CS 1675: Intro to Machine Learning

Unsupervised Learning (Clustering, Dimensionality Reduction)

Prof. Adriana Kovashka
University of Pittsburgh
September 6, 2018

Unsupervised Learning

- We only use the features X, not the labels Y
- This is useful because we may not have any labels but we can still detect patterns
- For example:
  - We can detect that news articles revolve around certain topics, and group them accordingly
  - Discover a distinct set of objects appear in a given environment, even if we don't know their names, then ask humans to label each group
  - Identify health factors that correlate with a disease

Plan for this lecture

- Clustering
  - Motivation and uses
  - K-means clustering
  - Other methods and evaluation
- Dimensionality reduction
  - PCA algorithm (briefly) and demo
  - Some applications of PCA

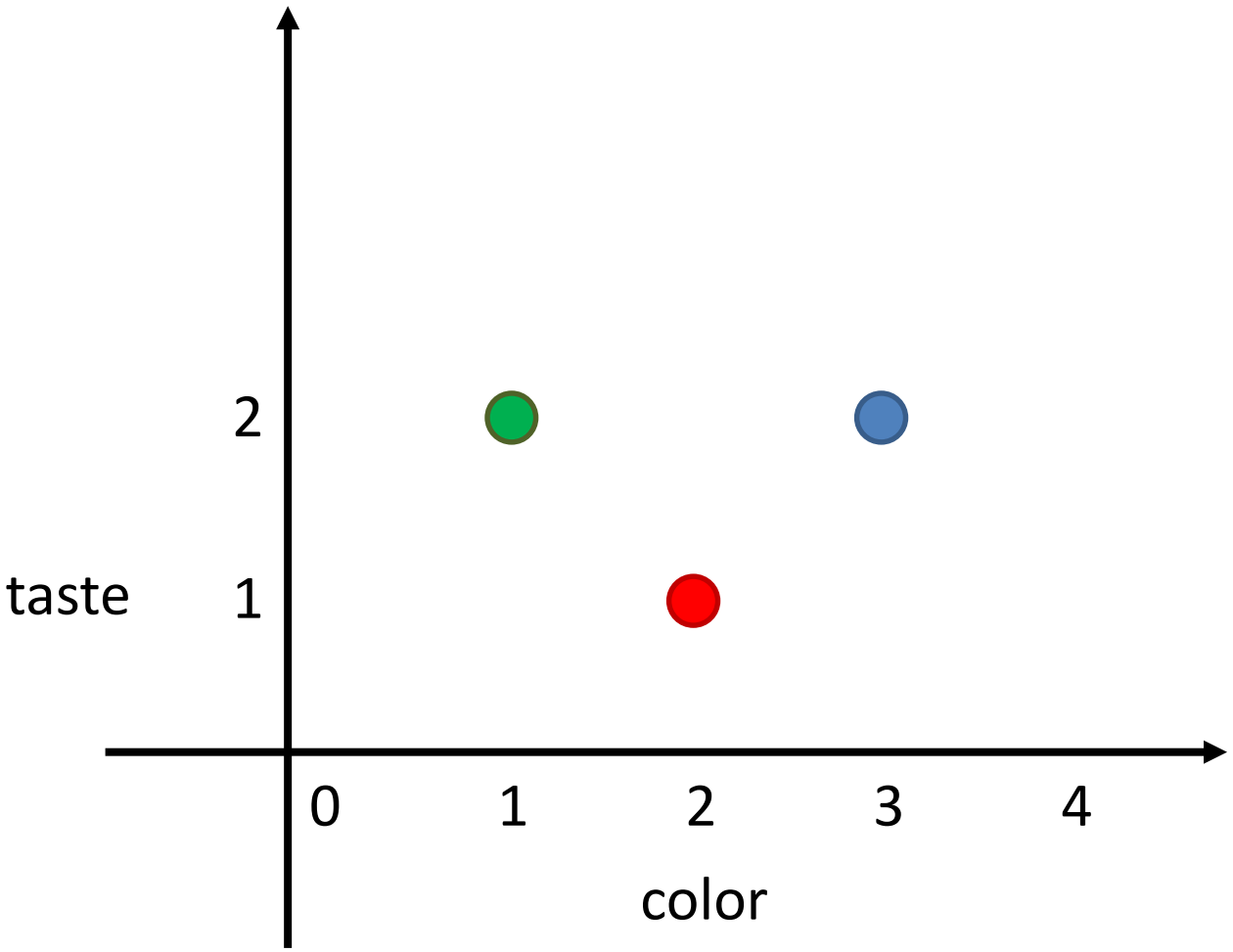
What is clustering?

- Grouping items that “belong together” (i.e. have similar features)

Feature representation (**x**)

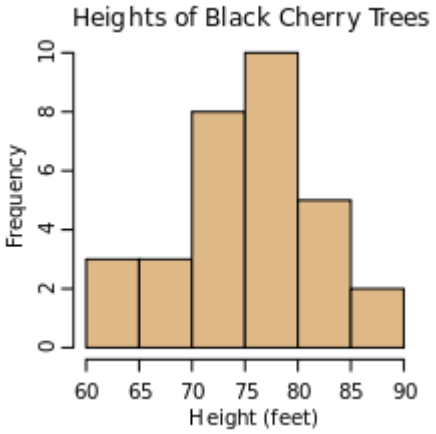
- A vector representing measurable characteristics of a data sample we have
- E.g. a glass of juice can be represented via its color = {yellow=1, red=2, green=3, purple=4} and taste = {sweet=1, sour=2}
- For a given glass i, this can be represented as a vector: $\mathbf{x}_i$ = [3 2] represents sour green juice
- For D features, this defines a D-dimensional space where we can measure similarity between samples

Feature representation (**x**)



Why do we cluster?

- Counting
  - Feature *histograms*: by grouping similar features and counting how many of each a data sample has
- Summarizing data
  - Look at large amounts of data
  - Represent a large continuous vector with the cluster number
- Prediction
  - Data points in the same cluster may have the same labels
  - Ask a human to label the clusters



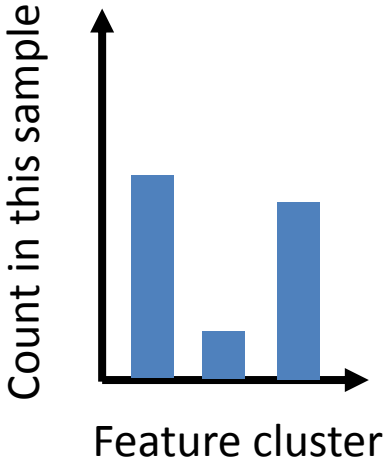
[3 2] → "juice type 3"

[3 2] → "kombucha"

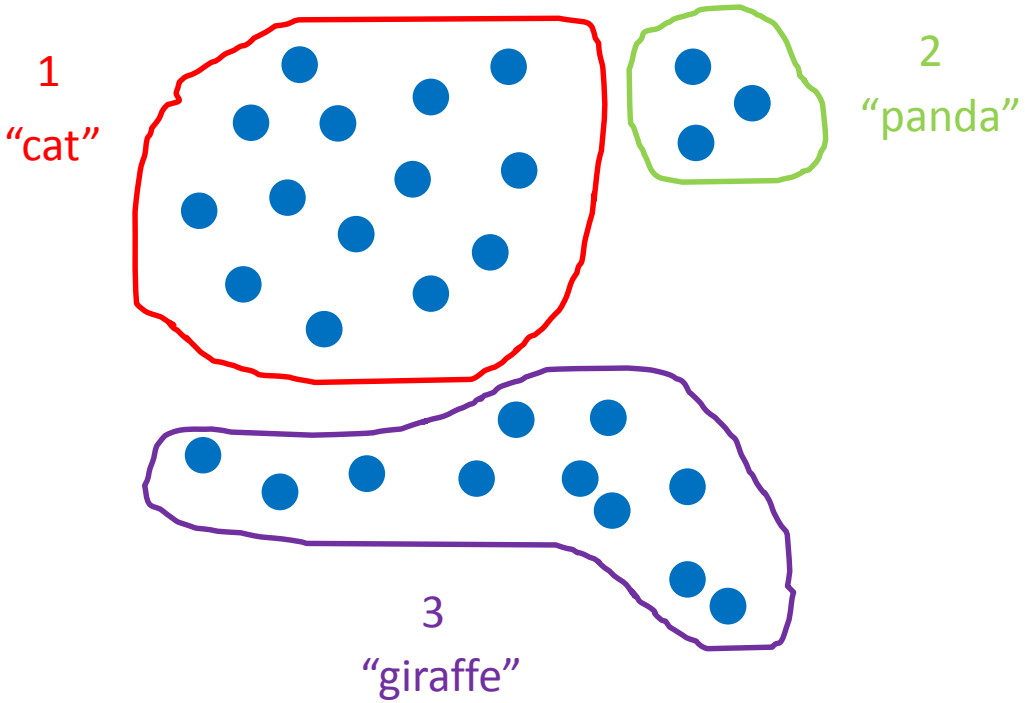
Slide credit: J. Hays, D. Hoiem

Two uses of clustering in one application

- Cluster, then ask human to label groups
- Compute a histogram to summarize the data

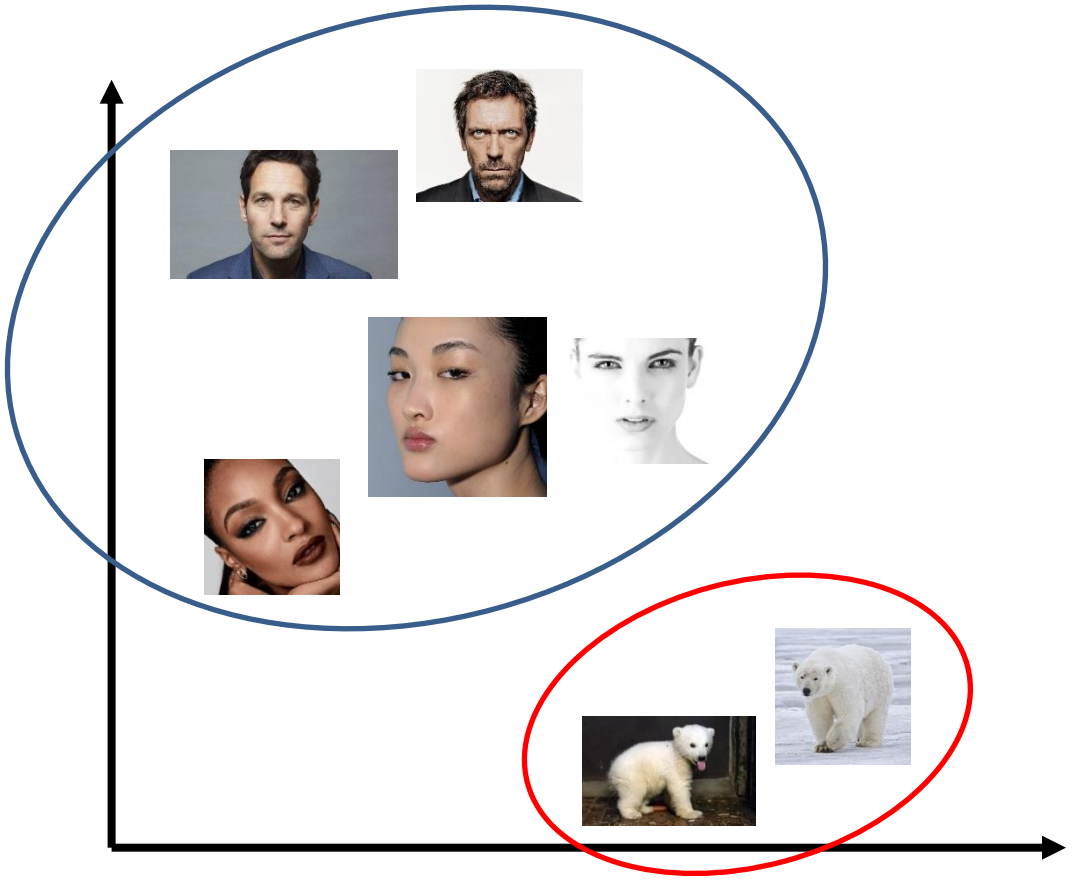


3d feature



2d feature

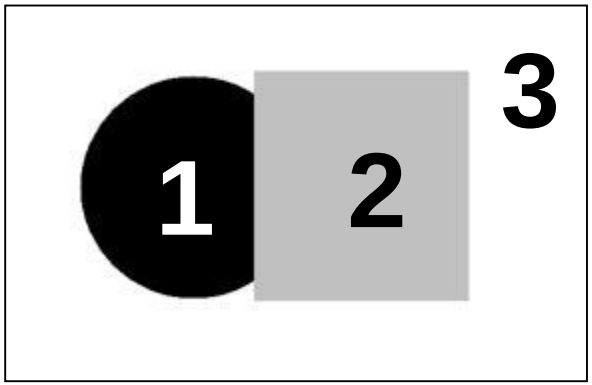
Unsupervised discovery



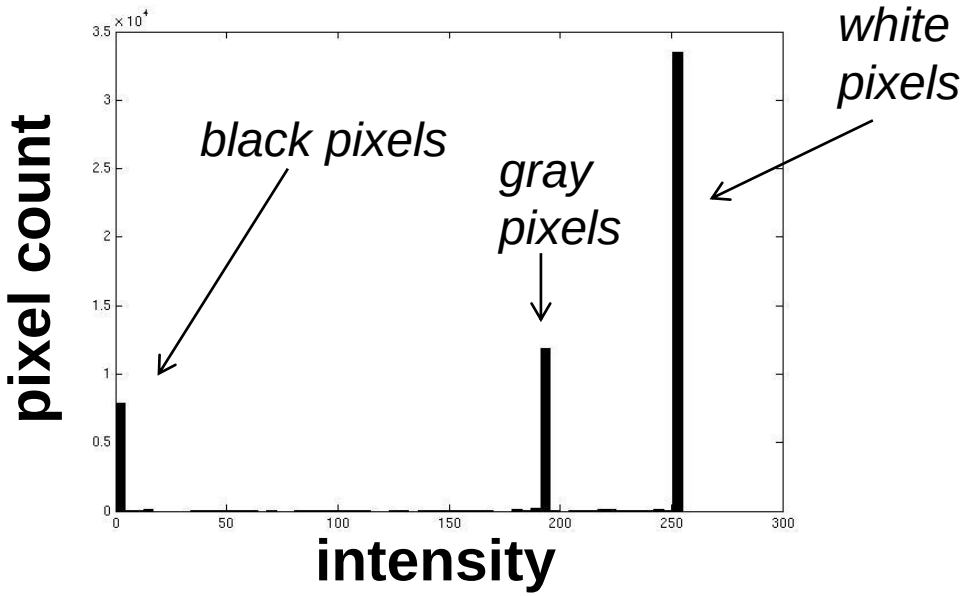
Clustering algorithms

- In depth
  - K-means (iterate between finding centers and assigning points)
- Briefly
  - Mean-shift (find modes in the data)
  - Hierarchical clustering (start with all points in separate clusters and merge)

Image segmentation: toy example



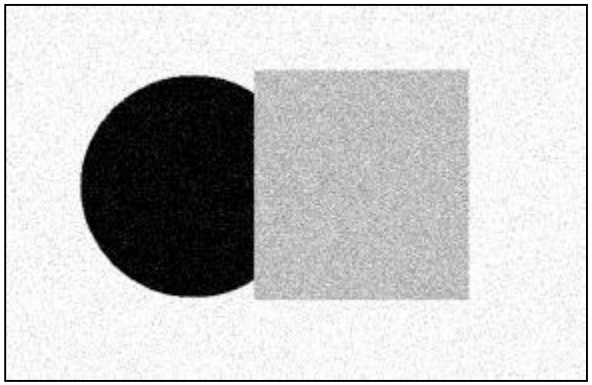
input image



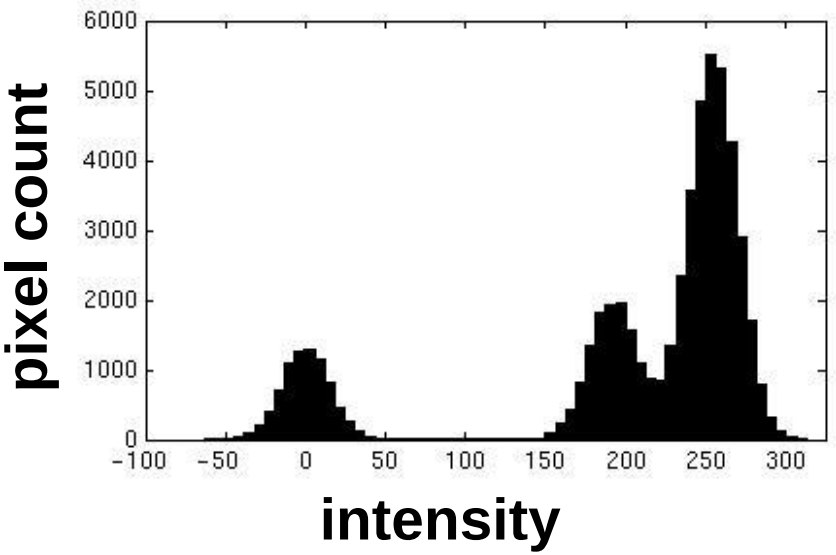
- These intensities define the three groups.
- We could label every pixel in the image according to which of these primary intensities it is.
  - i.e., *segment* the image based on the intensity feature.
- What if the image isn't quite so simple?

