UNIVERSITÄT BONN

Juergen Gall

Clustering and Segmentation
MA-INF 2201 - Computer Vision
WS24/25

The goals of segmentation

- Separate image into coherent “objects”

image | human segmentation

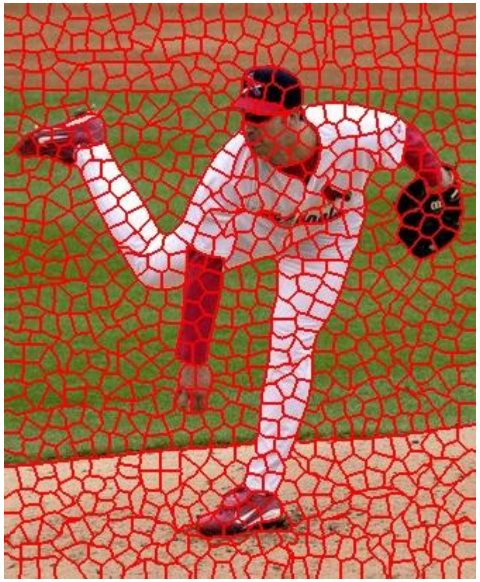
Source: S. Lazebnik

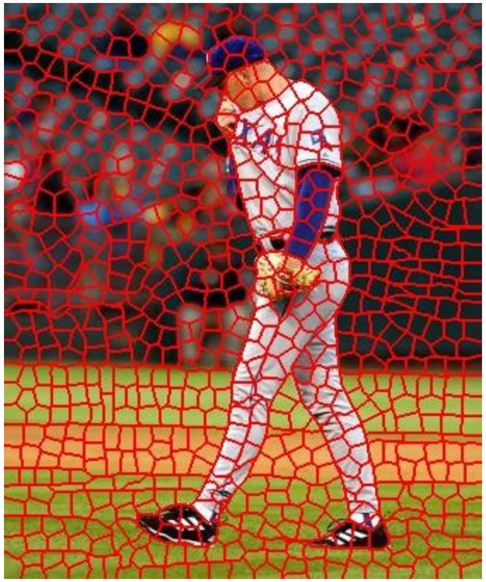
The goals of segmentation

- Separate image into coherent “objects”

- Group together similar-looking pixels for efficiency of further processing

“superpixels”

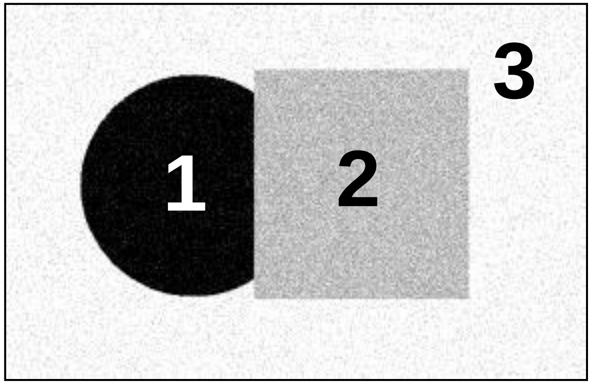




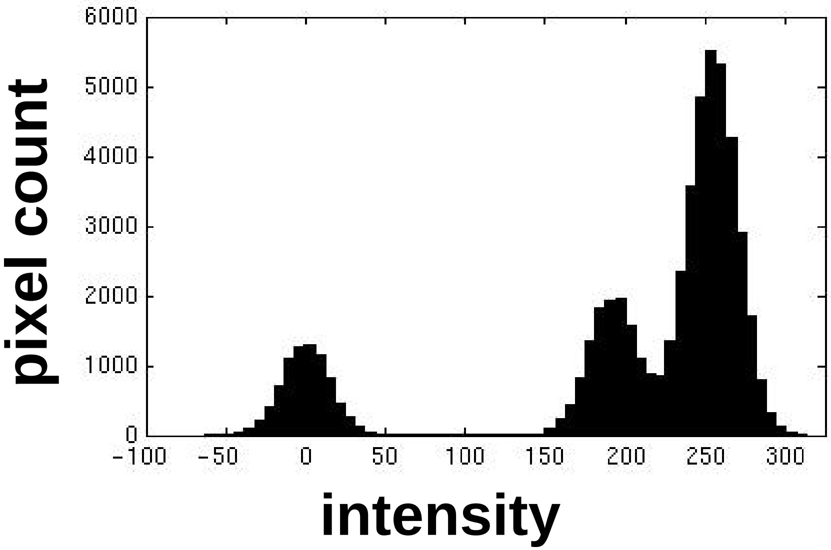
X. Ren and J. Malik. [Learning a classification model for segmentation.]() ICCV 2003.

Source: S. Lazebnik

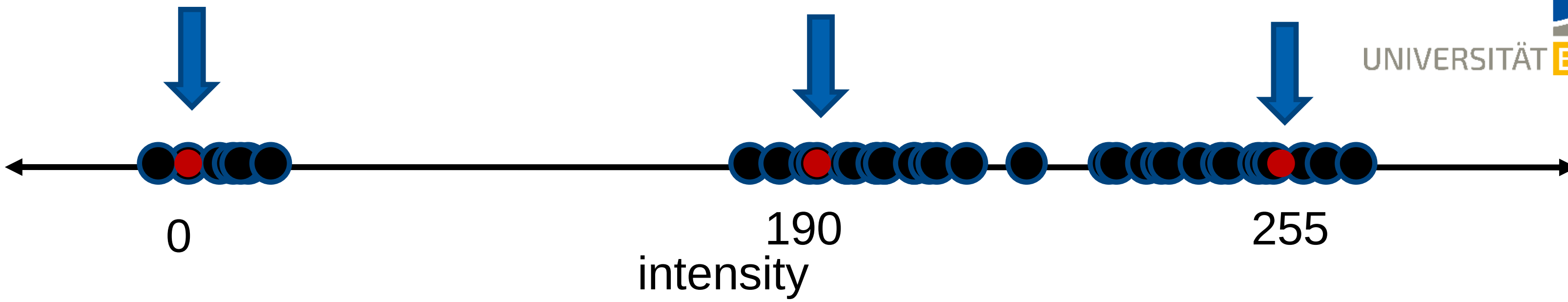
Toy example

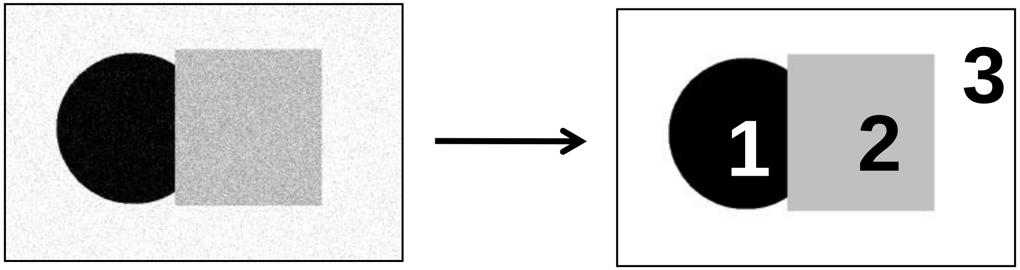


input image



How can we divide the image into three superpixels (black, gray, white)?





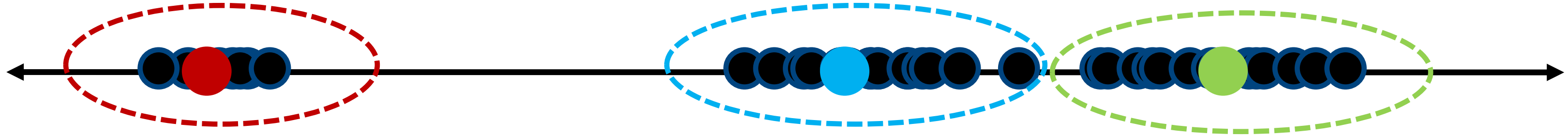
- Goal: choose three “centers” as the representative intensities, and label every pixel according to which of these centers it is nearest to.
- Best cluster centers are those that minimize SSD between all points and their nearest cluster center c_i:

$$\sum_{\text{clusters } i} \; \sum_{\text{points p in cluster } i} ||p - c_i||^2$$

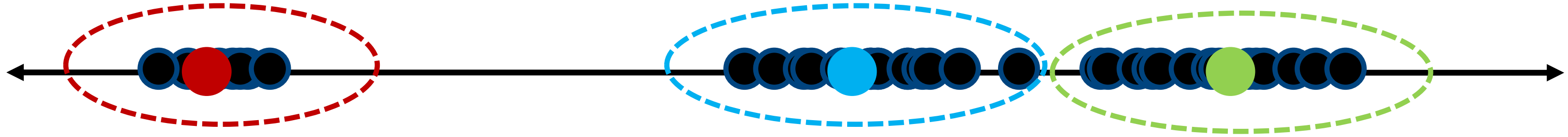
Source: K. Grauman

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.



Source: K. Grauman

K-means clustering

- Basic idea: randomly initialize the *k* cluster centers, and iterate between the two steps we just saw.
  1. Randomly initialize the cluster centers, $c_1, ..., c_K$
  2. Given cluster centers, determine points in each cluster
     - For each point p, find the closest c_i. Put p into cluster i
  3. Given points in each cluster, solve for c_i
     - Set c_i to be the mean of points in cluster i
  4. If c_i have changed, repeat Step 2

Properties

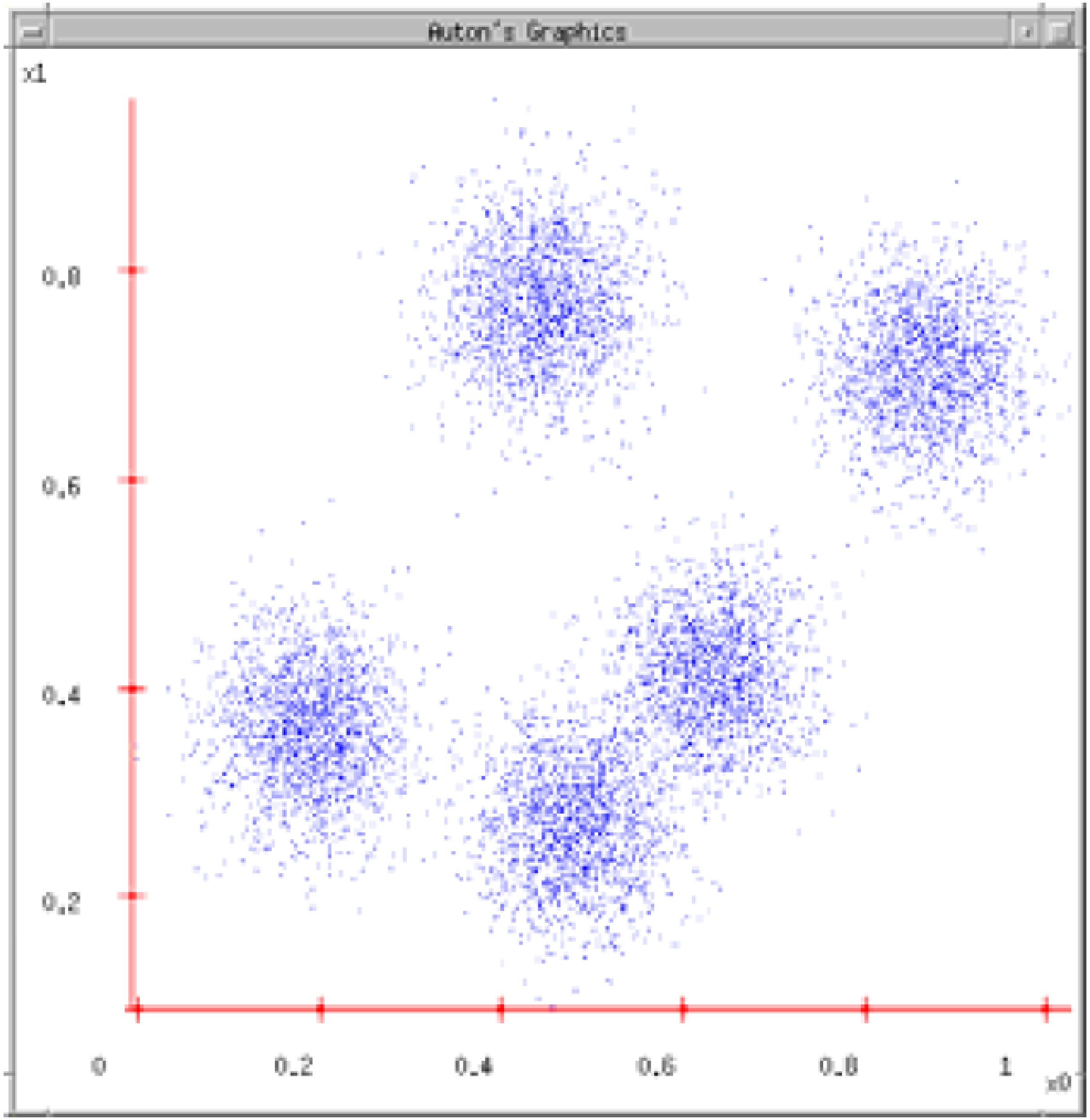
- Will always converge to *some* solution
- Finds only a local minimum of objective function:

$$\sum_{\text{clusters } i} \quad \sum_{\text{points p in cluster } i} ||p - c_i||^2$$

Source: S. Seitz

K-means

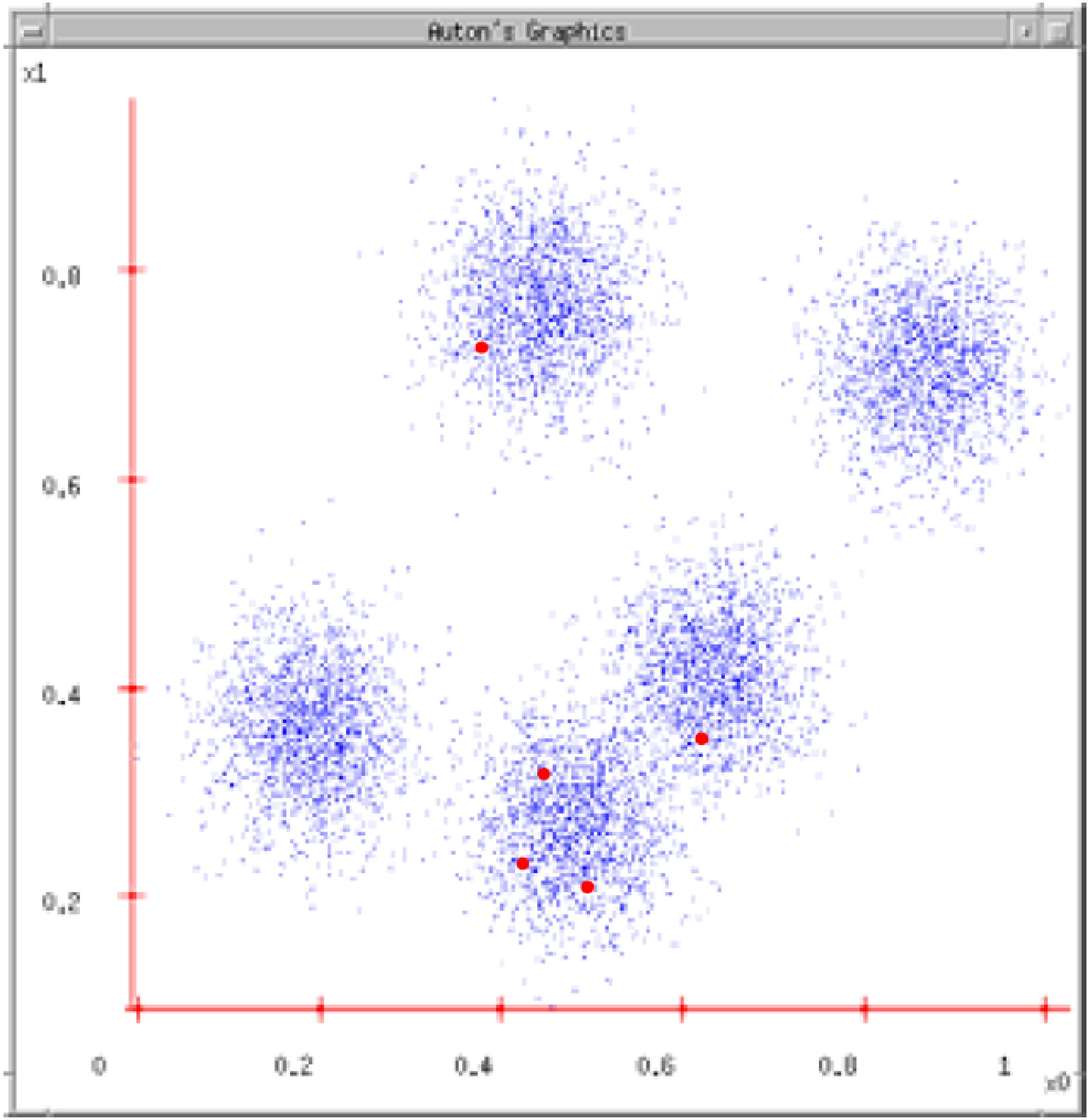
1. Ask user how many clusters they'd like. *(e.g. k=5)*



