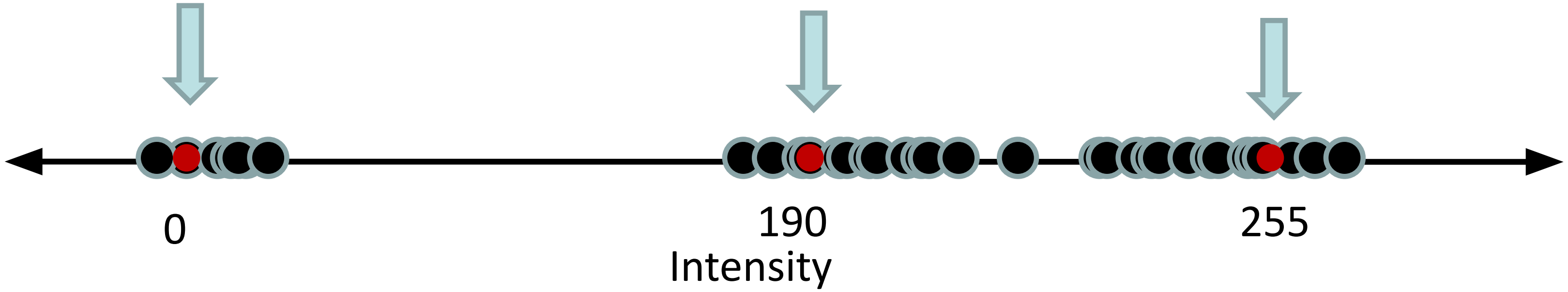


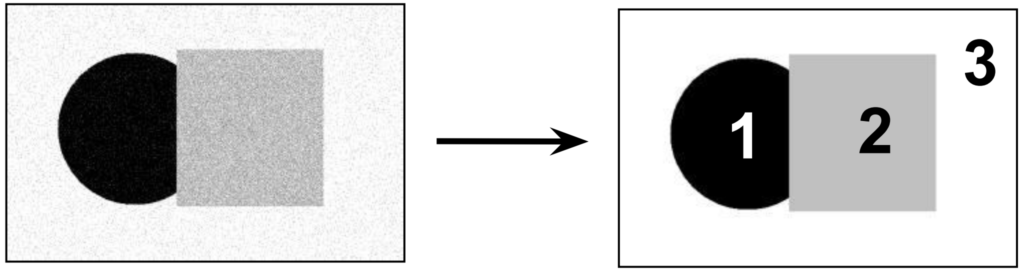


Input image



- How do we determine the three main intensities that define our groups?
- Each cluster has a cluster center
  - A mean cluster value.





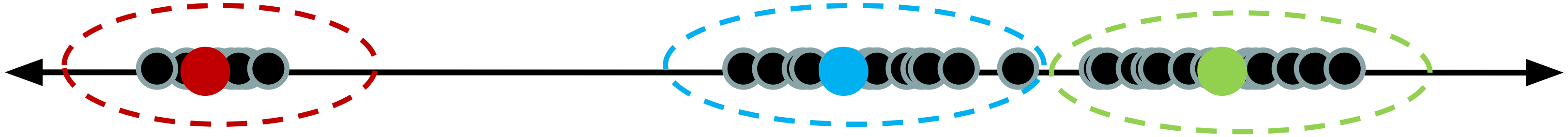
- Goal: choose three "centers" as the representative intensities and label every pixel according to which of these centers it is nearest to.

- Best cluster centers are those that minimize **Sum of Square Distance** (SSD) between all points and their nearest cluster center c_i:

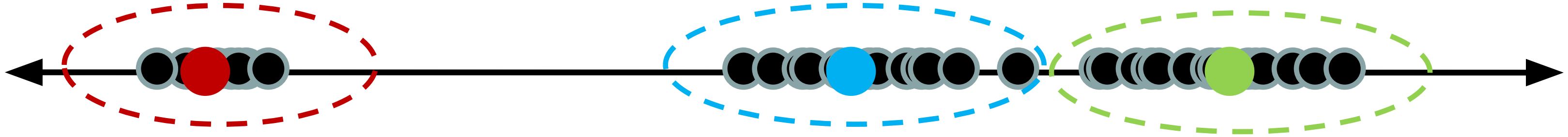
$$SSD = \sum_{C} \sum_{v \in C} (v - c_i)^2$$

Clustering

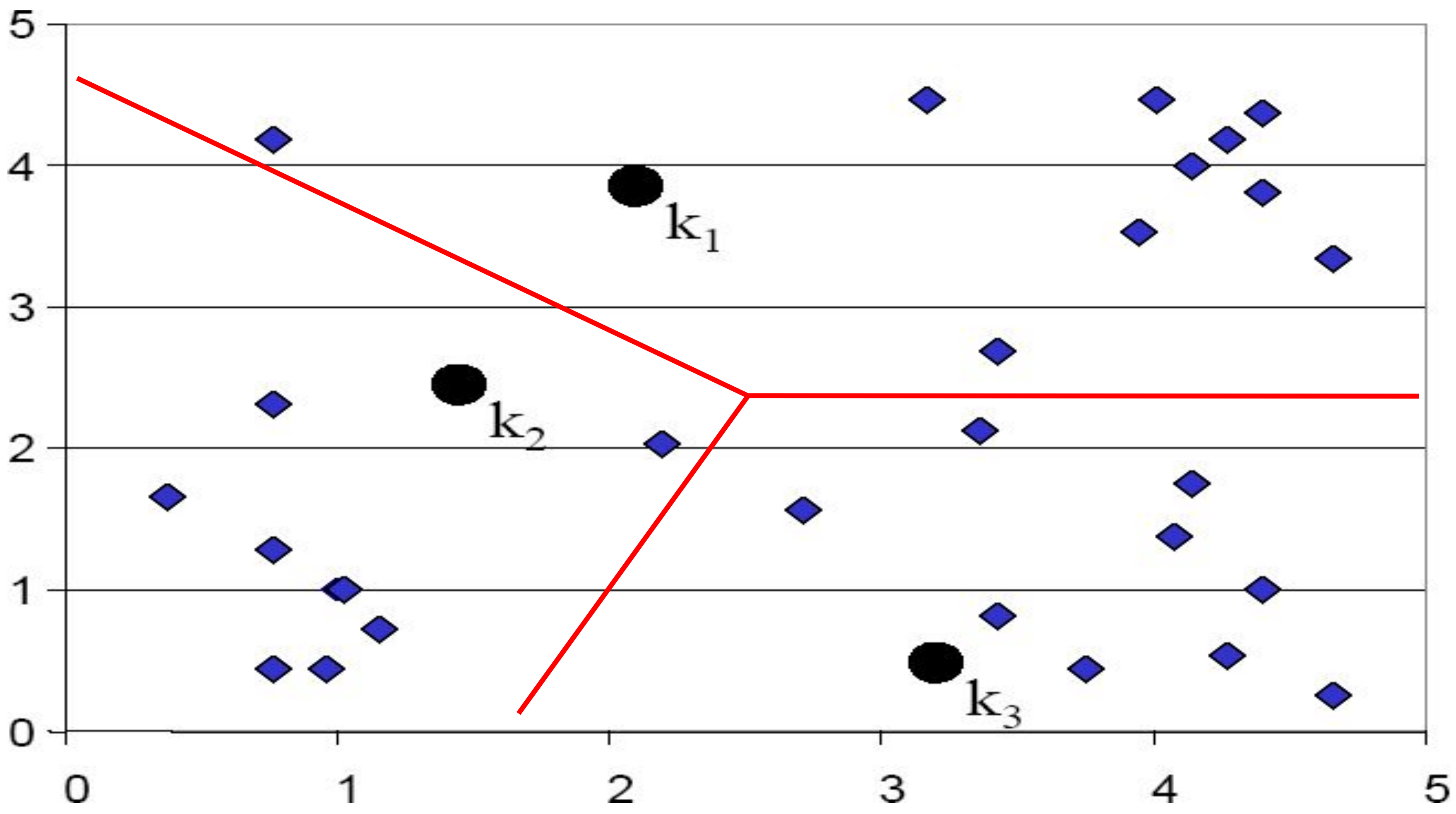
- With this objective, it is a “chicken and egg” problem:
  - If we knew the *cluster centers*, we could allocate points to groups by assigning each to its closest center.



  - If we knew the *group memberships*, we could get the centers by computing the mean per group.



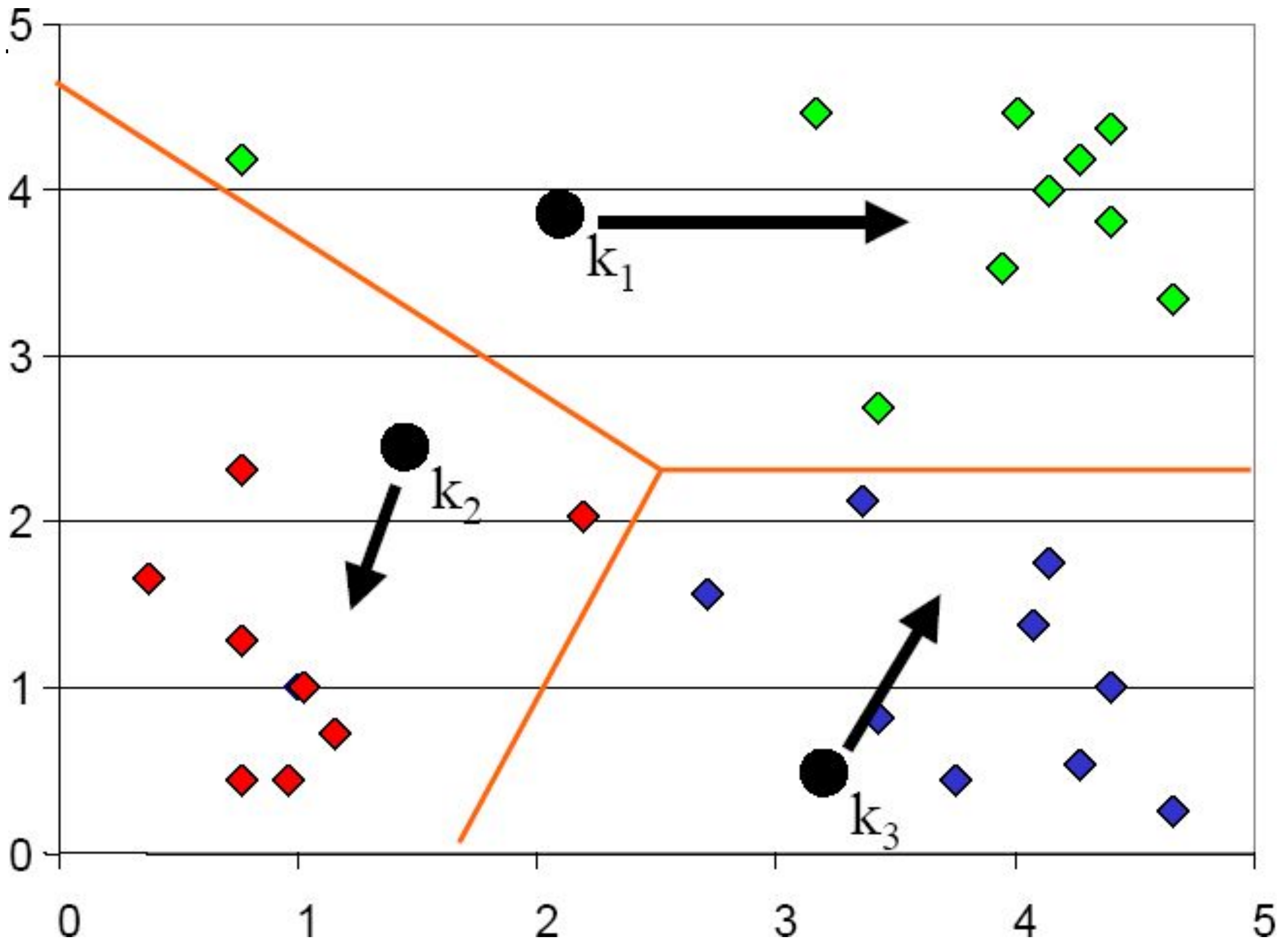
Given, a set of points, randomly select k=3 of them to be the cluster centers



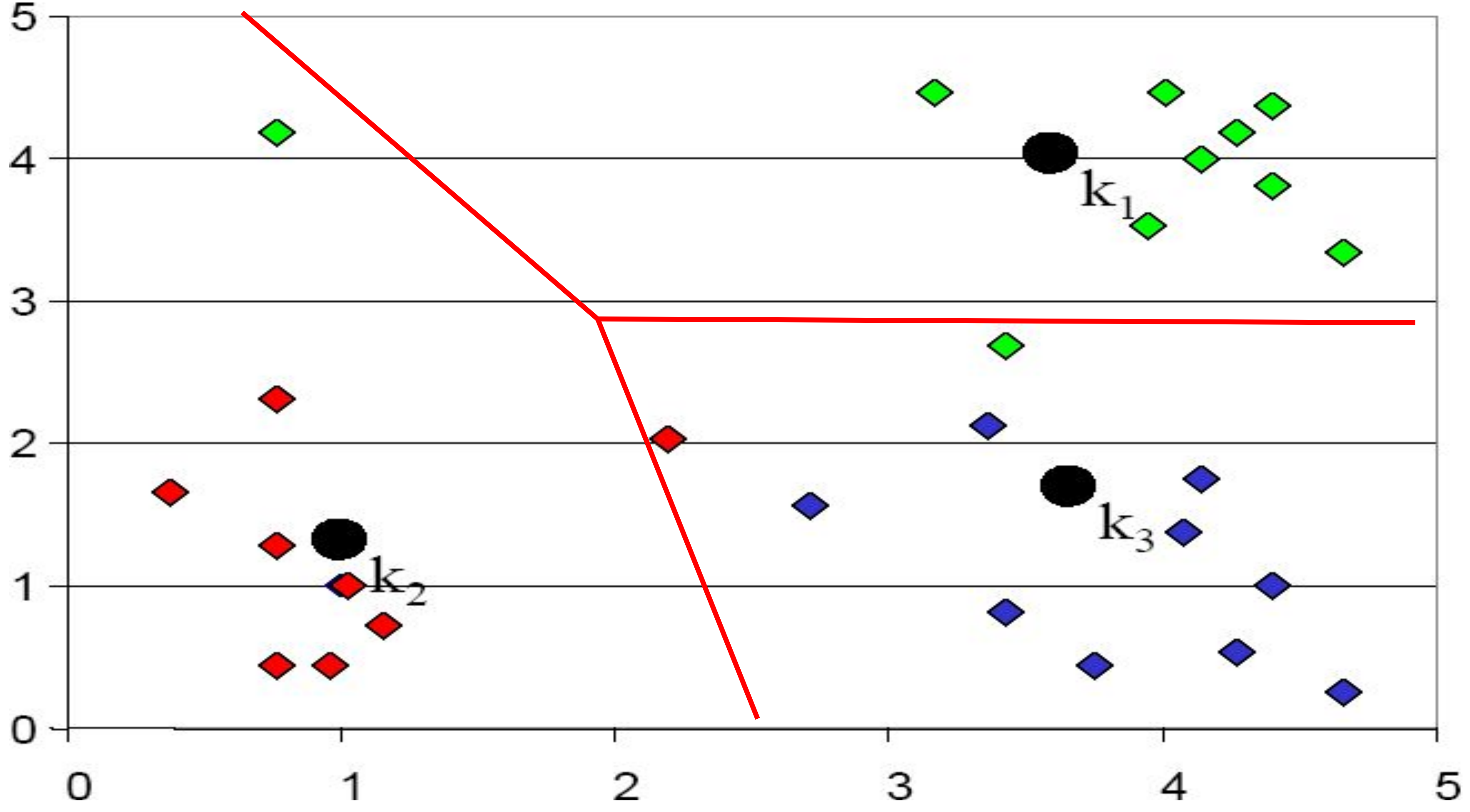
Voronoi diagram

Categorize each point into a cluster defined by its closest center.

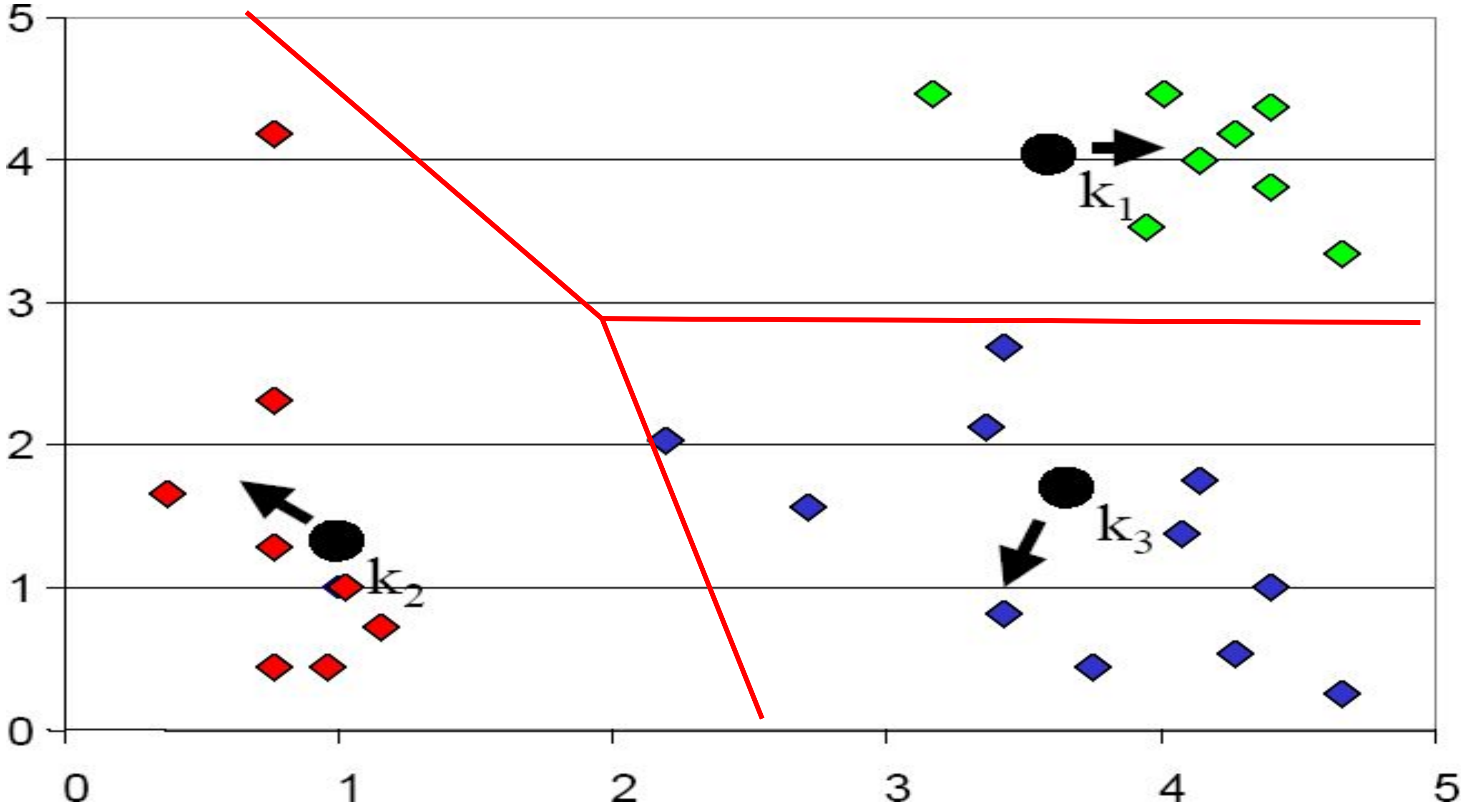
Next, move the cluster centers to the location amongst its cluster