Source: K. Grauman

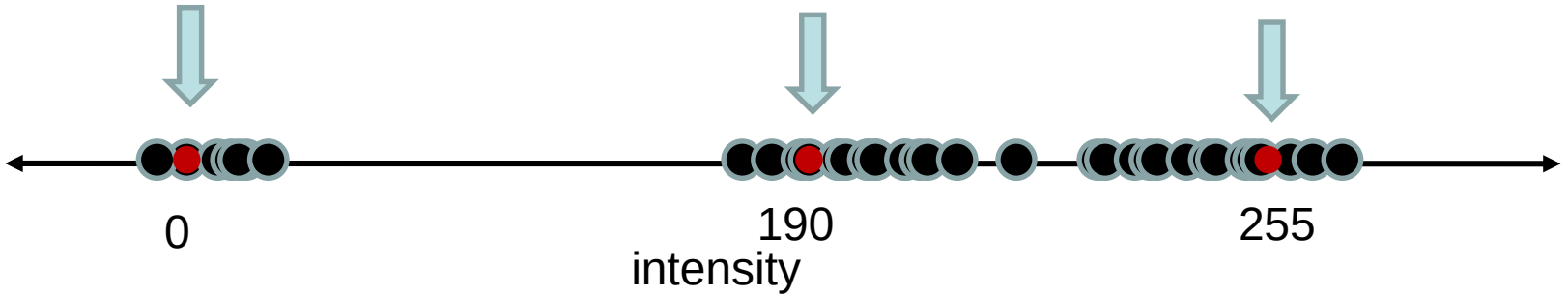
- Now how to determine the three main intensities that define our groups?
- We need to ***cluster.***

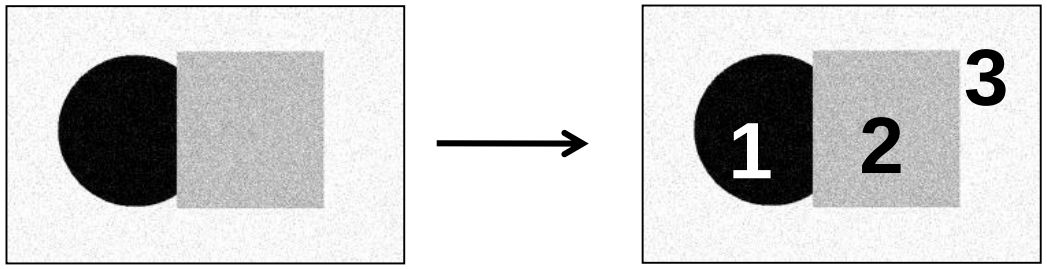


input image



Source: K. Grauman





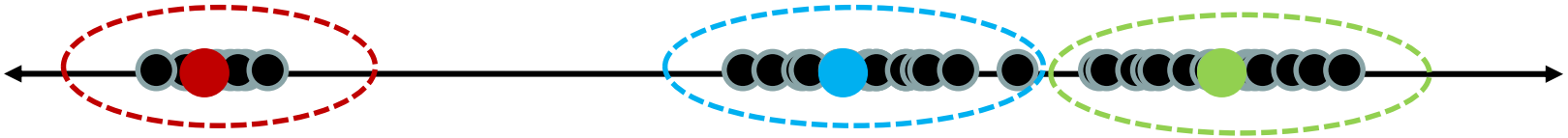
- Goal: choose three “centers” as the representative intensities, and label every pixel according to which of these centers it is nearest to.
- Best cluster centers are those that minimize SSD between all points and their nearest cluster center c_i:

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

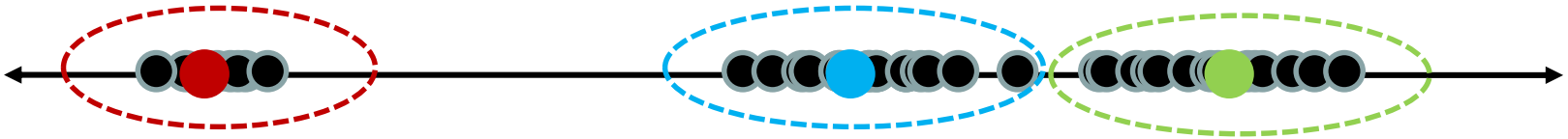
Source: K. Grauman

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.

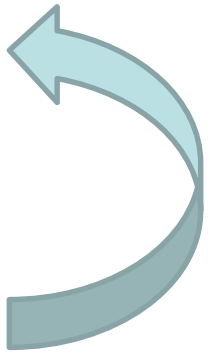


Source: K. Grauman

K-means clustering

- Basic idea: randomly initialize the *k* cluster centers, and iterate between the two steps we just saw.

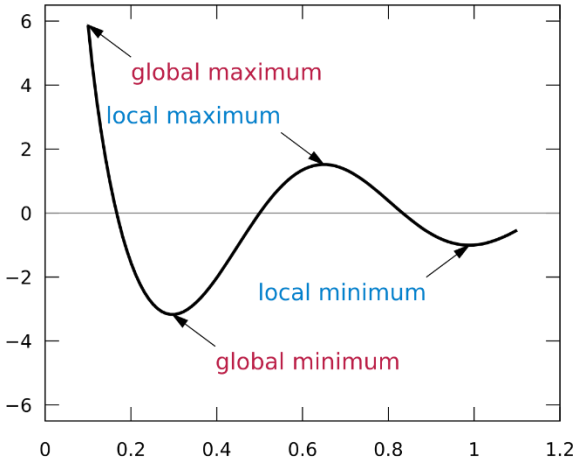
  1. *Randomly* initialize the cluster centers, c_1, ..., c_K
  2. *Given cluster centers*, determine points in each cluster
     - For each point p, find the closest c_i. Put p into cluster i
  3. *Given points in each cluster*, solve for c_i
     - Set c_i to be the mean of points in cluster i
  4. If c_i have changed, repeat Step 2



Properties

- Will always converge to *some* solution
- Can be a "local minimum" of objective:

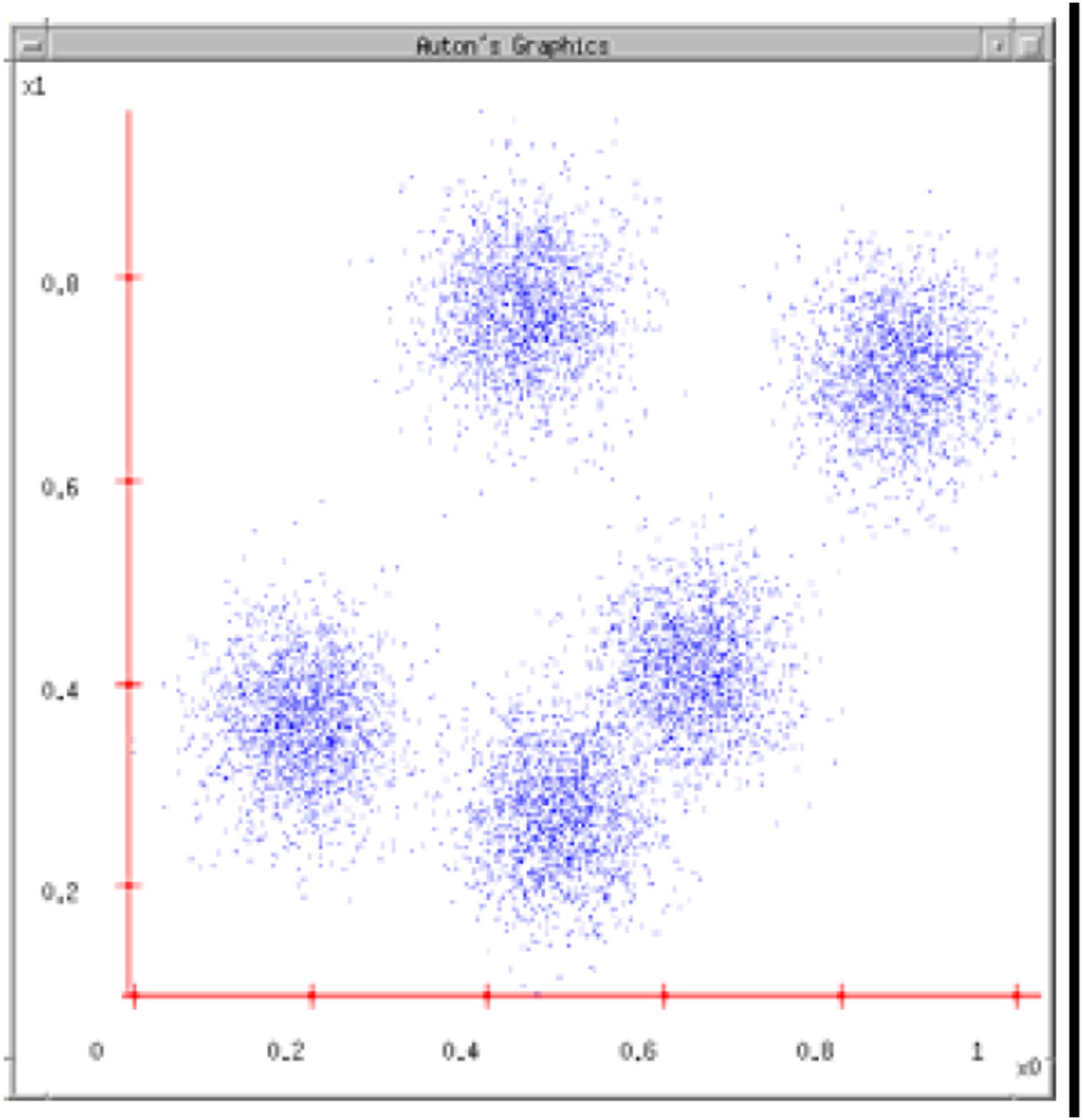
$$\sum_{\text{clusters } i} \quad \sum_{\text{points p in cluster } i} \|p - c_i\|^2$$



Slide: Steve Seitz, image: Wikipedia

K-means

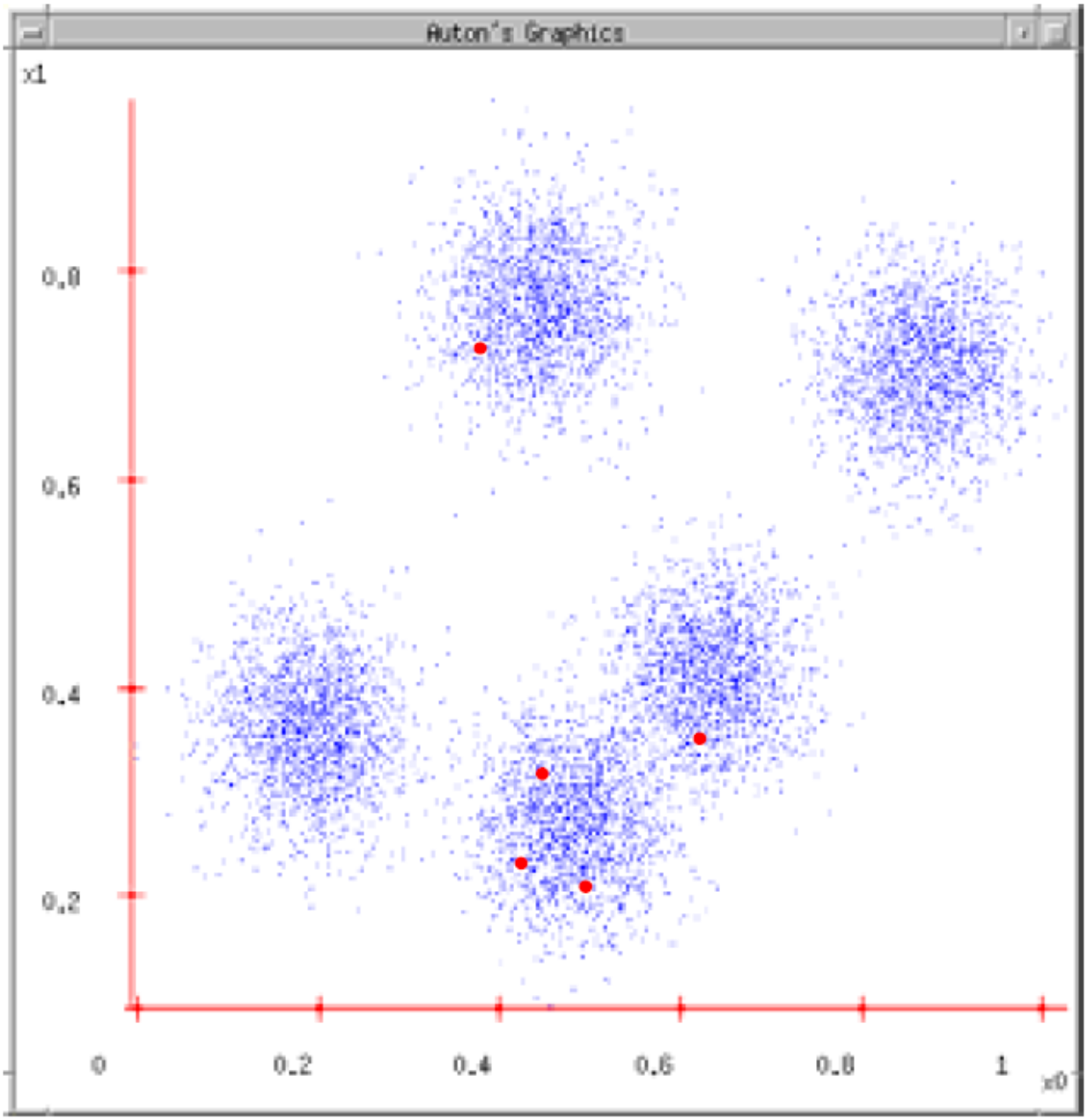
1. Ask user how many clusters they'd like. *(e.g. k=5)*