Source: A. Moore

K-means

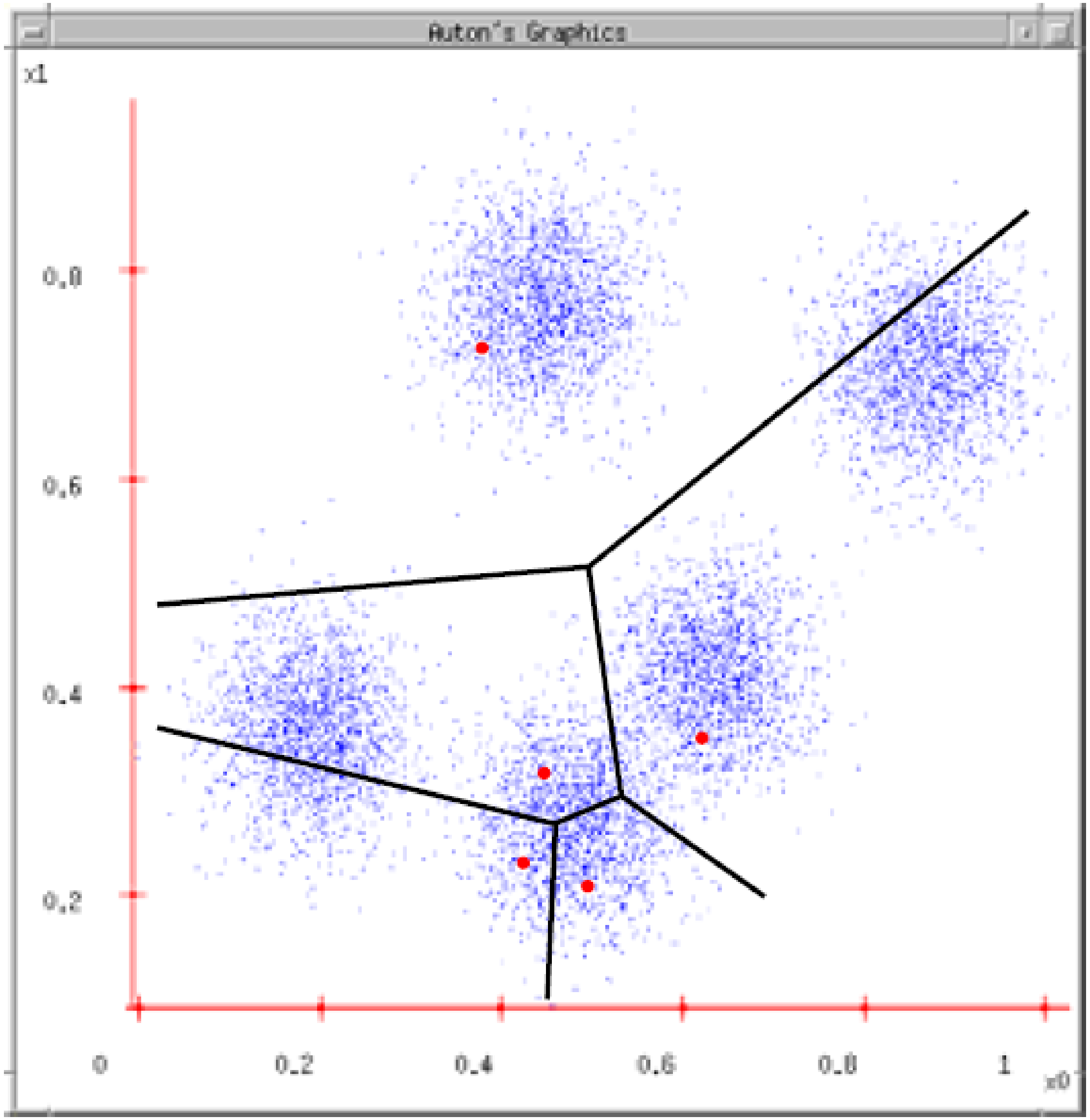
1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations



Source: A. Moore

K-means

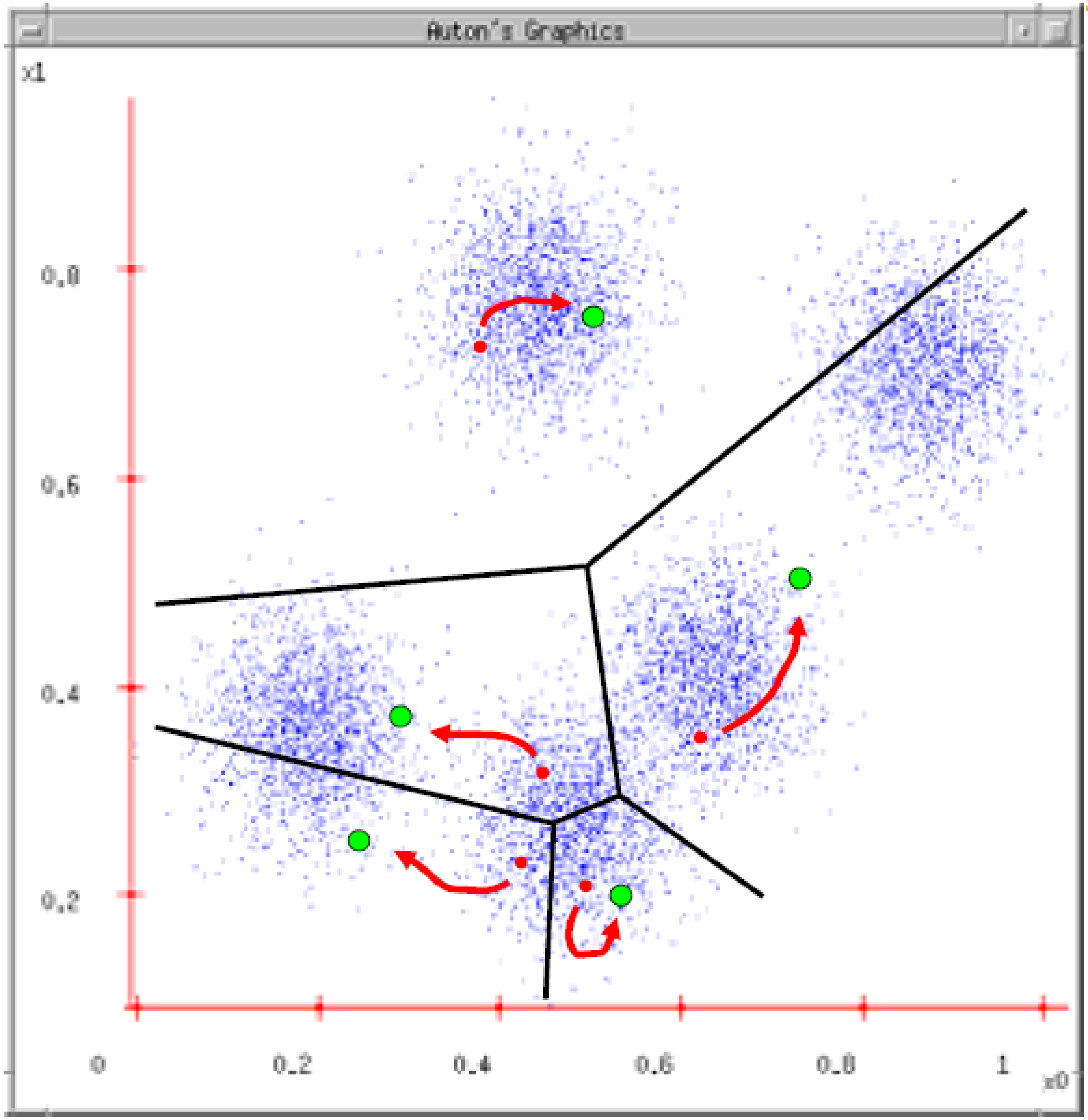
1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to. (Thus each Center "owns" a set of datapoints)



Source: A. Moore

K-means

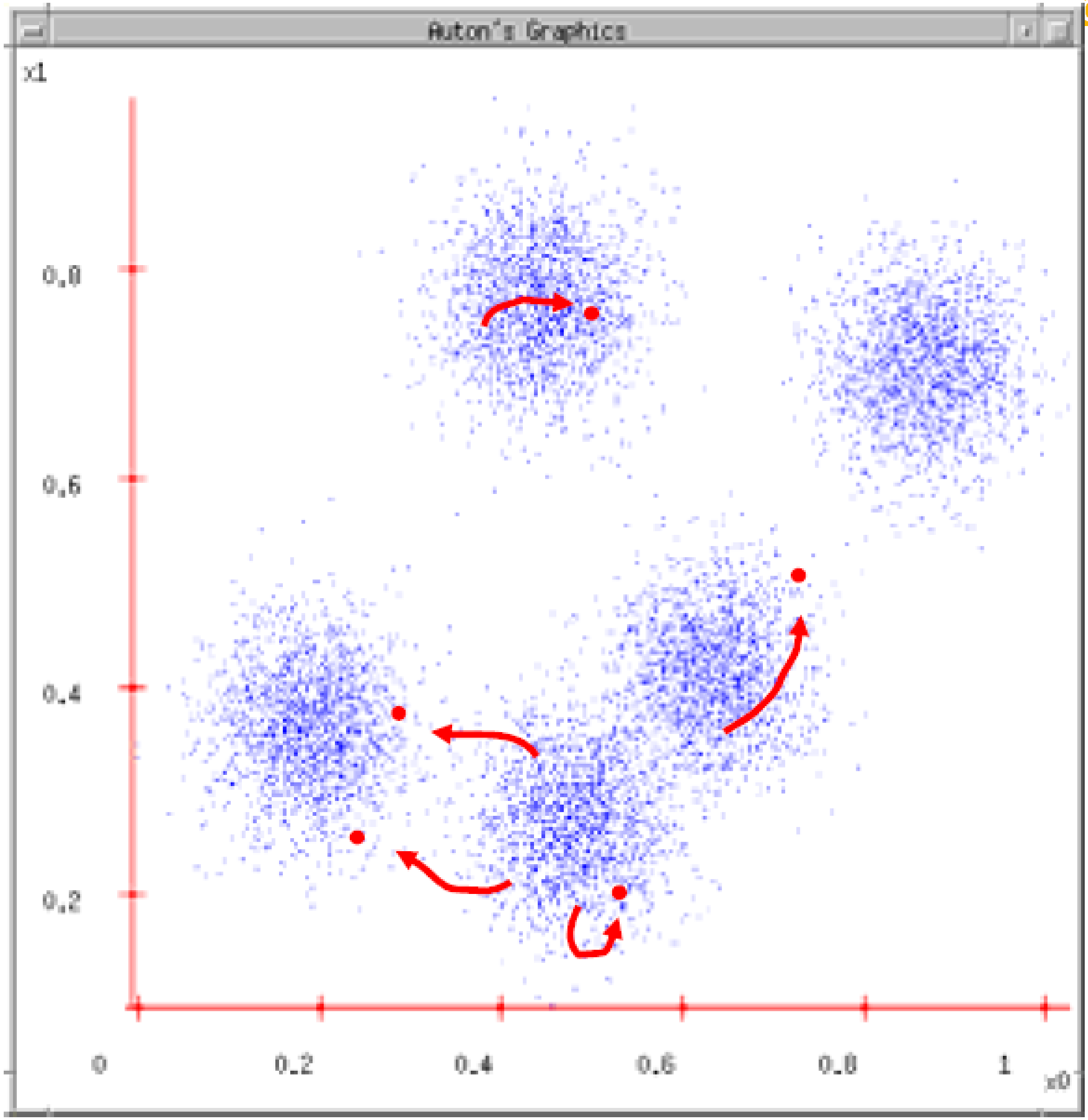
1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds the centroid of the points it owns



Source: A. Moore

K-means

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds the centroid of the points it owns...
5. ...and jumps there
6. ...Repeat until terminated!



Source: A. Moore

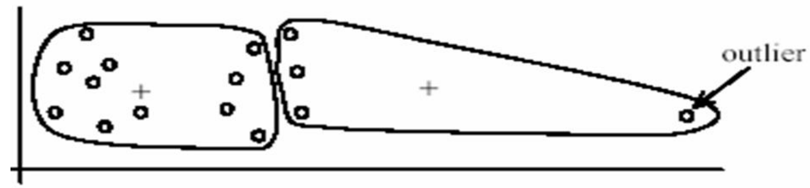
K-means: pros and cons

Pros

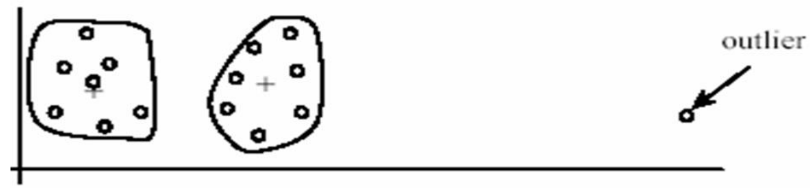
- Simple, fast to compute
- Converges to local minimum of within-cluster squared error

Cons/issues

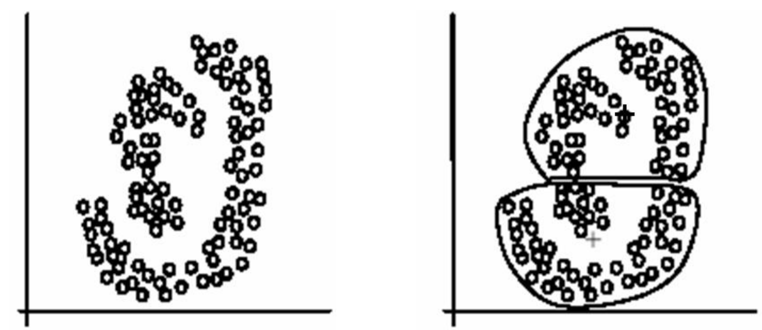
- Setting k?
- Sensitive to initial centers
- Sensitive to outliers
- Detects spherical clusters
- Assuming means can be computed



(A): Undesirable clusters



(B): Ideal clusters



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

Source: K. Grauman

Segmentation as clustering

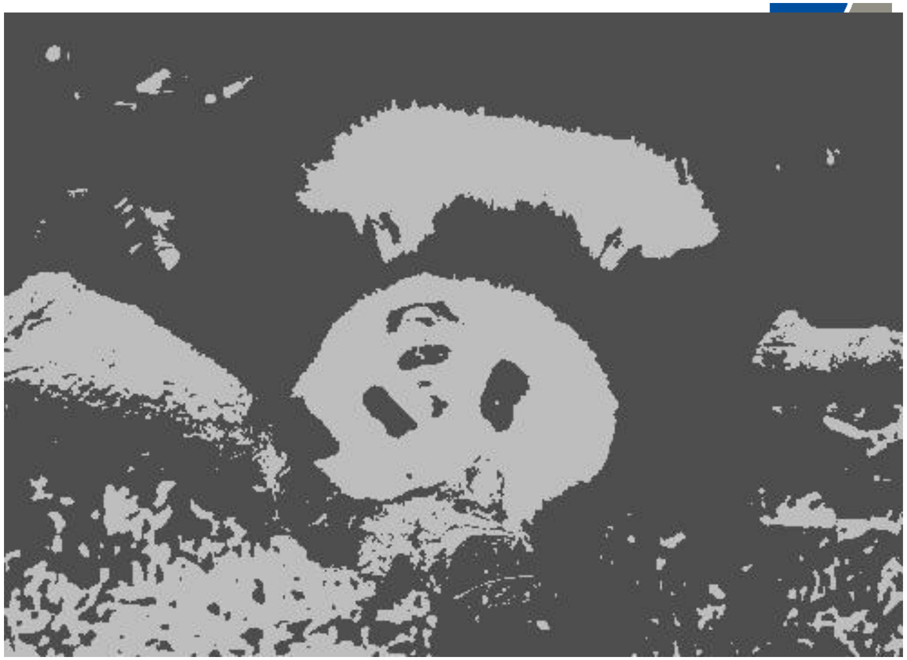
Depending on what we choose as the *feature space*, we can group pixels in different ways.

Grouping pixels based on **intensity** similarity

Feature space: intensity value (1-d)