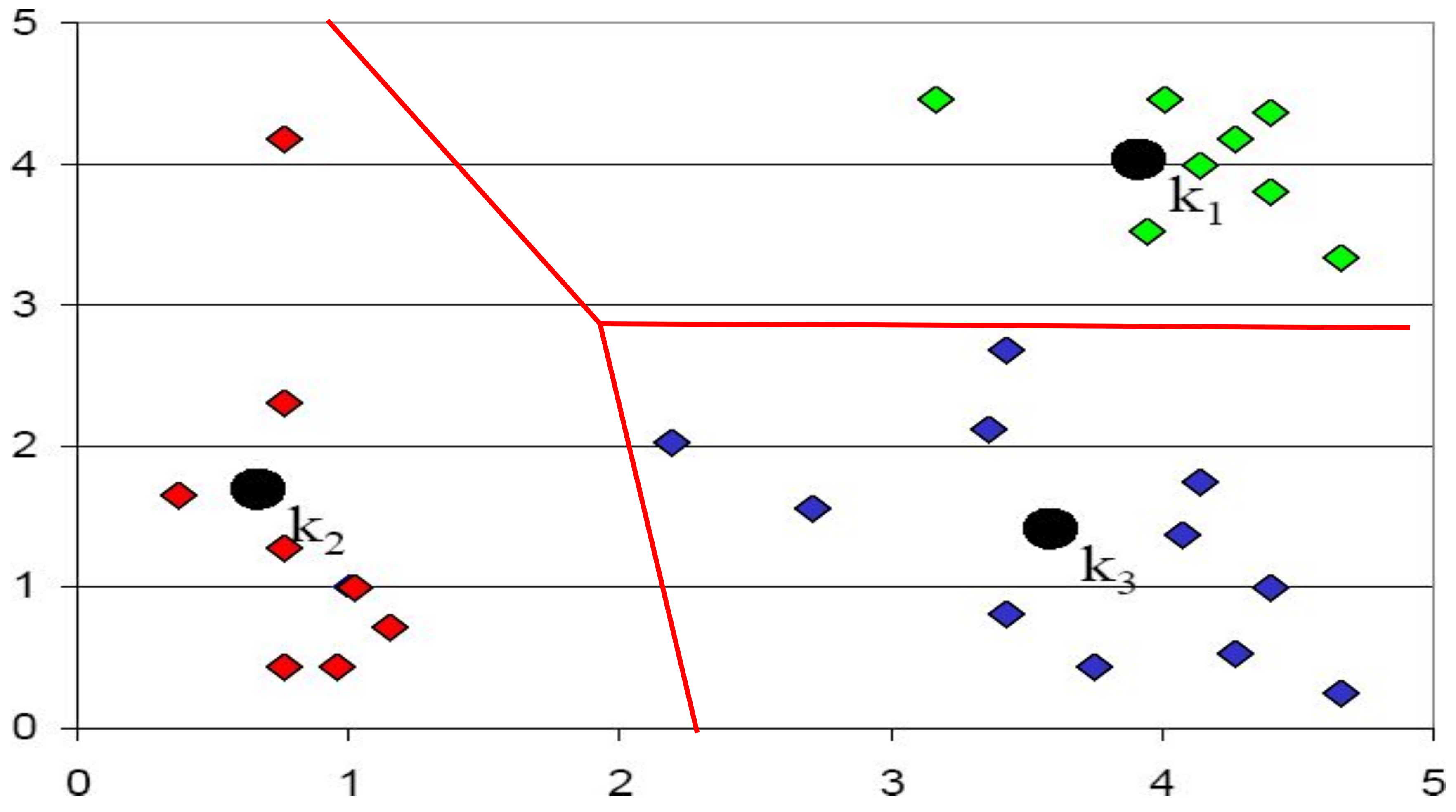
Repeat with new cluster center locations



Categorize into new clusters.
Move center to the mean



Repeat with new cluster centers



Computational Complexity

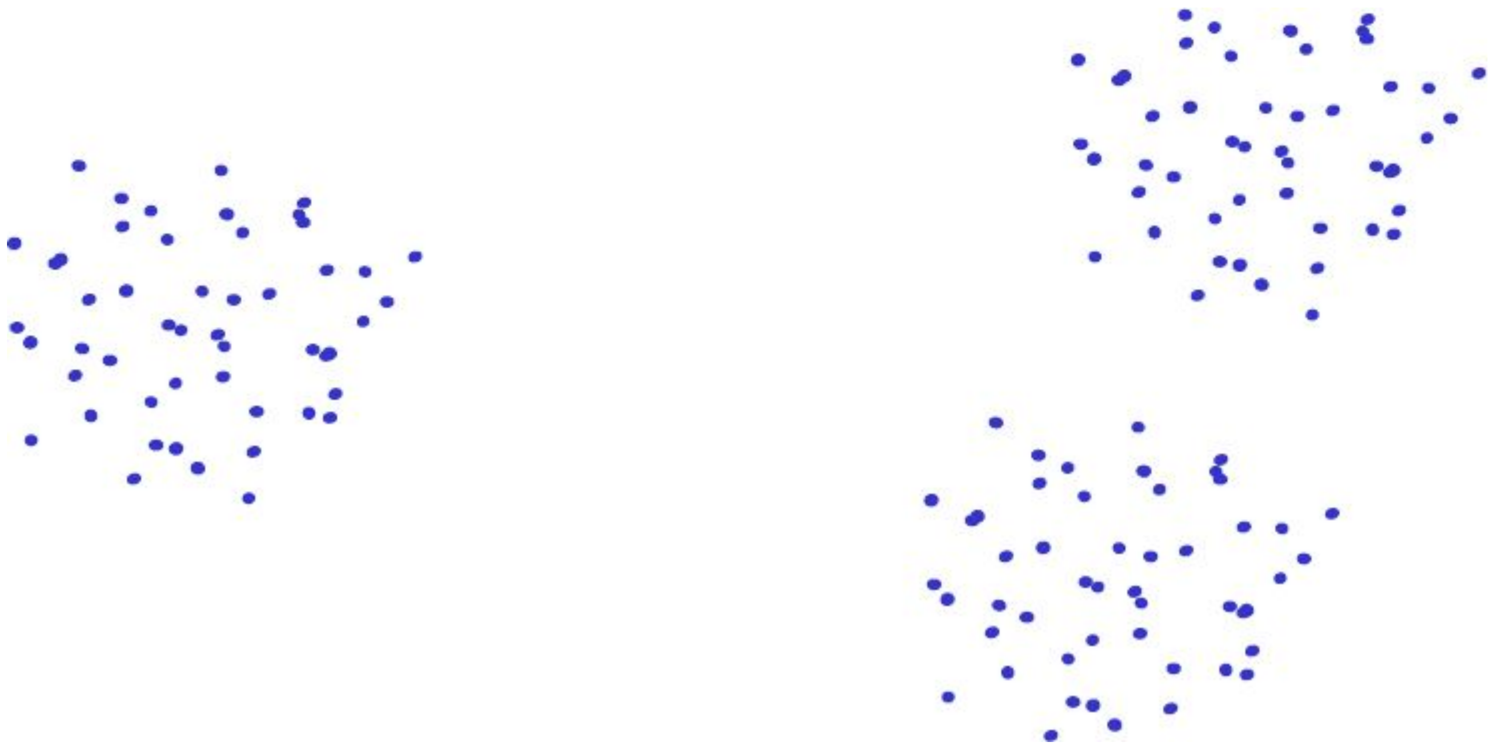
At each iteration,

- Computing distance between each of the n objects and the K cluster centers is O(Kn).
- Computing cluster centers: Each object gets added once to some cluster: O(n).

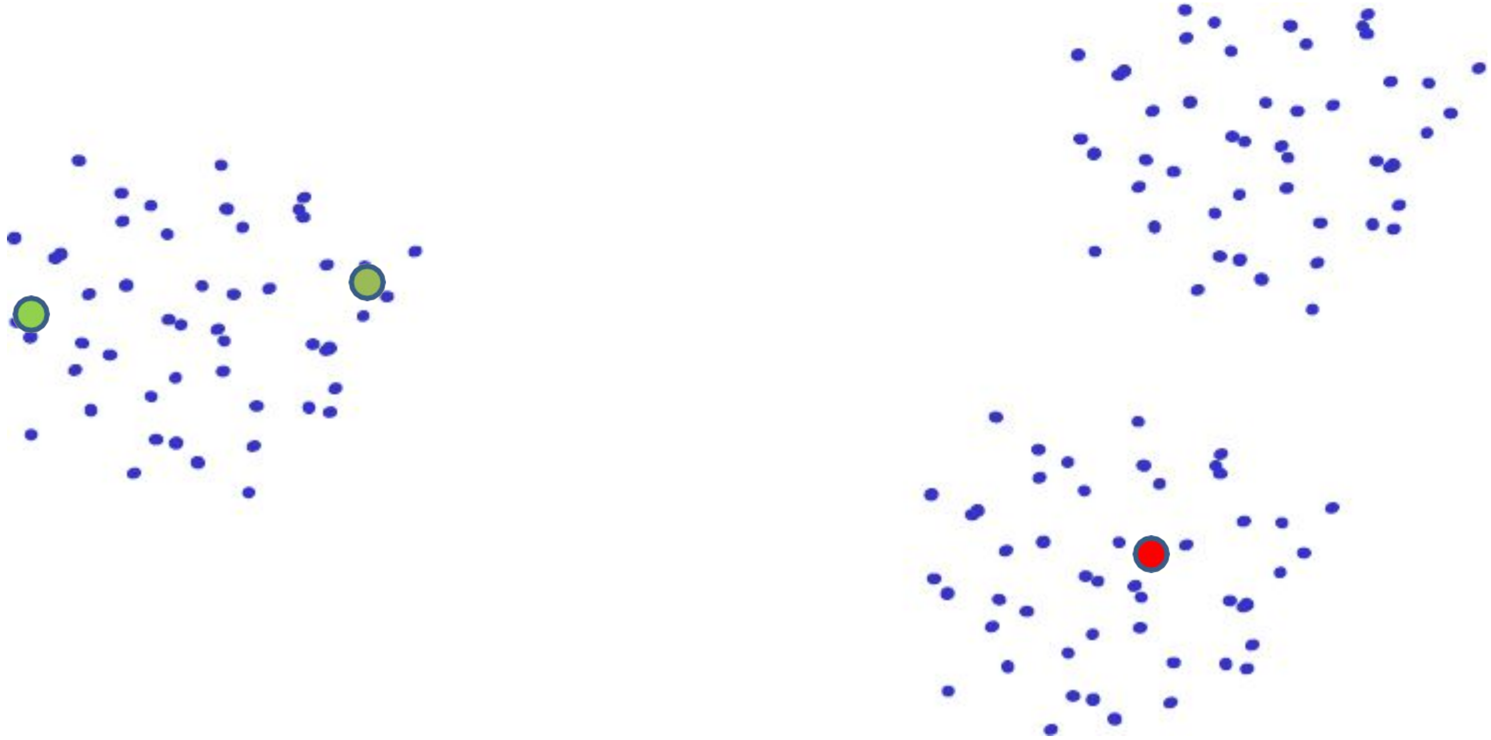
Assume these two steps are each done once for I iterations: O(IKn).

Q. Is K-means guaranteed to converge to a global maximum?

Results are quite sensitive to seed selection.



Results are quite sensitive to seed selection.



Results are quite sensitive to seed selection.

Results are quite sensitive to seed selection.

- Some seeds can result in poor convergence rate, or convergence to sub-optimal clustering.
- Select good seeds using a heuristic (e.g., object least similar to any existing mean)
- Try out multiple starting points (very important!!!)
- Initialize with the results of another method.

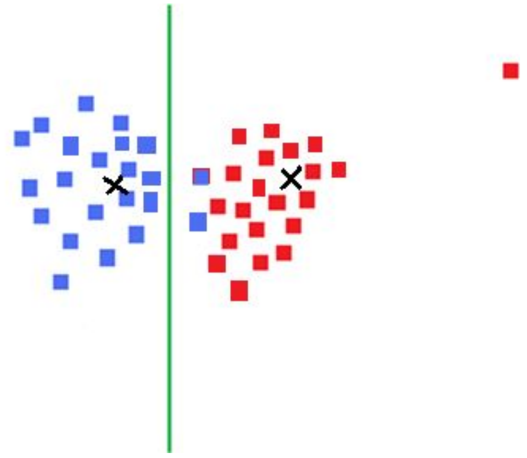
Other issues with k-means

Shape of clusters

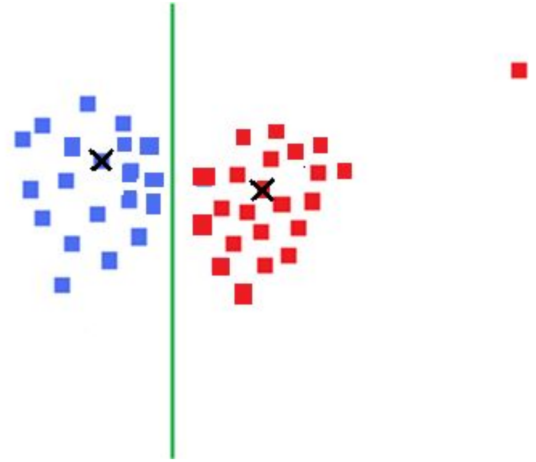
– Assumes isotopic, convex clusters

Sensitive to Outliers

Outlier causes misclassifications



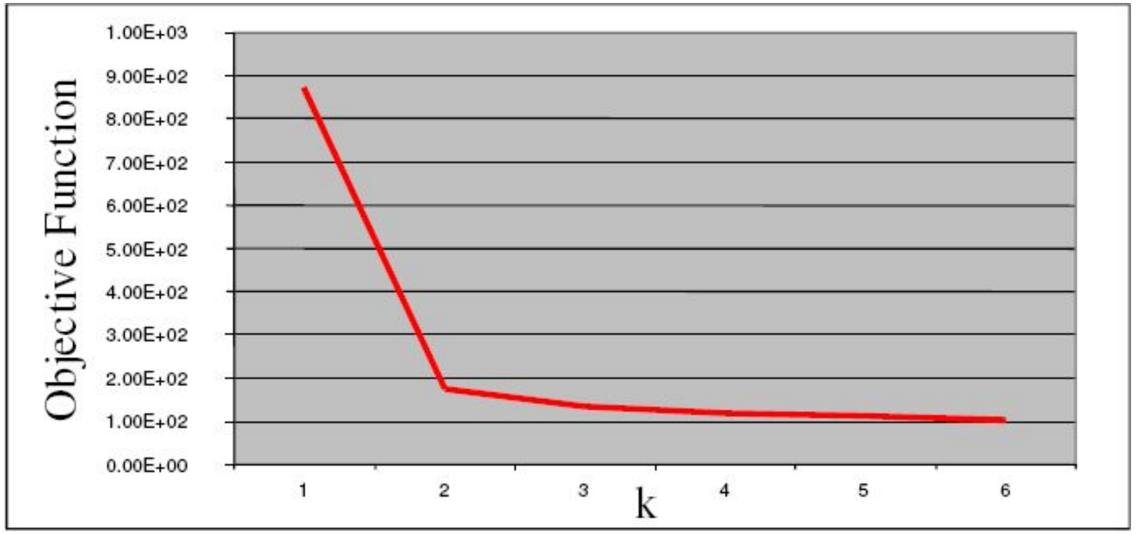
Ideal decision boundary



How to choose the value of k

- Number of clusters K
  - Objective function

  - Look for “Knee” in objective function



Clustering

Goal: cluster to minimize distance of pixels to their cluster centers

Cluster center — Data

$$c^*, \delta^* = \arg\min_{c,\delta} \sum_j^N \sum_i^N \delta_{ij}(c_i - v_j)^2$$

Whether v_j is assigned to c_i

K-means clustering

1. Initialize ($t = 0$): cluster centers $c_1, ..., c_K$

K-means clustering

1. Initialize ($t = 0$): cluster centers $c_1, ..., c_K$

2. Compute δ^t : assign each point to the closest center
   - δ^t denotes the set of assignment for each v_j to cluster c_i at iteration *t*

$$\delta^t = \arg\min_{\delta} \frac{1}{N} \sum_{j}^{N} \sum_{i}^{K} \delta_{ij}^{t-1} (c_i^{t-1} - v_j)^2$$

Slide: Derek Hoiem

K-means clustering

1. Initialize ($t=0$): cluster centers $c_1,...,c_K$

2. Compute δ^t : assign each point to the closest center
   - δ^t denotes the set of assignment for each v_j to cluster c_i at iteration *t*

$$\delta^t = \arg\min_{\delta} \frac{1}{N} \sum_{j}^{N} \sum_{i}^{K} \delta_{ij}^{t-1} (c_i^{t-1} - v_j)^2$$

3. Computer c^t : update cluster centers as the mean of the points

$$c^t = \arg\min_{c} \frac{1}{N} \sum_{j}^{N} \sum_{i}^{K} \delta_{ij}^{t} (c_i^{t-1} - v_j)^2$$

Slide: Derek Hoiem

K-means clustering

1. Initialize ($t = 0$): cluster centers $c_1, ..., c_K$

2. Compute δ^t : assign each point to the closest center
   - δ^t denotes the set of assignment for each v_j to cluster c_i at iteration *t*

$$\delta^t = \arg\min_{\delta} \frac{1}{N} \sum_{j}^{N} \sum_{i}^{K} \delta_{ij}^{t-1} (c_i^{t-1} - v_j)^2$$

3. Computer c^t : update cluster centers as the mean of the points

$$c^t = \arg\min_{c} \frac{1}{N} \sum_{j}^{N} \sum_{i}^{K} \delta_{ij}^{t} (c_i^{t-1} - v_j)^2$$

4. Update $t = t + 1$, Repeat Step 2-3 till stopped

Slide: Derek Hoiem

K-means clustering

1. Initialize ($t = 0$): cluster centers $c_1, ..., c_K$

2. Compute δ^t : assign each point to the closest center
   - δ^t denotes the set of assignment for each v_j to cluster c_i at iteration *t*

$$\delta^t = \arg\min_{\delta} \frac{1}{N} \sum_{j}^{N} \sum_{i}^{K} \delta_{ij}^{t-1} (c_i^{t-1} - v_j)^2$$

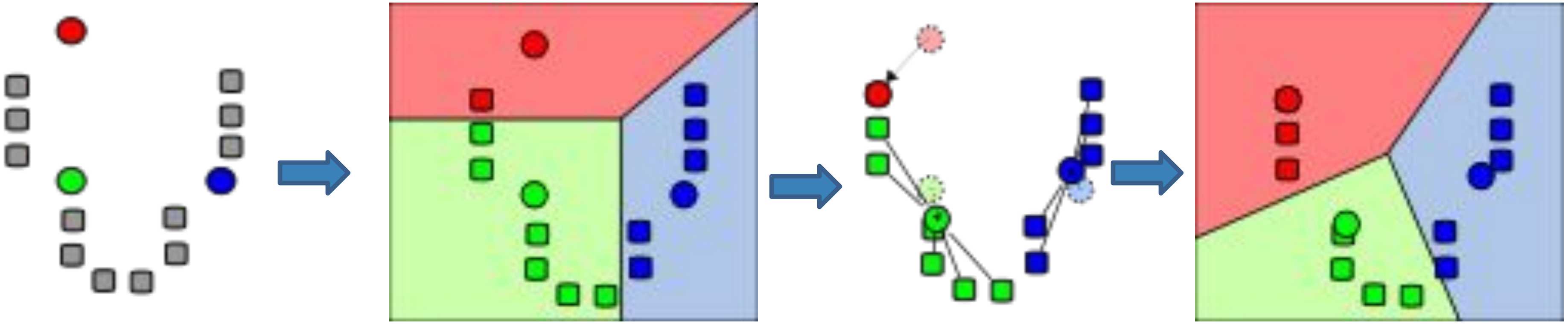
3. Computer c^t : update cluster centers as the mean of the points

$$c^t = \arg\min_{c} \frac{1}{N} \sum_{j}^{N} \sum_{i}^{K} \delta_{ij}^{t} (c_i^{t-1} - v_j)^2$$

4. Update $t = t + 1$, Repeat Step 2-3 till stopped

Slide: Derek Hoiem

K-means clustering



1. Initialize Cluster Centers

2. Assign Points to Clusters

3. Re-compute Means

Repeat (2) and (3)

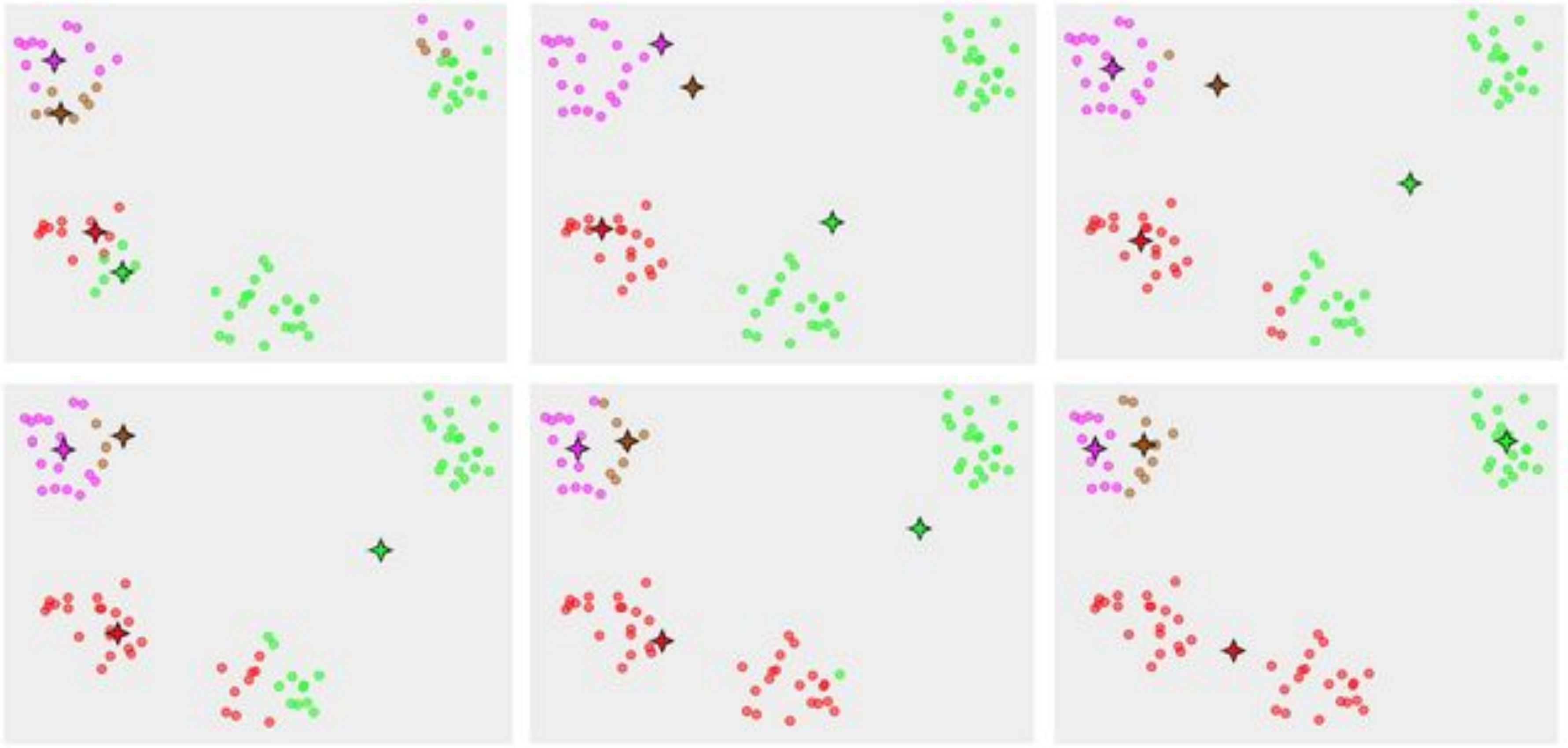
K-means clustering

Initial cluster centers are randomly initialized

- Can lead to bad initializations
- Can cause bad clusters

Another example of how K-means Converges to a local minimum solution

Initialize multiple runs!