Source: A. Moore

K-means

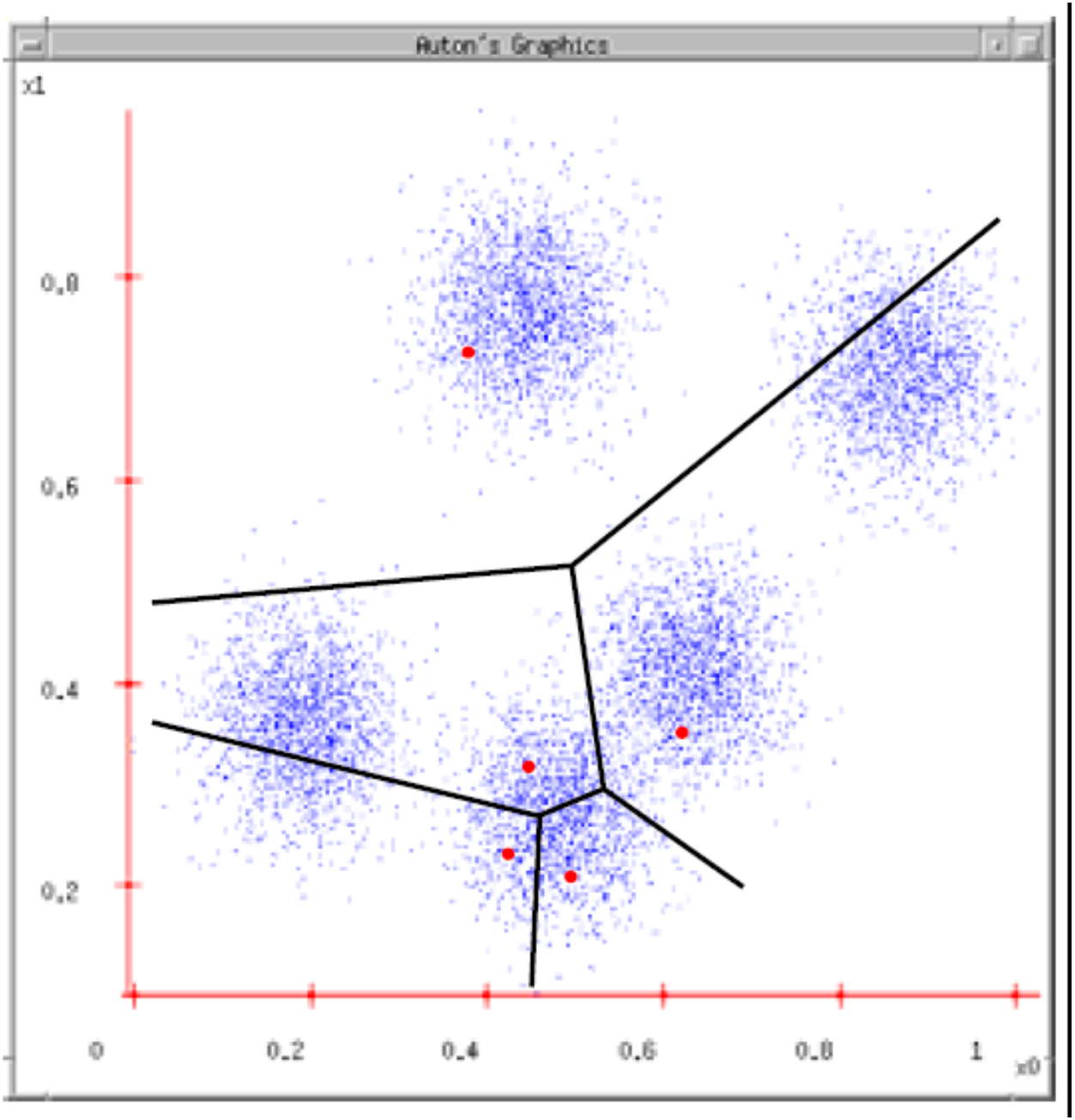
1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations



Source: A. Moore

K-means

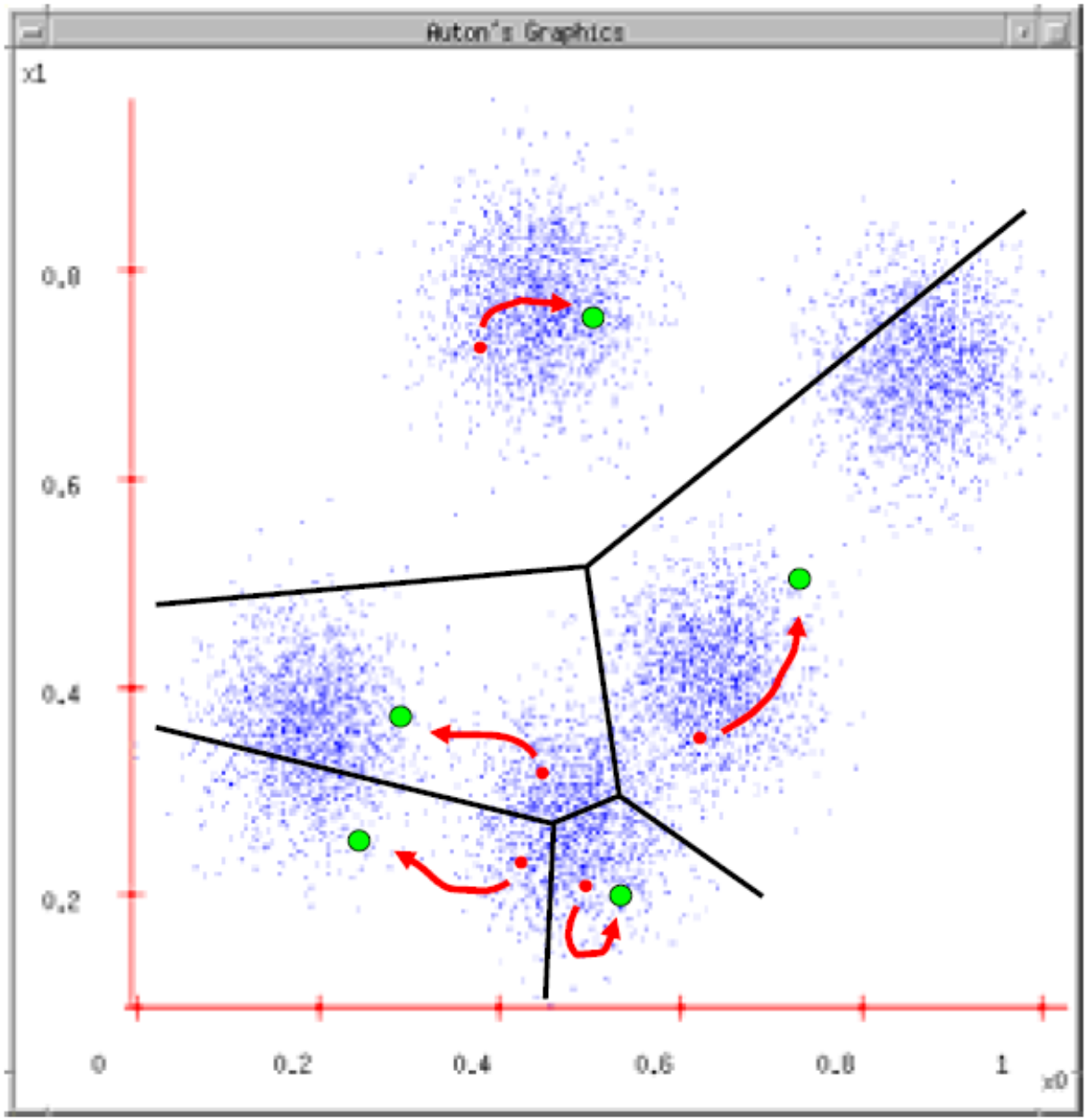
1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to. (Thus each Center "owns" a set of datapoints)



Source: A. Moore

K-means

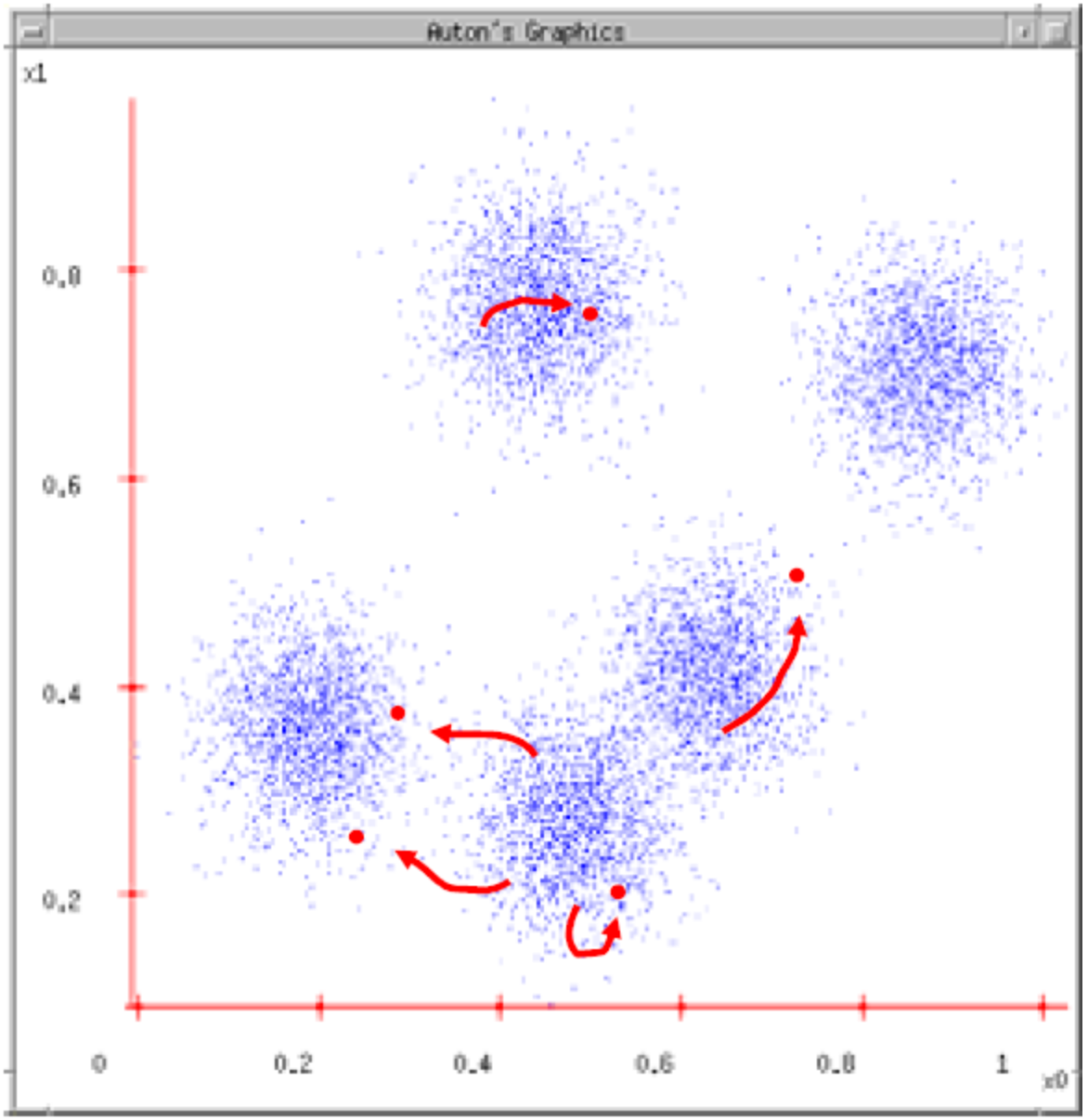
1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds the centroid of the points it owns



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 converges to a local minimum

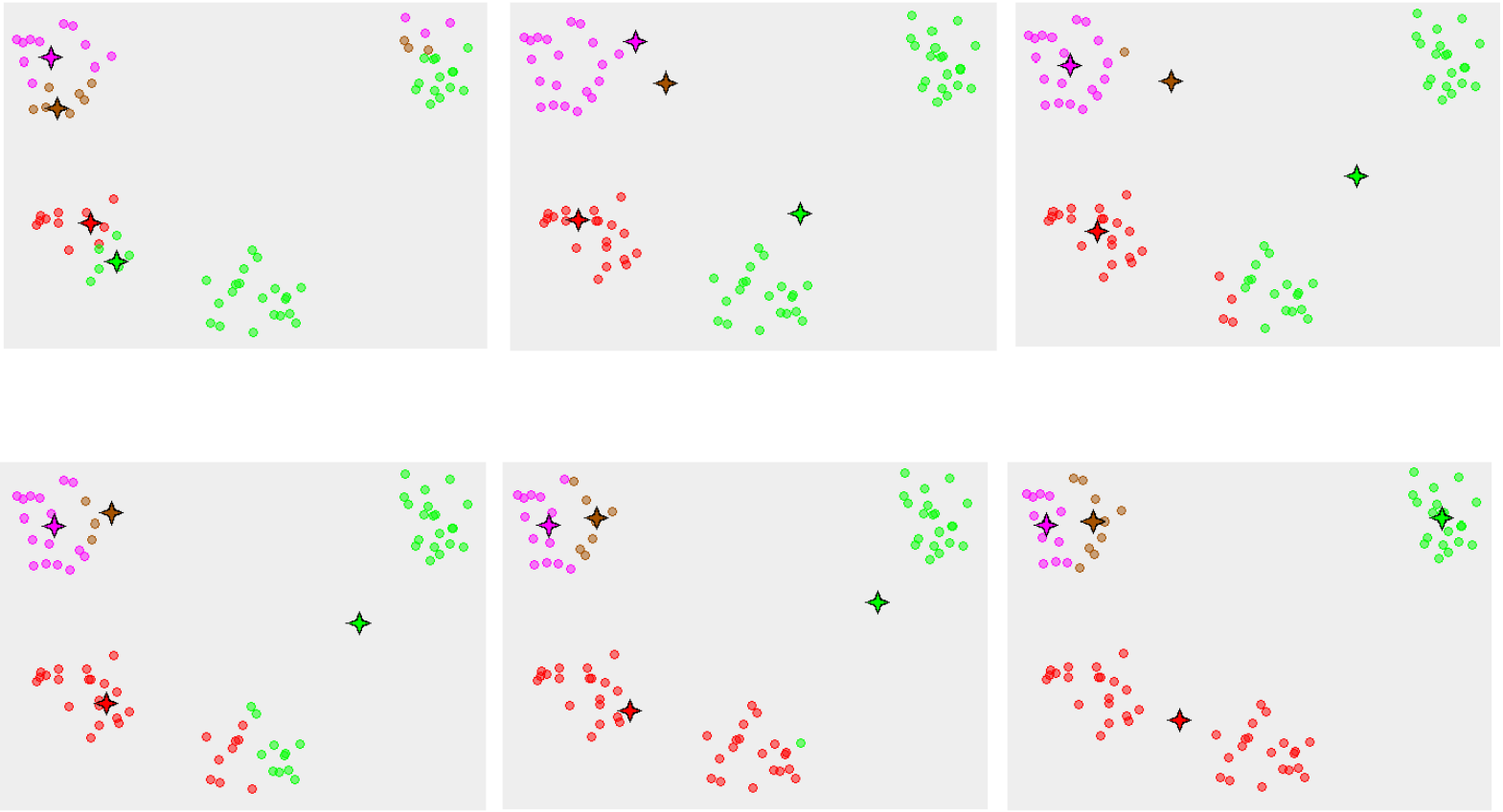


Figure from Wikipedia

K-means clustering

- Visualization

https://www.naftaliharris.com/blog/visualizing-k-means-clustering/

- Java demo

http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/AppletKM.html

- Matlab demo

http://www.cs.pitt.edu/~kovashka/cs1699_fa15/kmeans_demo.m

Time Complexity

- Let n = number of instances, d = dimensionality of the features, k = number of clusters
- Assume computing distance between two instances is O(d)
- Reassigning clusters:
  - O(kn) distance computations, or O(knd)
- Computing centroids:
  - Each instance vector gets added once to a centroid: O(nd)
- Assume these two steps are each done once for a fixed number of iterations I: O($Iknd$)
  - Linear in all relevant factors

Adapted from Ray Mooney

Another way of writing objective

- K-means: Let r_{nk} = 1 if instance *n* belongs to cluster *k*, 0 otherwise

$$\boldsymbol{\mu}_k = \frac{\sum_n r_{nk}\mathbf{x}_n}{\sum_n r_{nk}}$$

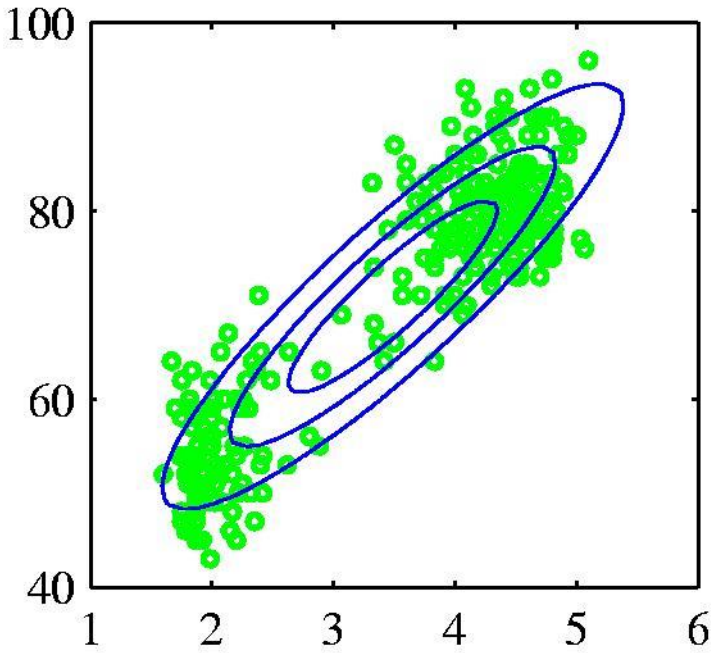
$$J = \sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk}\|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2$$