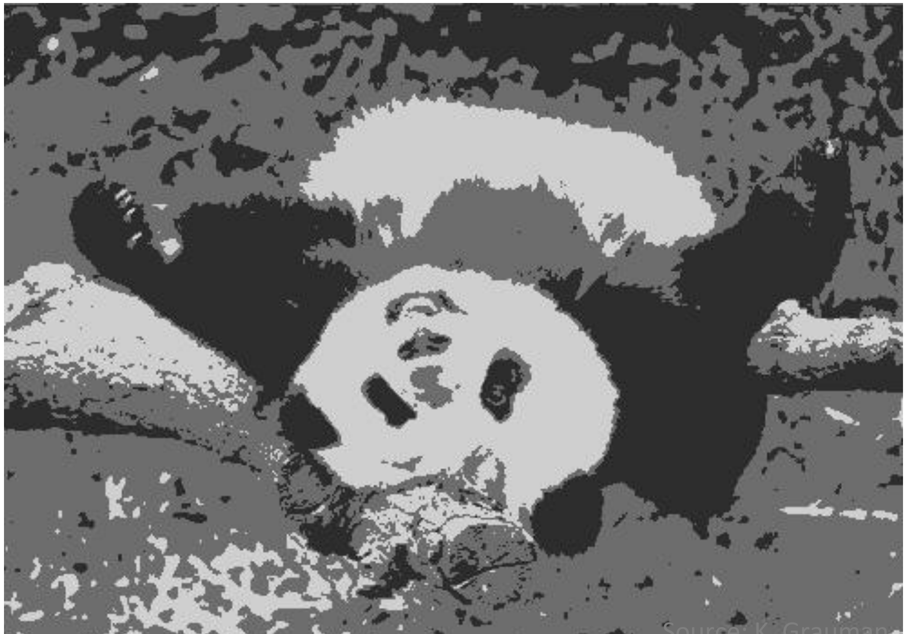
Source: K. Grauman

K=2



K=3

quantization of the feature space;
segmentation label map

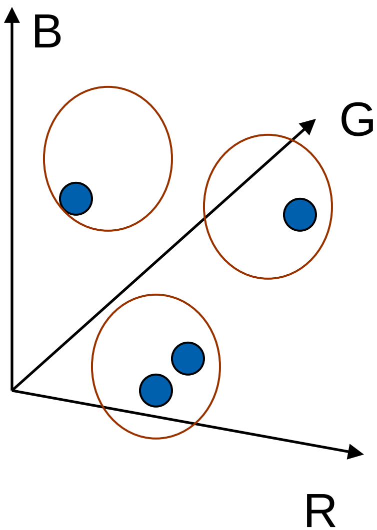


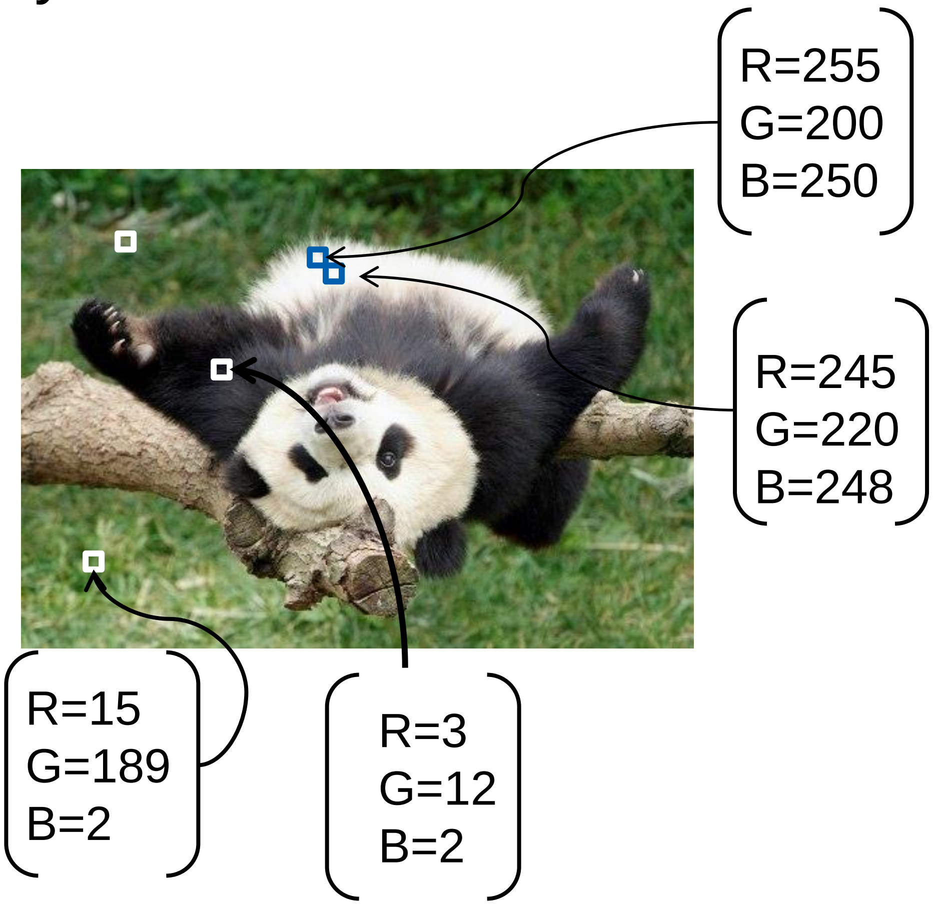
Source: K. Grauman

Segmentation as clustering

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

Grouping pixels based on **color** similarity





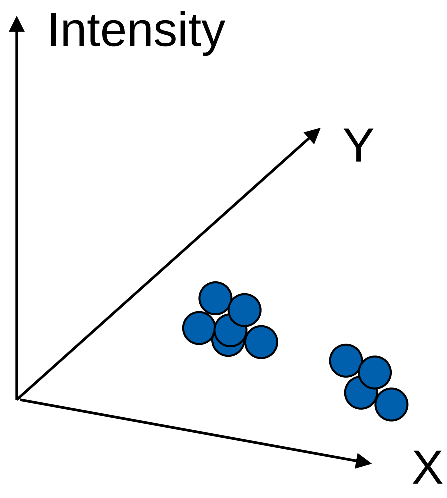
Feature space: color value (3-d)

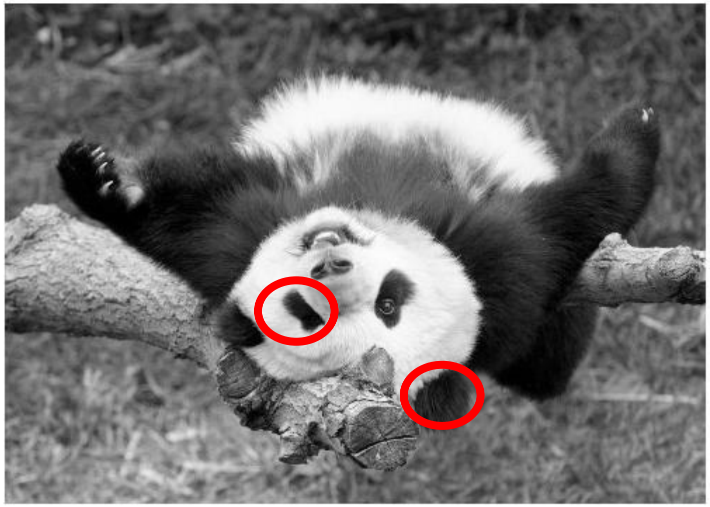
Source: K. Grauman

Segmentation as clustering

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

Grouping pixels based on **intensity+position** similarity





Both regions are black, but if we also include **position (x,y),** then we could group the two into distinct segments; way to encode both similarity & proximity.

Source: K. Grauman

Segmentation as clustering

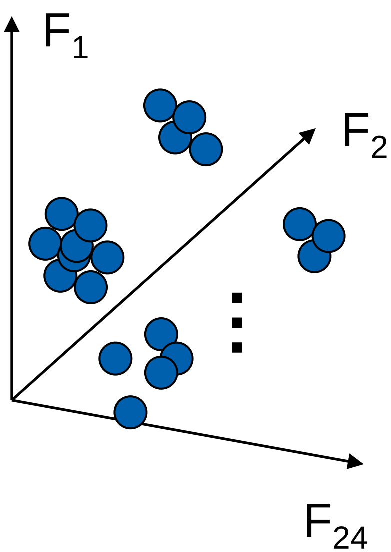
- Color, brightness, position alone are not enough to distinguish all regions…

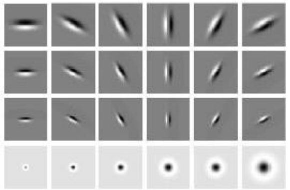
Source: K. Grauman

Segmentation as clustering

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

Grouping pixels based on **texture** similarity





Filter bank of 24 filters

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

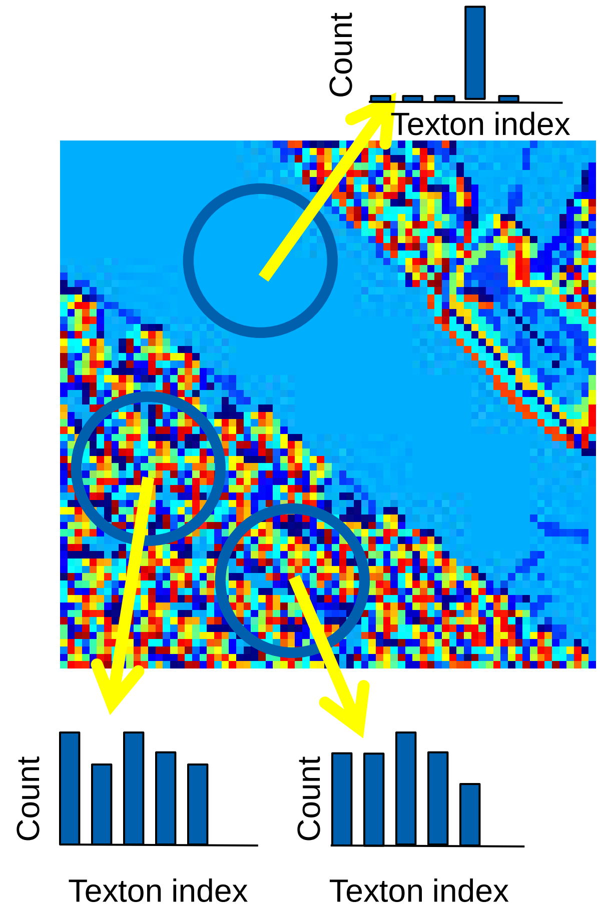
Source: K. Grauman

Segmentation with texture features

- Find “textons” by **clustering** vectors of filter bank outputs
- Describe texture in a window based on *texton histogram*

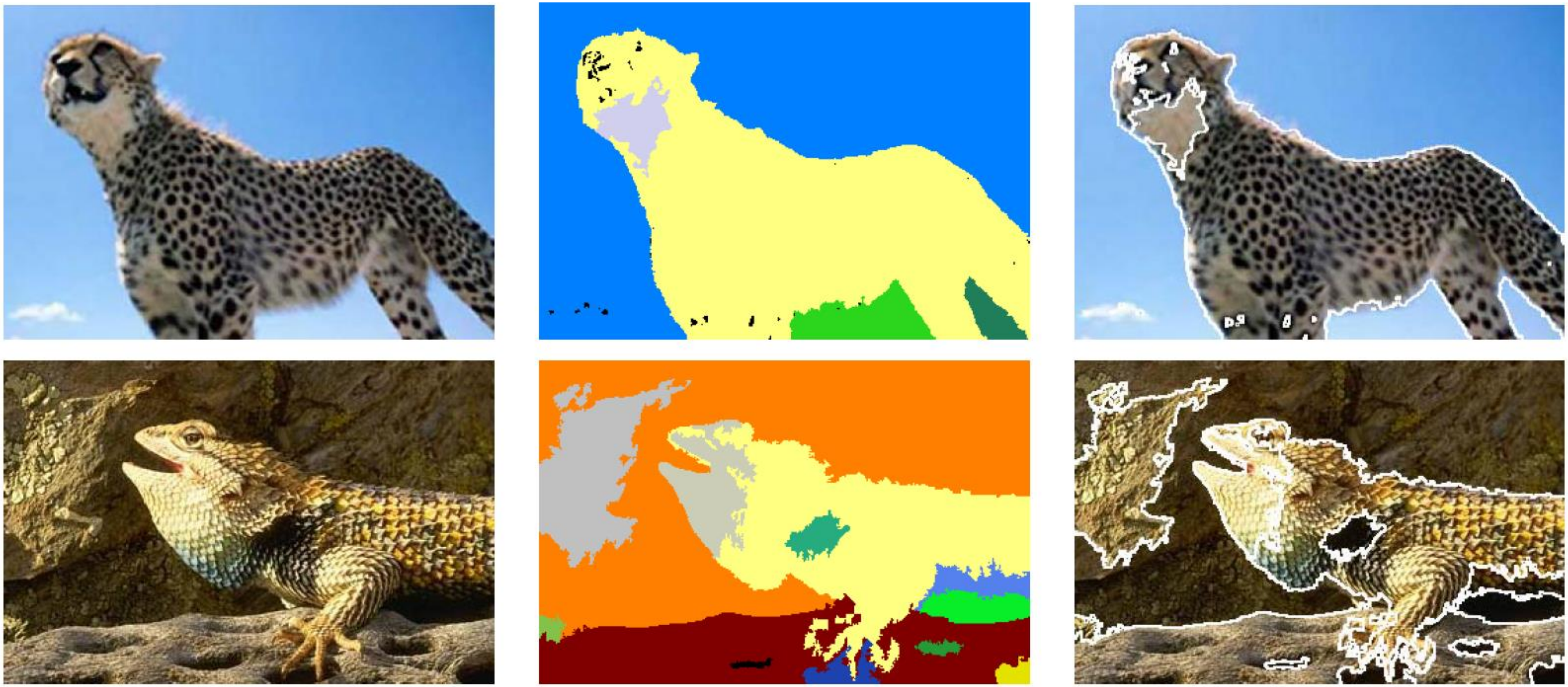
Image

Texton map



Source: S. Lazebnik

Image segmentation example



Material classification example

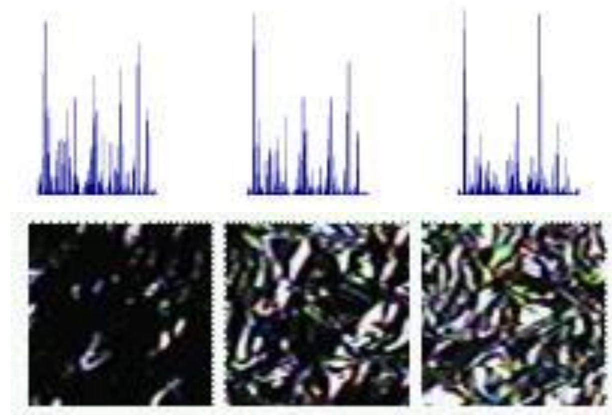
For an image of a single texture, we can classify it according to its global (image-wide) texton histogram.

Source: M. Varma

Material classification example

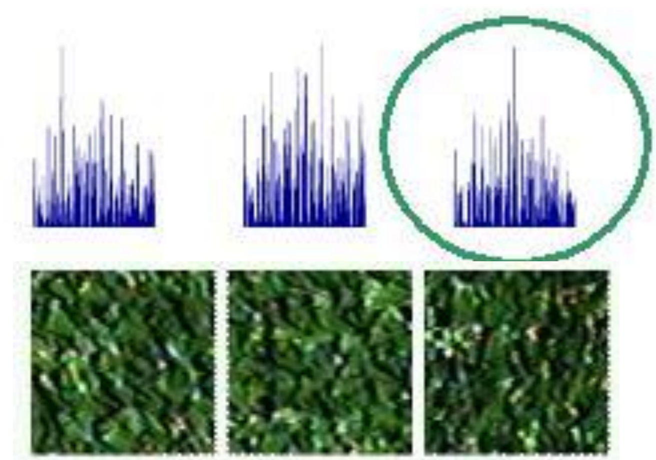
Nearest neighbor classification: label the input according to the nearest known example's label.

Foil



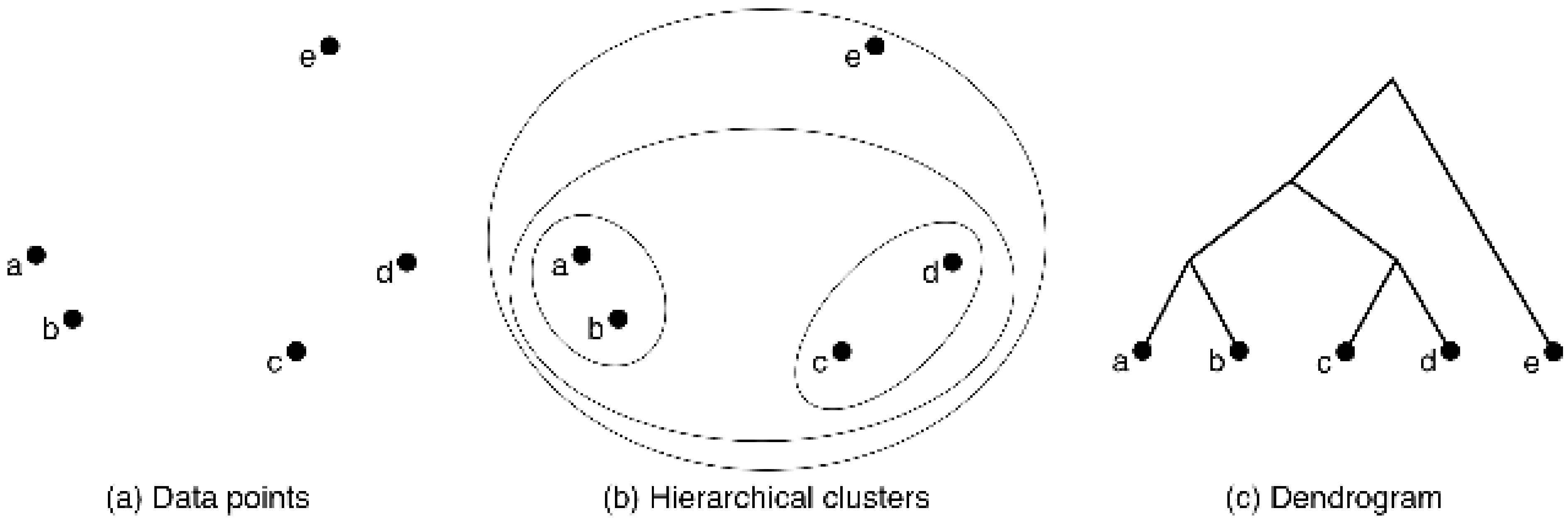
$$\chi^2(h_i, h_j) = \frac{1}{2}\sum_{k=1}^{K} \frac{[h_i(k) - h_j(k)]^2}{h_i(k) + h_j(k)}$$

Grass



Source: M. Varma

Agglomerative Clustering



Source: R. Akella

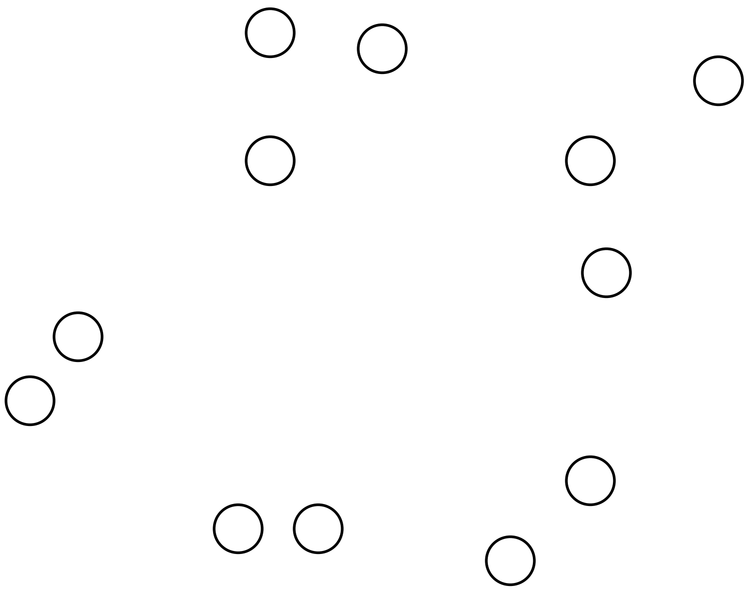
Agglomerative Clustering

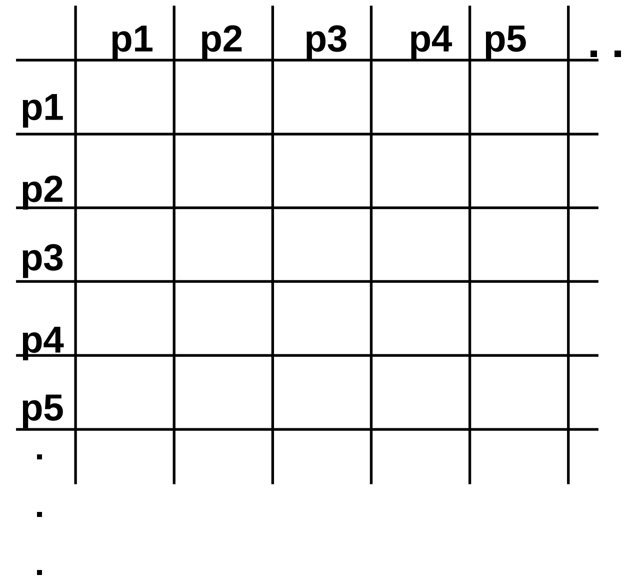
- Start with the points as individual clusters
- At each step, merge the closest pair of clusters until only one cluster (or k clusters) left
- Key operation is the computation of the proximity of two clusters
    - Different approaches to defining the distance between clusters distinguish the different algorithms