K-Means++

Tries to prevent arbitrarily bad local minima?

1. Randomly choose first center.
2. Pick new center with prob. proportional to $(c_i - v_j)^2$
   a. Basically we want to find as good of an initialization as possible
3. Repeat until *K* centers.

K-means clustering

Initial cluster centers are randomly initialized

- Can lead to bad initializations
- Can cause bad clusters

Different distance measures can change K-Means clusters

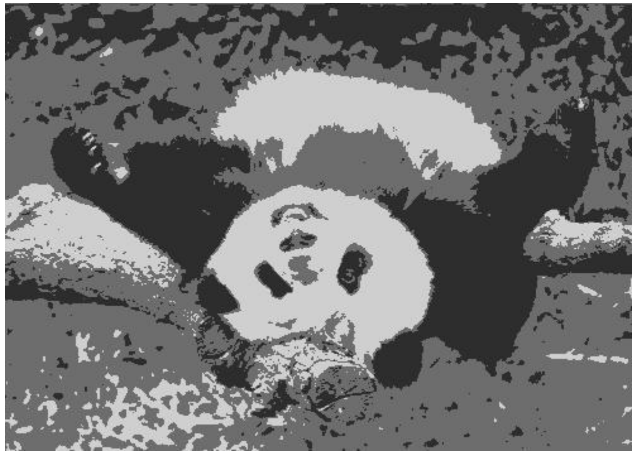
- Euclidean distance of cosine distance.

Different feature space can lead to different cluster

Segmentation as Clustering

Original image

2 clusters



3 clusters

Feature Space: pixel value

- Feature space: what measurements do we include in x_i?
- Depending on what we choose as the *feature space*, we can group pixels in different ways.

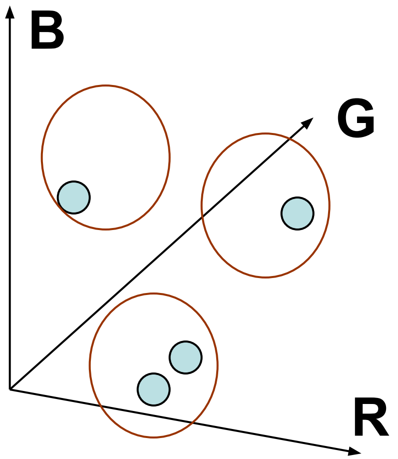
- Grouping pixels based on intensity similarity

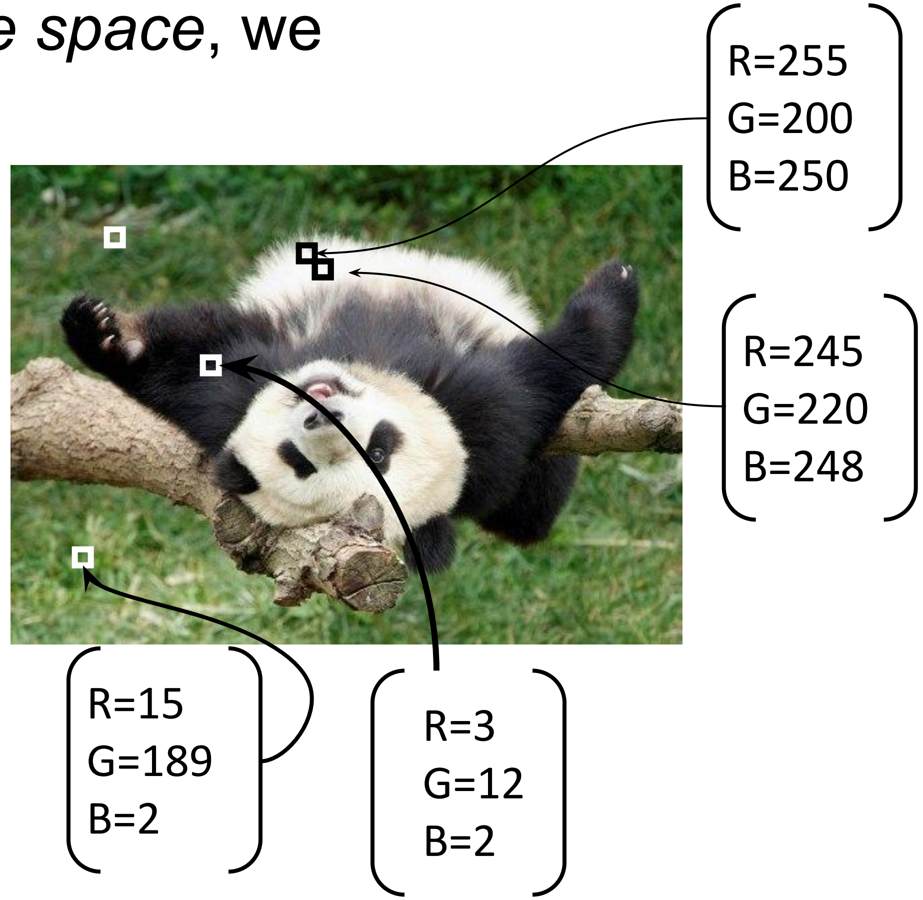
- Feature space: intensity value (1D)

Feature Space: RGB

- Depending on what we choose as the *feature space*, we can group pixels in different ways.

- Grouping pixels based on color similarity

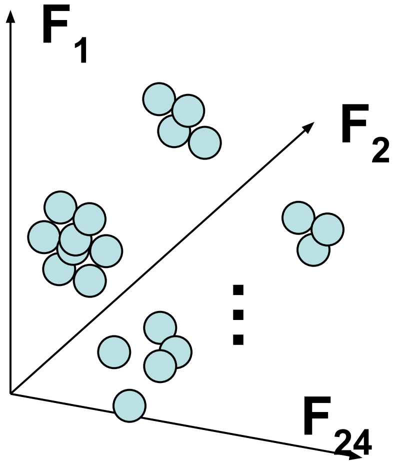


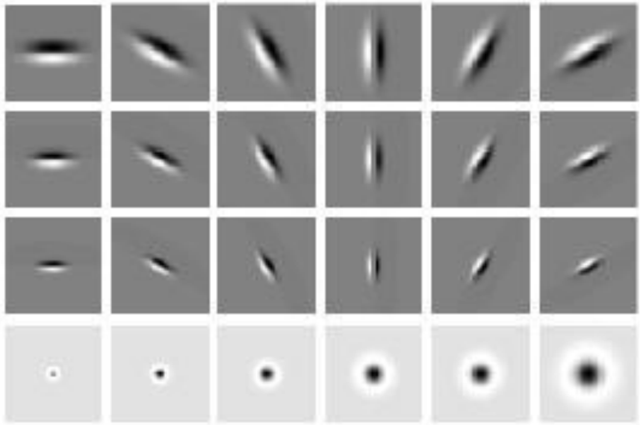


- Feature space: color value (3-dim)

Feature Space: edges and blobs

- Depending on what we choose as the *feature space*, we can group pixels in different ways.
- Grouping pixels based on oriented gradient similarity



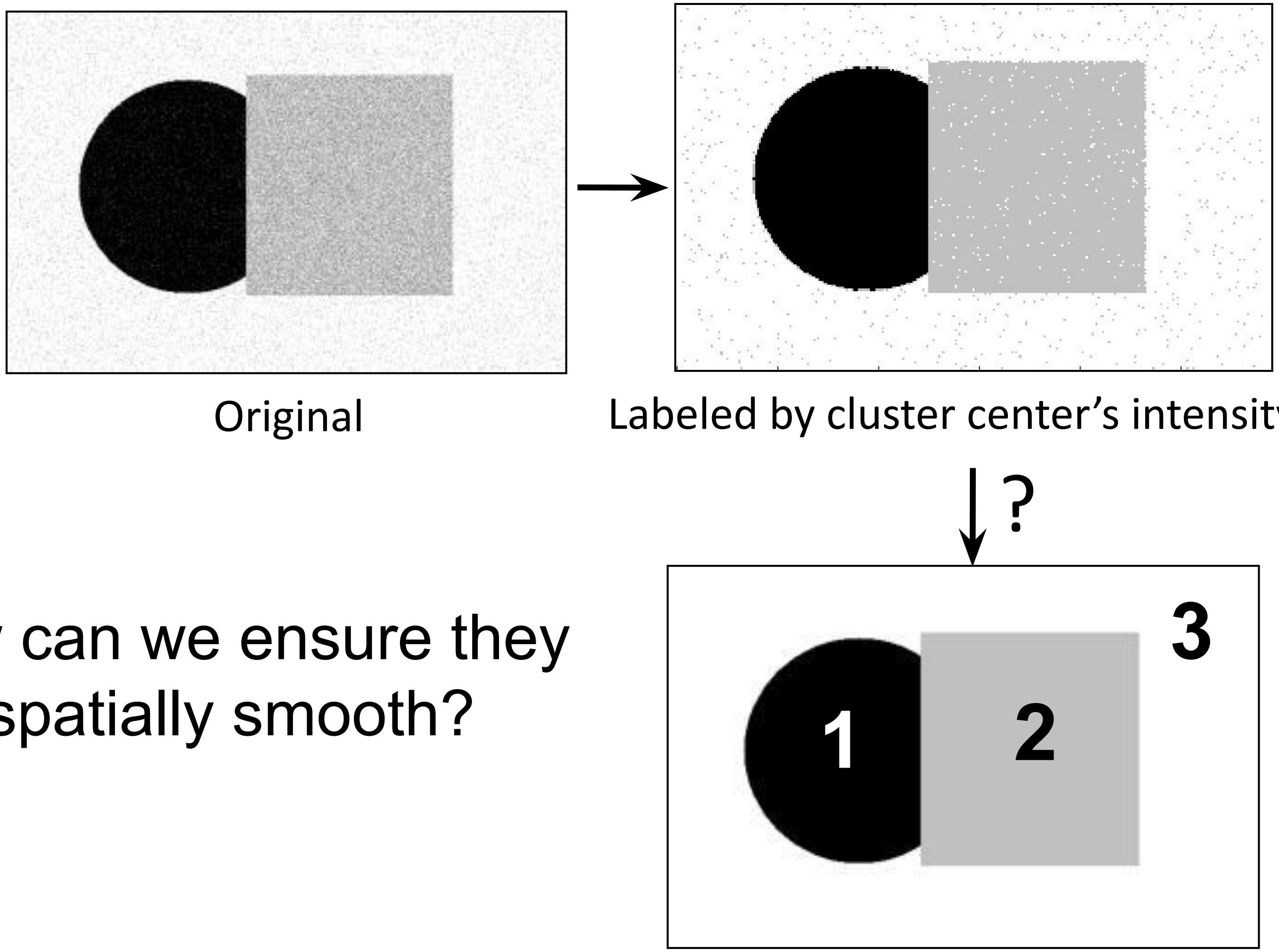


24 edge & blog filters

- Feature space: filter bank responses (e.g., 24D)

Smoothing Out Cluster Assignments

- Assigning a cluster label per pixel may yield outliers:



Original

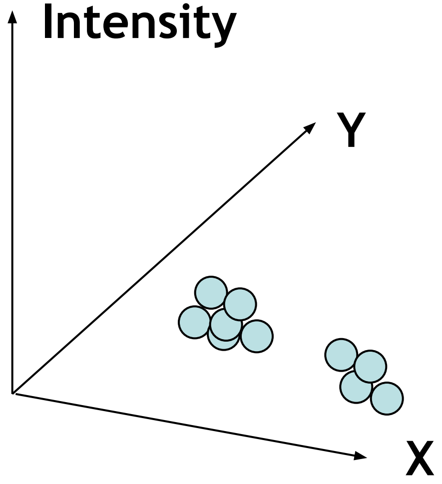
Labeled by cluster center's intensity

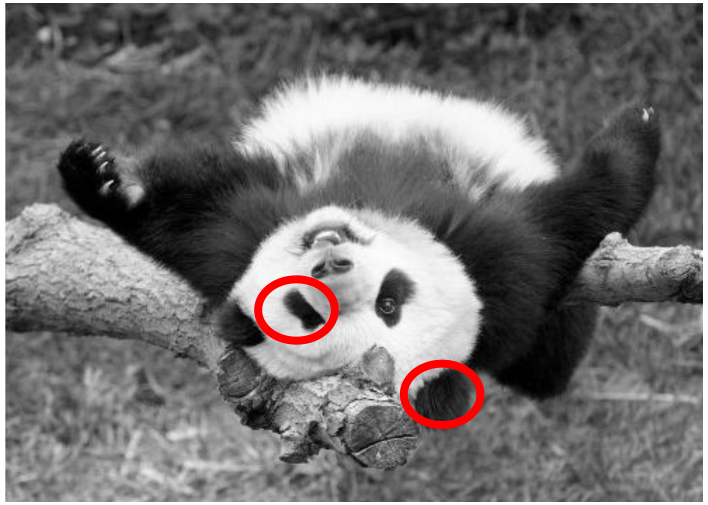
- How can we ensure they are spatially smooth?

Feature Space: RGB + XY location

- Depending on what we choose as the *feature space*, we can group pixels in different ways.

- Grouping pixels based on *intensity+position* similarity





⇒ Way to encode both *similarity* and *proximity.*

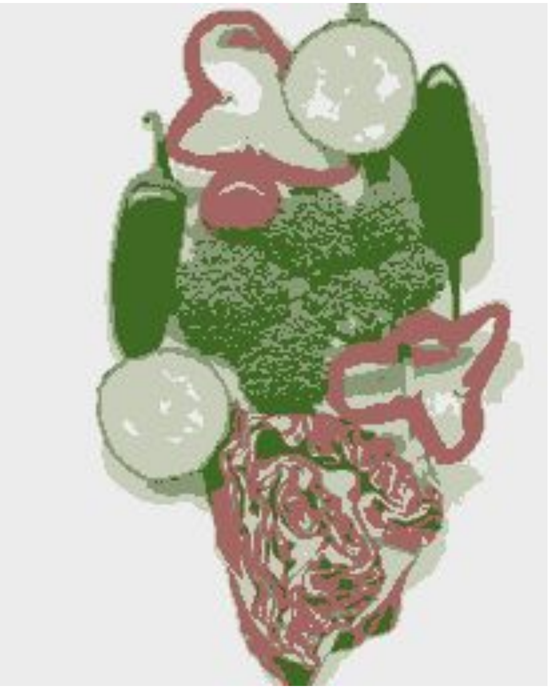
K-Means Clustering Results

- Clusters don’t have to be spatially coherent

Image

grayscale clusters

Color-based clusters



K-Means Clustering Results

- Clustering based on (r,g,b,x,y) values enforces more spatial coherence

How to evaluate clusters?

- **Generative**
  - How well are points reconstructed from the clusters?

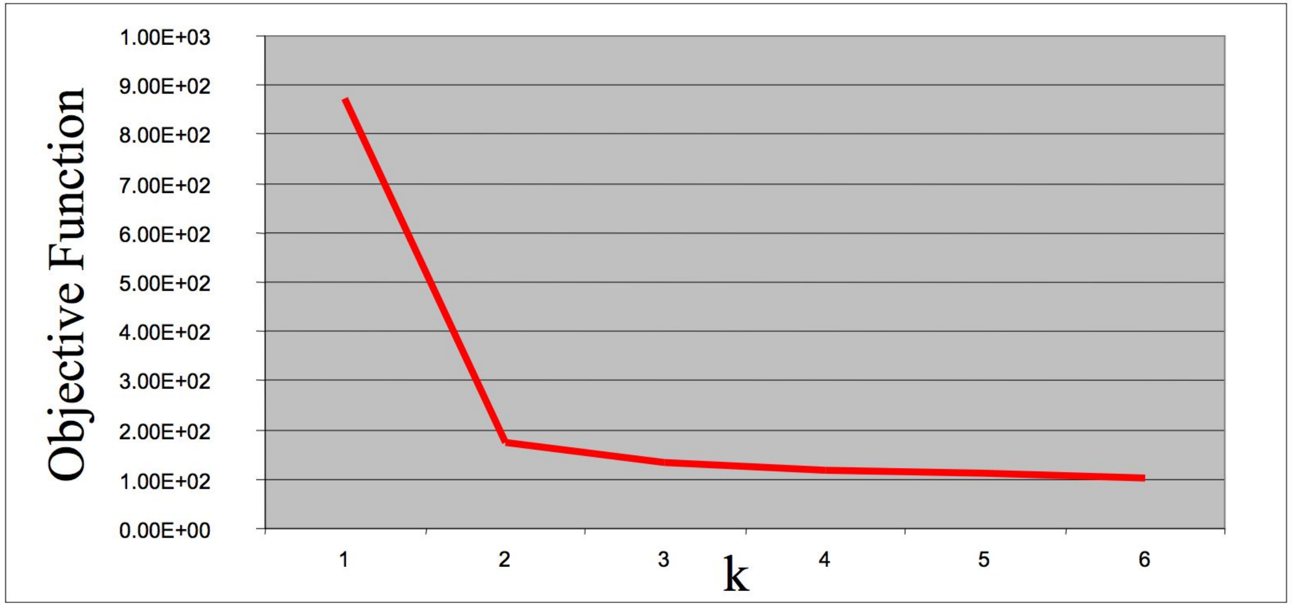
- **Discriminative**
  - How well do the clusters correspond to labels?
    - Can we correctly classify which pixels belong to the panda?
  - Note: unsupervised clustering does not aim to be discriminative as we don't have the labels.

How to choose the number of clusters?

Try different numbers of clusters in a validation set and look at performance.