- K-medoids (more general distances):

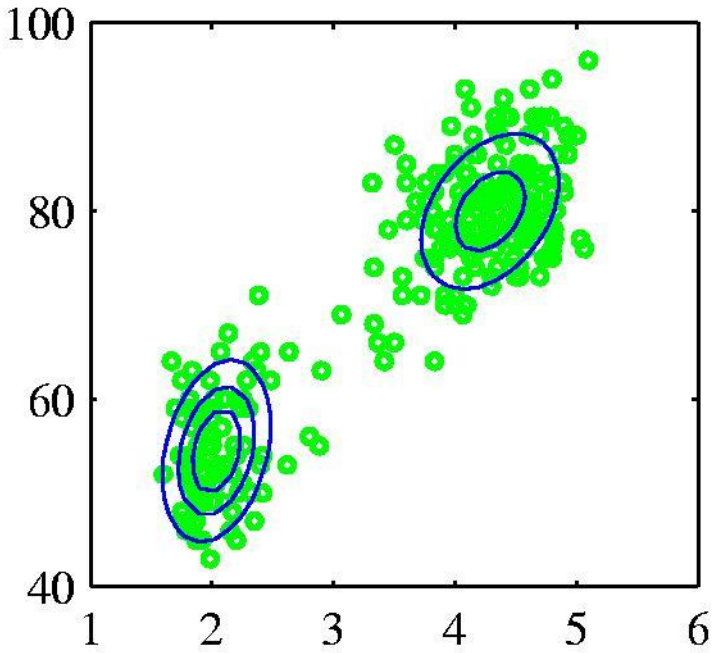
$$\widetilde{J} = \sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk}\mathcal{V}(\mathbf{x}_n, \boldsymbol{\mu}_k)$$

Probabilistic version: Mixtures of Gaussians

- Old Faithful data set

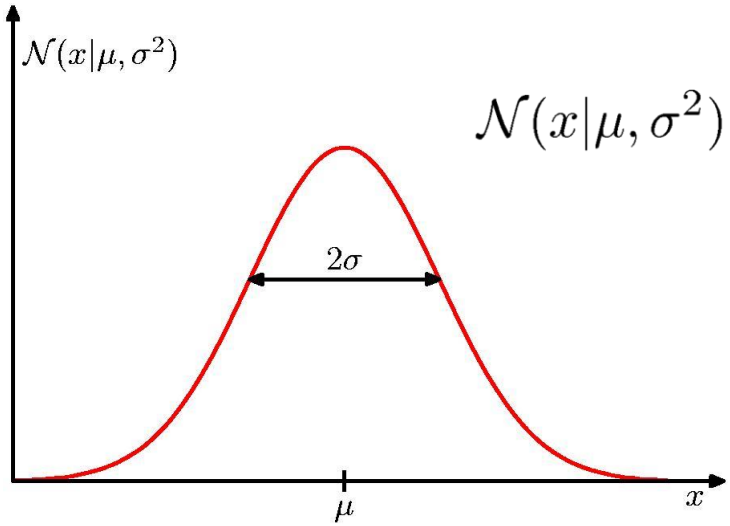


Single Gaussian

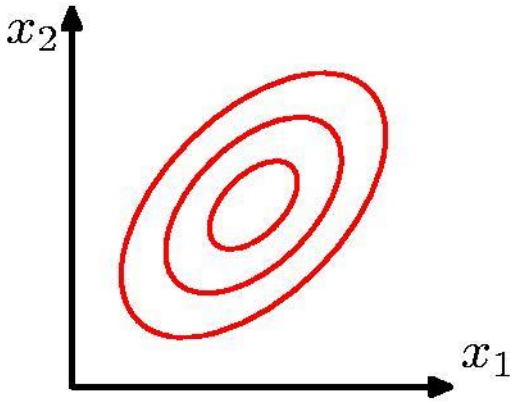


Mixture of two Gaussians

Review: Gaussian Distribution



$$\mathcal{N}(x|\mu,\sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\}$$



$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\}$$

Mixtures of Gaussians

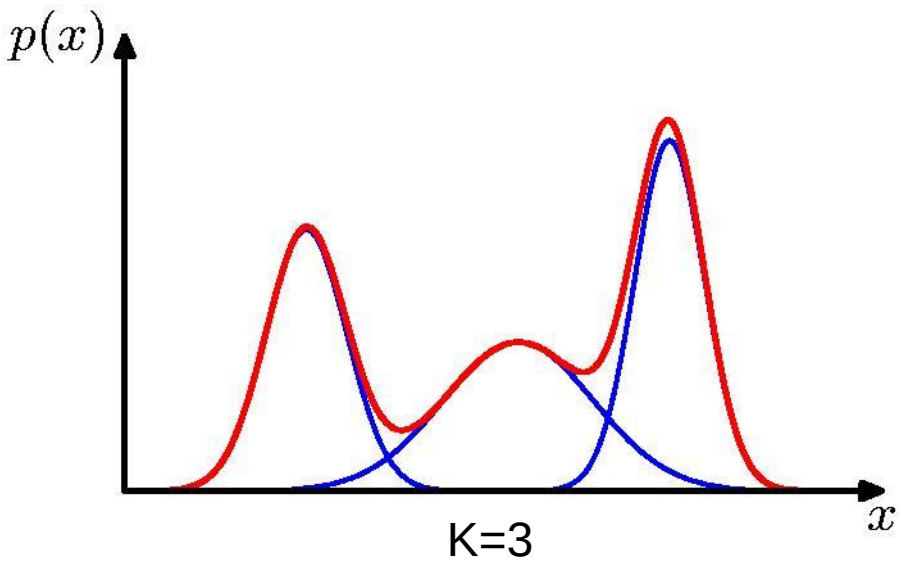
- Combine simple models into a complex model:

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

Component

Mixing coefficient

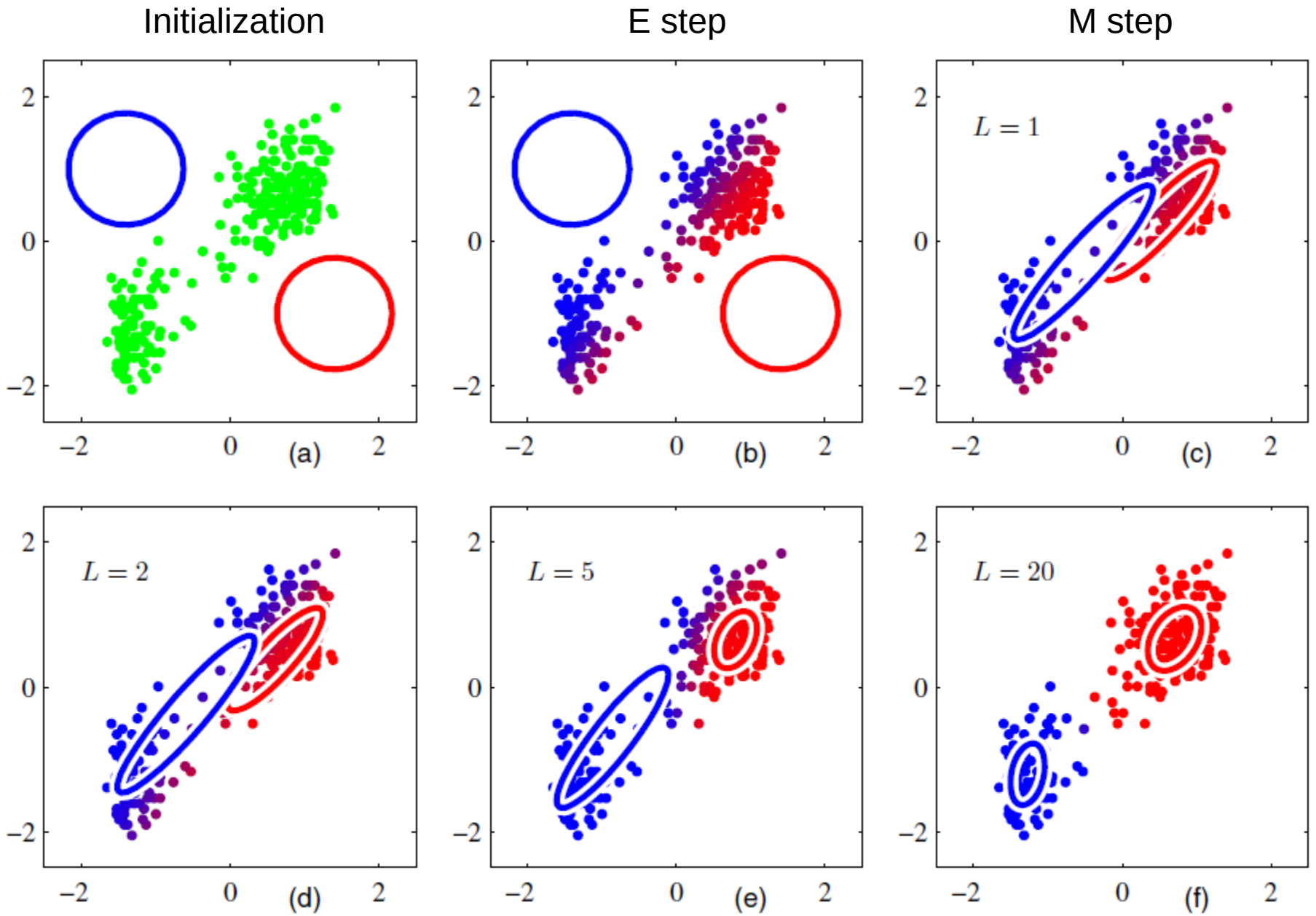
$$\forall k : \pi_k \geqslant 0 \qquad \sum_{k=1}^{K} \pi_k = 1$$



K=3

- Find parameters through EM (Expectation Maximization) algorithm

Adapted from Chris Bishop



Figures from Chris Bishop

Segmentation as clustering

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

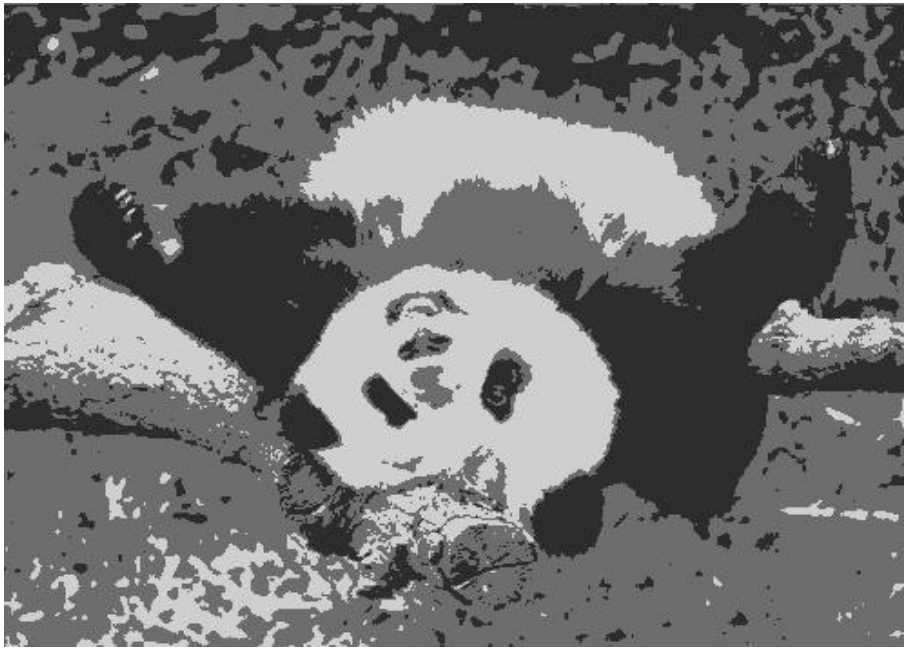
Grouping pixels based on **intensity** similarity

Feature space: intensity value (1-d)

Source: K. Grauman

K=2

K=3



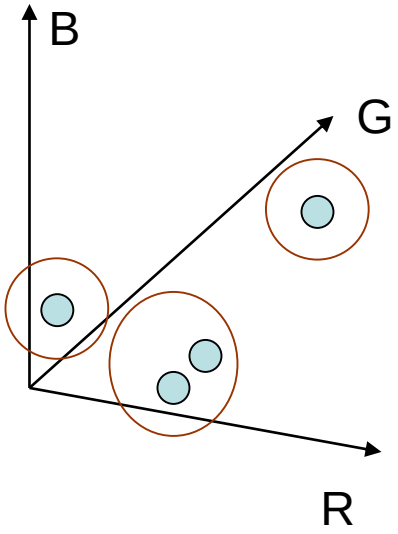
quantization of the feature space; segmentation label map

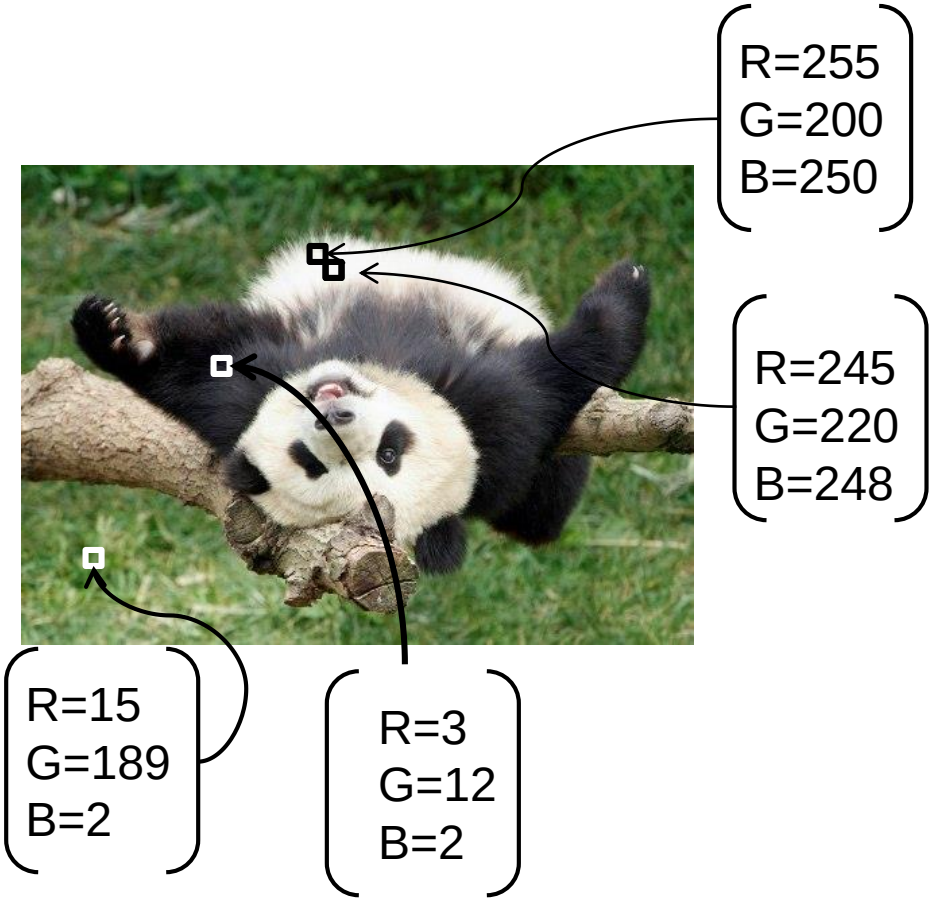
Source: K. Grauman

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

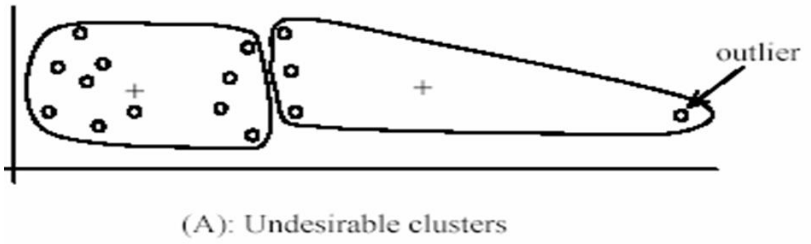
K-means: pros and cons

Pros

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

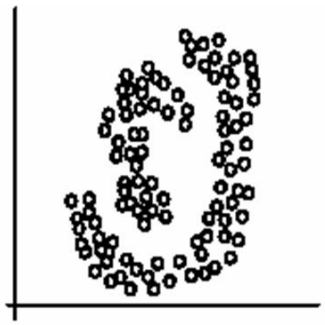
Cons/issues

- Setting k?
  - One way: silhouette coefficient
- Sensitive to initial centers
  - Use heuristics or output of another method
- Sensitive to outliers
- Detects spherical clusters