Source: R. Akella

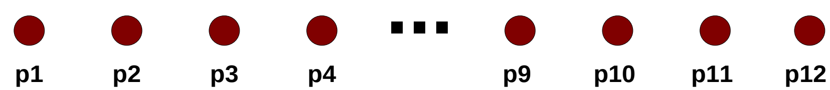
Starting Situation

Start with clusters of individual points and a proximity matrix





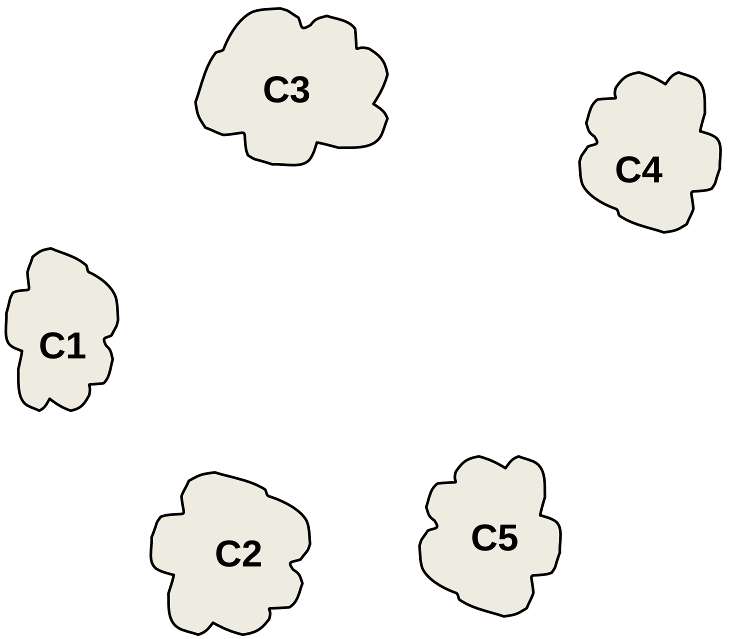
Proximity Matrix



Source: R. Akella

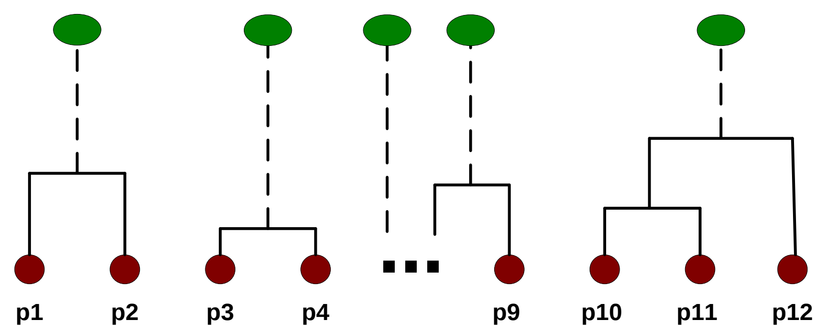
Intermediate Situation

After some merging steps, we have some clusters



	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

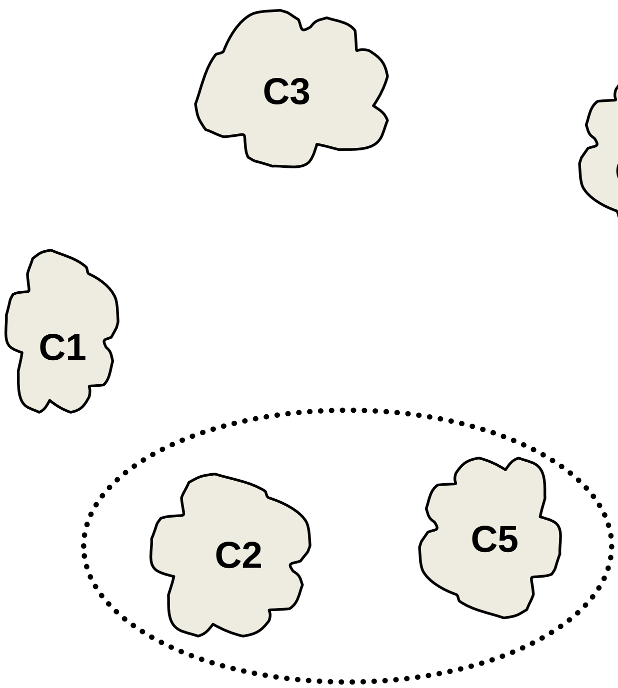
Proximity Matrix

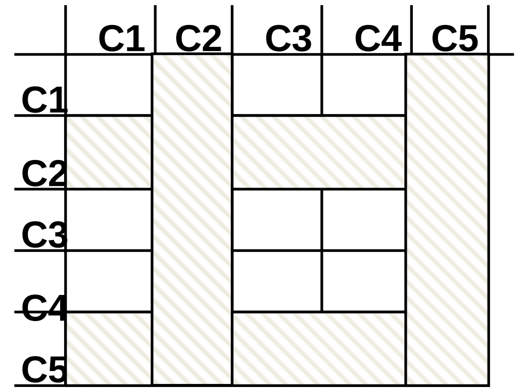


Source: R. Akella

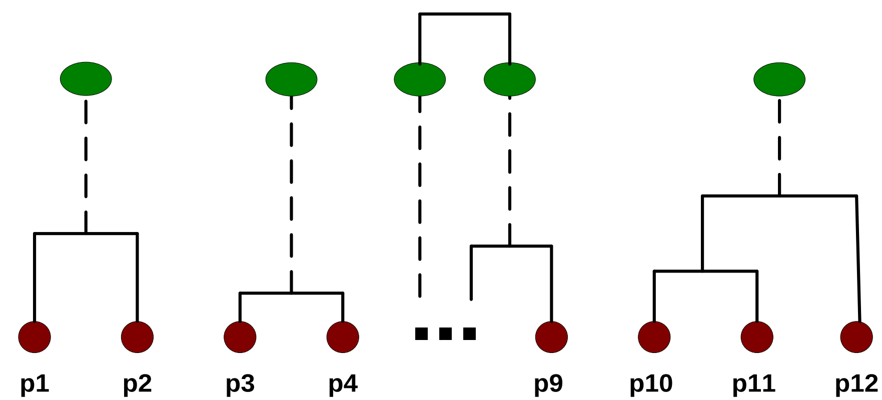
Intermediate Situation

We want to merge the two closest clusters (C2 and C5) and update the proximity matrix.





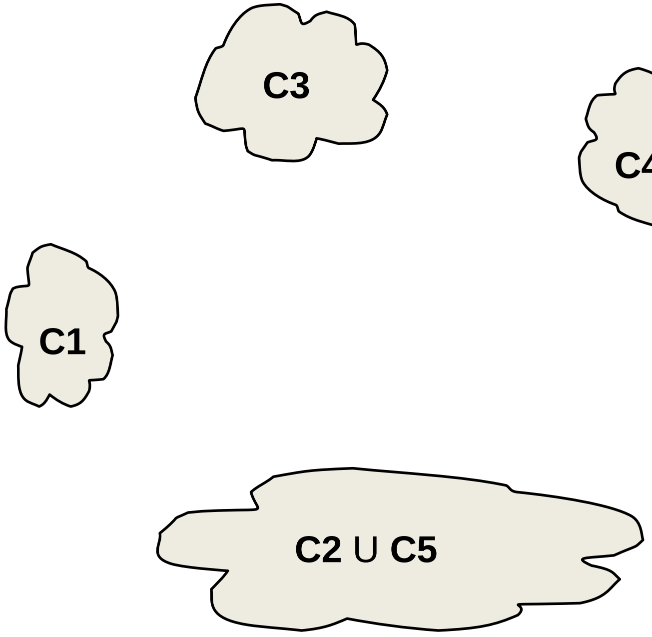
Proximity Matrix



Source: R. Akella

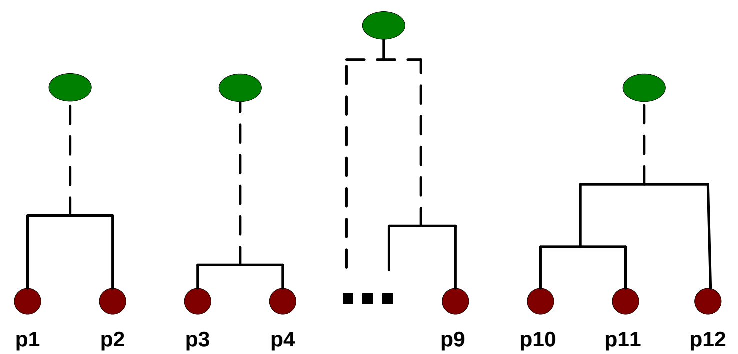
After Merging

The question is: “How do we update the proximity matrix?”



	C1	C2 U C5	C3	C4
C1		?		
C2 U C5	?	?	?	?
C3		?		
C4		?		

Proximity Matrix



Source: R. Akella

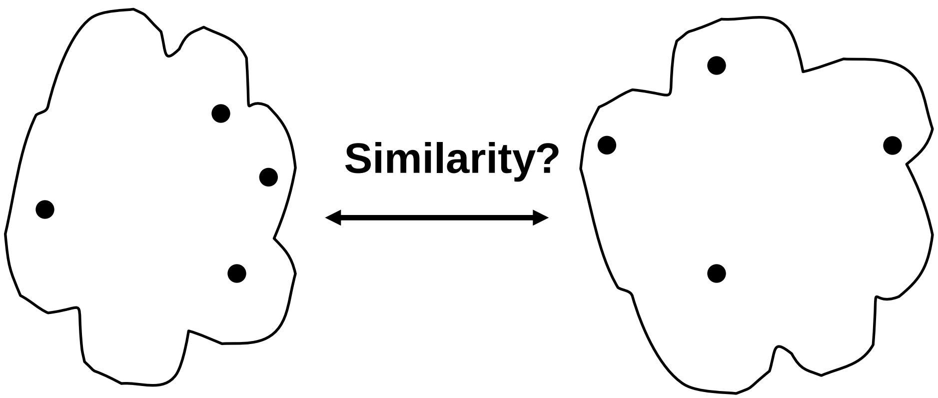
Agglomerative Clustering Algorithm

Basic algorithm is:

1. Compute the proximity matrix
2. Let each data point be a cluster
3. **Repeat**
4. Merge the two closest clusters
5. Update the proximity matrix
6. **Until** only a single cluster remains

Source: R. Akella

How to Define Inter-Cluster Similarity



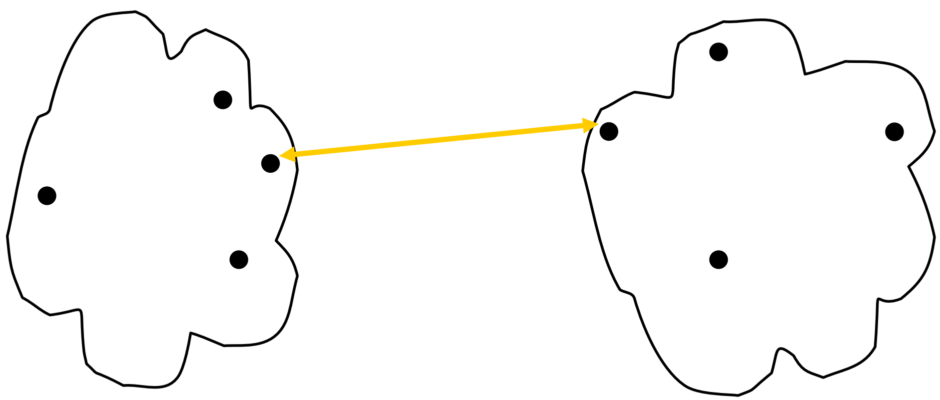
	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Ward’s Method

Source: R. Akella

How to Define Inter-Cluster Similarity



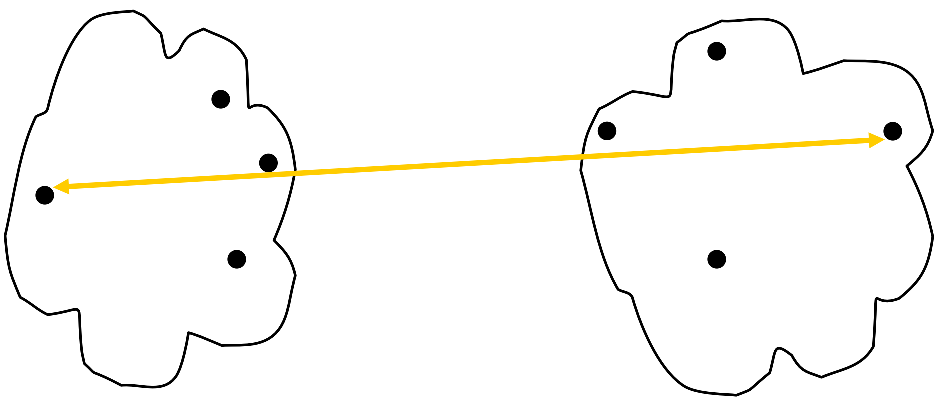
	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

- MIN $d_S(R, Q) = \min_{i \in R, j \in Q} d(i, j)$
- MAX
- Group Average
- Distance Between Centroids
- Ward’s Method

Source: R. Akella

How to Define Inter-Cluster Similarity



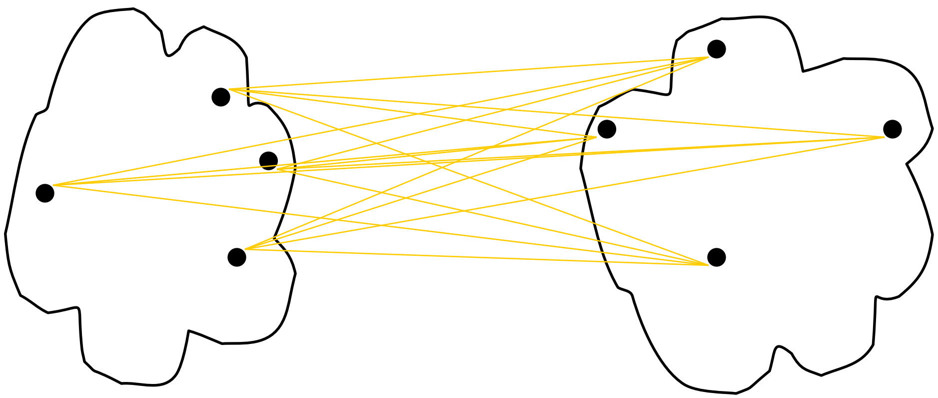
	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
. . .						