Plot of SSD versus values of k

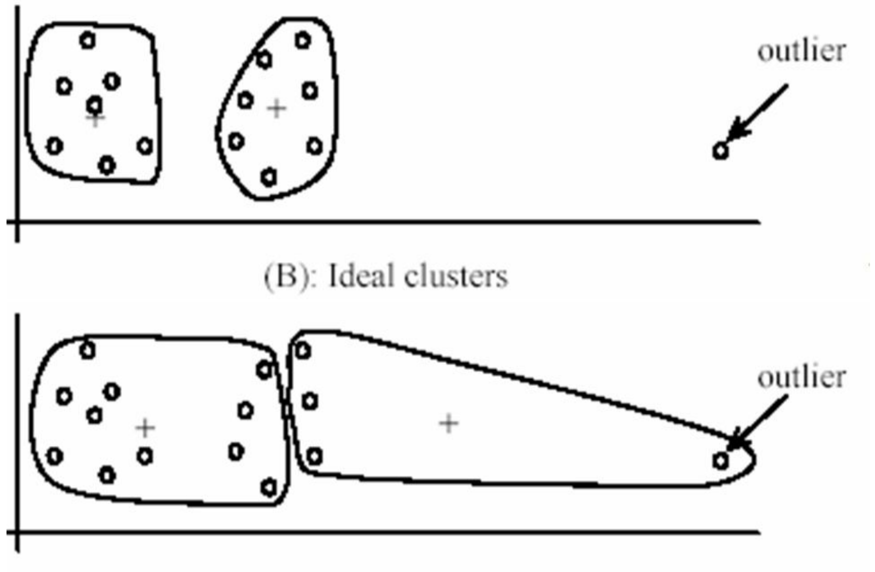
abrupt change at k=2 is suggestive of two clusters in the data

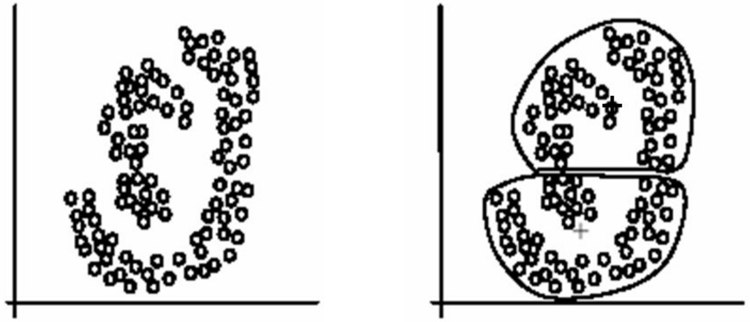


Slide: Derek Hoiem

K-Means pros and cons

- **Pros**
  - Good representation of data
  - Simple and fast, Easy to implement
- **Cons**
  - Need to choose K
  - Sensitive to outliers
  - Prone to local minima
  - All clusters have the same parameters (e.g., distance measure is non-adaptive)
  - Can still be slow: each iteration is O(KNd) for N d-dimensional pixels



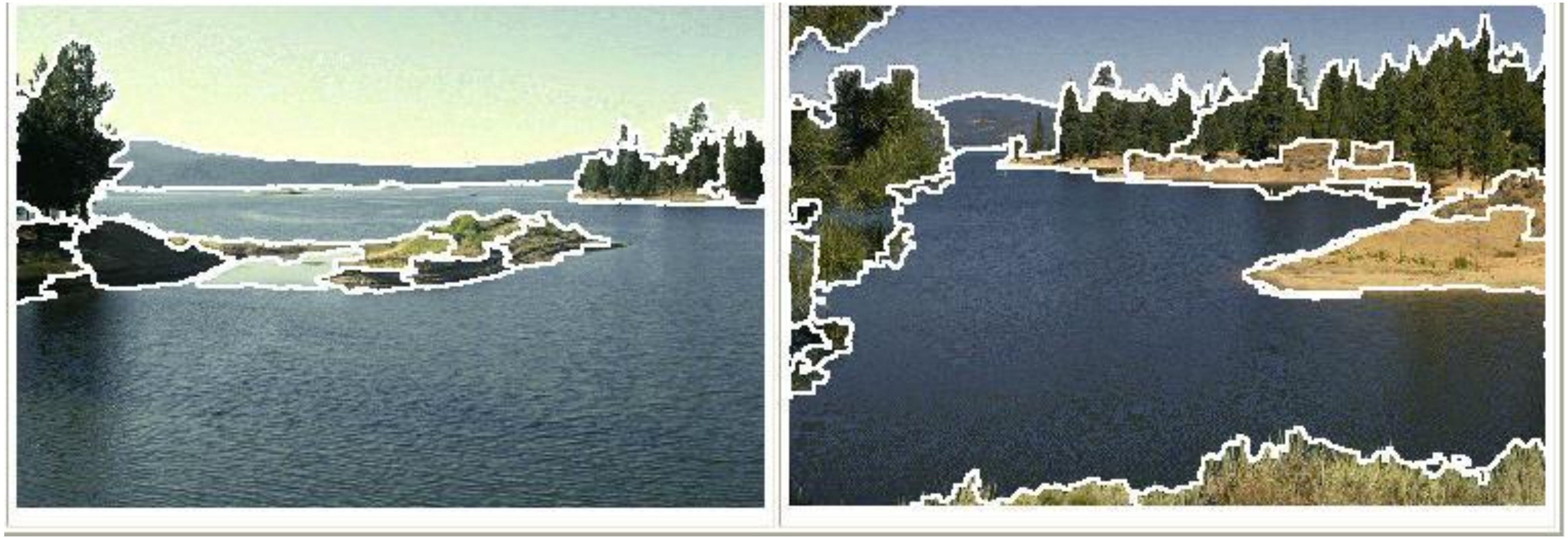

(A): Two natural clusters (B): k-means clusters

What will we learn today?

- K-means clustering
- Mean-shift clustering
- Normalized cuts

Mean-Shift Segmentation

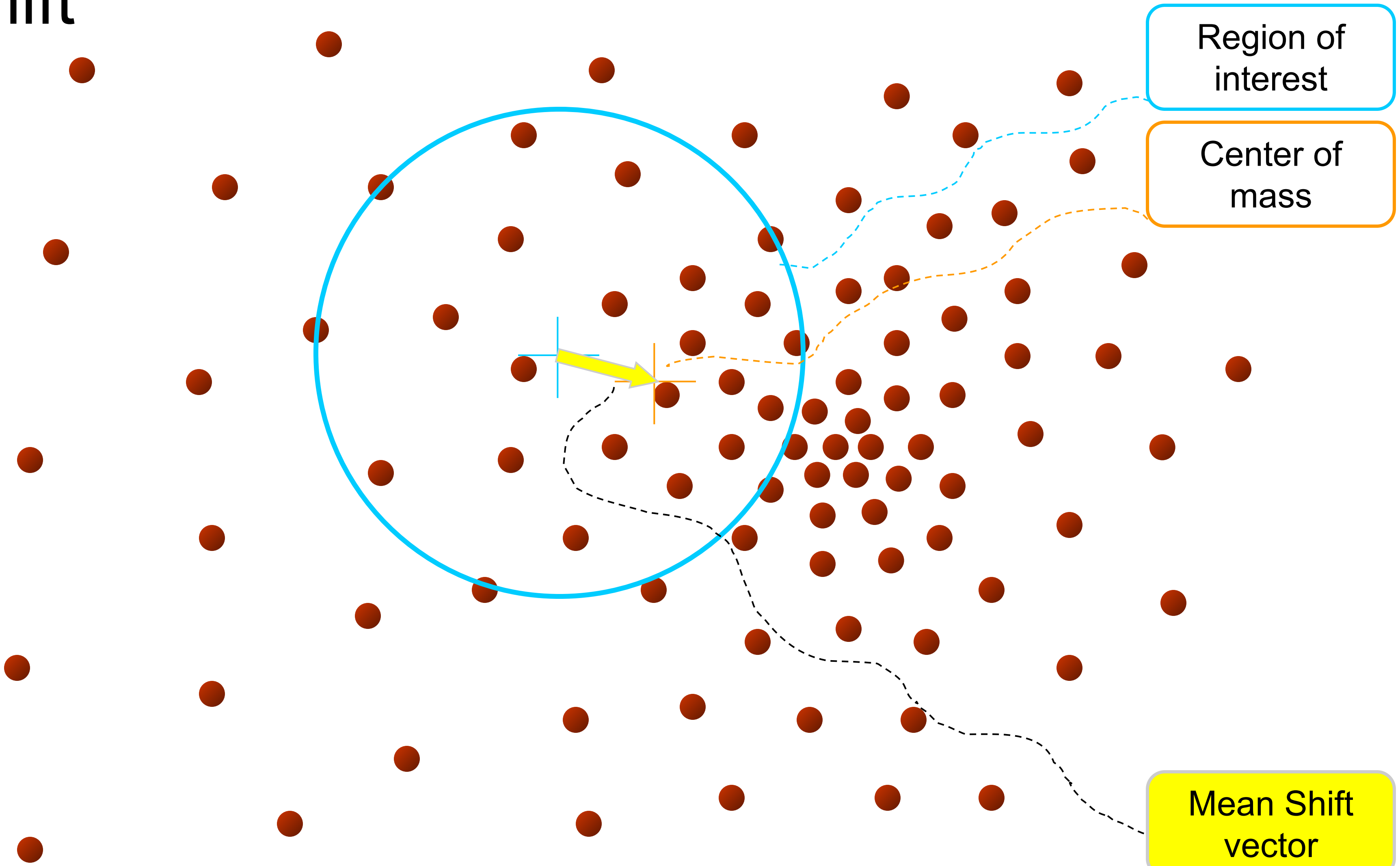
- An advanced and versatile technique for clustering-based segmentation



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

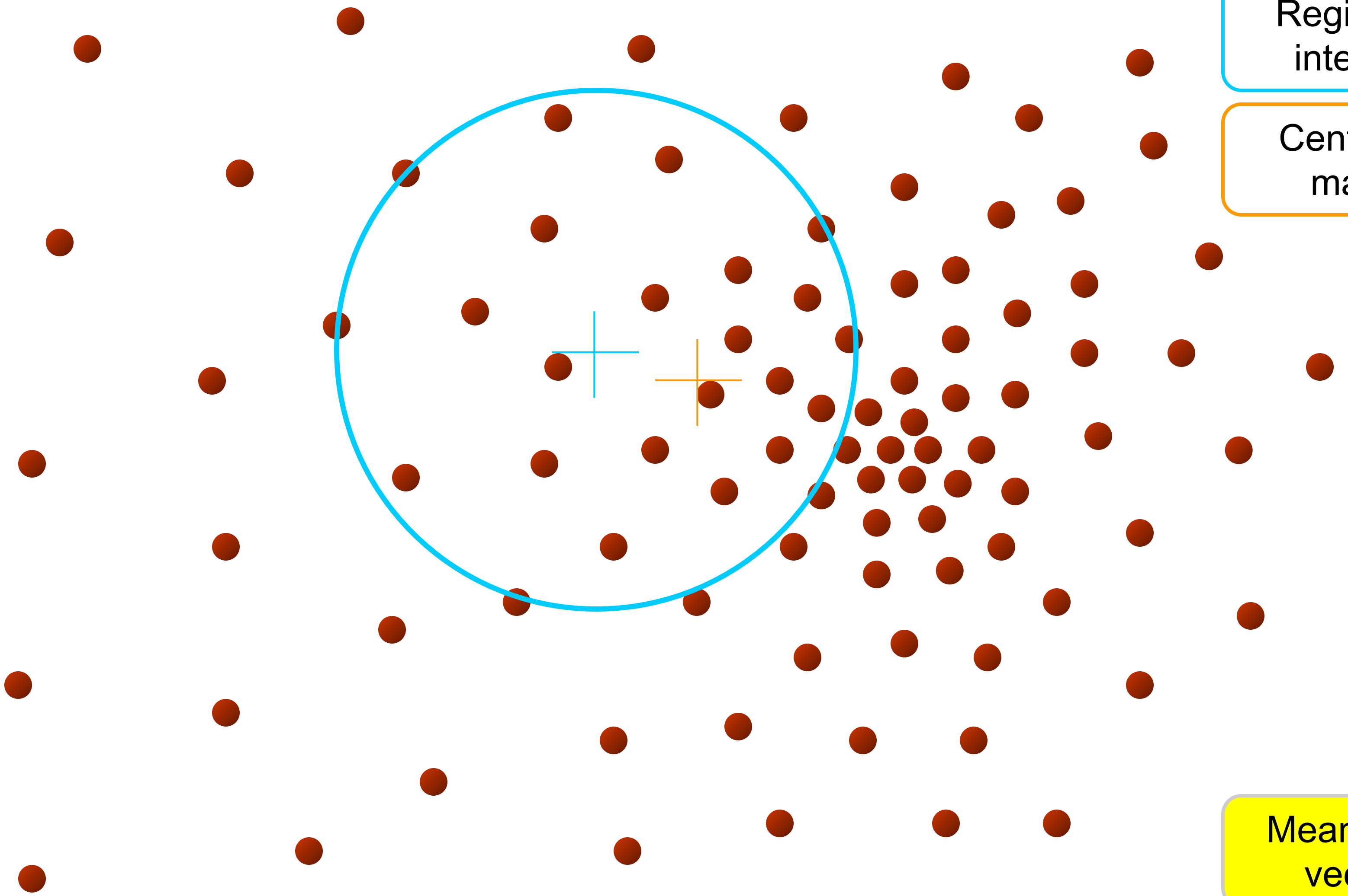
D. Comaniciu and P. Meer, Mean Shift: A Robust Approach toward Feature Space Analysis, PAMI 2002.