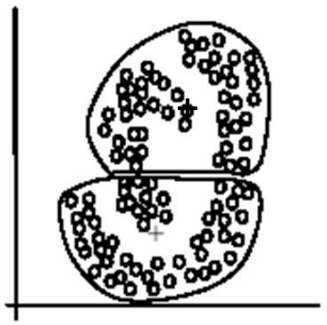


(A): Undesirable clusters

(B): Ideal clusters



(A): Two natural clusters



(B): *k*-means clusters

Adapted from K. Grauman

Clustering algorithms

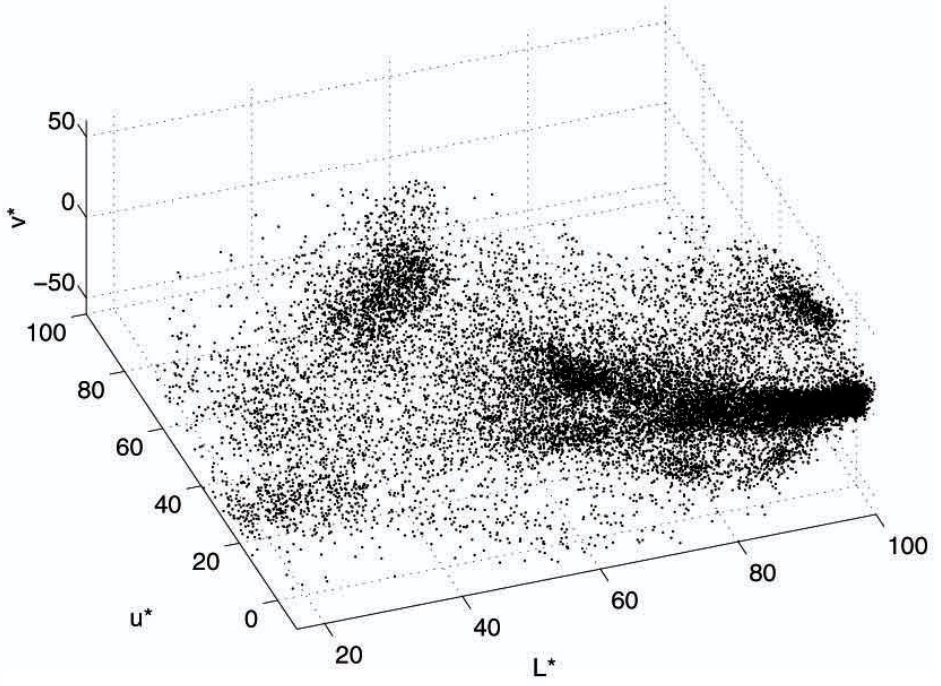
- In depth
  - K-means (iterate between finding centers and assigning points)
- Briefly
  - Mean-shift (find modes in the data)
  - Hierarchical clustering (start with all points in separate clusters and merge)