Proximity Matrix

- MIN
- MAX $d_C(R,Q) = \max_{i \in R, j \in Q} d(i,j)$
- Group Average
- Distance Between Centroids
- Ward's Method

Source: R. Akella

How to Define Inter-Cluster Similarity



	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

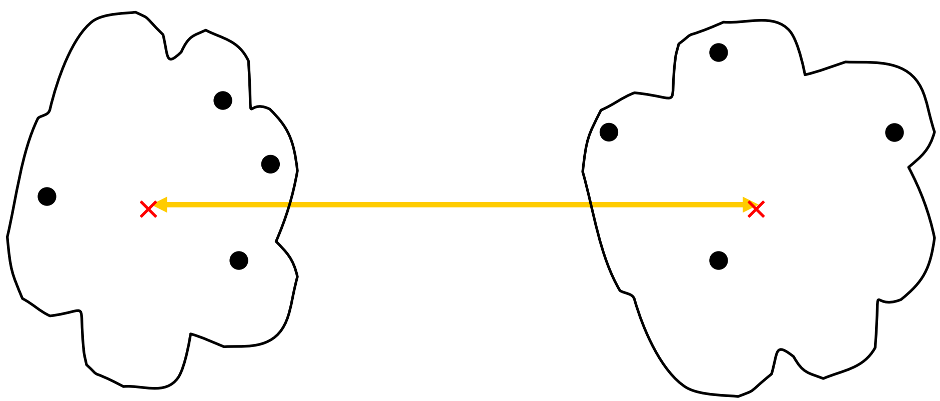
Proximity Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Ward's Method

$$d_A(R, Q) = \frac{1}{|R||Q|} \sum_{i \in R, j \in Q} d(i, j)$$

Source: R. Akella

How to Define Inter-Cluster Similarity



	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Ward's Method

Source: R. Akella

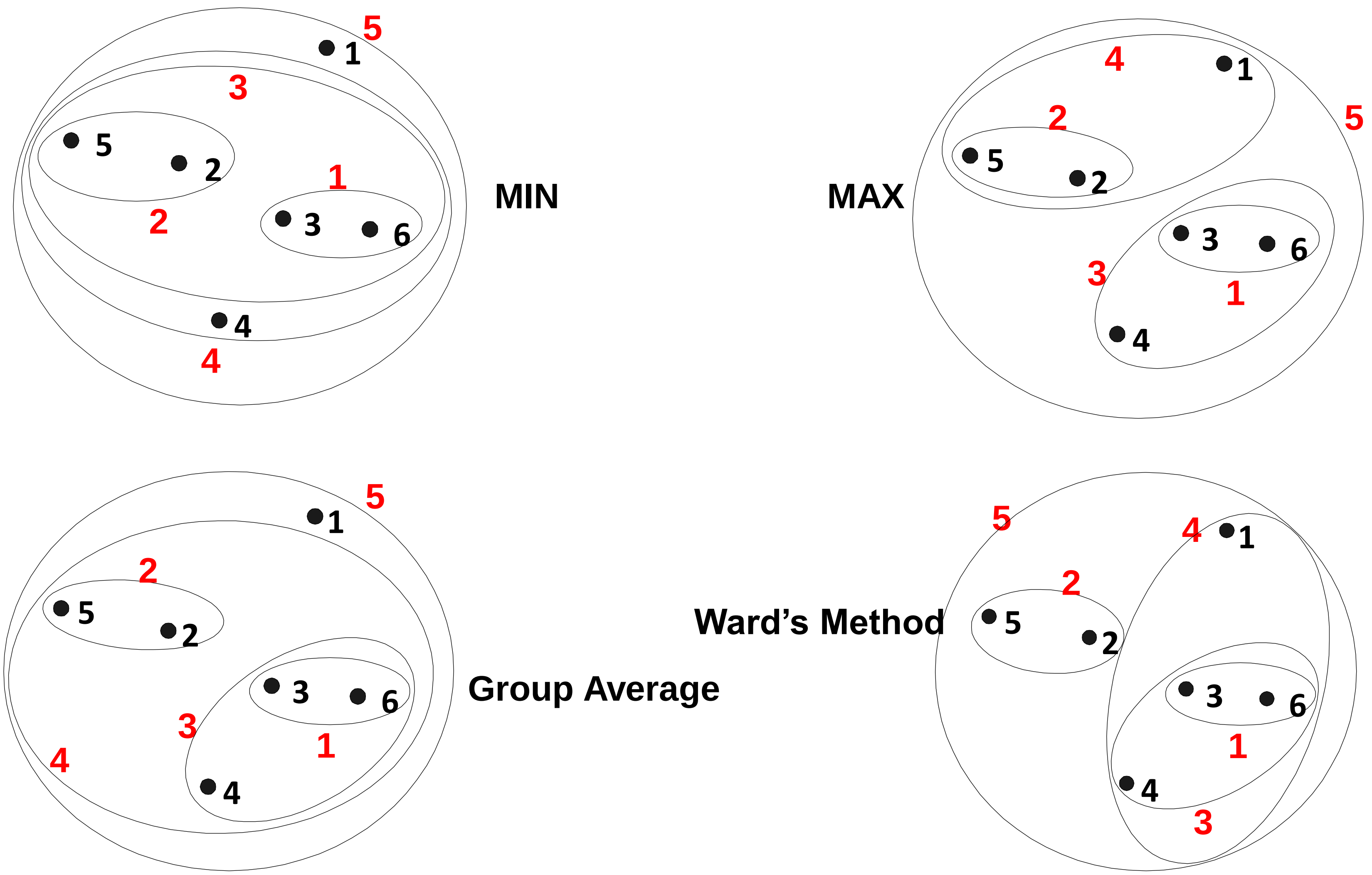
Ward's Method

- Similarity of two clusters is based on the increase in squared error when two clusters are merged

$$\Delta(A,B) \;=\; \sum_{i \in A \cup B} \|\vec{x}_i - \vec{m}_{A \cup B}\|^2 - \sum_{i \in A} \|\vec{x}_i - \vec{m}_A\|^2 - \sum_{i \in B} \|\vec{x}_i - \vec{m}_B\|^2$$

- Hierarchical analogue of K-means

Hierarchical Clustering: Comparison

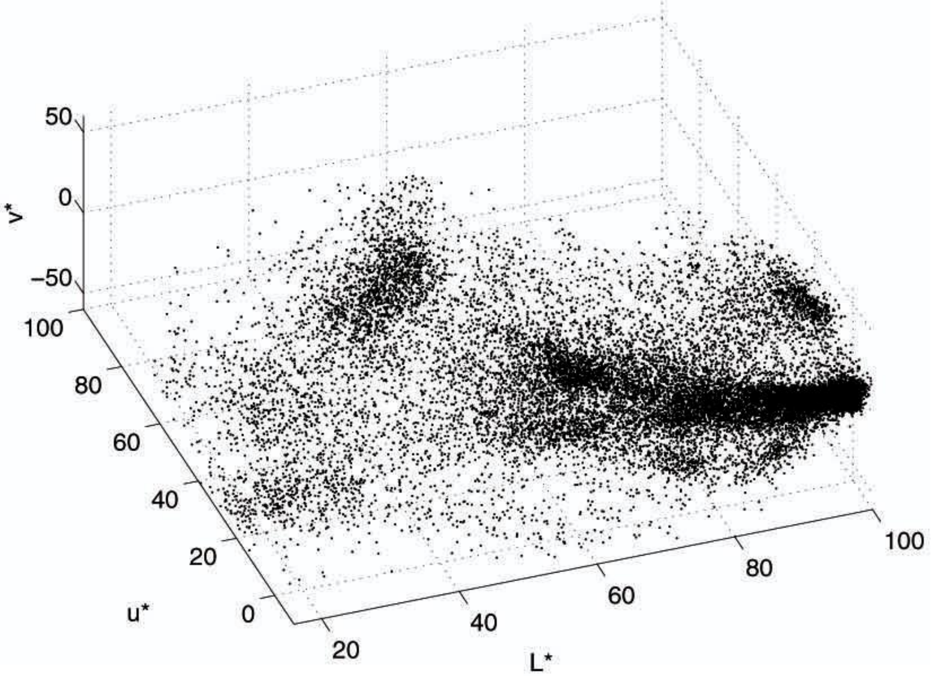


Mean shift algorithm

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

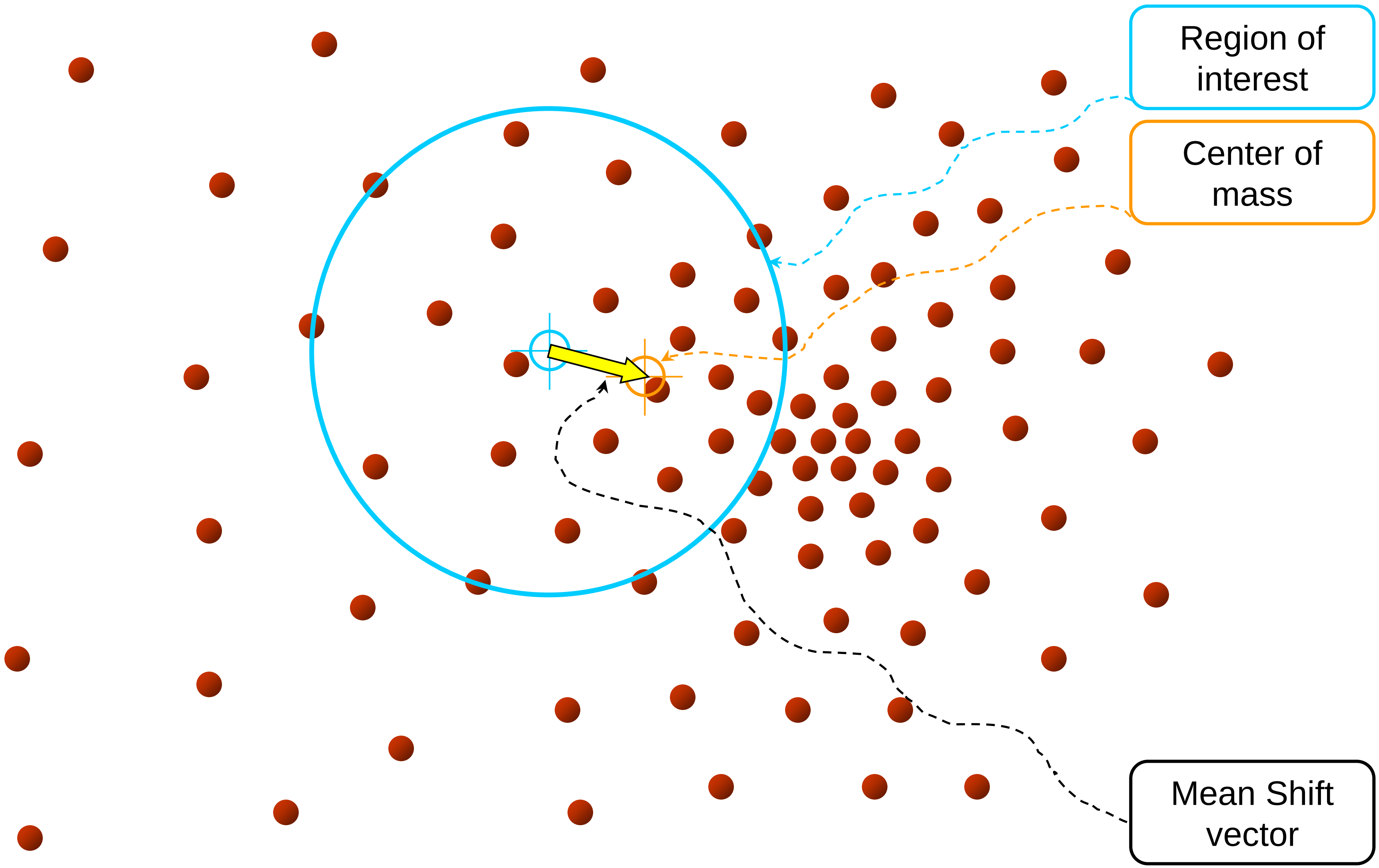
image

Feature space (L*u*v* color values)



Source: K. Grauman

Recall: Mean Shift

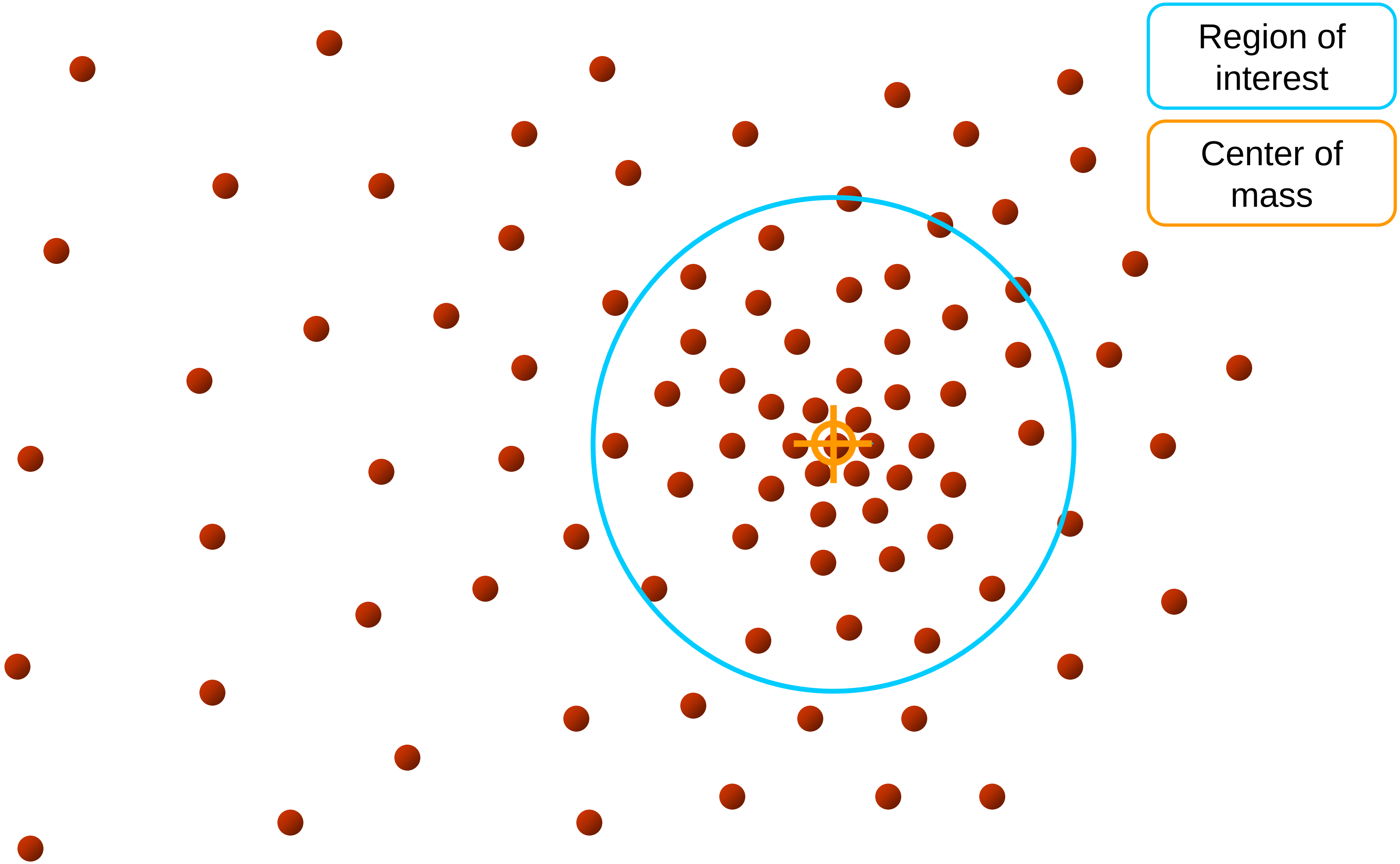


<u>Objective</u> : Find the densest region

Distribution of identical billiard balls

Source: Y. Ukrainitz and B. Sarel

Recall: Mean Shift



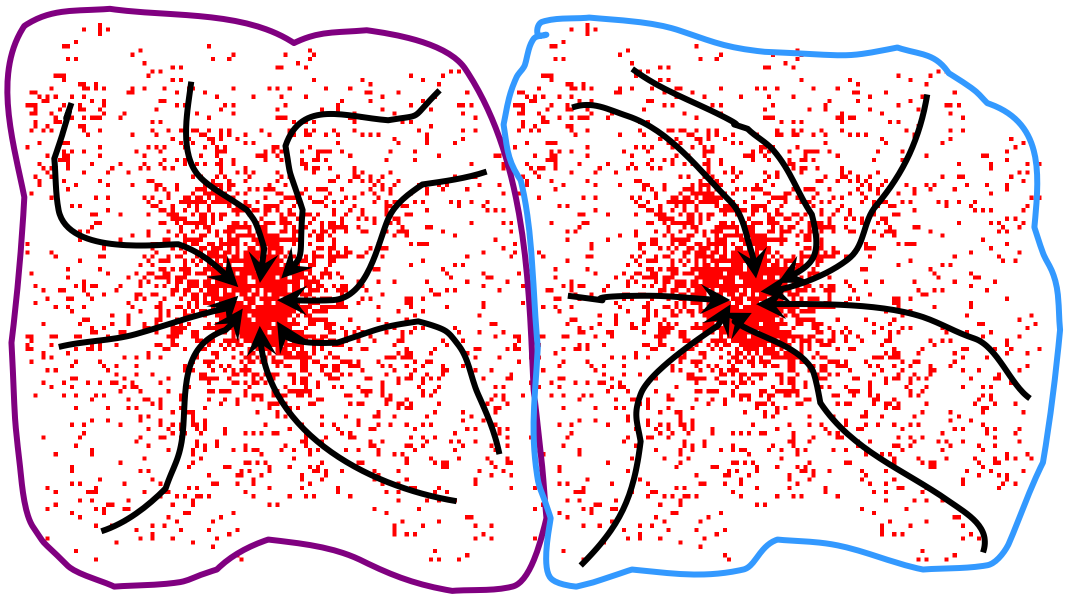
<u>Objective</u> : Find the densest region

Distribution of identical billiard balls

Source: Y. Ukrainitz and B. Sarel

Mean shift clustering

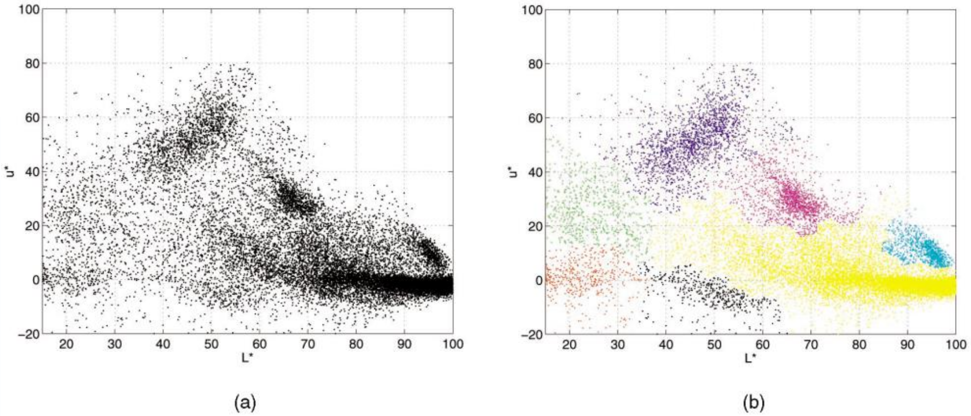
- Cluster: all data points in the attraction basin of a mode
- Attraction basin: the region for which all trajectories lead to the same mode