Mean-Shift



Slide by Y. Ukrainitz & B. Sarel

Mean-Shift



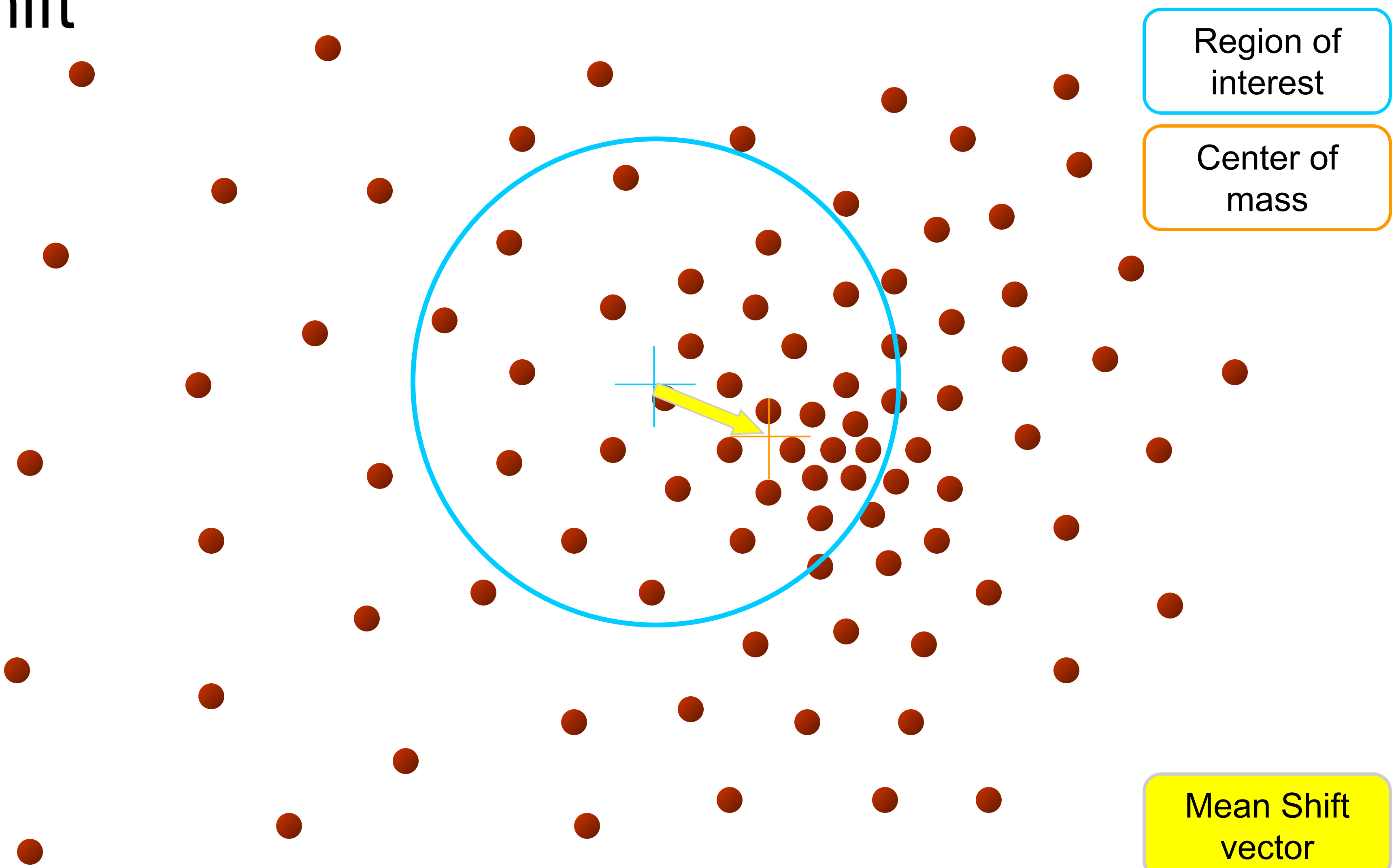
Region of interest

Center of mass

Mean Shift vector

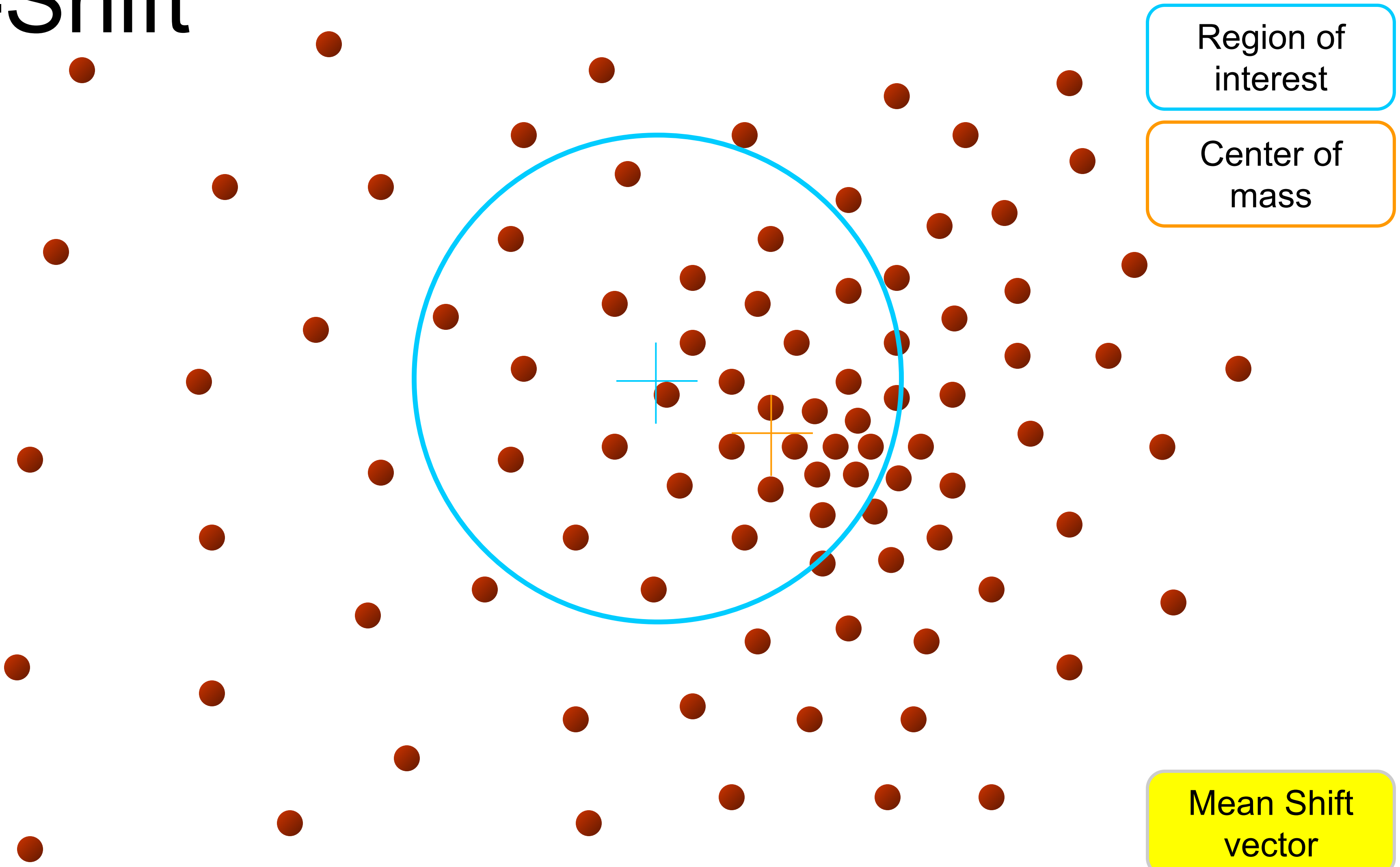
Slide by Y. Ukrainitz & B. Sarel

Mean-Shift



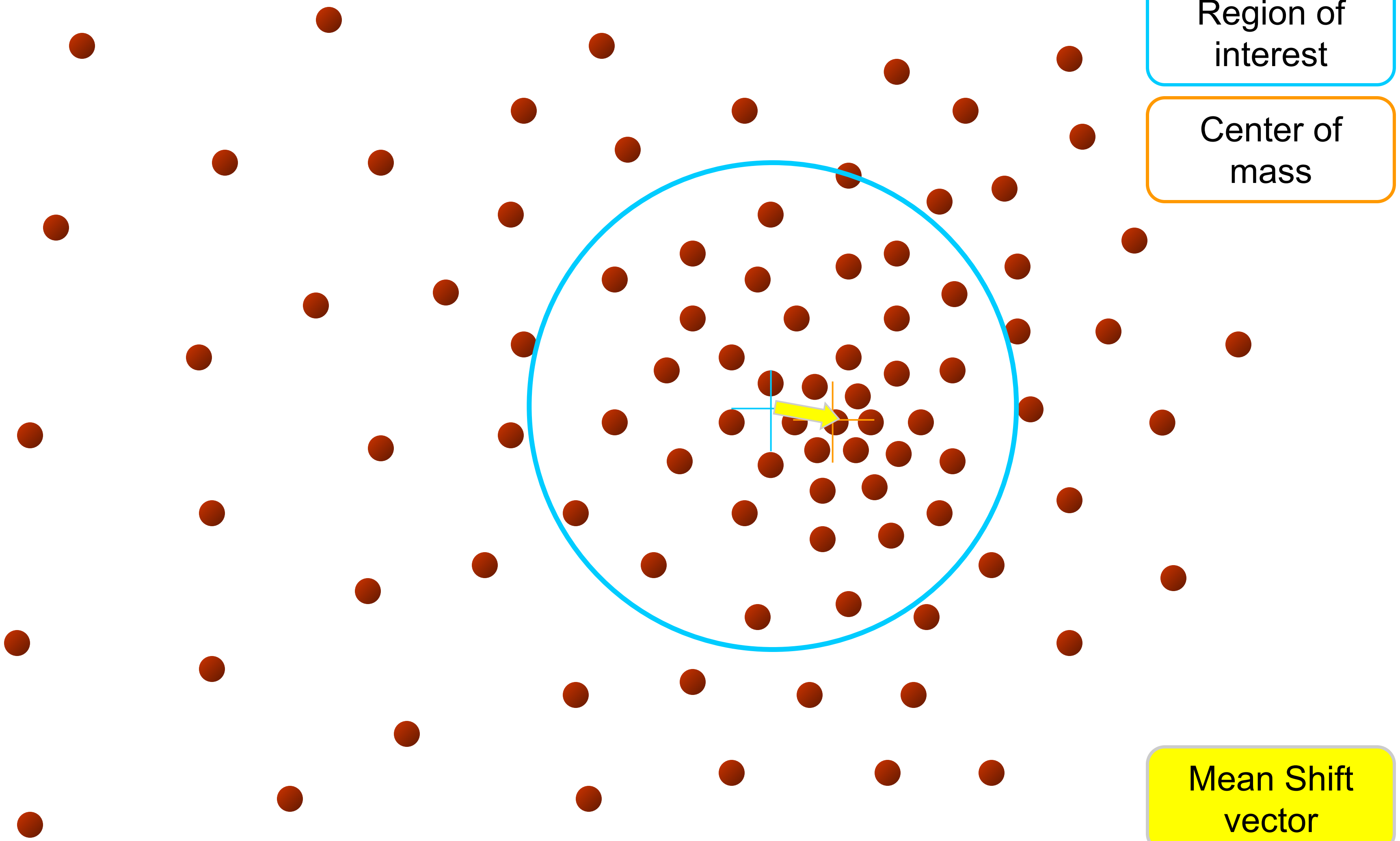
Slide by Y. Ukrainitz & B. Sarel

Mean-Shift



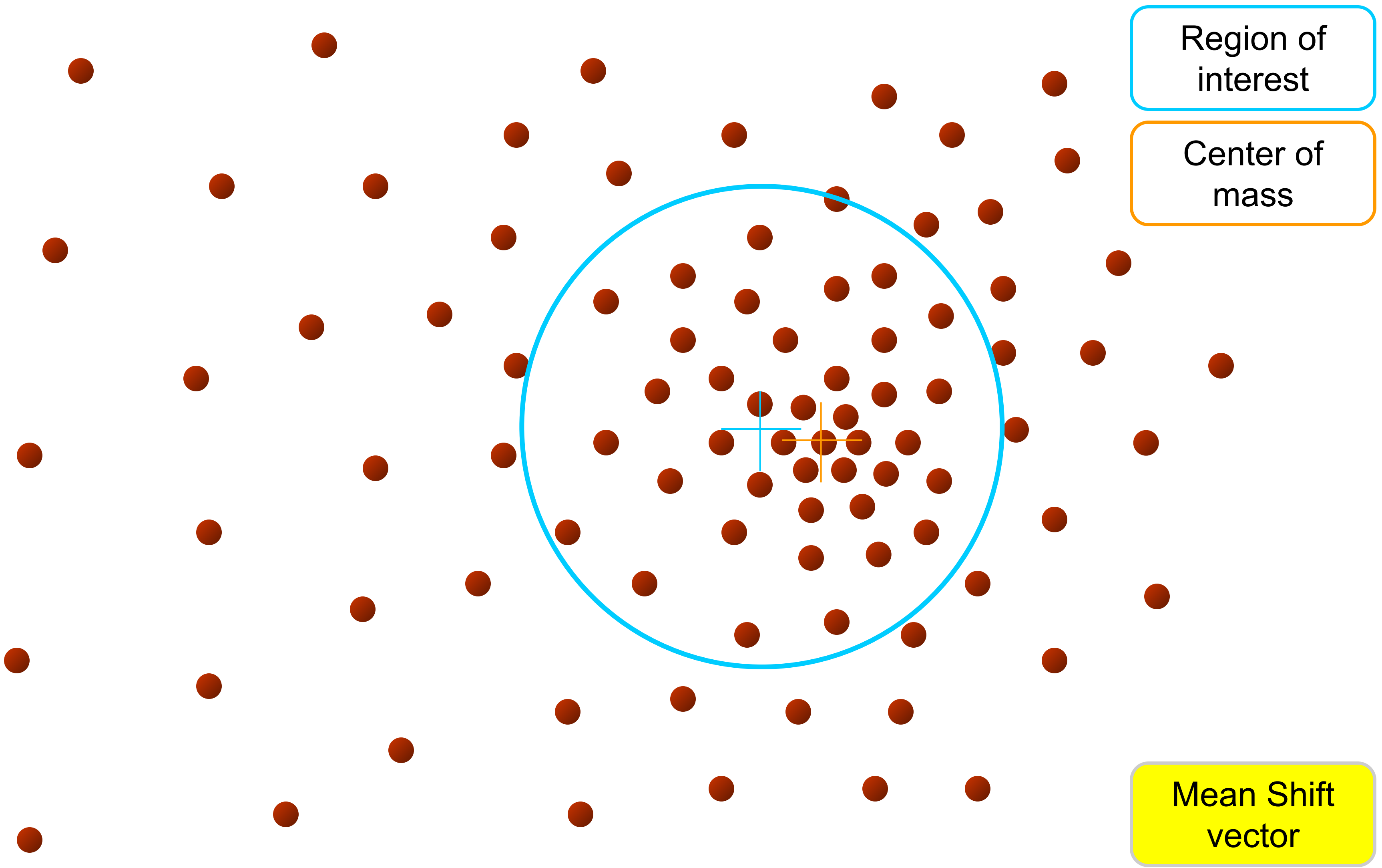
Slide by Y. Ukrainitz & B. Sarel

Mean-Shift



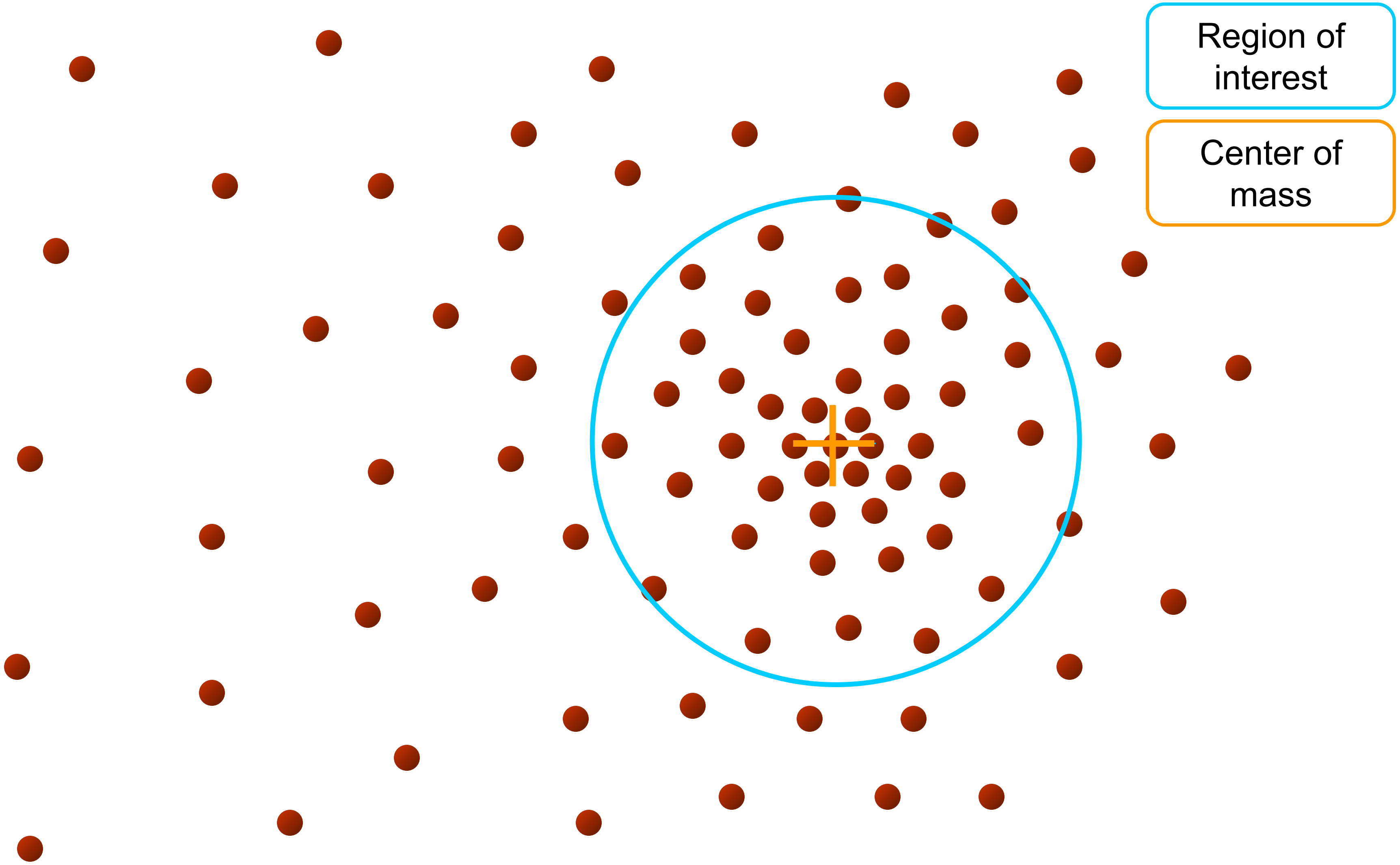
Slide by Y. Ukrainitz & B. Sarel

Mean-Shift



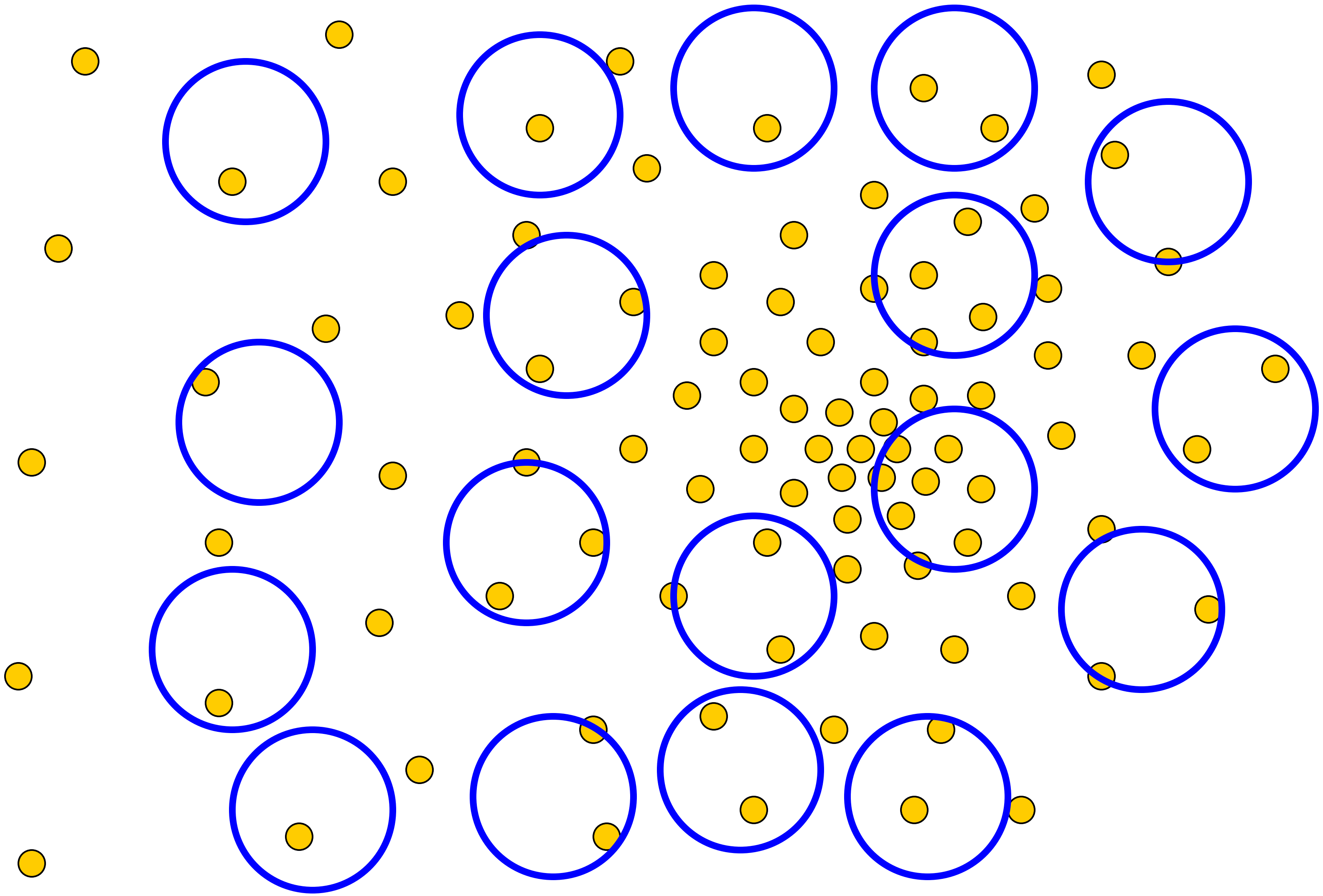
Slide by Y. Ukrainitz & B. Sarel

Mean-Shift



Slide by Y. Ukrainitz & B. Sarel

Real Modality Analysis

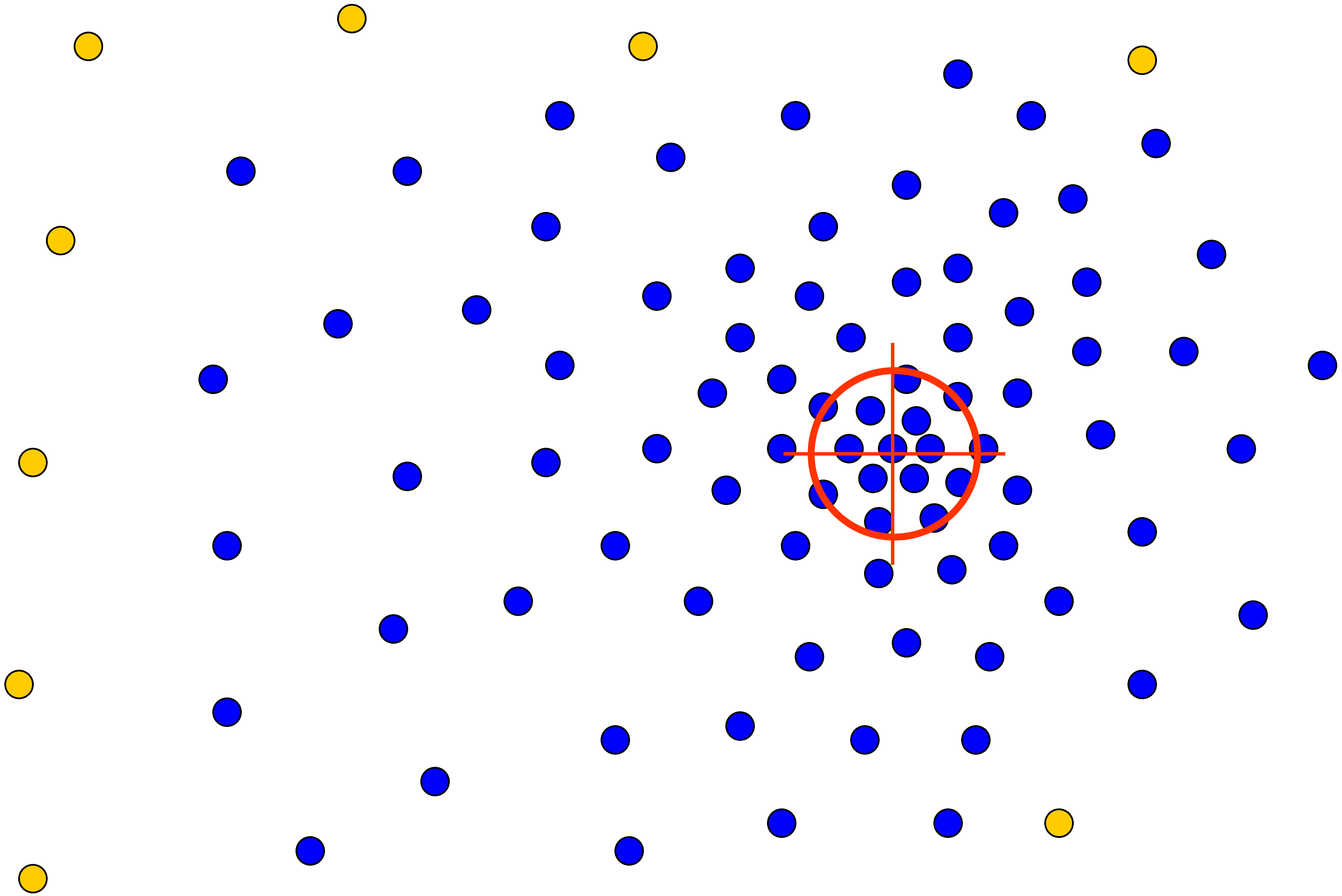


Tessellate the space with windows

Run the procedure in parallel

Slide by Y. Ukrainitz & B. Sarel

Real Modality Analysis



Slide by Y. Ukrainitz & B. Sarel

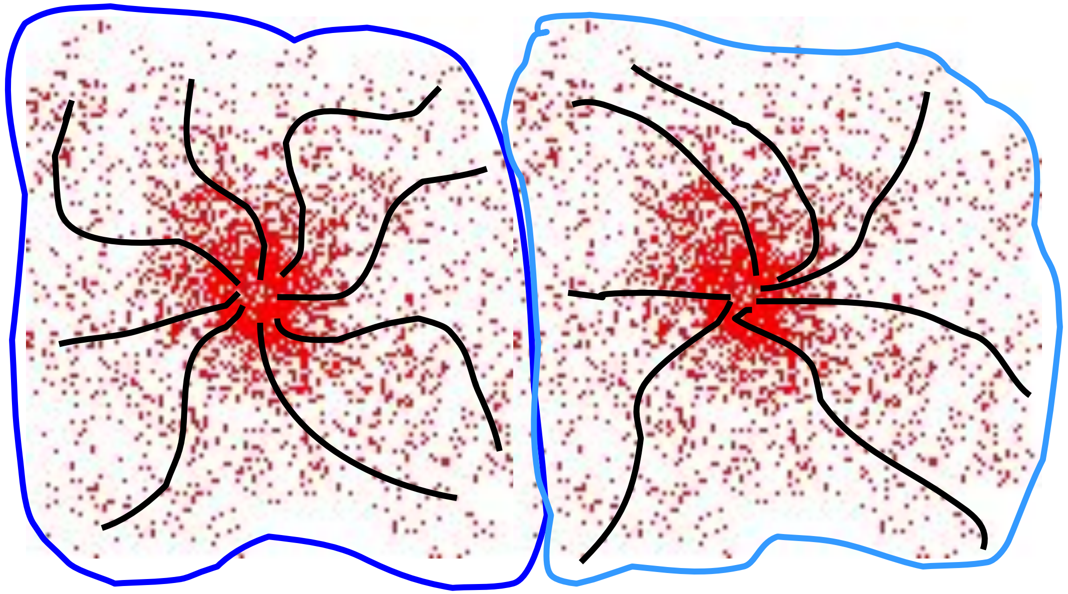
The blue data points were traversed by the windows towards the mode.