Mean shift algorithm

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

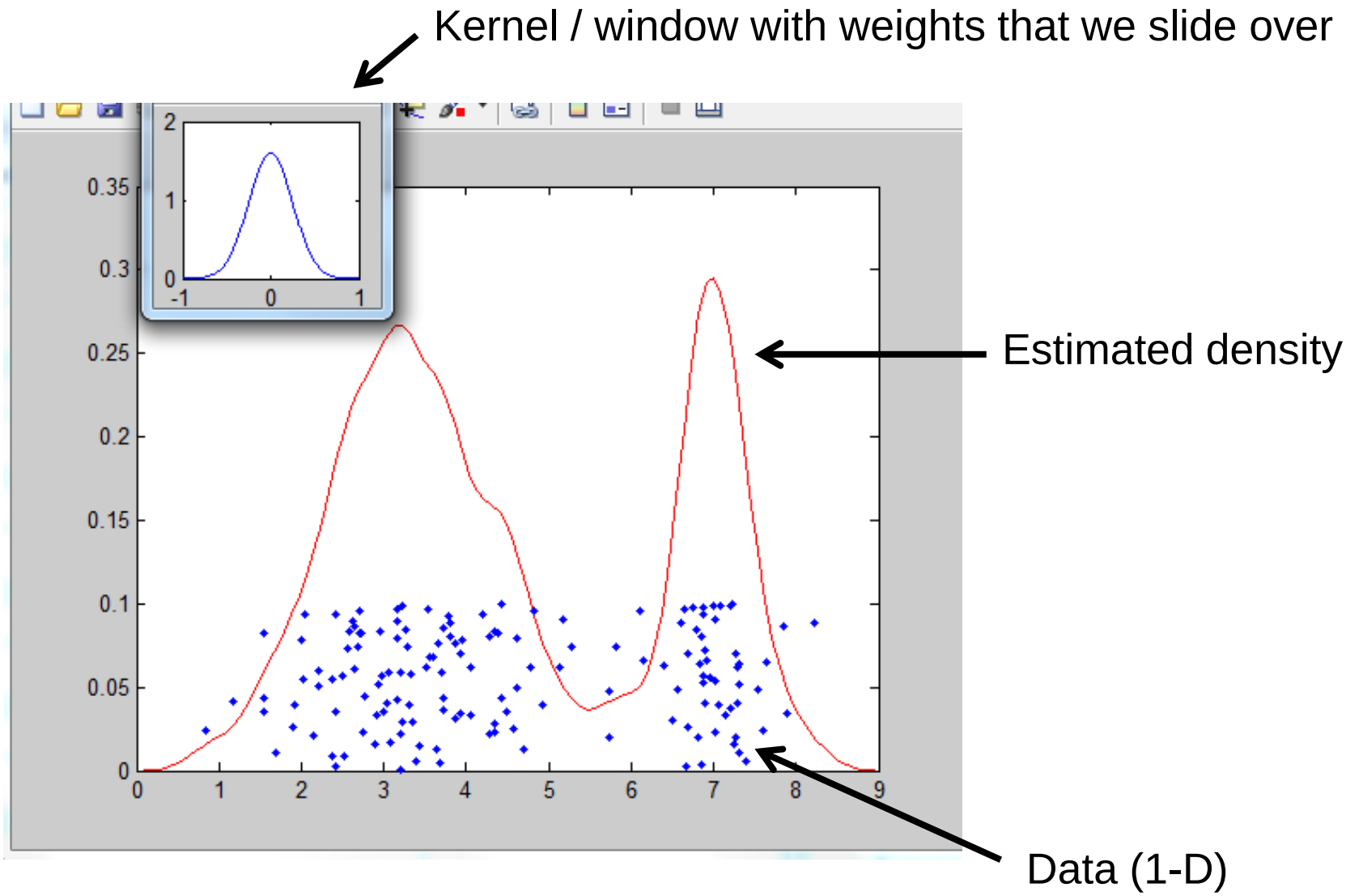
image

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



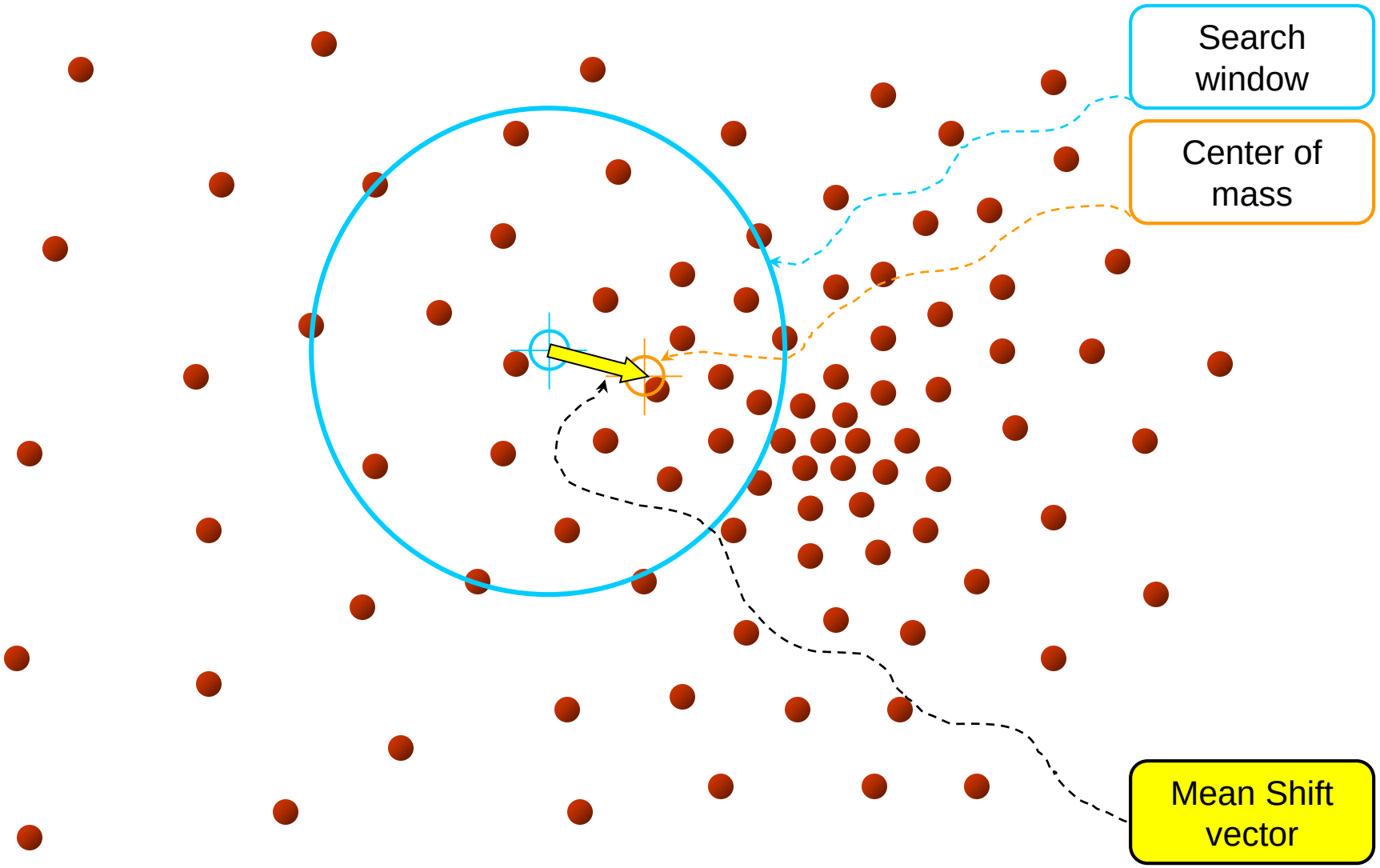
Source: K. Grauman

Density estimation



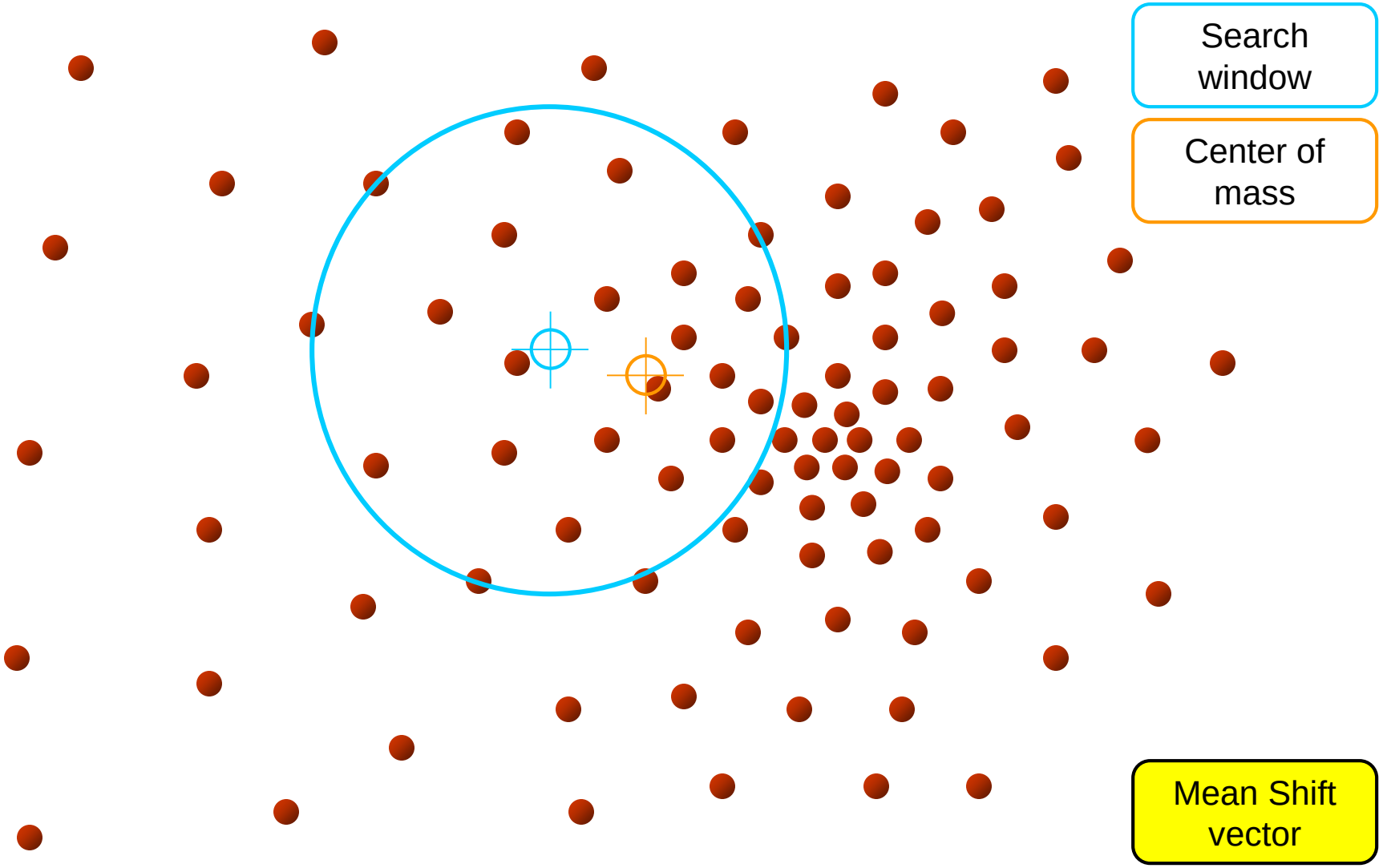
Adapted from D. Hoiem

Mean shift



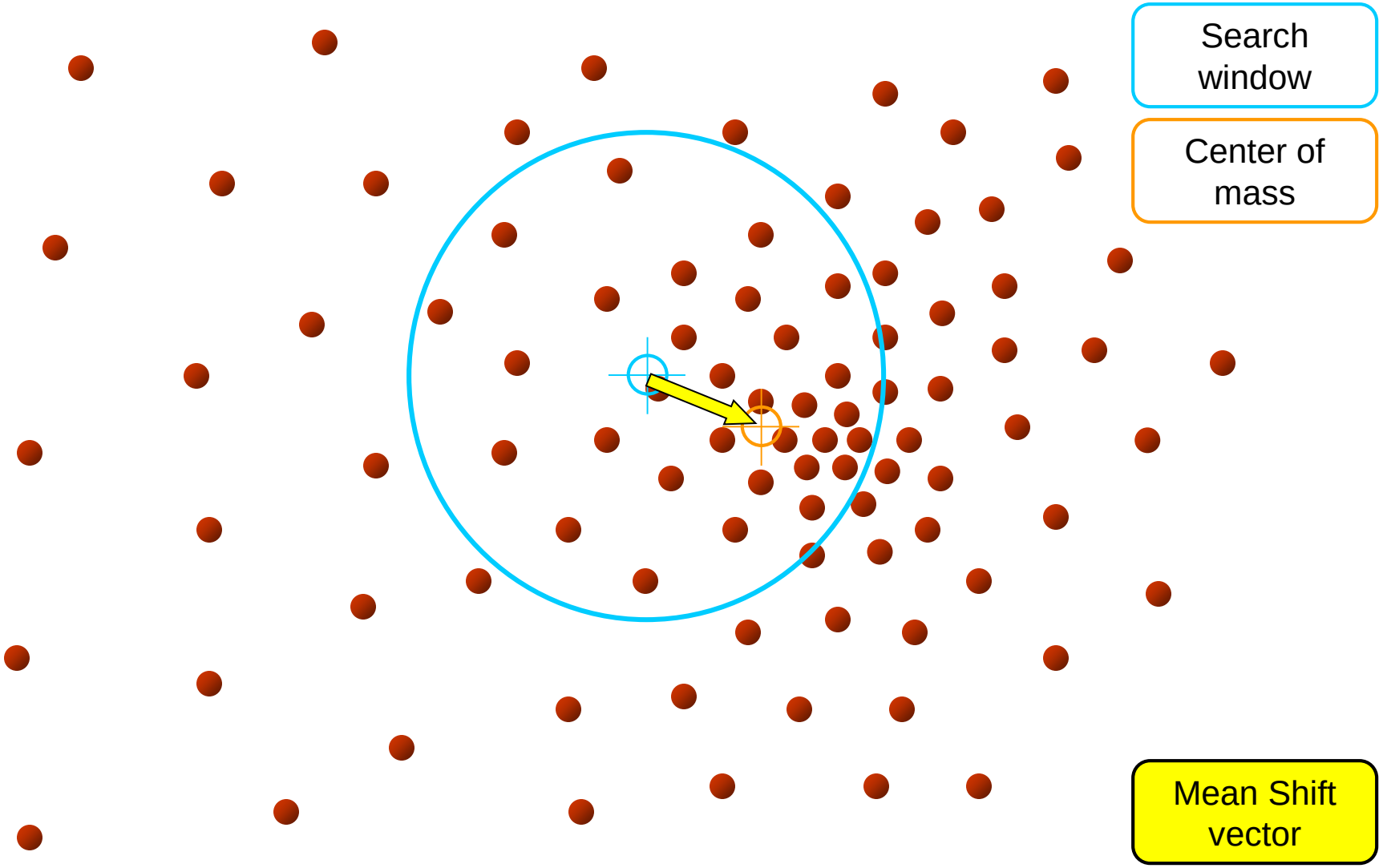
Slide by Y. Ukrainitz & B. Sarel

Mean shift



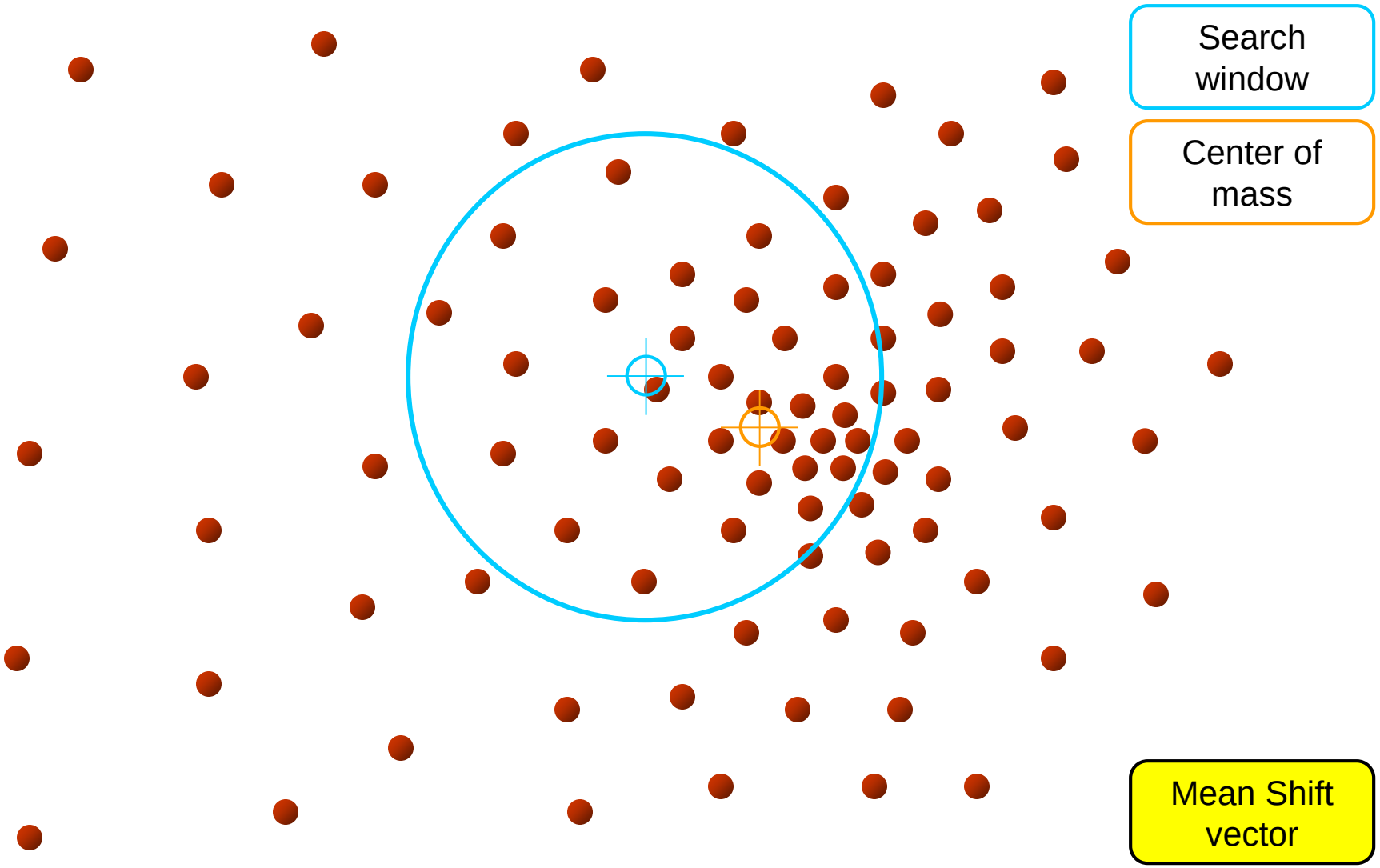
Slide by Y. Ukrainitz & B. Sarel

Mean shift



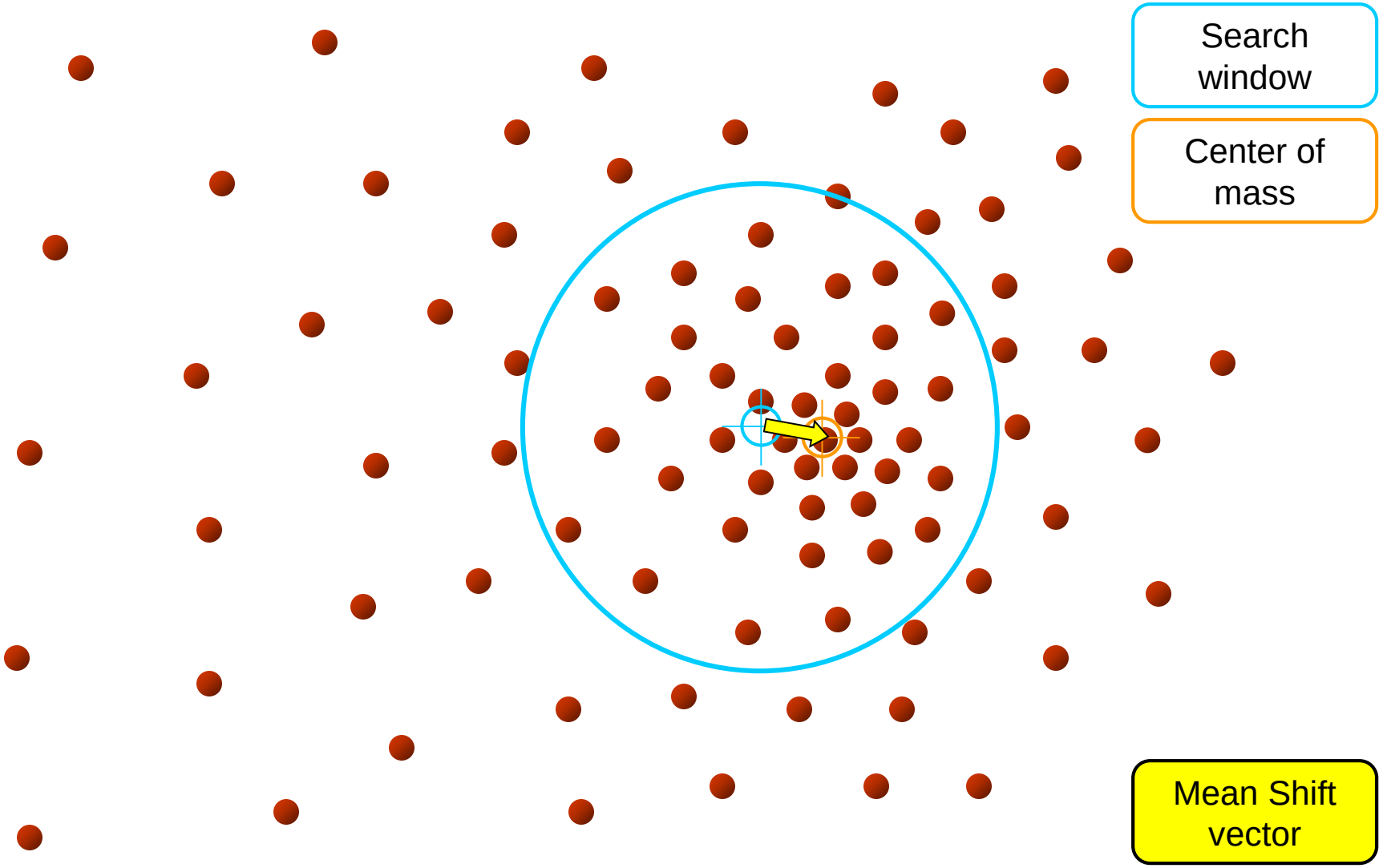
Slide by Y. Ukrainitz & B. Sarel

Mean shift



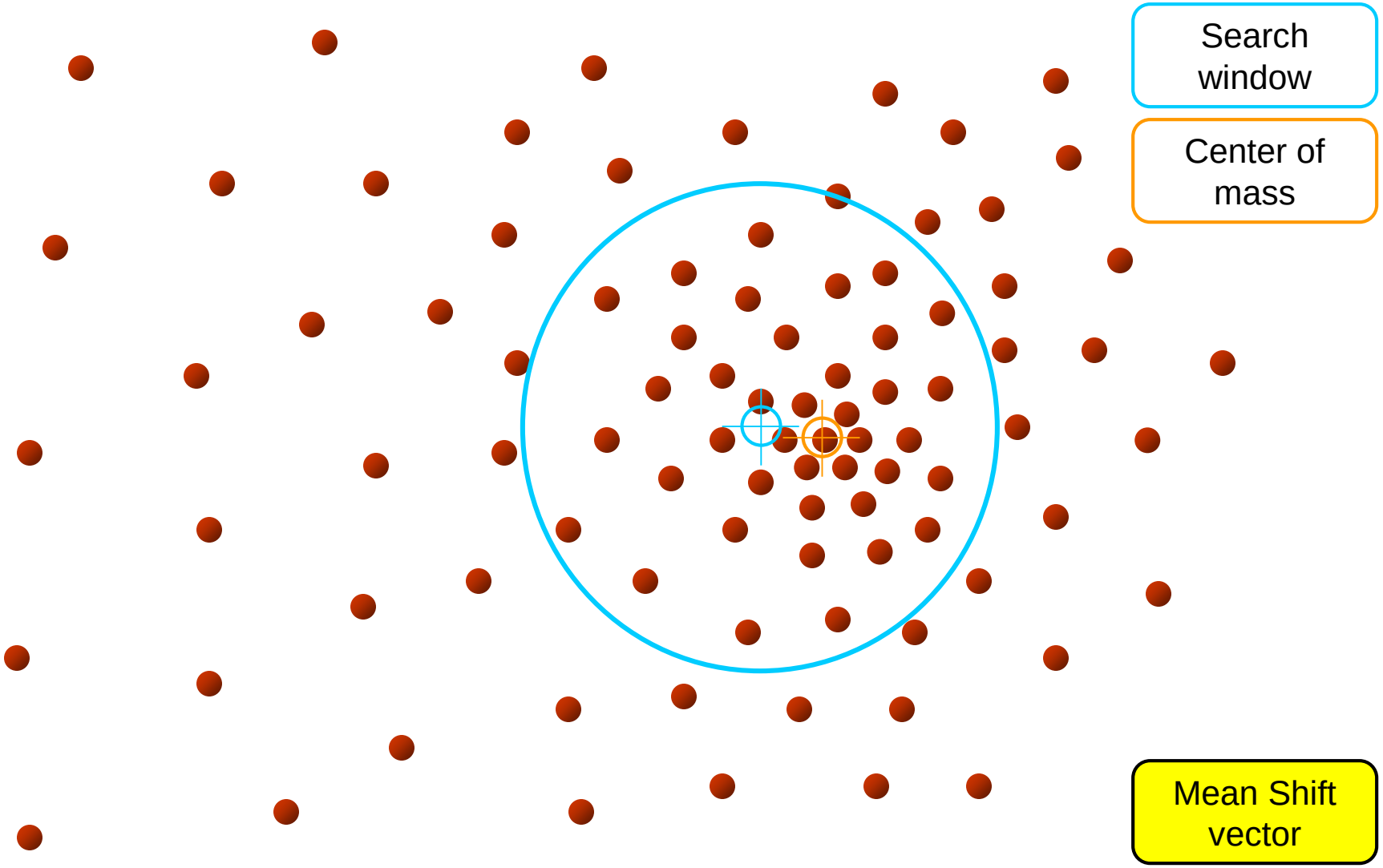
Slide by Y. Ukrainitz & B. Sarel

Mean shift



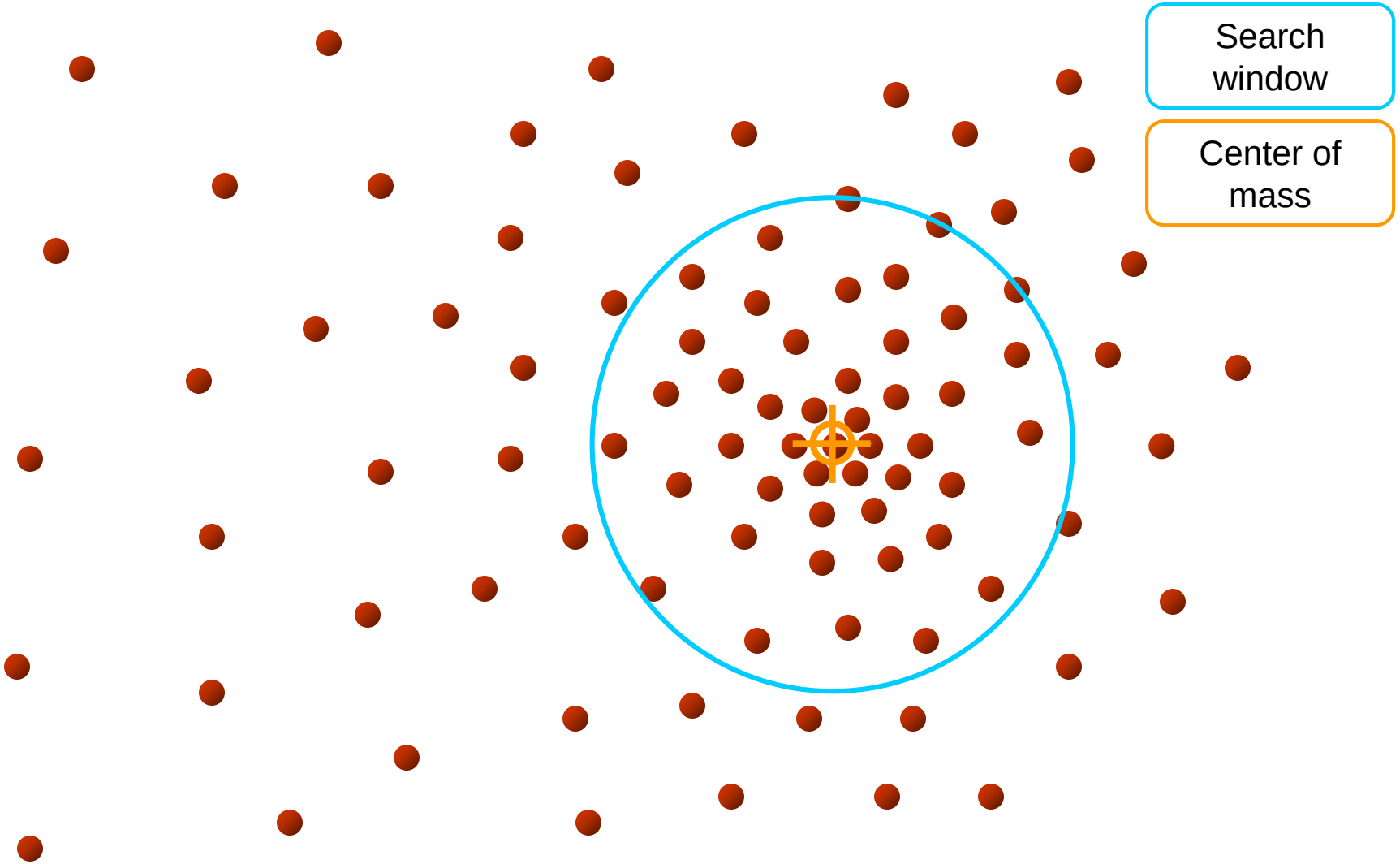
Slide by Y. Ukrainitz & B. Sarel

Mean shift



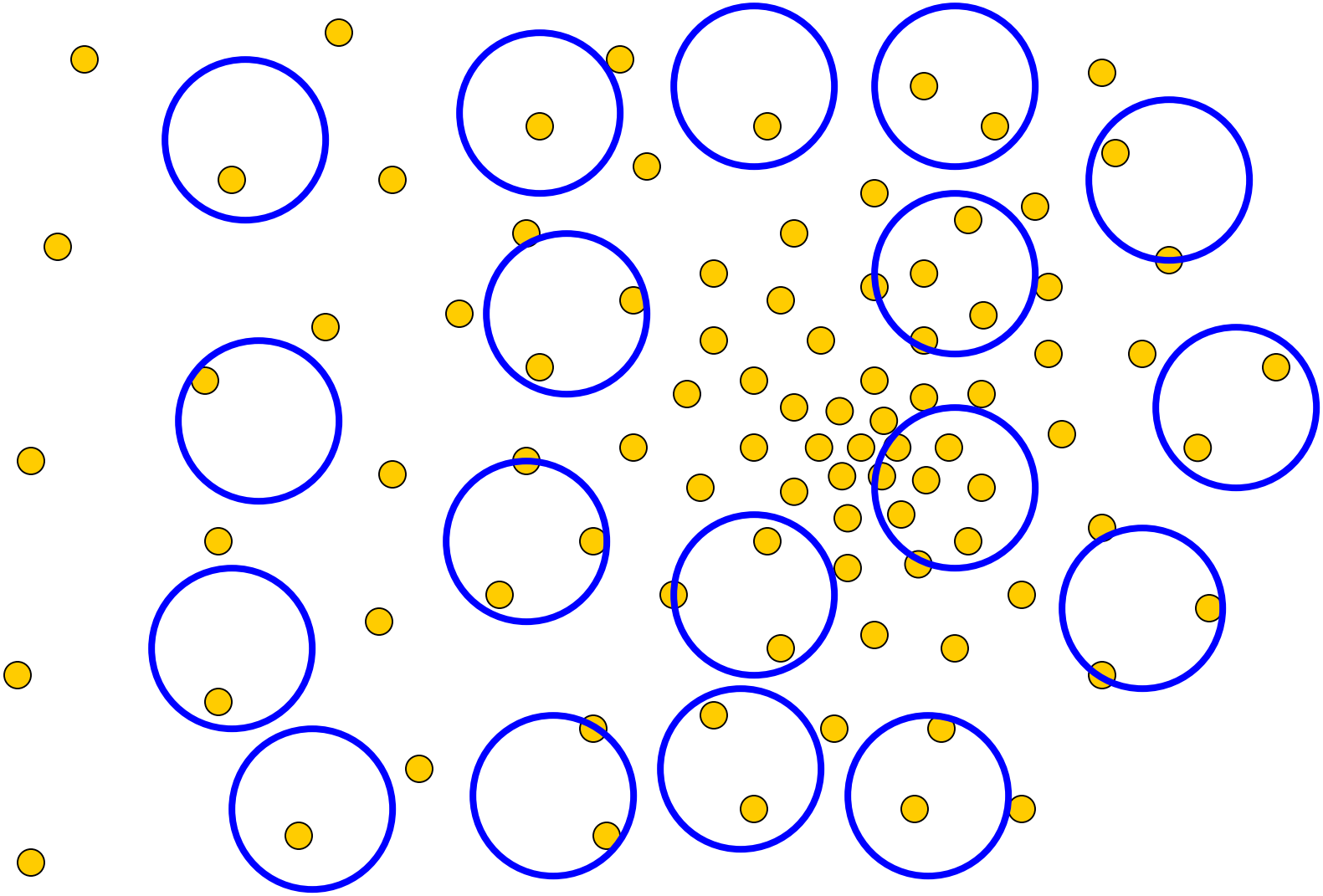
Slide by Y. Ukrainitz & B. Sarel

Mean shift



Slide by Y. Ukrainitz & B. Sarel

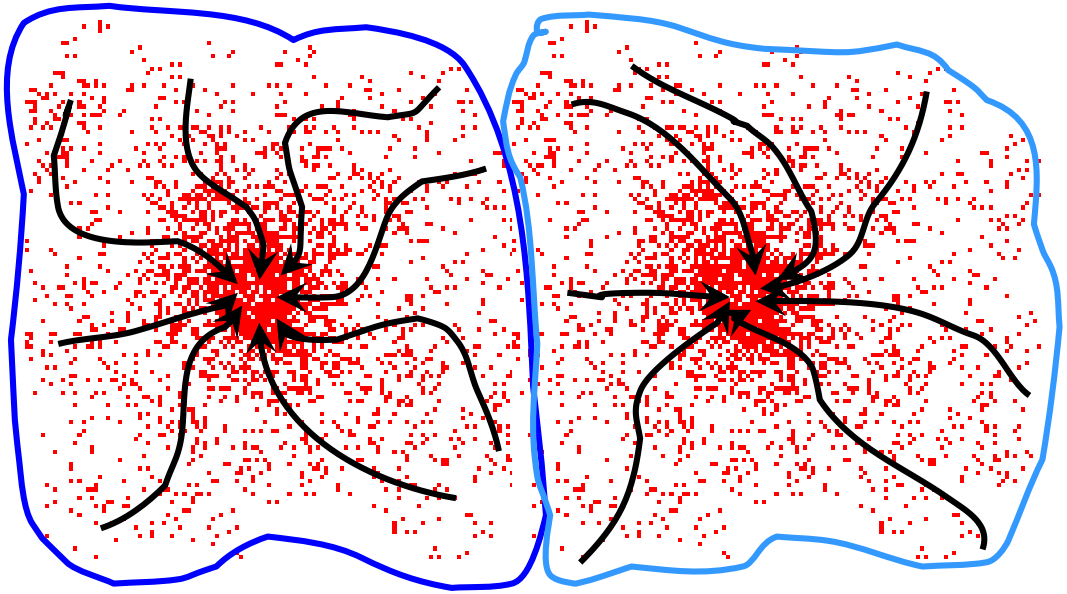
Points in same cluster converge



Source: D. Hoiem

Mean shift clustering

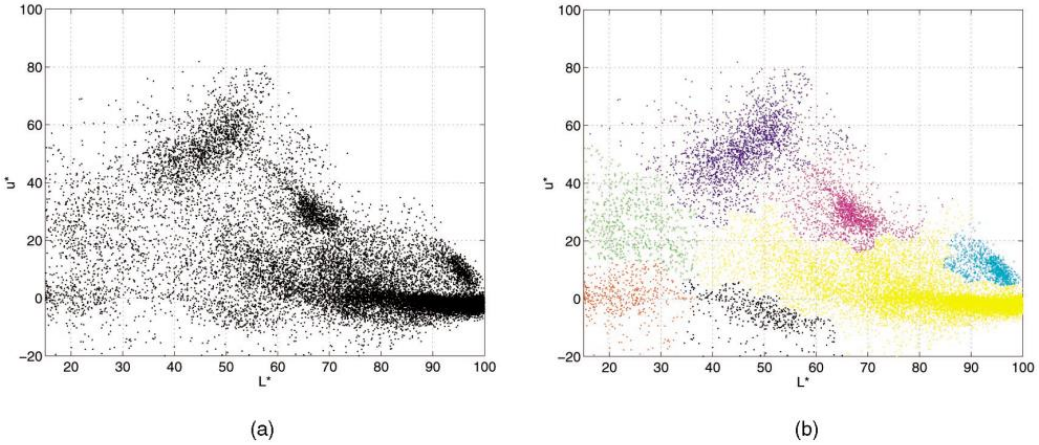
- Cluster: all data points in the attraction basin of a mode
- Attraction basin: the region for which all trajectories lead to the same mode

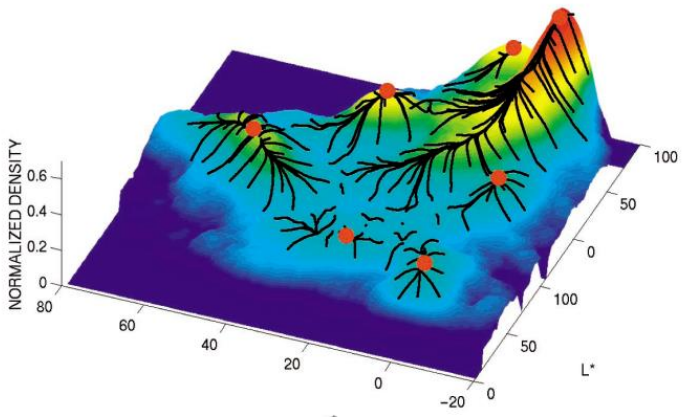


Slide by Y. Ukrainitz & B. Sarel

Mean shift clustering/segmentation

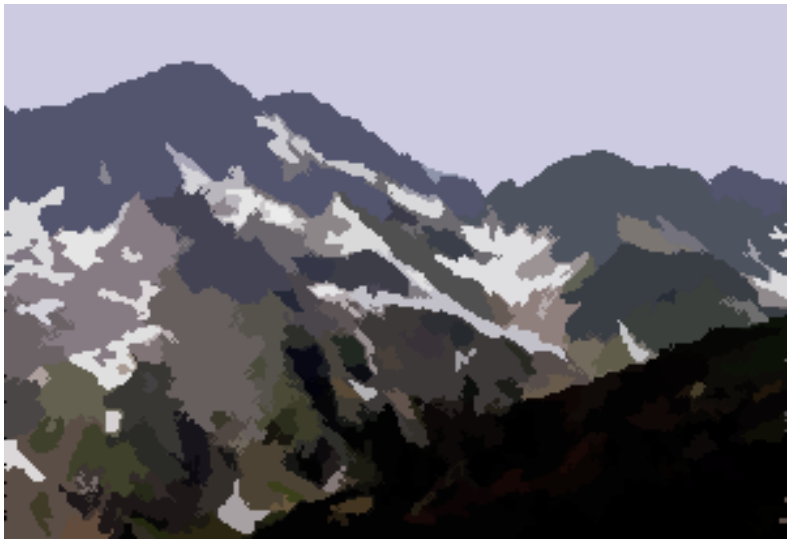
- Compute features for each point (intensity, word counts, etc)
- Initialize windows at individual feature points
- Perform mean shift for each window until convergence