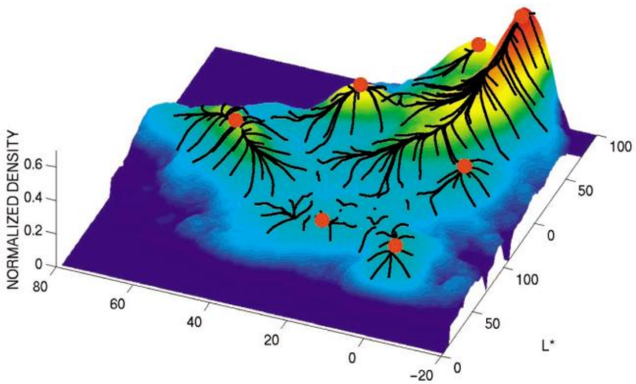


Source: Y. Ukrainitz and B. Sarel

Mean shift clustering/segmentation

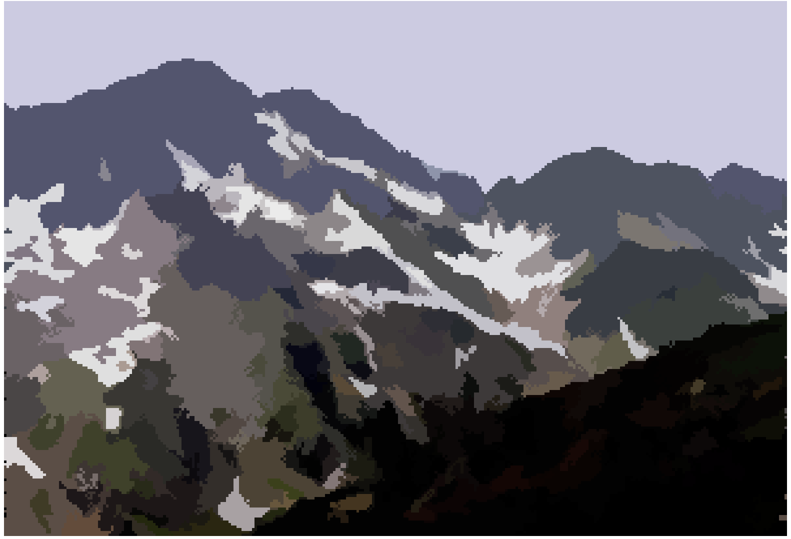
- Find features (color, gradients, texture, etc)
- Initialize windows at individual feature points
- Perform mean shift for each window until convergence
- Merge windows that end up near the same "peak" or mode





Source: K. Grauman

Mean shift segmentation results



Source: D. Comaniciu

Mean shift segmentation results

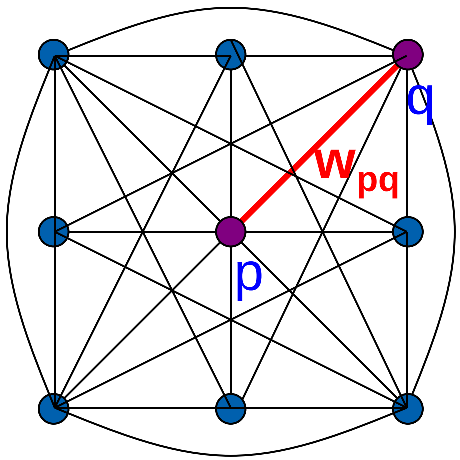
Source: D. Comaniciu

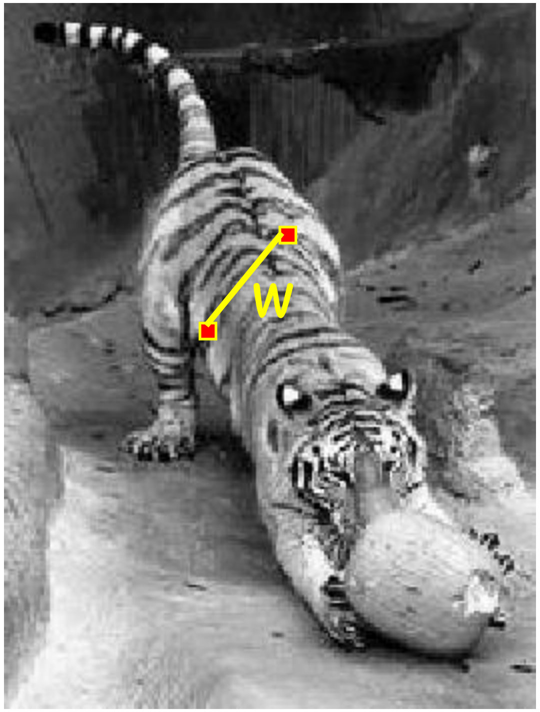
Mean shift

- <u>Pros</u>:
  - Does not assume shape on clusters
  - One parameter choice (window size)
  - Generic technique
  - Find multiple modes
- <u>Cons</u>:
  - Selection of window size
  - Does not scale well with dimension of feature space

Source: K. Grauman

Images as graphs





- *Fully-connected* graph
  - node (vertex) for every pixel
  - link between *every* pair of pixels, **p**,**q**
  - affinity weight $\mathbf{w_{pq}}$ for each link (edge)
    - $\mathbf{w_{pq}}$ measures *similarity*
      - similarity is *inversely proportional* to difference (in color and position…)

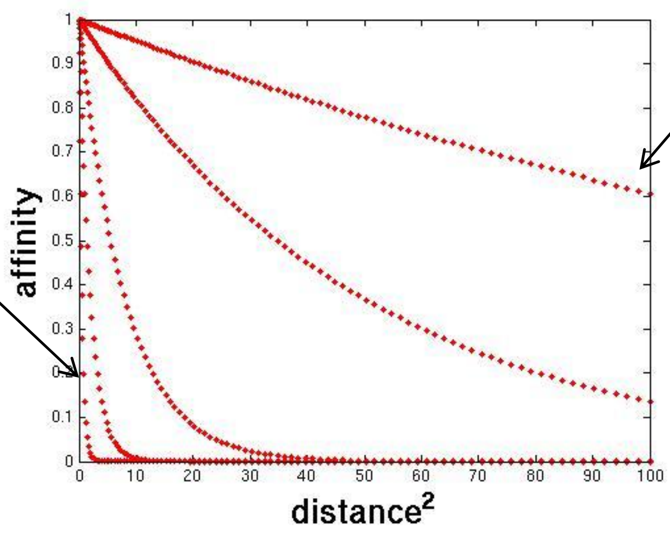
Source: S. Seitz

Measuring affinity

- One possibility:

$$W_{ij} = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$$

Small sigma: group only nearby points

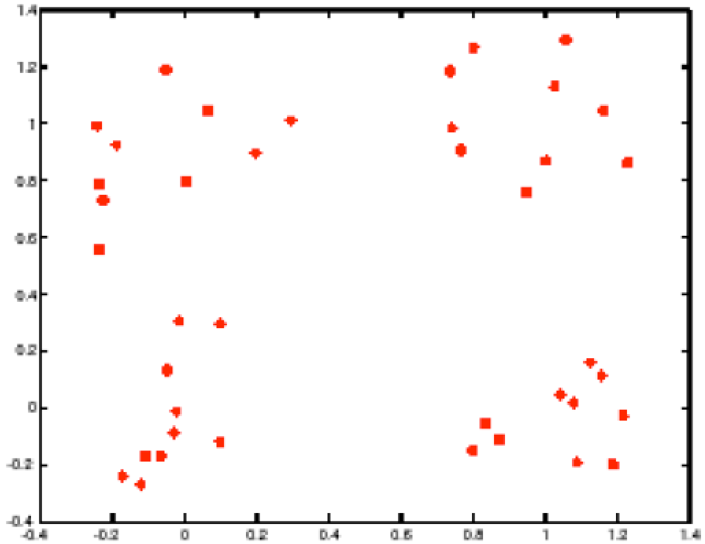


Large sigma: group distant points

Source: K. Grauman

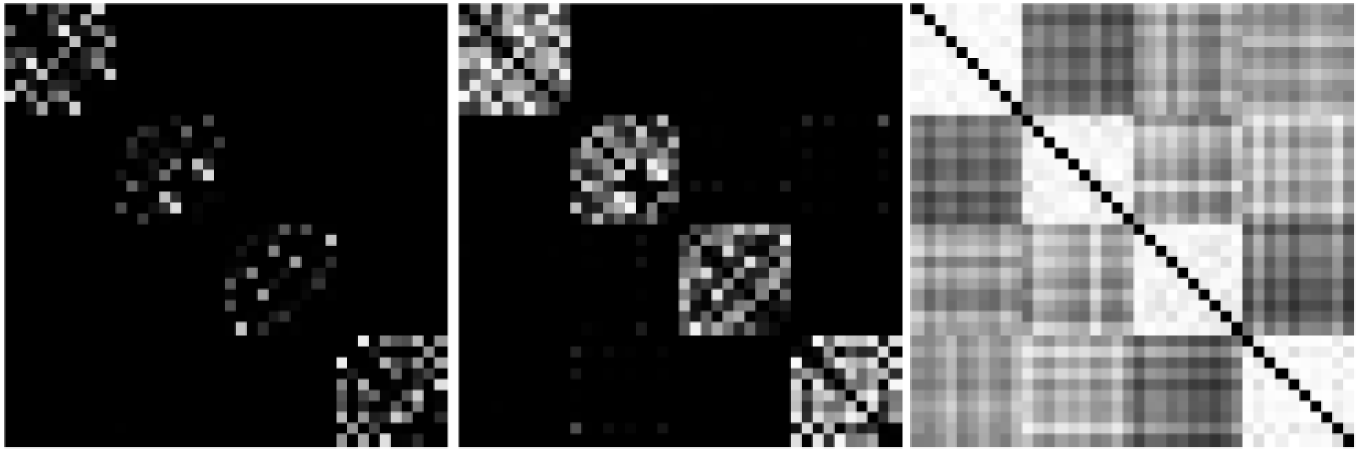
Measuring affinity

Data points



σ=.2

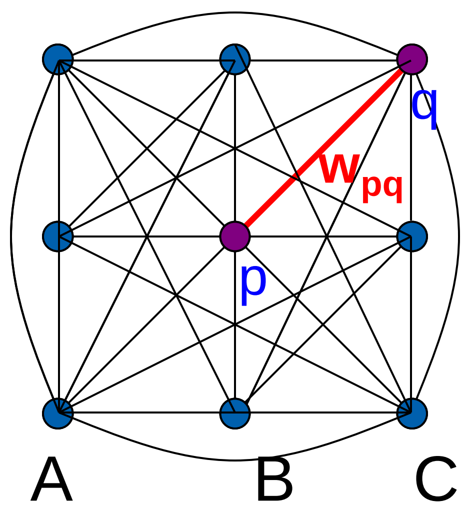
Affinity matrices



σ=.1 σ=.2 σ=1

Source: K. Grauman

Segmentation by Graph Cuts

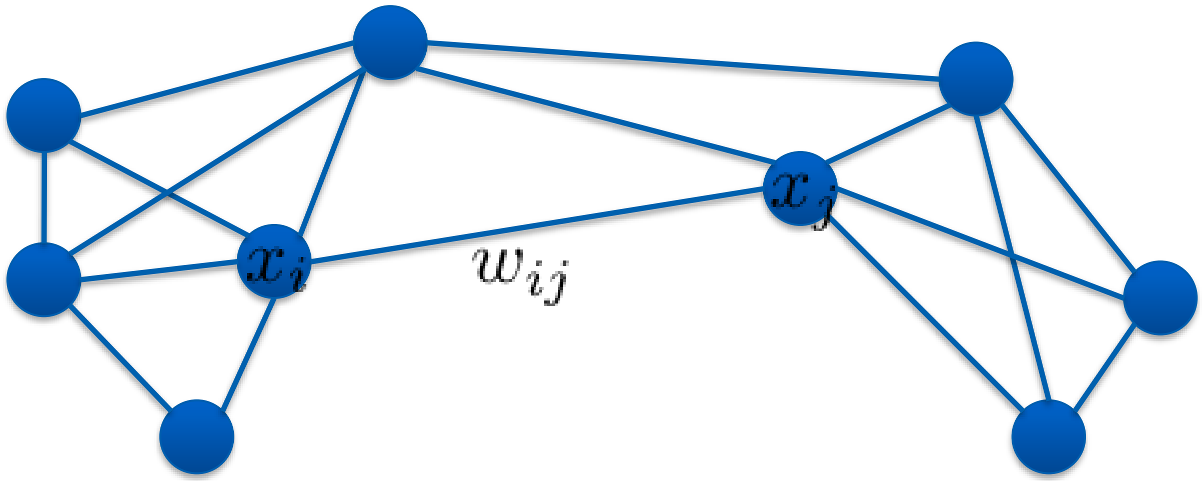


- Break Graph into Segments
  - Want to delete links that cross **between** segments
  - Easiest to break links that have low similarity (low weight)
    - similar pixels should be in the same segments
    - dissimilar pixels should be in different segments

Source: S. Seitz

Graphs

- Nodes and Edges
- Edges can be directed or undirected
- Edges can have weights associated with them



- Here the weights correspond to **pairwise affinity**

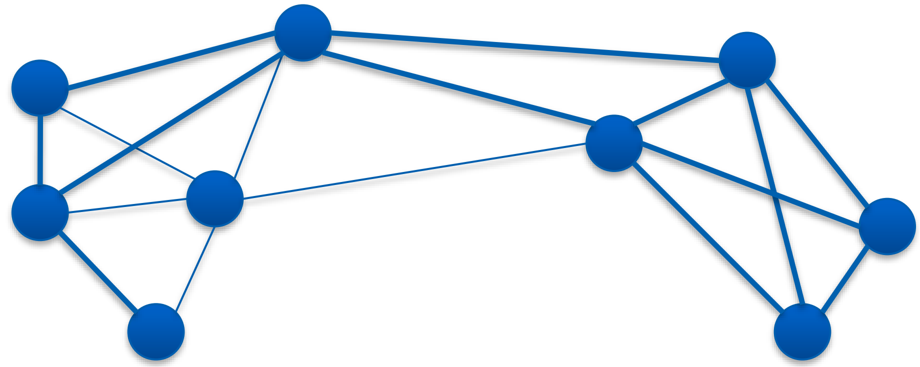
$$w_{ij} = d(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$$

Source: A. Rosenberg

Graphs

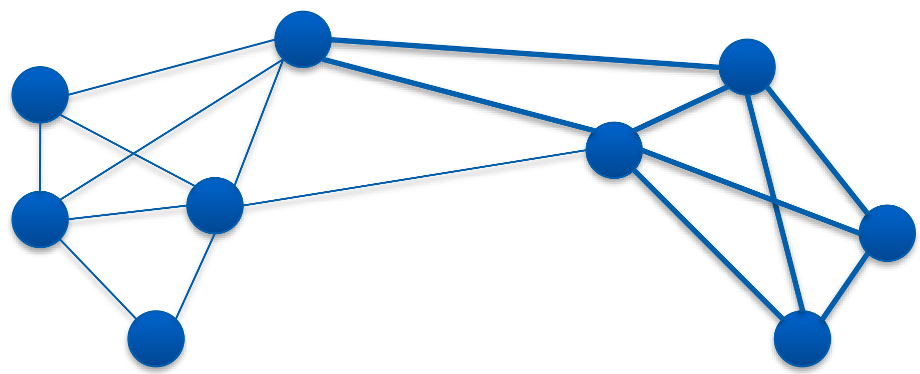
- Degree

$$D(x_i) = \sum_{j \in V} w_{ij}$$



- Volume of a set

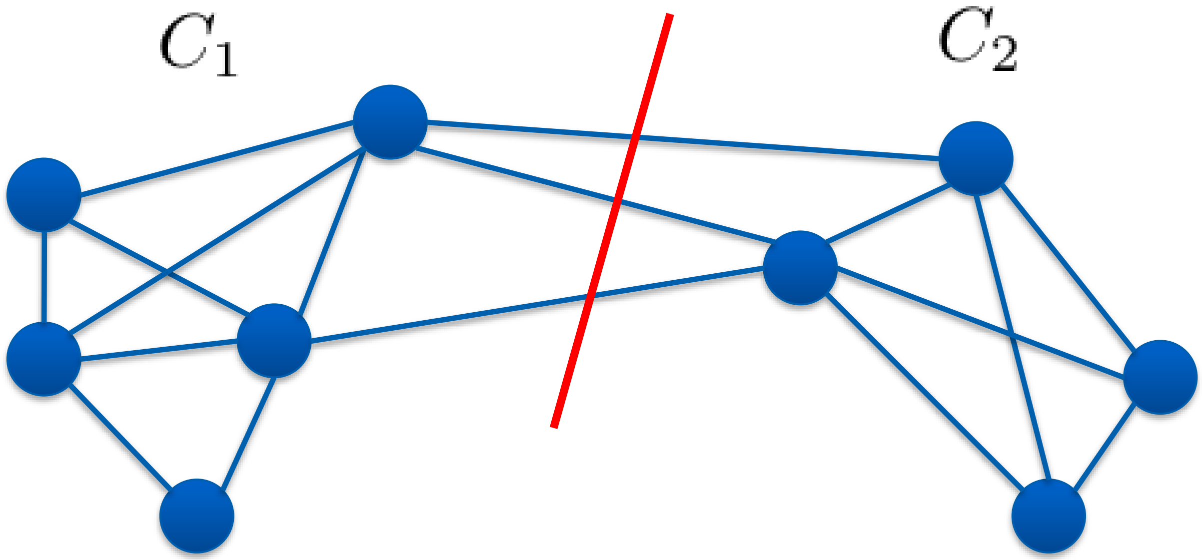
$$Vol(C) = \sum_{i \in C} D(x_i)$$



Source: A. Rosenberg

Graph Cuts

- The **cut** between two subgraphs is calculated as follows



$$Cut(C_1, C_2) = \sum_{i \in C_1} \sum_{j \in C_2} w_{ij}$$

Source: A. Rosenberg

Intuition

- The minimum cut of a graph identifies an optimal partitioning of the data.