Mean-Shift Algorithm

1. Represent each pixel *i* using some feature vector v_i
2. Generate a window **W** as a random pixel feature v_w
3. Identify all the pixels within a radius *r* of v_w
4. Calculate the mean ("center of gravity") amongst the neighbors of **W**
5. Translate the window **W** to the mean feature location
6. Repeat Step 2 until convergence

Mean-Shift Clustering

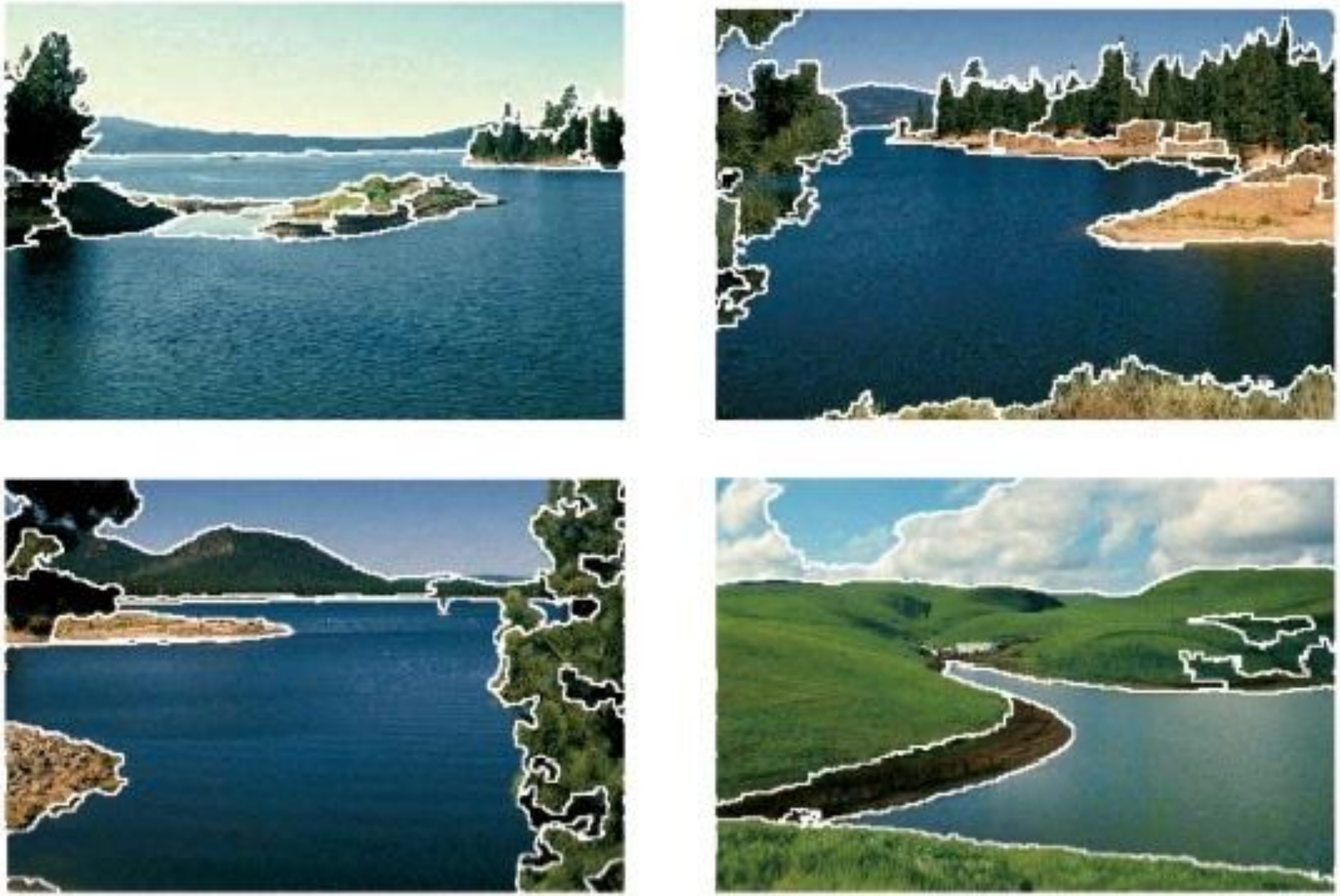
- Initialize not just 1 window but a multiple windows at random
- All pixels that end up in the same location belong to the same **cluster**
- **Attraction basin**: the feature region for which all windows end up in the same location



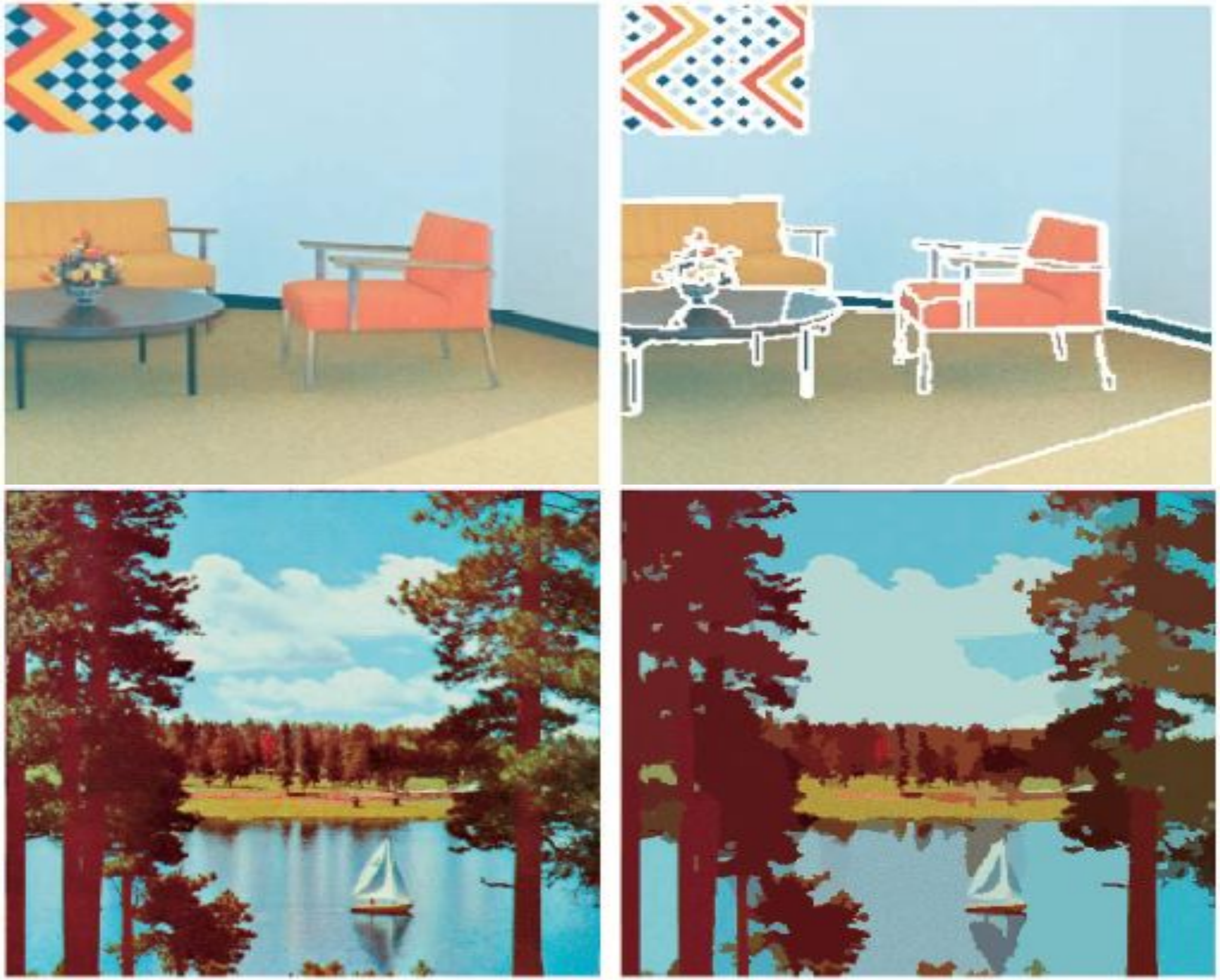
Mean-Shift Segmentation Results

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

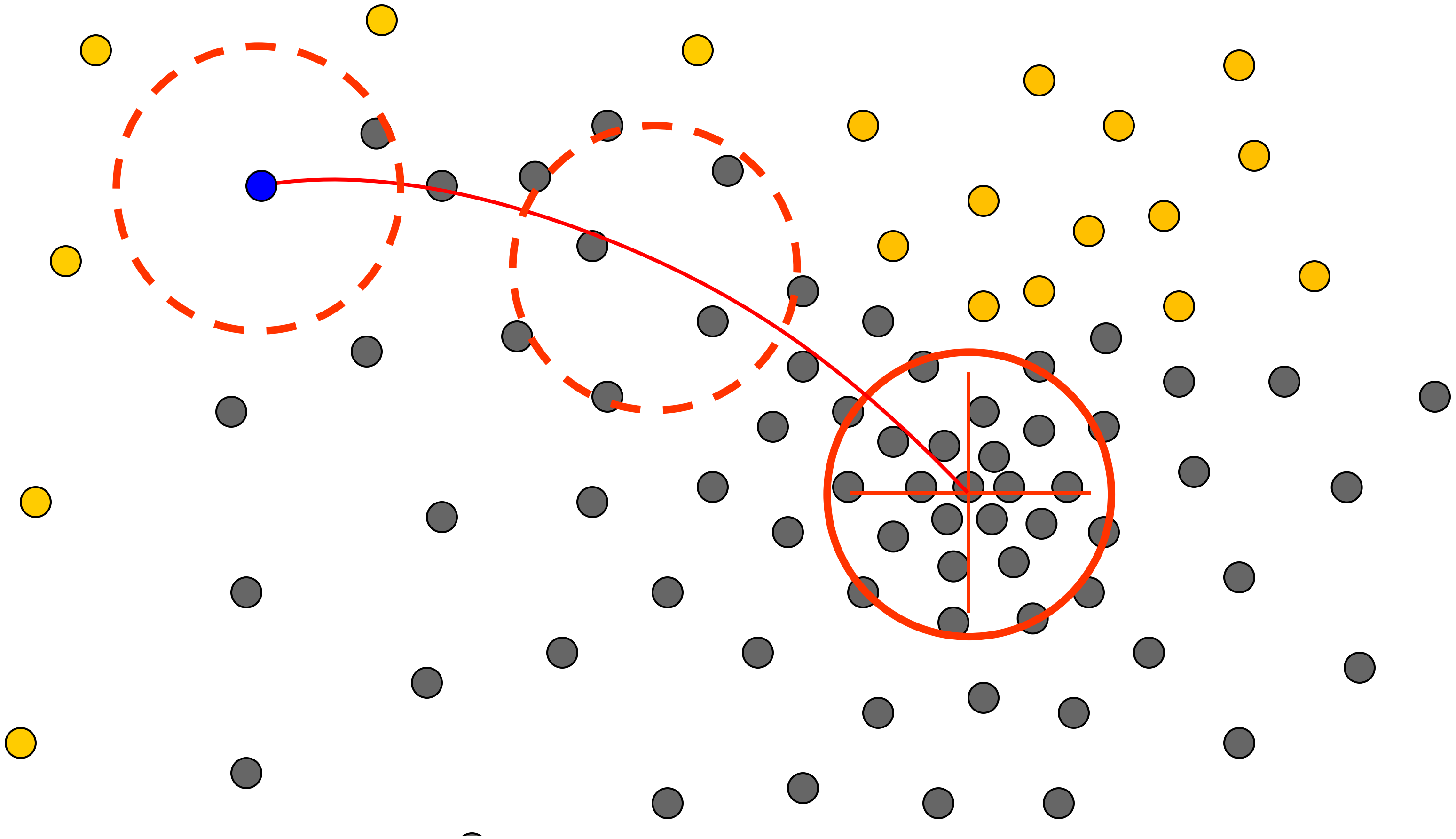
More Results



More Results

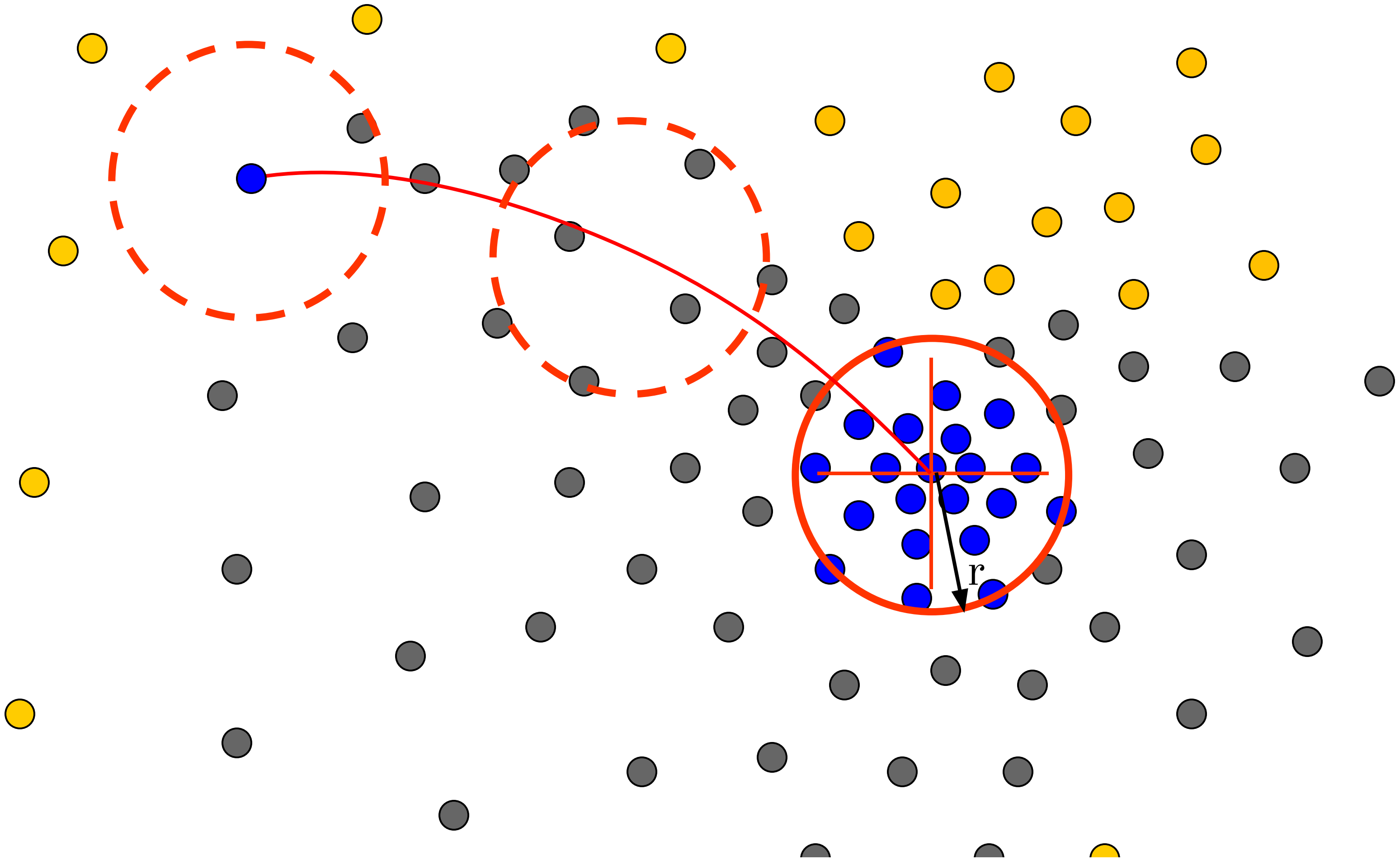


Problem: Computational Complexity



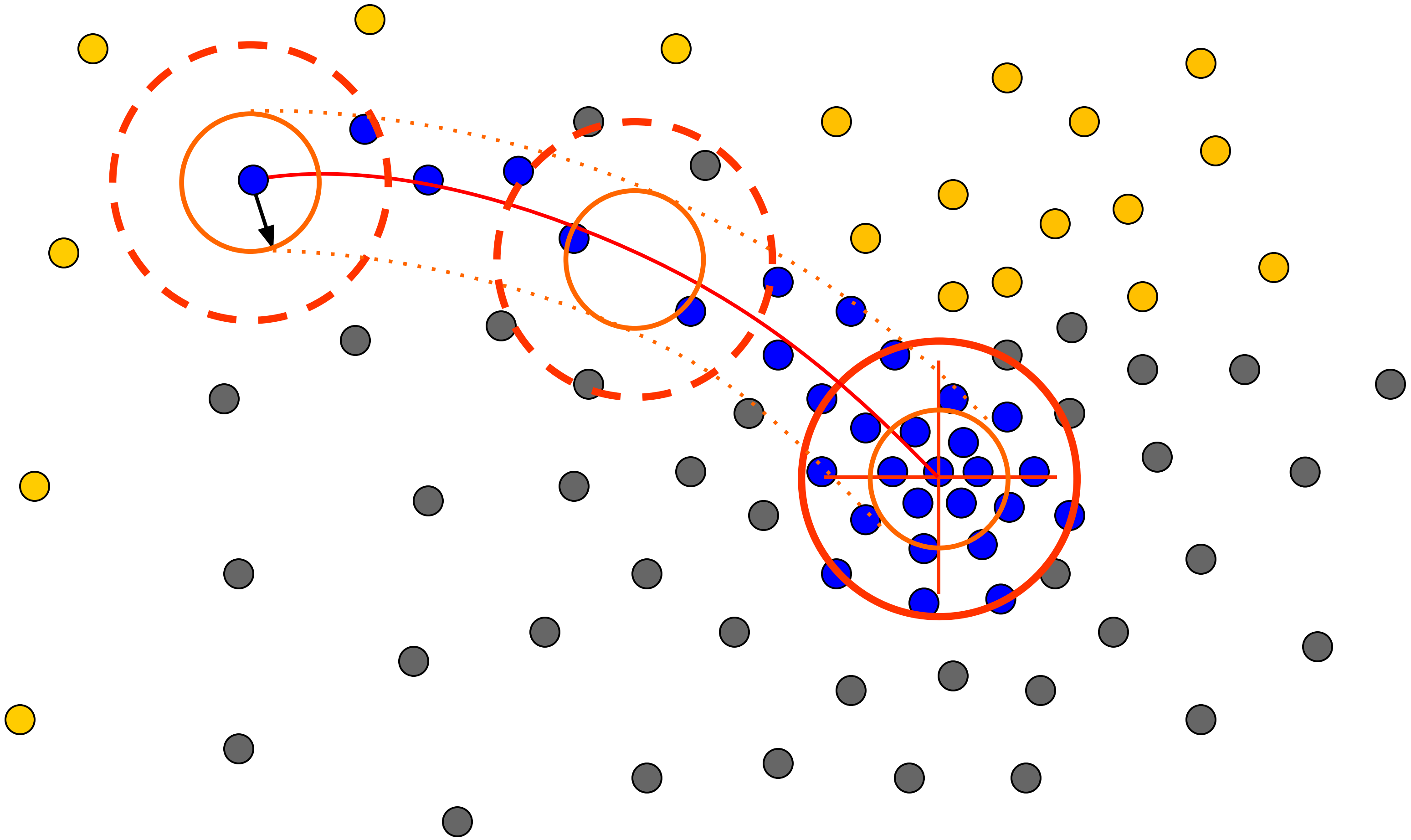
- Need to shift one window for every pixel
- Many computations will be redundant.