- Merge windows that end up near the same “peak” or mode





Source: D. Hoiem

Mean shift segmentation results



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

Mean shift: Pros and cons

- Pros:
  - Does not assume shape on clusters
  - Robust to outliers
- Cons:
  - Need to choose window size
  - Quadratic in the number of samples

Hierarchical Agglomerative Clustering (HAC)

- Assumes a *similarity function* for determining the similarity of two instances.
- Starts with all instances in separate clusters and then repeatedly joins the two clusters that are most similar until there is only one cluster.
- The history of merging forms a binary tree or hierarchy.

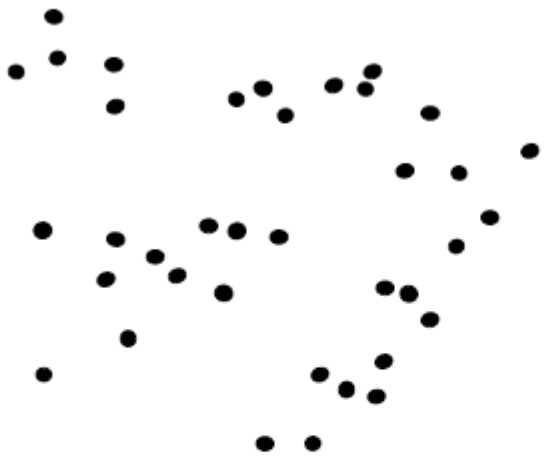
Slide credit: Ray Mooney

HAC Algorithm

Start with all instances in their own cluster.
Until there is only one cluster:
    Among the current clusters, determine the two clusters, c_i and c_j, that are most similar.
    Replace c_i and c_j with a single cluster $c_i \cup c_j$

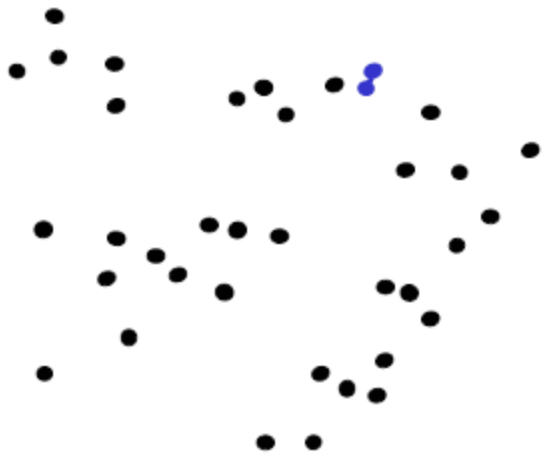
Slide credit: Ray Mooney

Agglomerative clustering



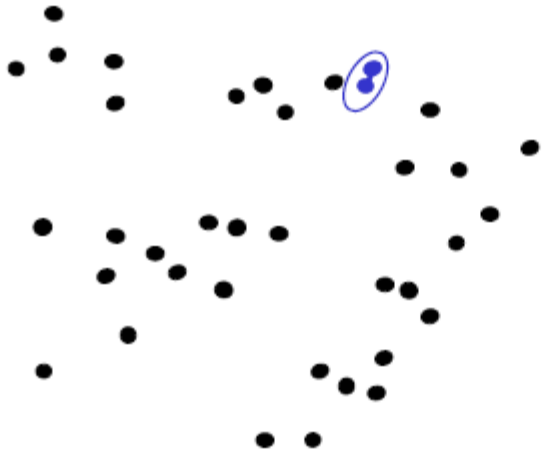
1. Say "Every point is its own cluster"

Agglomerative clustering



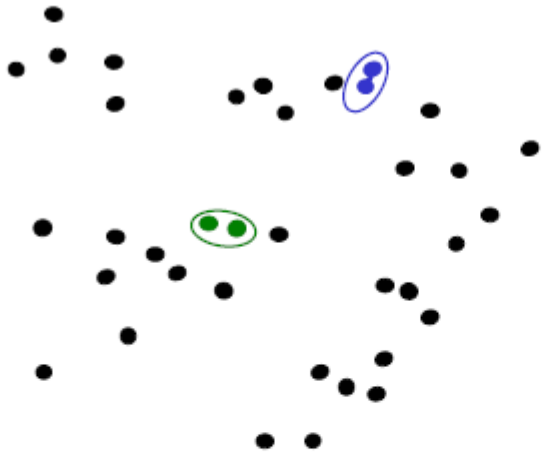
1. Say "Every point is its own cluster"
2. Find "most similar" pair of clusters

Agglomerative clustering



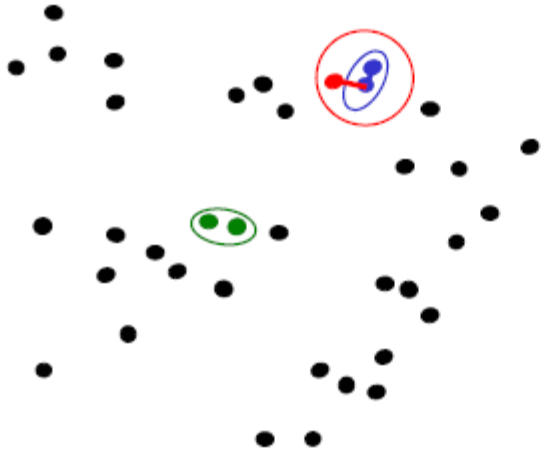
1. Say "Every point is its own cluster"
2. Find "most similar" pair of clusters
3. Merge it into a parent cluster