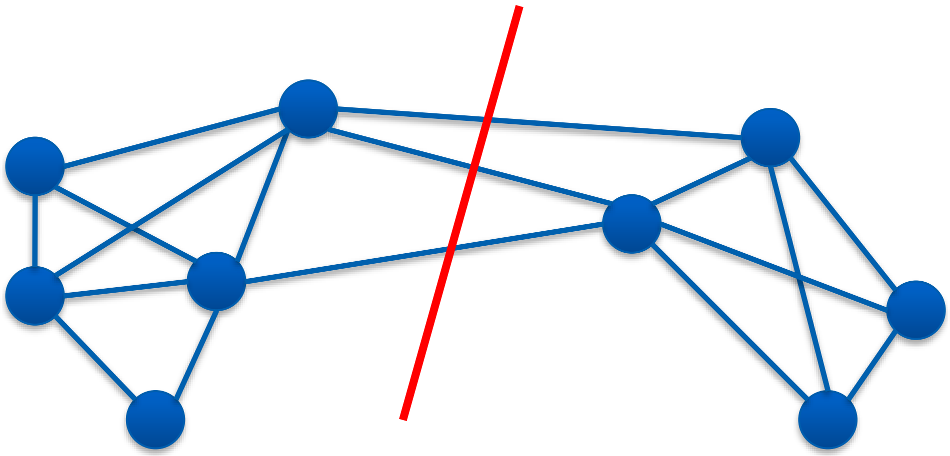
- Clustering
  - Recursively partition the data set
    - Identify the minimum cut
    - Remove edges
    - Repeat until k clusters are identified

Source: A. Rosenberg

Graph Cuts

- Minimum (bipartitional) cut

$$\min Cut(C_1, C_2) = \sum_{i \in C_1} \sum_{j \in C_2} w_{ij}$$

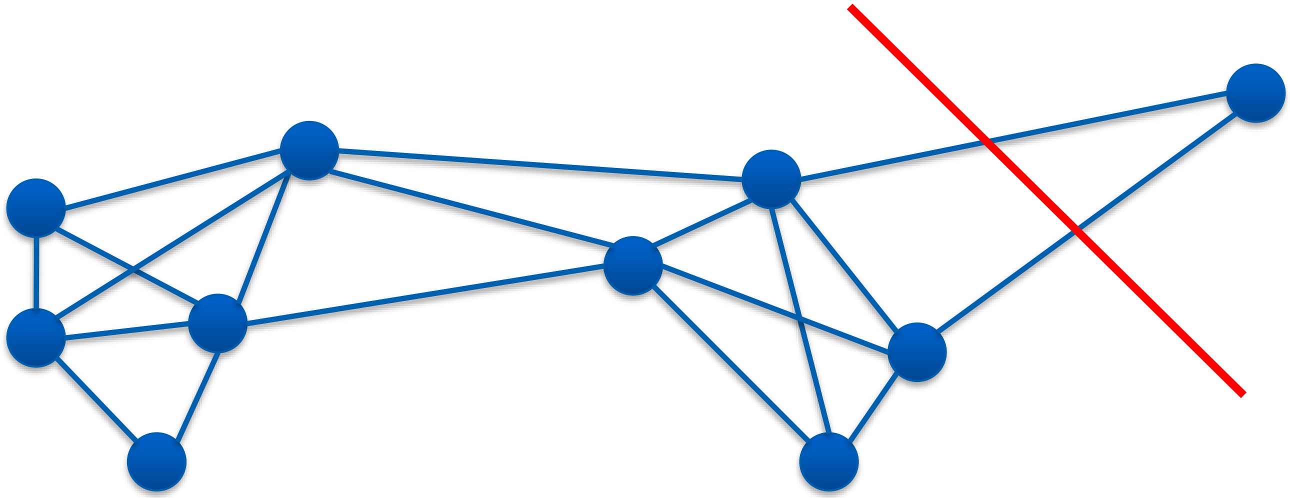


Source: A. Rosenberg

Graph Cuts

- Minimum (bipartitional) cut

$$\min Cut(C_1, C_2) = \sum_{i \in C_1} \sum_{j \in C_2} w_{ij}$$

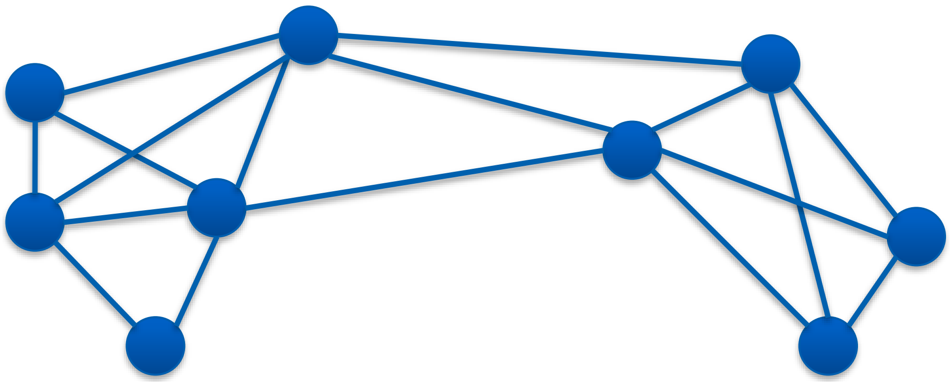


Source: A. Rosenberg

Graph Cuts

- Minimal (bipartitional) normalized cut.

$$\min \frac{Cut(C_1, C_2)}{Vol(C_1)} + \frac{Cut(C_1, C_2)}{Vol(C_2)} = \min \left(\frac{1}{Vol(C_1)} + \frac{1}{Vol(C_2)} \right) Cut(C_1, C_2)$$

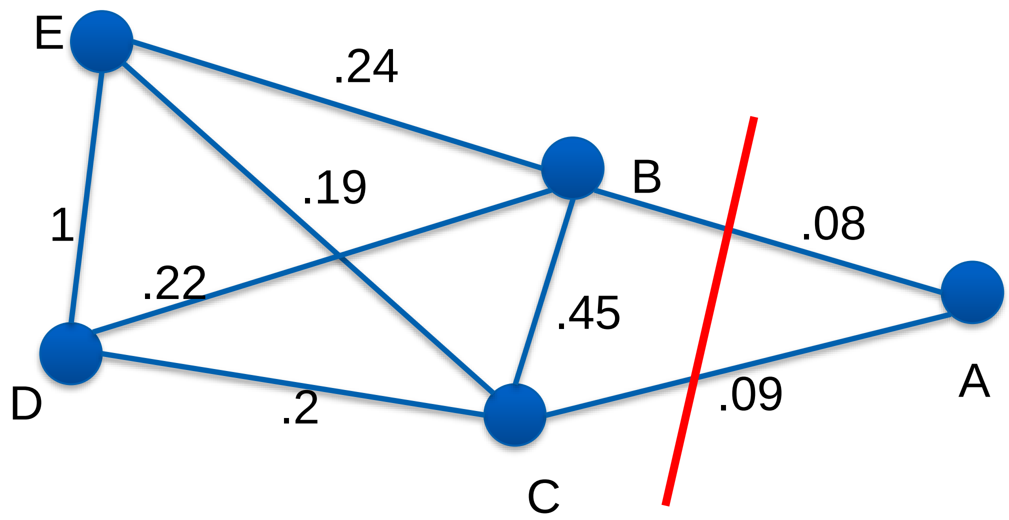


- Unnormalized cuts are attracted to outliers.

Source: A. Rosenberg

Spectral Clustering Example

- Minimum Cut

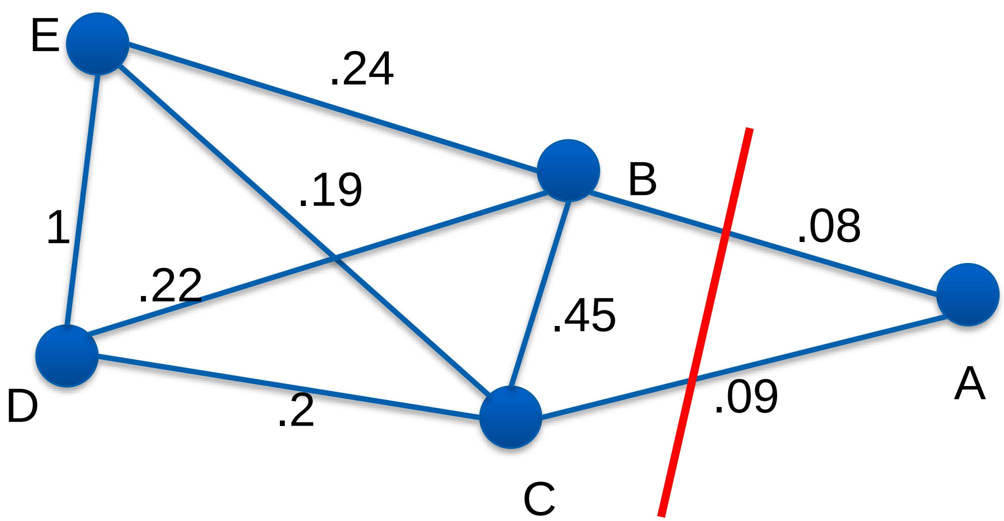


Cut(BCDE,A)=0.08+0.09=0.17

Spectral Clustering Example

- Normalized Minimum Cut

$$NormCut(C_1, C_2) = \frac{Cut(C_1, C_2)}{Vol(C_1)} + \frac{Cut(C_1, C_2)}{Vol(C_2)}$$

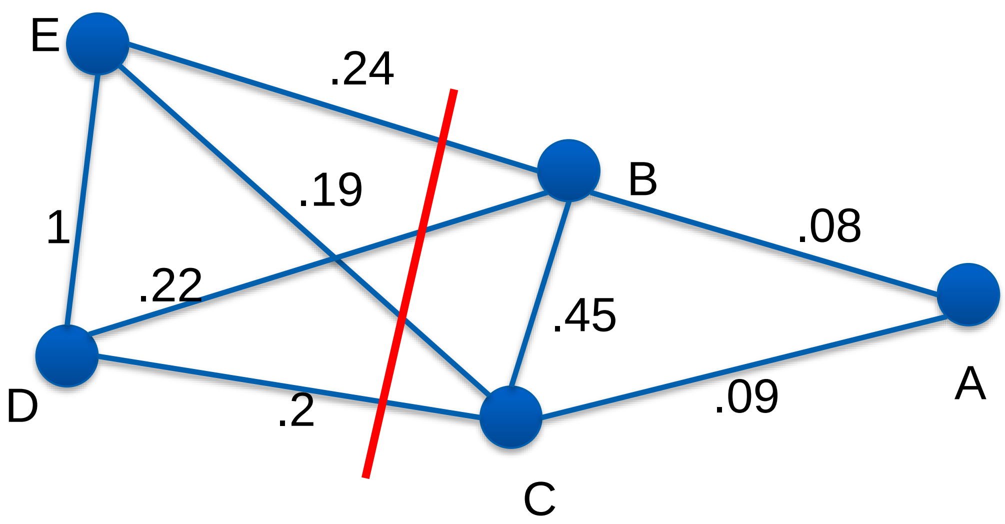


Cut(BCDE,A)=0.17/(0.99+0.93+1.42+1.43) + 0.17/0.17 = 1.036

Spectral Clustering Example

- Normalized Minimum Cut

$$NormCut(C_1, C_2) = \frac{Cut(C_1, C_2)}{Vol(C_1)} + \frac{Cut(C_1, C_2)}{Vol(C_2)}$$



Cut(BCDE,A)=0.17/(0.99+0.93+1.42+1.43) + 0.17/0.17 = 1.036

Cut(DE,ABC)=0.85/(1.42+1.43) + 0.85/(0.17+0.99+0.93) = 0.705

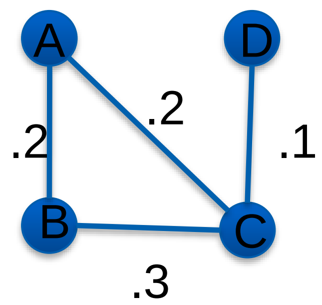
Problem

- Identifying a minimum cut is NP-hard.
- There are efficient approximations using linear algebra.
- Based on the Laplacian Matrix, or **graph Laplacian**

Source: A. Rosenberg

Spectral Clustering

- Construct an **affinity** matrix



W

	A	**B**	**C**	**D**
A	0	.2	.2	0
B	.2	0	.3	0
C	.2	.3	0	.1
D	0	0	.1	0

- Graph Laplacian

$$L = D - W$$

	A	**B**	**C**	**D**
A	.4	-.2	-.2	-0
B	-.2	.5	-.3	-0
C	-.2	-.3	.6	-.1
D	0	0	-.1	.1

Spectral Clustering

UNIVERSITÄT BONN

- Reformulate problem:

$$min_x Ncut(x) = min_y \frac{y^T(\mathbf{D}-\mathbf{W})y}{y^T\mathbf{D}y}$$

$$y(i) \in \{1, -b\} \text{ and } y^T\mathbf{D1} = 0 \qquad b = \frac{k}{1-k} = \frac{\sum_{x_i>0} \boldsymbol{d}_i}{\sum_{x_i<0} \boldsymbol{d}_i}$$

- If *y* is relaxed to be real valued, it becomes the generalized eigenvalue problem

$$(\mathbf{D}-\mathbf{W})\boldsymbol{y} = \lambda \mathbf{D}y$$

or standard eigenvalue problem

$$\mathbf{D}^{-\frac{1}{2}}(\mathbf{D}-\mathbf{W})\mathbf{D}^{-\frac{1}{2}}z = \lambda z \qquad z = \mathbf{D}^{\frac{1}{2}}\boldsymbol{y}$$

[Jianbo Shi and Jitendra Malik. Normalized Cuts and Image Segmentation. TPAMI 2000]

Spectral Clustering

- If y is relaxed to be real valued, it becomes the generalized eigenvalue problem

$$(\mathbf{D} - \mathbf{W})\boldsymbol{y} = \lambda \mathbf{D} \boldsymbol{y}$$

  or standard eigenvalue problem

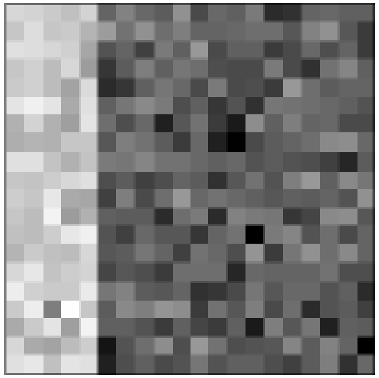
$$\mathbf{D}^{-\frac{1}{2}}(\mathbf{D} - \mathbf{W})\mathbf{D}^{-\frac{1}{2}} z = \lambda z \qquad z = \mathbf{D}^{\frac{1}{2}} \boldsymbol{y}$$

- Trivial solution eigenvector with eigenvalue 0: $y_0 = 1$
- More interesting: eigenvector corresponding to the second smallest eigenvalue (by definition orthogonal to the first eigenvector)

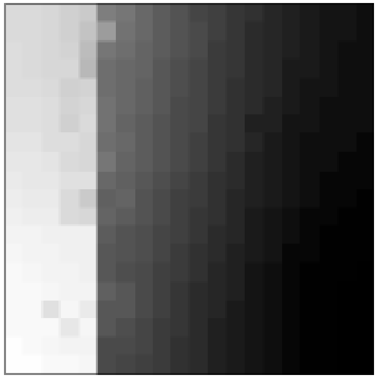
[Jianbo Shi and Jitendra Malik. Normalized Cuts and Image Segmentation. TPAMI 2000]

Binary segmentation

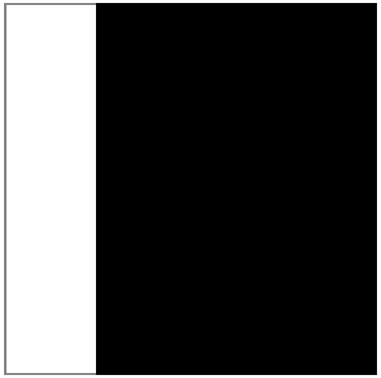
- Why is the solution only an approximation?
- The eigenvector is real-valued. Where is our segmentation/clustering?
- We solved the relaxed problem
  → generally no clear cut of the graph, just soft indicators
- We get a (generally suboptimal) solution of the binary segmentation problem by thresholding the eigenvector



Input image



2^{nd} Eigenvector



After thresholding

Source: T. Brox

Eigensolver

- The considered matrices can get huge
  → usual numerical methods for computing eigenvalues with cubic complexity $O(N^3)$ are very slow

- We may not always deal with fully connected graphs
  → matrix will be sparse (many zero entries)

- We just require the smallest eigenvalues and their corresponding eigenvectors

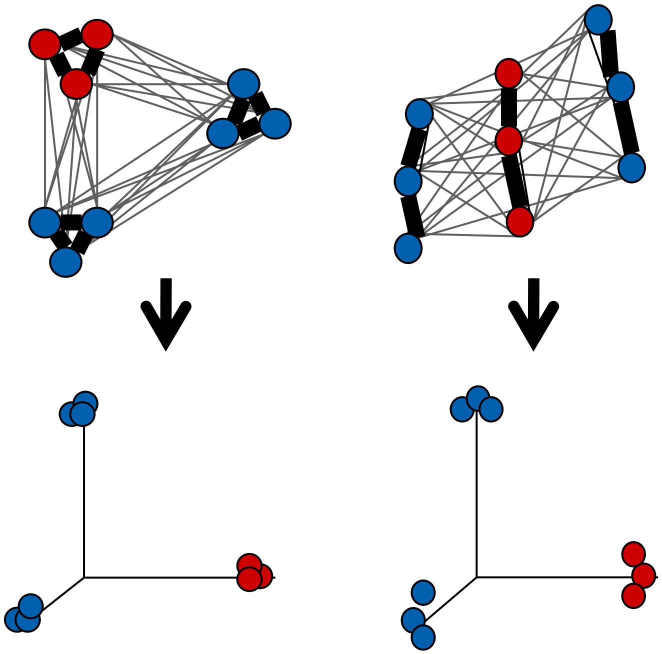
- Indeed there is the Lanczos method, which efficiently computes just the smallest (or largest) eigenvalues of a matrix in $O(N^2)$

Source: T. Brox

More than two clusters, Laplacian eigenmaps

- The eigenvector to the third smallest eigenvalue indicates an alternative partitioning of the graph
- In the general case we compute the k smallest eigenvalues
- The corresponding eigenvectors span a subspace
  - $Y = (y_1, y_2, \ldots, y_k) \in \mathbb{R}^{Nxk}$ matrix
  - Each data point $1, 2, \ldots, N$ maps to a k-dimensional vector (i-th row of Y)
  - Mapping is called **Laplacian eigenmap**