Speedups: Basin of Attraction



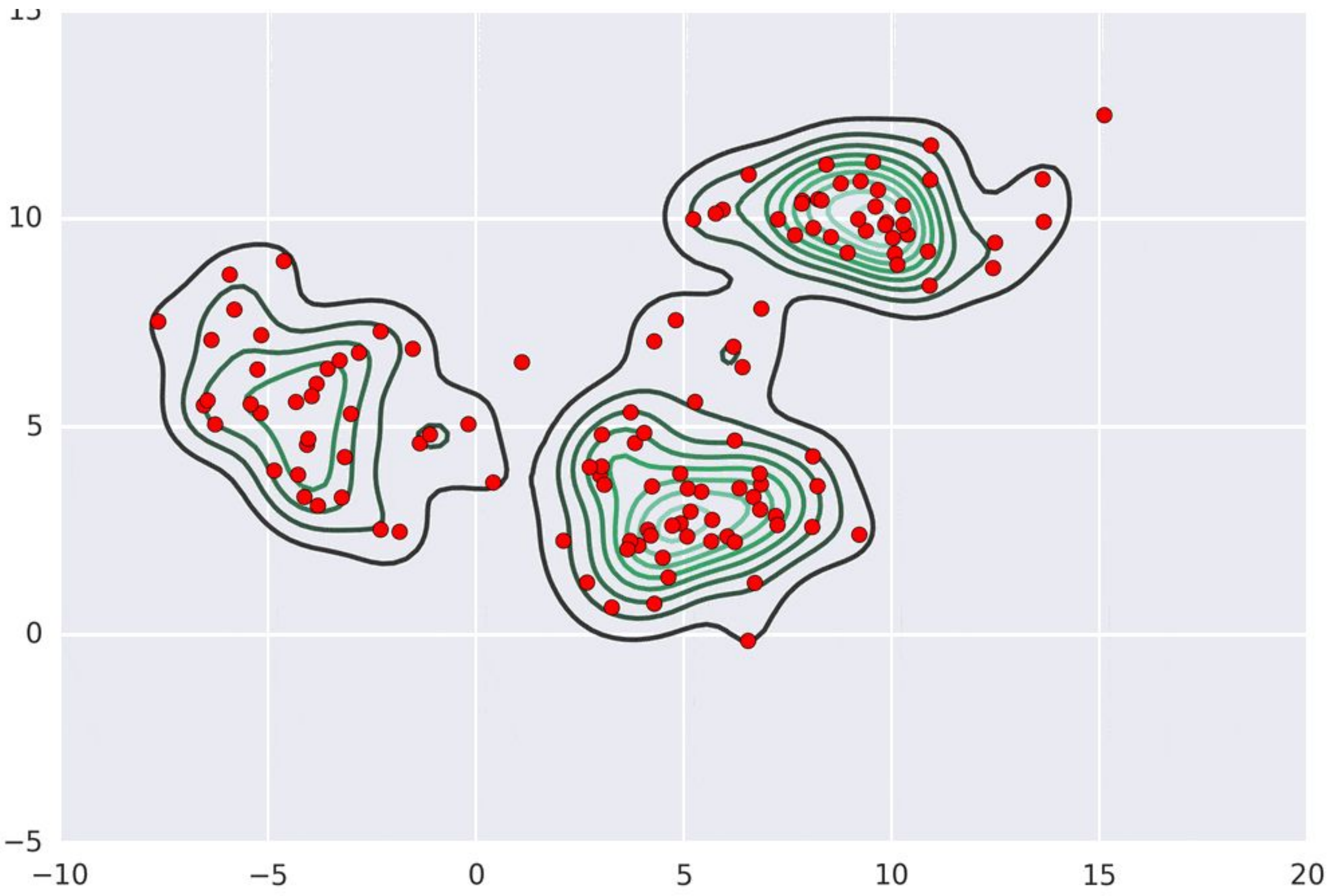
1. Assign all points within radius r of end point to the mode.

Speedups

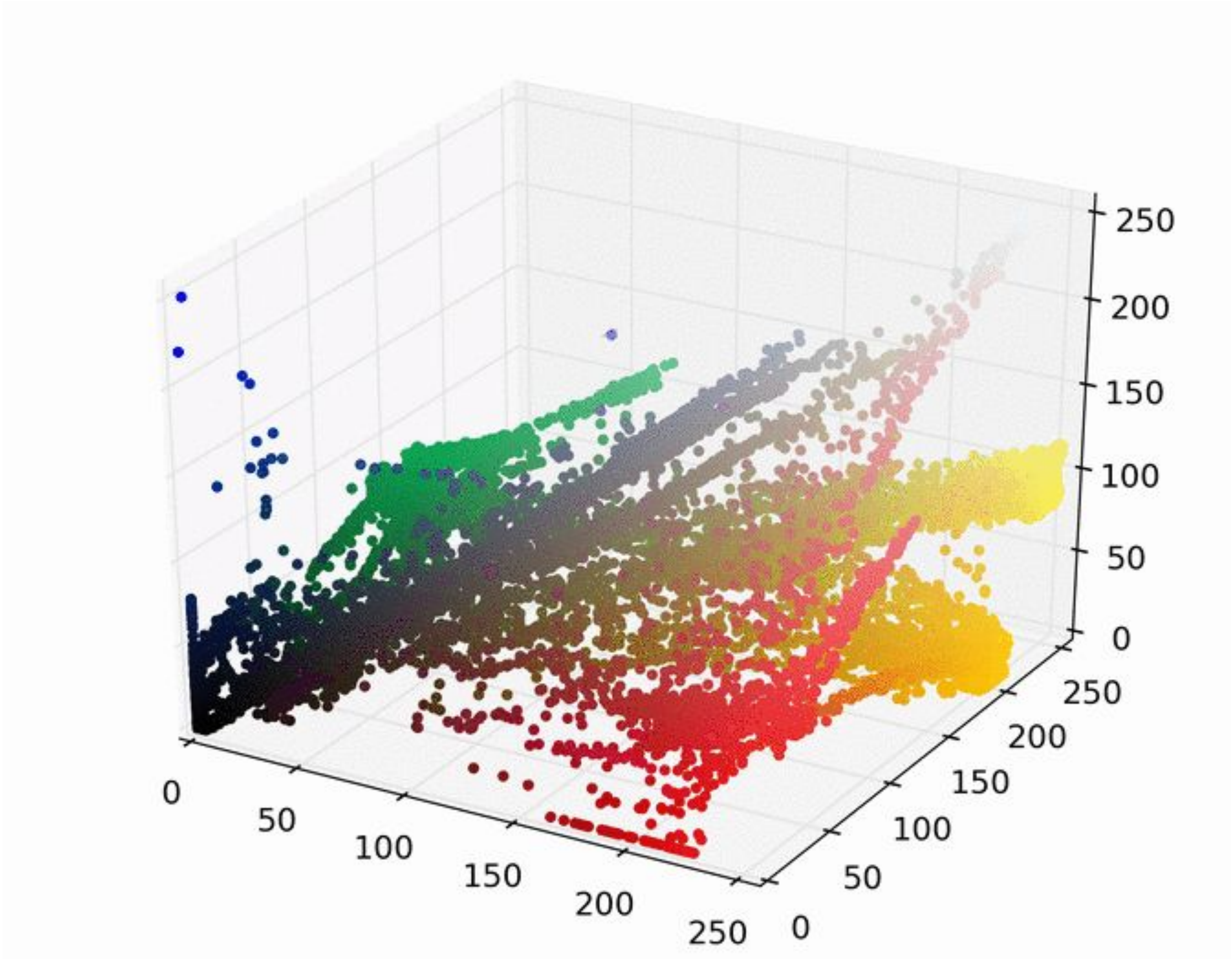


2. Assign all points within radius r/c of the search path to the mode -> reduce the number of data points to search.

Example of what running mean shift looks like

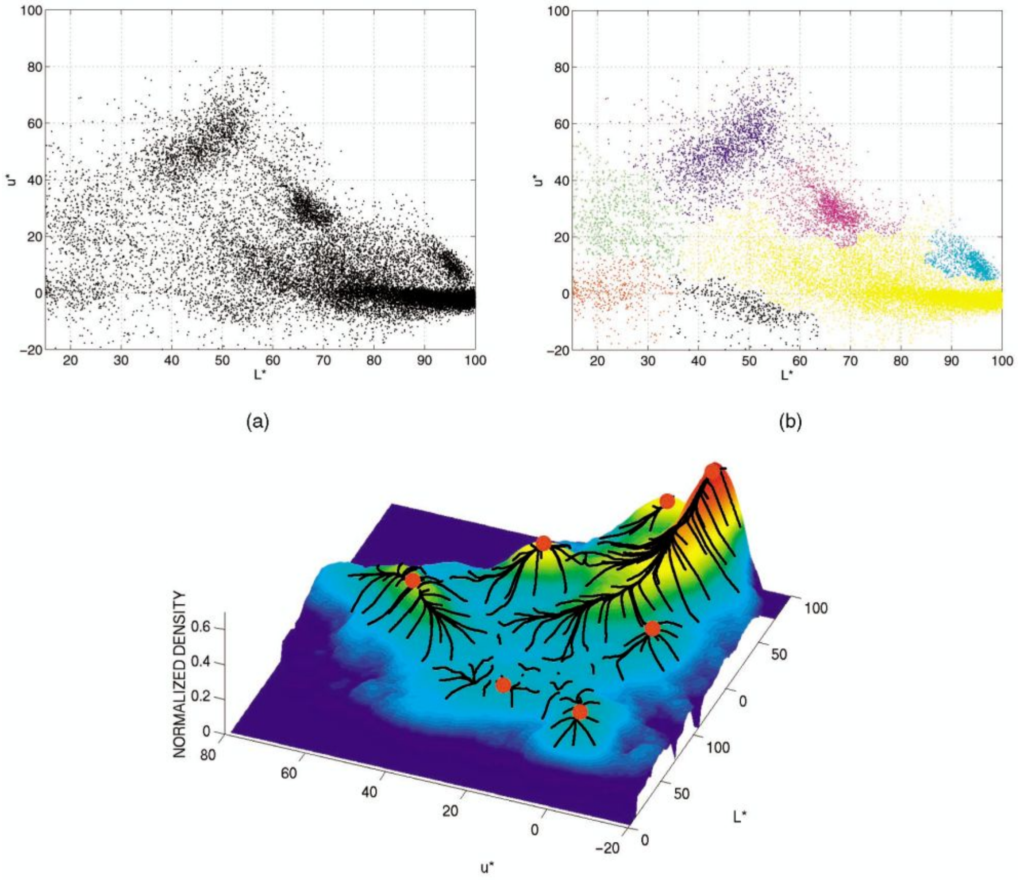


Another example



Mean-Shift Clustering

- Find features (color, gradients, texture, etc)
- Initialize windows at individual pixel locations
- Perform mean shift for each window until convergence
- At every step, merge windows that have high overlap to reduce computation



(a) (b)

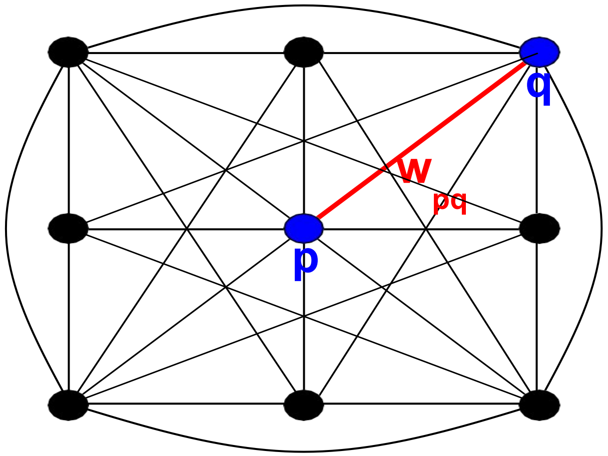
Mean-Shift pros and cons

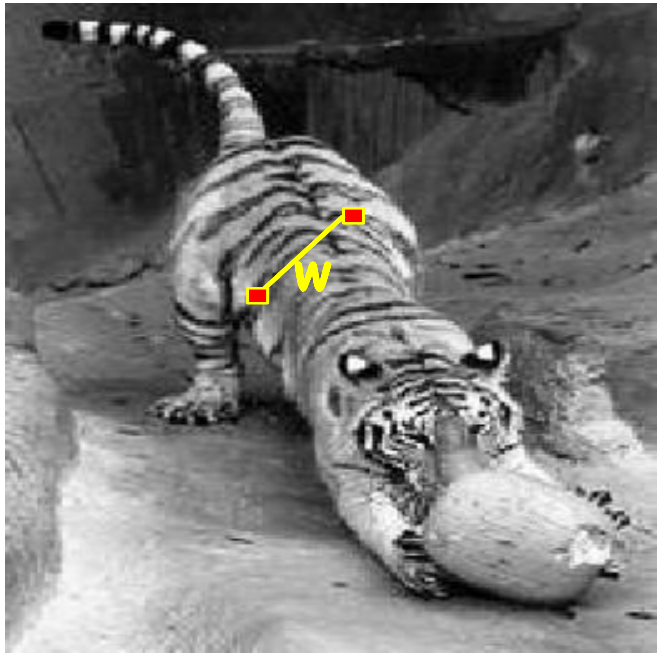
- **Pros**
  - General, application-independent algorithm
  - Model-free, does not assume any prior shape (spherical, elliptical, etc.) of data clusters
  - Just a single parameter (window size r)
    - r has a physical meaning (unlike k-means)
  - Finds variable number of modes
  - Robust to outliers
- **Cons**
  - Output depends on window size
  - Window size (bandwidth) selection is not easy
  - Computationally (relatively) expensive (~2s/image)
  - Does not scale well with dimension of feature space

Today's agenda

- K-means clustering
- Mean-shift clustering
- Normalized cuts

Images as Graphs





- Node (vertex) for every pixel
- Edge between pairs of pixels, (p,q)
- Affinity weight w_{pq} for each edge
  - w_{pq} measures similarity
  - Similarity is inversely proportional to difference (in color and position…)

slide credit: Steve Seitz

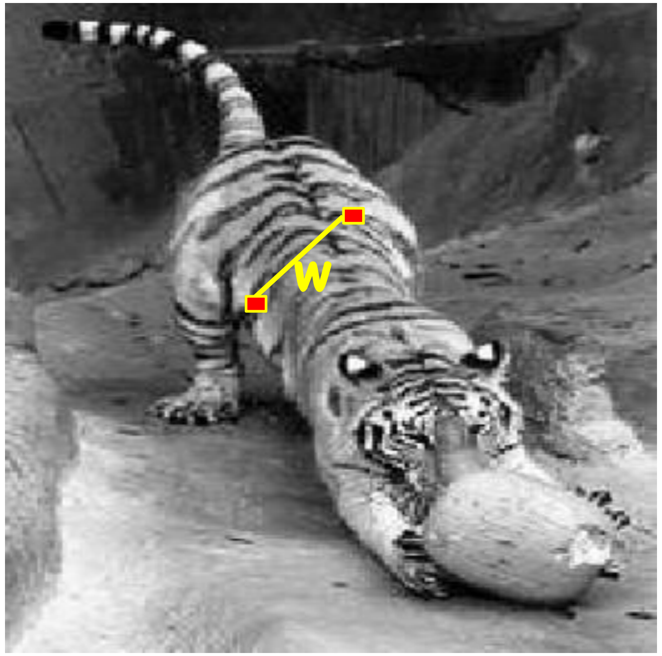
Images as Graphs

Which edges to include?

Fully connected:
- Captures all pairwise similarities
- Infeasible for most images

Neighboring pixels:
- Very fast to compute
- Only captures very local interactions

Local neighborhood:
- Reasonably fast, graph still very sparse
- Good tradeoff



Measuring Affinity

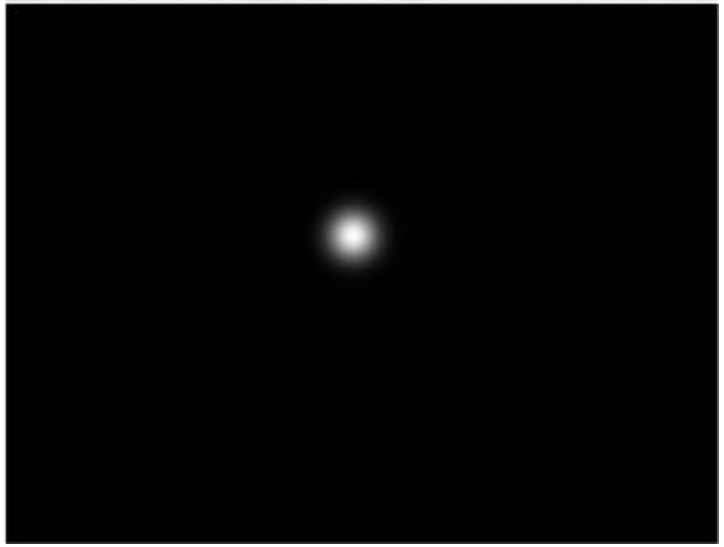
- Distance: $$aff(x,y) = \exp\left(-\frac{1}{2\sigma_d^2}\|f(x) - f(y)\|^2\right)$$

- Examples:
  - Distance: $f(x) = location(x)$
  - Intensity: $f(x) = intensity(x)$
  - Color: $f(x) = color(x)$
  - Texture: $f(x) = filterbank(x)$
-

Measuring Affinity

Distance:

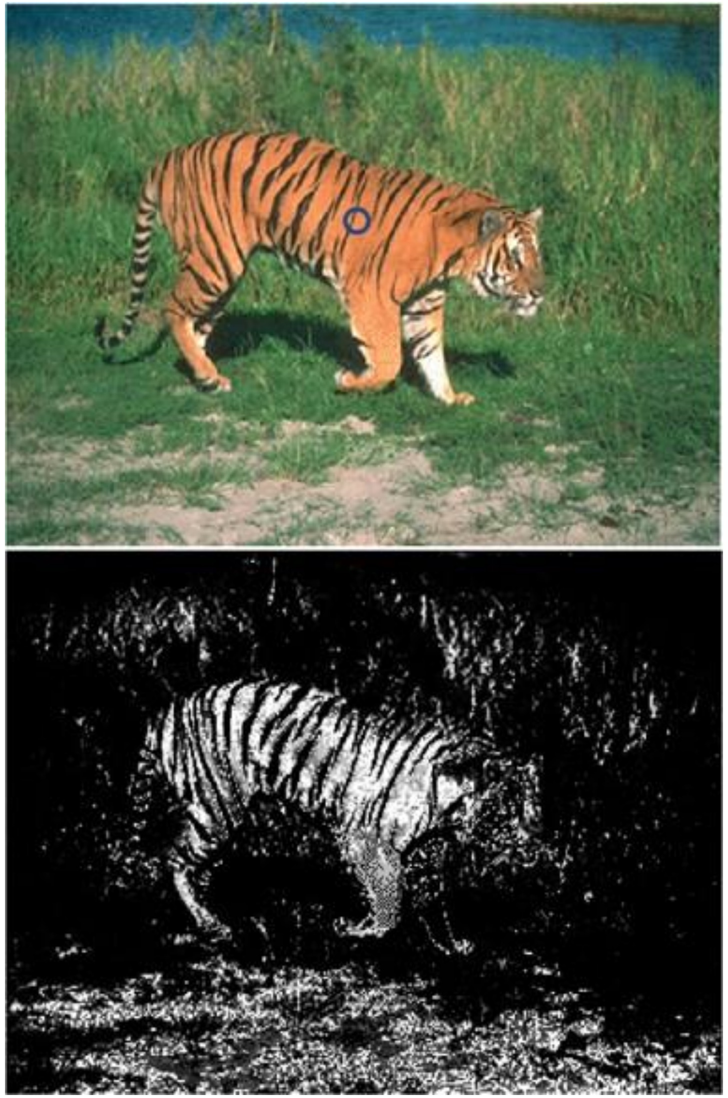
$$f(x) = location(x)$$



Measuring Affinity

Intensity:

$$f(x) = intensity(x)$$



Measuring Affinity

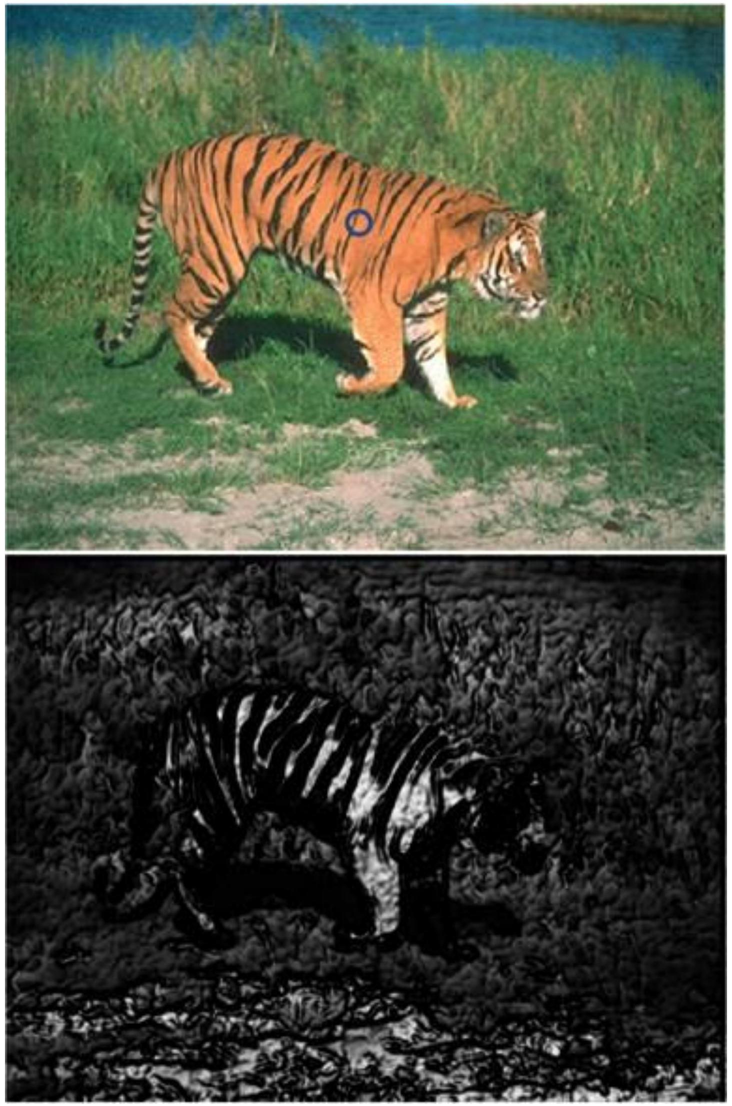
Color:

$$f(x) = color(x)$$

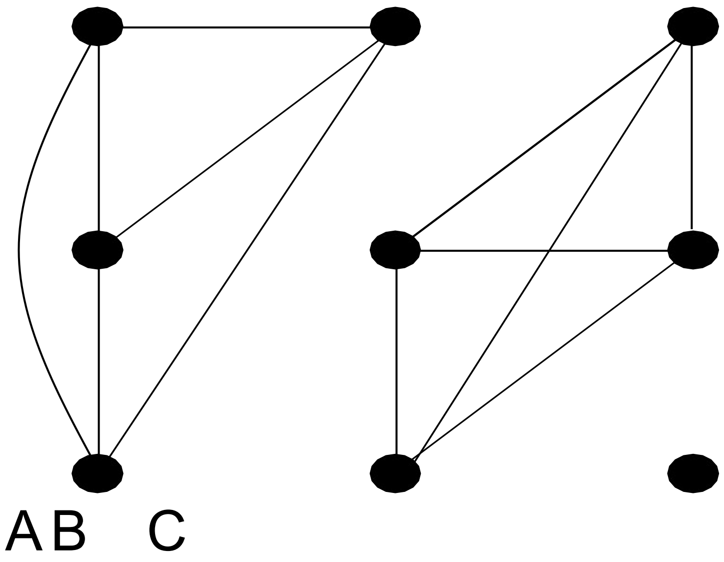
Measuring Affinity

Texture:

$$f(x) = filterbank(x)$$



Segmentation as Graph Cuts



Break Graph into Segments

- Delete links that cross between segments
- Easiest to break links that have low similarity (low weight)
  - Similar pixels should be in the same segments
  - Dissimilar pixels should be in different segments

slide credit: Steve Seitz

Graph Cut with Eigenvalues

- Given: Affinity matrix *W*
- Goal: Extract a single good cluster *v*
  - *v(i):* score for point *i* for cluster *v*

$$\max_{v} \; v^T W v \quad \text{s.t.} \; v^T v = 1$$

Optimizing

$$\max_v \; v^T W v \quad \text{s.t.} \; v^T v = 1 \quad \longleftrightarrow \quad \min_v \; -\tfrac{1}{2} v^T W v \quad \text{s.t} \; v^T v = 1$$

Lagrangian: $$-\frac{1}{2} v^T W v + \lambda (v^T v - 1)$$

$$-Wv + \lambda v = 0$$

$$Wv = \lambda v$$

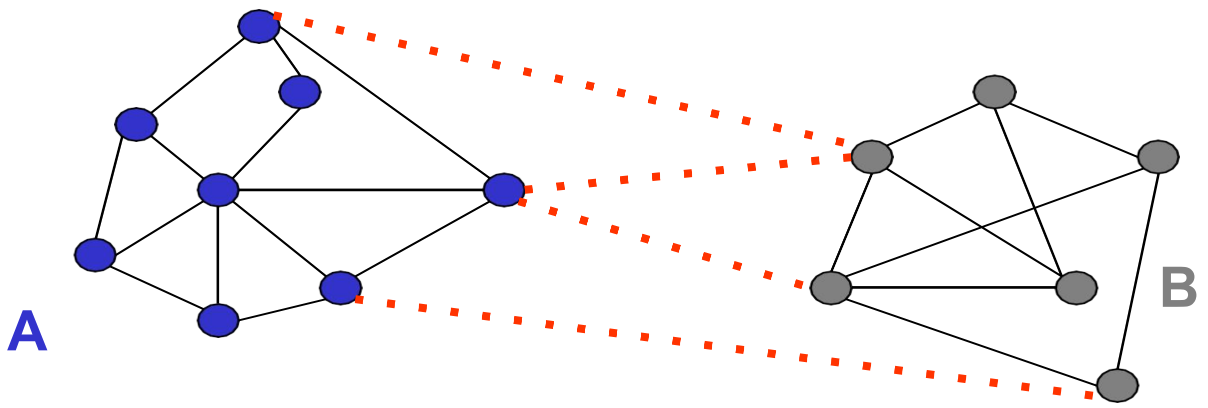
v is an eigenvector of *W*

Clustering via Eigenvalues

1. Construct affinity matrix W
2. Computeeigenvalues and vectors of W
3. Until done
   1. Take eigenvector of largest unprocessed eigenvalue
   2. Zero all components of elements that have already been clustered
   3. Threshold remaining components to determine cluster membership

Note: This is an example of a *spectral clustering* algorithm

Graph Cuts - Another Look



- Set of edges whose removal makes a graph disconnected
- Cost of a cut
  - Sum of weights of cut edges: $$cut(A,B) = \sum_{p \in A, q \in B} w_{pq}$$

- A graph cut gives us a segmentation
  - What is a "good" graph cut and how do we find one?

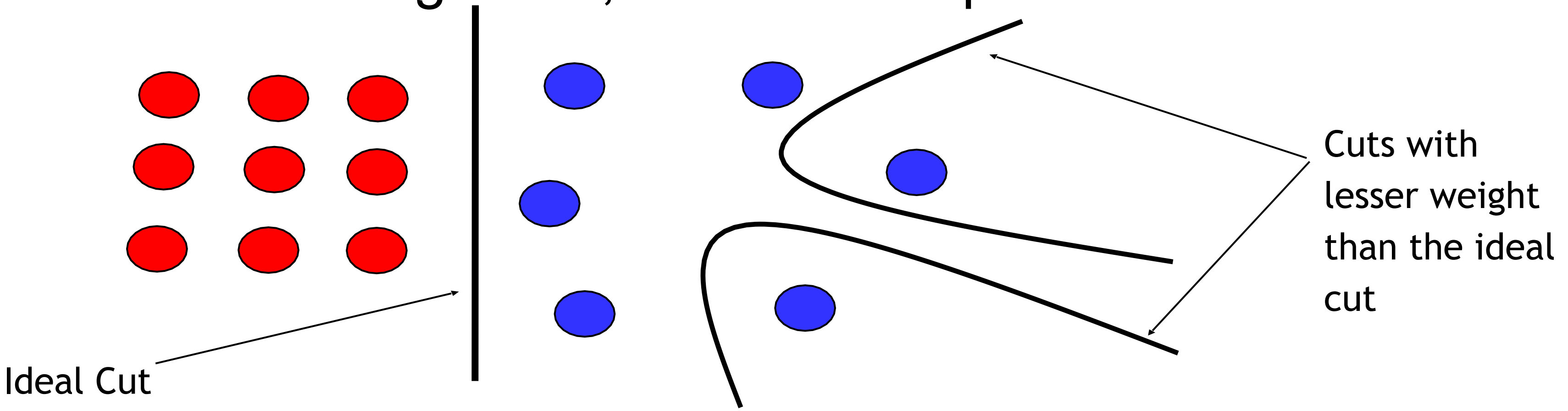
Formulation: Min Cut

We can do segmentation by finding the *minimum cut*

- either smallest number of elements (unweighted) or smallest sum of weights (weighted)
- efficient algorithms exist

Drawback

- Weight of cut proportional to number of edges
- Biased towards cutting small, isolated components



Solution: Normalized Cuts

1. Construct weighted graph $G = (V, E)$
2. Construct affinity matrix W
3. Solve for smallest few eigenvectors. $(D - W)y = \lambda D y$
4. Threshold eigenvectors to get a discrete cut
   - This is the approximation
   - As before, several heuristics for doing this
5. Recursively subdivide as desired.

Formulation: Normalized Cuts

- Key idea: normalize segment size
  - Fixes min cut's bias
- Formulation:

$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)}$$

$$= cut(A,B)\left[\frac{1}{\sum_{p \in A} w_{p,q}} + \frac{1}{\sum_{q \in B} w_{p,q}}\right]$$

$assoc(A,V)$ = sum of weights of edges in *V* that touch *A*

- NP-hard, but can approximate

J. Shi and J. Malik. [Normalized cuts and image segmentation.](#) PAMI 2000

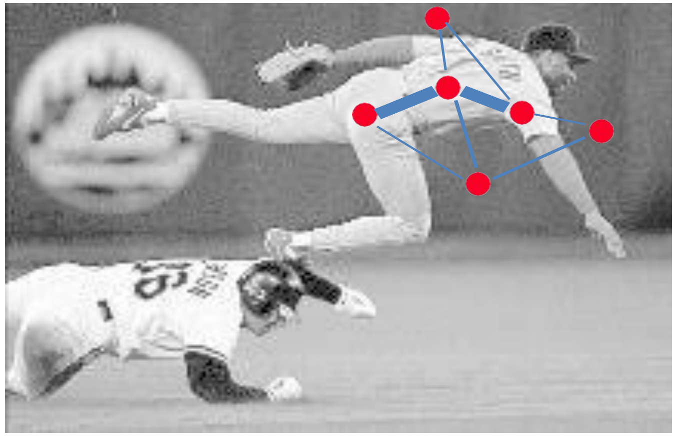
NCuts as Generalized Eigenvector Problem

Definitions:

W : **affinity** matrix

D : **diagonal** matrix $D(i,i) = \sum_j w_{i,j}$

z : **vector** in $\{-1, 1\}^N, z_i = 1 \Leftrightarrow i \in A$



In matrix form:

$$NCut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)}$$

$$= \frac{(1+z)^T(D-W)(1+z)}{k1^TD1} + \frac{(1-z)(D-W)(1-z)}{(1-k)1^TD1}; \quad k = \frac{\sum_{z_i>0} D(i,i)}{\sum_i D(i,i)}$$

$$= \ldots$$

After a lot of math…

- After simplification, we get

$$NCut(A,B) = \frac{y^T(D-W)y}{y^TDy}, \qquad y_i \in \{1, -b\},\ y^TD1 = 0$$

This is hard, y is discrete!

Relaxation: continuous y

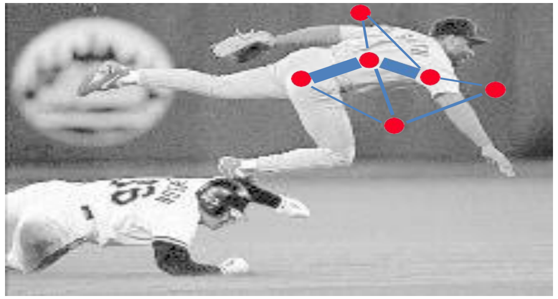
- This is a Rayleigh Quotient
  – Solution given by the "generalized" eigenvalue problem

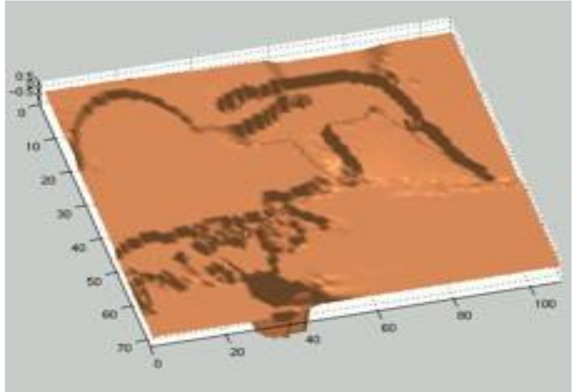
$$(D-W)y = \lambda Dy$$

- Subtleties
  _ Optimal solution is second smallest eigenvector
  _ Gives continuous result—must convert into discrete values of y

Slide credit: Alyosha Efros

Normalized Cuts example



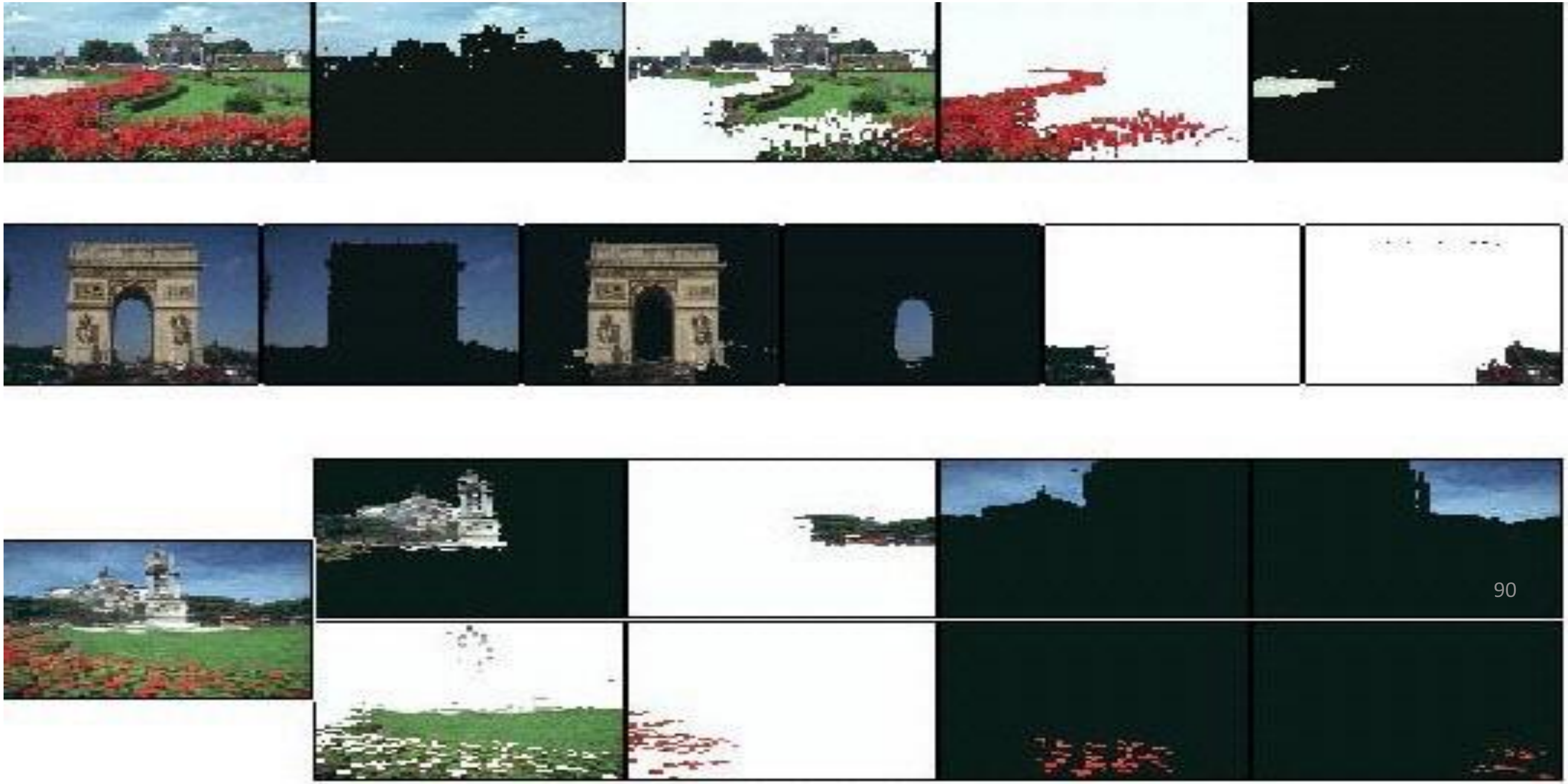


Smallest eigenvectors

NCuts segments

Image source: Shi & Malik

Normalized Cuts example



Normalized Cuts example

Normalized Cuts summary

- Pro
  - Flexible to choice of affinity matrix
  - Generally works better than other methods we've seen so far
- Con
  - Can be expensive, especially with many cuts.
  - Bias toward balanced partitions
  - Constrained by affinity matrix model

Today’s agenda

- K-means clustering
- Mean-shift clustering
- Normalized cuts

Next time

Cameras and Calibration

Other Kernels

A kernel is a function that satisfies the following requirements :

1. $\int_{R^d} \phi(x) = 1$

2. $\phi(x) \geq 0$

Some examples of kernels include :

1. Rectangular $\phi(x) = \begin{cases} 1 & a \leq x \leq b \\ 0 & else \end{cases}$

2. Gaussian $\phi(x) = e^{-\frac{x^2}{2\sigma^2}}$

3. Epanechnikov $\phi(x) = \begin{cases} \frac{3}{4}(1 - x^2) & if\ |x| \leq 1 \\ 0 & else \end{cases}$

source

Technical Details

Taking the derivative of: $\hat{f}_K = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right)$

$$\nabla \hat{f}(\mathbf{x}) = \underbrace{\frac{2c_{k,d}}{nh^{d+2}} \left[\sum_{i=1}^{n} g\left(\left\| \frac{\mathbf{x} - \mathbf{x}_i}{h} \right\|^2 \right) \right]}_{\text{term 1}} \underbrace{\left[\frac{\sum_{i=1}^{n} \mathbf{x}_i g\left(\left\| \frac{\mathbf{x} - \mathbf{x}_i}{h} \right\|^2 \right)}{\sum_{i=1}^{n} g\left(\left\| \frac{\mathbf{x} - \mathbf{x}_i}{h} \right\|^2 \right)} - \mathbf{x} \right]}_{\text{term 2}}, \qquad (3)$$

where $g(x) = -k'(x)$ denotes the derivative of the selected kernel profile.

- Term1: this is proportional to the density estimate at x (similar to equation 1 from two slides ago).
- Term2: this is the mean-shift vector that points towards the direction of maximum density.

Comaniciu & Meer, 2002

Technical Details

Finally, the mean shift procedure from a given point x_t is:

1. Compute the mean shift vector **m**:

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

2. Translate the density window:

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \mathbf{m}(\mathbf{x}_i^t).$$

3. Iterate steps 1 and 2 until convergence.

$$\nabla f(\mathbf{x}_i) = 0.$$

Comaniciu & Meer, 2002

Technical Details

Given n data points $\mathbf{x}_i \in \mathbb{R}^d$, the multivariate kernel density estimate using a radially symmetric kernel[1] (e.g., Epanechnikov and Gaussian kernels), $K(\mathbf{x})$, is given by,

$$\hat{f}_K = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right), \tag{1}$$

where h (termed the *bandwidth* parameter) defines the radius of kernel. The radially symmetric kernel is defined as,

$$K(\mathbf{x}) = c_k k(\|\mathbf{x}\|^2), \tag{2}$$

where c_k represents a normalization constant.

Comaniciu & Meer, 2002