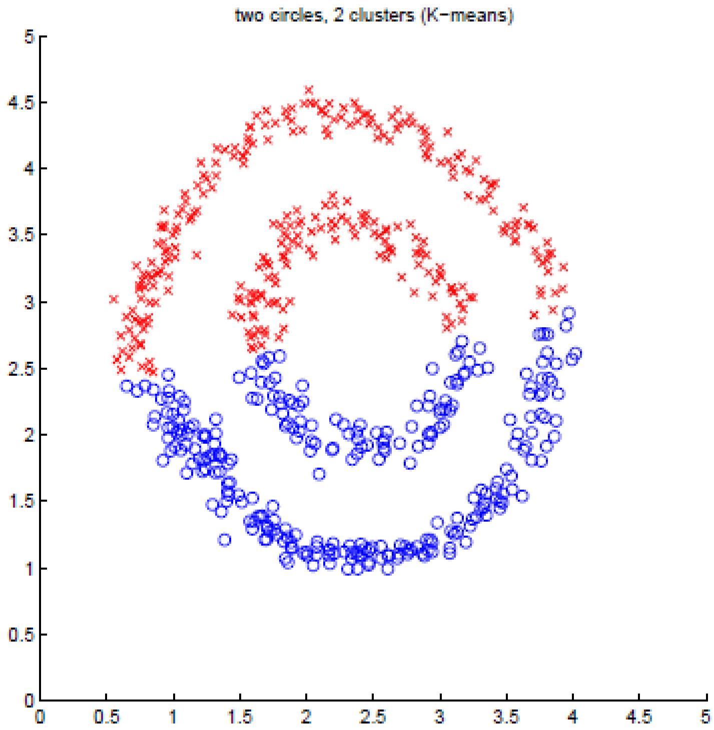
Standard clustering techniques (k-means) to convert the real-valued vectors into integer labels
→ **spectral clustering**



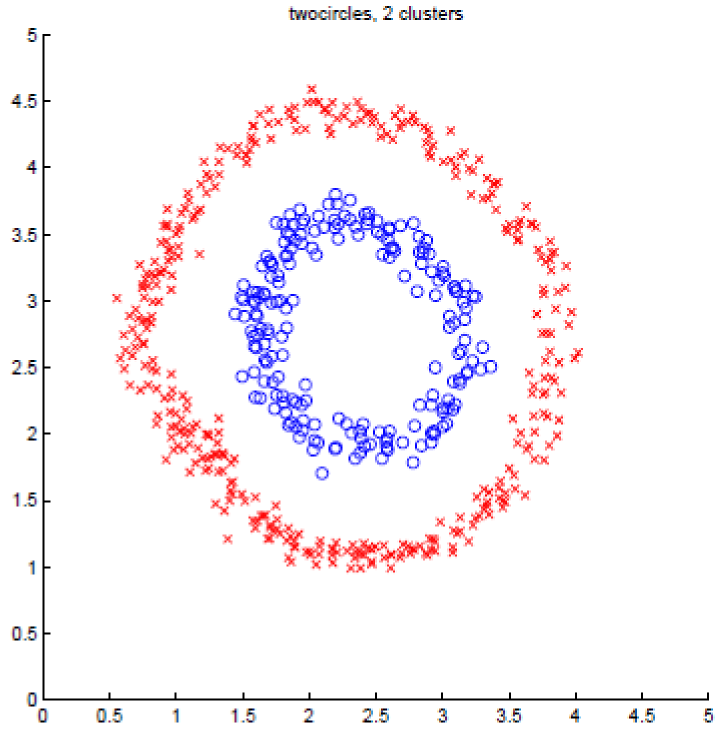
Mappings from a graph to its eigenspace, clustering in the eigenspace

[Ulrike von Luxburg. **A Tutorial on Spectral Clustering.** Statistics and Computing 2007]

K means vs. spectral clustering



K means



Spectral clustering

[A. Ng et al. **On spectral clustering: Analysis and an algorithm.** NIPS 2001]

Example results

Eigenvalue Problem

- Reformulate problem:

$$min_x Ncut(x) = min_y \frac{y^T(\mathbf{D}-\mathbf{W})y}{y^T\mathbf{D}y}$$

- If *y* is relaxed to be real valued, it becomes the generalized eigenvalue problem

$$(\mathbf{D}-\mathbf{W})\boldsymbol{y} = \lambda \mathbf{D} y$$

Why is this the case?

Eigenvalue Problem

- Minimize or Maximize objective J:

$$J(\mathbf{w}) = \frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_W \mathbf{w}}$$

- S symmetric positive definite
- Known as Rayleigh Quotient
- Scale invariant: $J(w)=J(\alpha\, w)$ for scalar α
- Thus, we can set:

$$\mathbf{w}^T S_W \mathbf{w} = 1$$

Eigenvalue Problem

- Maximize objective J (Rayleigh Quotient):

$$J(\mathbf{w}) = \frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_W \mathbf{w}}$$

- Equivalent to

$$\begin{array}{ll} \min_{\mathbf{w}} & -\frac{1}{2}\mathbf{w}^T S_B \mathbf{w} \\ \text{s.t.} & \mathbf{w}^T S_W \mathbf{w} = 1 \end{array}$$

Eigenvalue Problem

$$
\begin{aligned}
\min_{\mathbf{w}} \quad & -\frac{1}{2}\mathbf{w}^T S_B \mathbf{w} \\
\text{s.t.} \quad & \mathbf{w}^T S_W \mathbf{w} = 1
\end{aligned}
$$

- Lagrangian multiplier:

$$L(\mathbf{w}, \lambda) = -\frac{1}{2}\mathbf{w}^T S_B \mathbf{w} + \frac{1}{2}\lambda(\mathbf{w}^T S_W \mathbf{w} - 1)$$

Eigenvalue Problem

$$L(\mathbf{w}, \lambda) = -\frac{1}{2}\mathbf{w}^T S_B \mathbf{w} + \frac{1}{2}\lambda(\mathbf{w}^T S_W \mathbf{w} - 1)$$

$$L(\mathbf{w}, \lambda) = -\frac{1}{2}\mathbf{w}^T (S_B - \lambda S_W)\mathbf{w} - \frac{1}{2}\lambda$$

$$\frac{\partial}{\partial \mathbf{w}} L(\mathbf{w}, \lambda) = -(S_B - \lambda S_W)\mathbf{w} = 0$$

$$\frac{\partial \mathbf{x}^T \mathbf{B} \mathbf{x}}{\partial \mathbf{x}} = (\mathbf{B} + \mathbf{B}^T)\mathbf{x}$$

Eigenvalue Problem

$$L(\mathbf{w}, \lambda) = -\frac{1}{2}\mathbf{w}^T S_B \mathbf{w} + \frac{1}{2}\lambda(\mathbf{w}^T S_W \mathbf{w} - 1)$$

$$L(\mathbf{w}, \lambda) = -\frac{1}{2}\mathbf{w}^T (S_B - \lambda S_W)\mathbf{w} - \frac{1}{2}\lambda$$

$$\frac{\partial}{\partial \mathbf{w}} L(\mathbf{w}, \lambda) = -(S_B - \lambda S_W)\mathbf{w} = 0$$

$$S_B \mathbf{w} = \lambda S_W \mathbf{w}$$

(Generalized eigenvalue problem)

Eigenvalue Problem

- Generalized eigenvalue problem

$$S_B\mathbf{w} = \lambda S_W\mathbf{w} \quad \Rightarrow \quad S_W^{-1}S_B\mathbf{w} = \lambda\mathbf{w}$$

- $S_W^{-1}S_B$ is not symmetric
- Eigen decomposition (S_B is symmetric positive definite):

$$S_B = U\Lambda U^T \rightarrow S_B^{\frac{1}{2}} = U\Lambda^{\frac{1}{2}}U^T$$

- This gives eigenvalue problem:

$$S_B^{\frac{1}{2}}S_W^{-1}S_B^{\frac{1}{2}}\mathbf{v} = \lambda\mathbf{v} \qquad \mathbf{v} = S_B^{\frac{1}{2}}\mathbf{w}$$

Spectral Clustering

Normalized Minimum Cut (Relaxed):

$$min_x Ncut(x) = min_y \frac{y^T(\mathbf{D} - \mathbf{W})y}{y^T\mathbf{D}y}$$

Generalized eigenvalue problem:

$$(\mathbf{D} - \mathbf{W})\boldsymbol{y} = \lambda \mathbf{D}\boldsymbol{y}$$

Equal eigenvalue problem:

$$\mathbf{D}^{-\frac{1}{2}}(\mathbf{D} - \mathbf{W})\mathbf{D}^{-\frac{1}{2}}z = \lambda z \qquad z = \mathbf{D}^{\frac{1}{2}}\boldsymbol{y}$$

Eigenvector corresponding to the second smallest eigenvalue

Eigenvalue Problem

- Minimize objective J (Rayleigh Quotient):

$$J(\mathbf{w}) = \frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_W \mathbf{w}}$$

- Generalized eigenvalue problem:

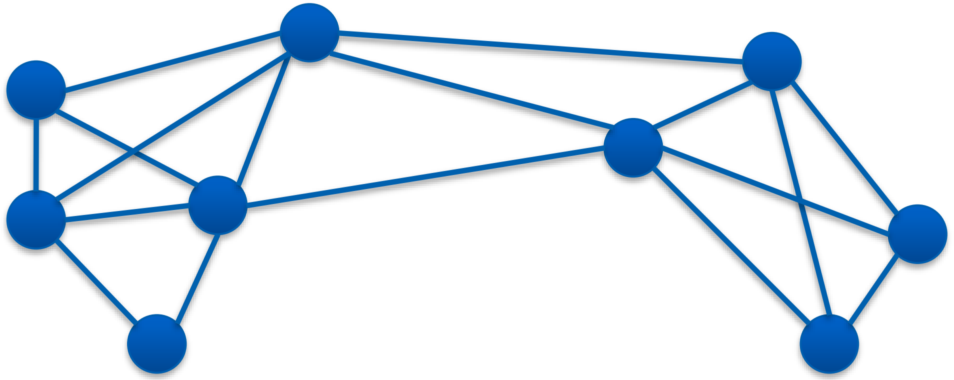
$$S_B \mathbf{w} = \lambda S_W \mathbf{w}$$

- Equal eigenvalue problem:

$$S_B^{\frac{1}{2}} S_W^{-1} S_B^{\frac{1}{2}} \mathbf{v} = \lambda \mathbf{v} \qquad \mathbf{v} = S_B^{\frac{1}{2}} \mathbf{w}$$

Random walk view of clustering

- In a random walk, you start at a node, and move to another node with some probability.
- The intuition is that if two nodes are in the same cluster, a random walk is likely to reach both points.

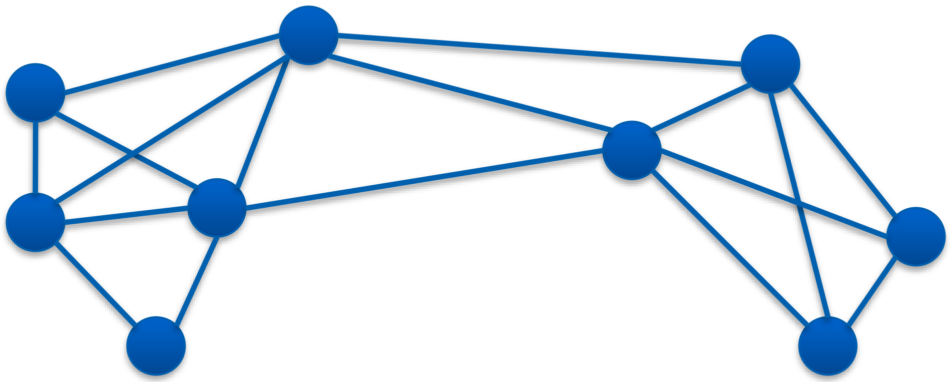


Source: A. Rosenberg

Random walk view of clustering

- Transition probabilities: $D^{-1}W$
- The transition probability is related to the weight of given transition and the inverse degree of the current node.

$$NCut(A, \bar{A}) = P_{A\bar{A}} + P_{\bar{A}A}$$



[Marina Meila and Jianbo Shi. A Random Walks View of Spectral Segmentation. NIPS 2001]

Source: A. Rosenberg

Normalized cuts: pros and cons

Pros:

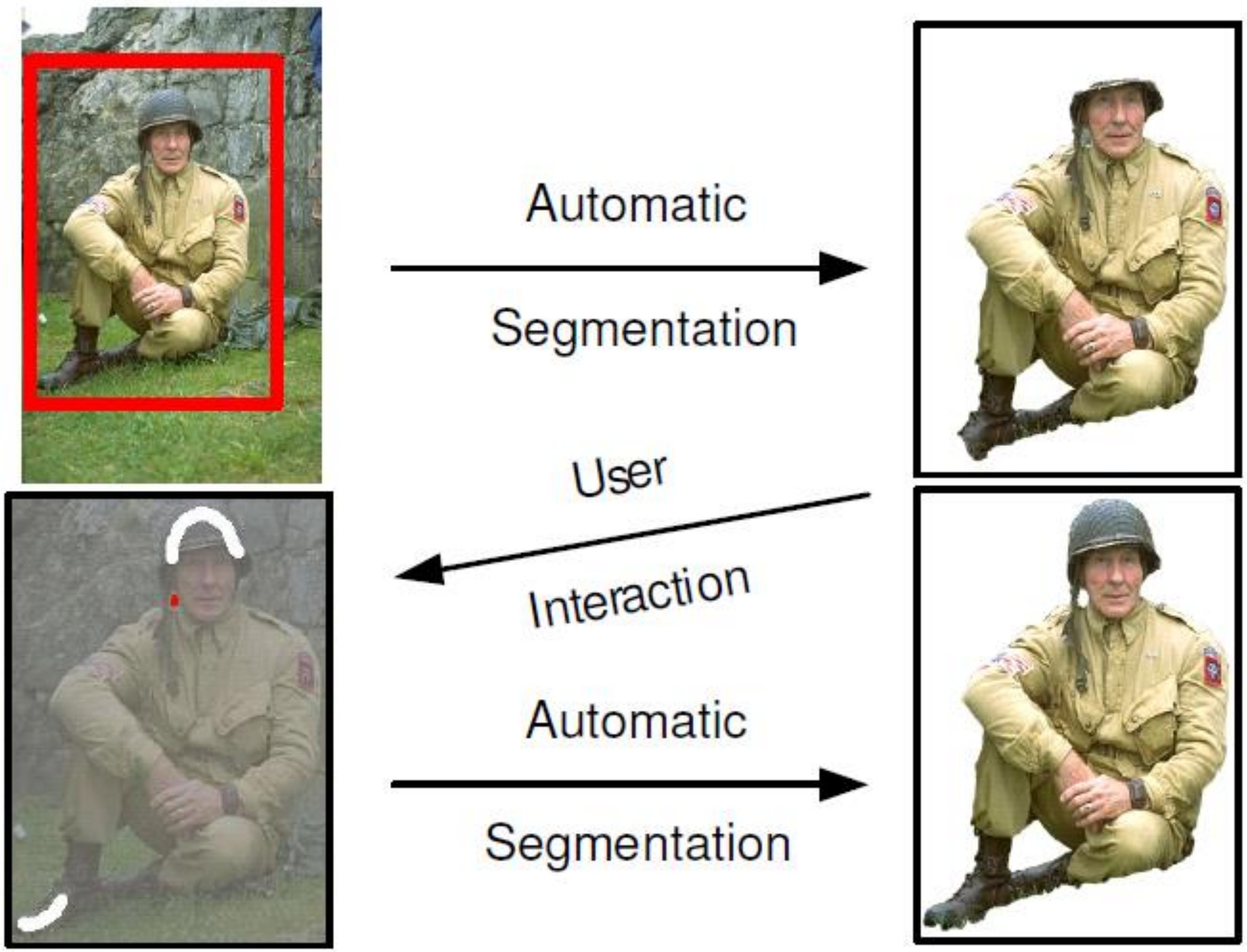
- Generic framework, flexible to choice of function that computes weights ("affinities") between nodes
- Does not require model of the data distribution

Cons:

- Time complexity can be high
  - Dense, highly connected graphs → many affinity computations
  - Solving eigenvalue problem
- Preference for balanced partitions

Source: S. Seitz

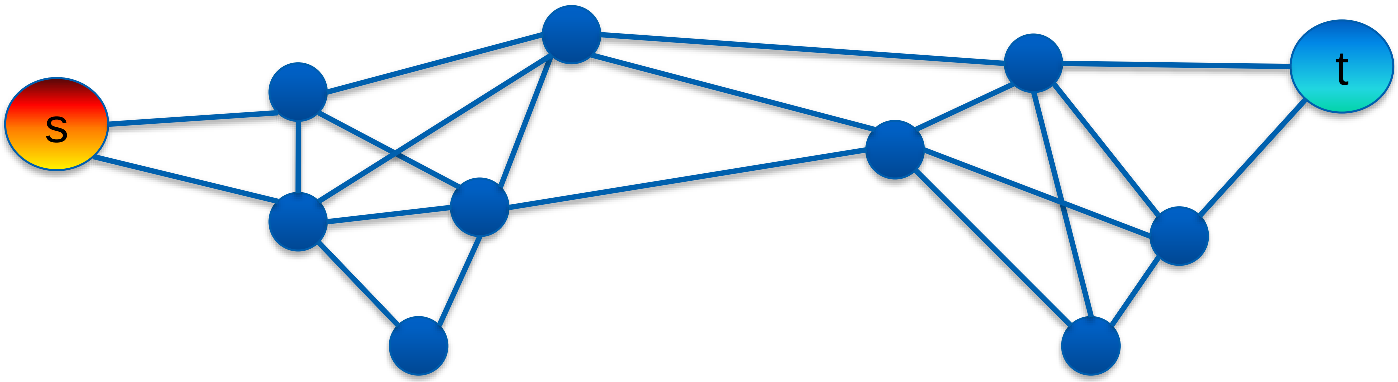
Application: Interactive segmentation



Y. Boykov and M.-P. Jolly. **Interactive Graph Cuts for Optimal Boundary & Region Segmentation of Objects in N-D images.** ICCV 2001
C. Rother et al. **GrabCut - Interactive Foreground Extraction using Iterated Graph Cuts.** SIGGRAPH 2004

Using minimum cut with labels?

- Construct a graph representation of unseen data.
- Insert imaginary nodes s and t connected to labeled points with infinite similarity.
- Treat the min cut as a maximum flow problem from s to t



Source: A. Rosenberg