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

Agglomerative clustering

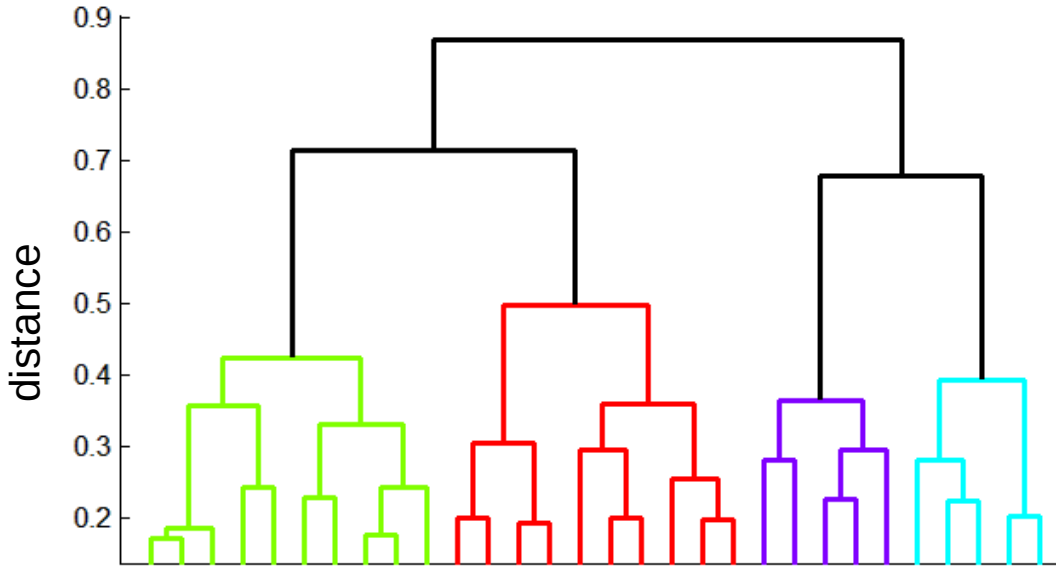


1. Say "Every point is its own cluster"
2. Find "most similar" pair of clusters
3. Merge it into a parent cluster
4. Repeat

Agglomerative clustering

How many clusters?

- Clustering creates a dendrogram (a tree)
- To get final clusters, pick a threshold
  - max number of clusters or
  - max distance within clusters (y axis)



Cluster Similarity

- How to compute similarity of two clusters each possibly containing multiple instances?
  - Single Link: Similarity of *two most similar* members.

$$sim(c_i, c_j) = \max_{x \in c_i, y \in c_j} sim(x, y)$$

  - Complete Link: Similarity of *two least similar* members.

$$sim(c_i, c_j) = \min_{x \in c_i, y \in c_j} sim(x, y)$$

  - Group Average: Average similarity between members.

Adapted from Ray Mooney

Agglomerative clustering: pros & cons

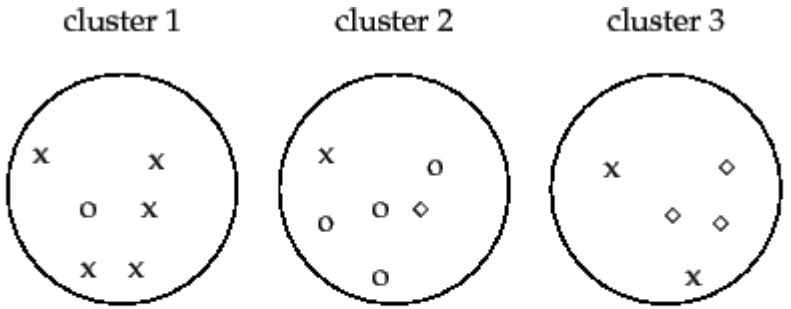
- Pros
  - Deterministic
  - Flexible (can use any cutoff to declare clusters)
  - Interpretable?
- Cons
  - Some variants sensitive to noise
  - Quadratic in the number of samples

How to evaluate clustering?

- Might depend on application
- Purity $$\text{purity}(\Omega, \mathbb{C}) = \frac{1}{N} \sum_k \max_j |\omega_k \cap c_j|$$

where $\Omega = \{\omega_1, \omega_2, \ldots, \omega_K\}$ is the set of clusters

and $\mathbb{C} = \{c_1, c_2, \ldots, c_J\}$ is the set of classes



► **Figure 16.1** Purity as an external evaluation criterion for cluster quality. Majority class and number of members of the majority class for the three clusters are: x, 5 (cluster 1); o, 4 (cluster 2); and ⋄, 3 (cluster 3). Purity is $(1/17) \times (5+4+3) \approx 0.71$.

http://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-clustering-1.html

Summary of Clustering Strategies

- K-means
  - Iteratively re-assign points to the nearest cluster center
- Mean-shift clustering
  - Estimate modes
- Agglomerative clustering
  - Start with each point as its own cluster and iteratively merge the closest clusters

Dimensionality reduction

- Motivation
- Principal Component Analysis (PCA)
- Applications
- Other methods for dimensionality reduction

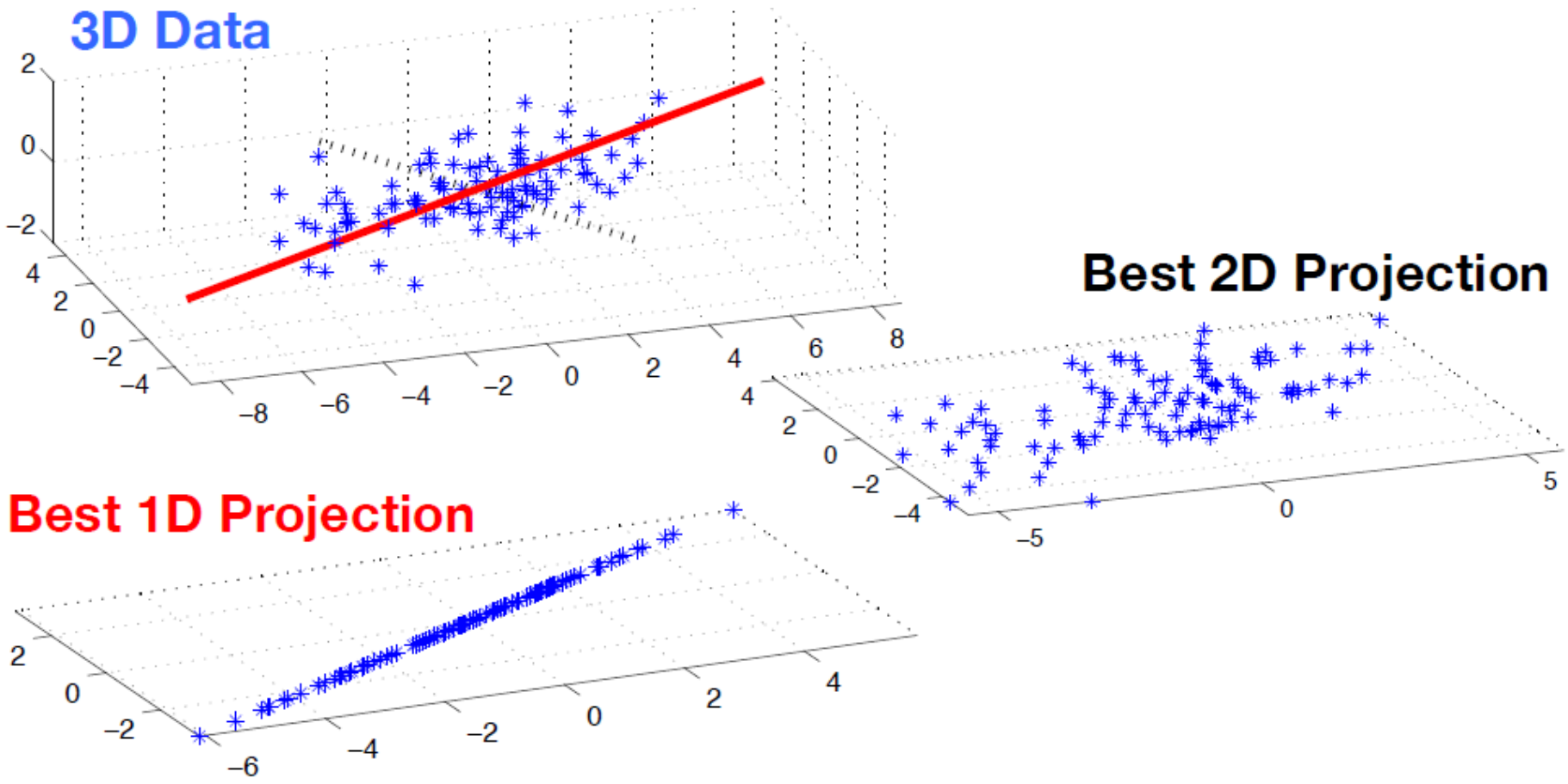
Why reduce dimensionality?

- Data may intrinsically live in a lower-dim space
- Too many features and too few data
- Lower computational expense (memory, train/test time)
- Want to visualize the data in a lower-dim space
- Want to use data of different dimensionality

Goal

- Input: Data in a high-dim feature space
- Output: Projection of same data into a lower-dim space
- Function: high-dim X → low-dim X

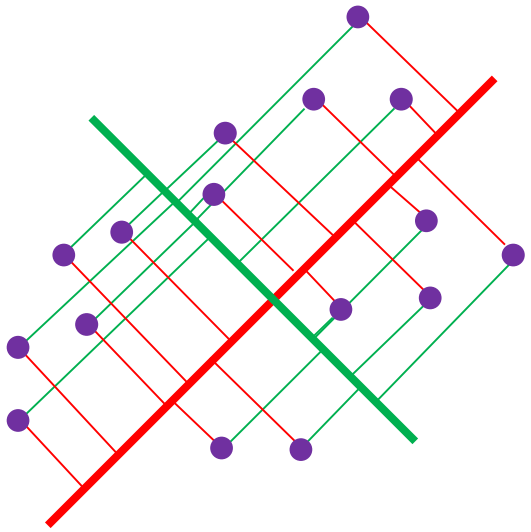
Goal



Slide credit: Erik Sudderth

Some criteria for success

- Find a projection where the data has:
  - Low reconstruction error
  - High variance of the data



Principal Components Analysis

- The eigenvectors give you the projection directions — to compute the embeddings you have to multiply the data by the projections
- For completeness here is the Matlab code:

```
function [E, U, lambda] = PCA(X, K)
mu = mean(X);                    % Data mean
N = size(X,1);                   % Number of data
X = X - ones(N,1)*mu;            % Center the data
covX = X'*X;                     % Data covaraince
[U,lambda] = eigs(covX, K); % Compute top K eigenvalues
E = X*U;                         % Compute embeddings
```

Slide credit: Subhransu Maji

Demo

- http://www.cs.pitt.edu/~kovashka/cs1675_fa18/PCA_demo.m
- http://www.cs.pitt.edu/~kovashka/cs1675_fa18/PCA.m

Application: Face Recognition

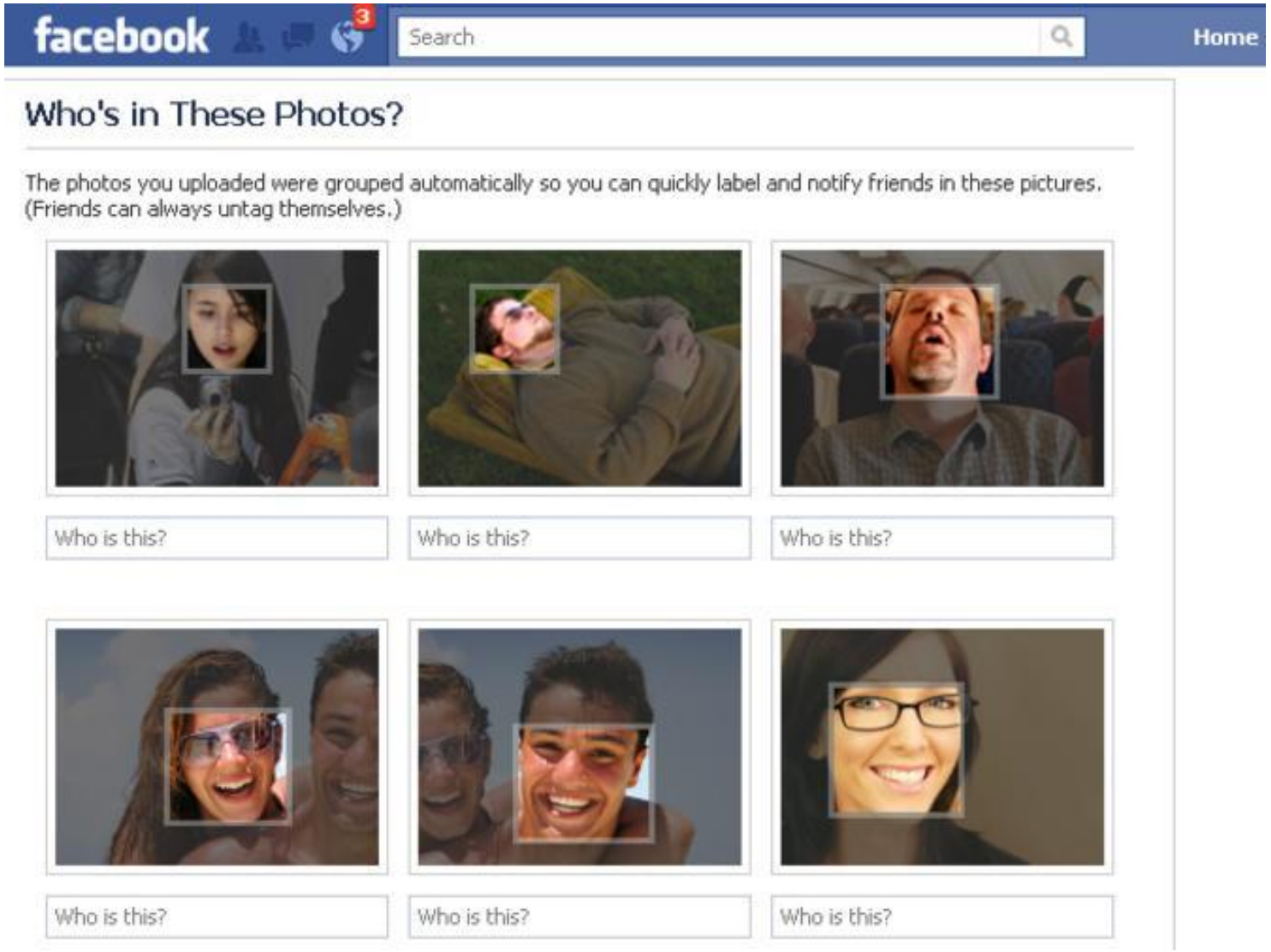


Image from cnet.com

The space of all face images

- When viewed as vectors of pixel values, face images are extremely high-dimensional
  - 24x24 image = 576 dimensions
  - Slow and lots of storage
- But few 576-dimensional vectors are valid face images
- We want to effectively model the subspace of face images

Adapted from Derek Hoiem

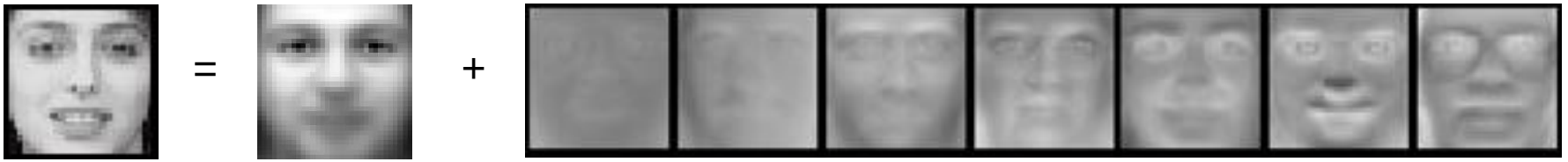
M. Turk and A. Pentland, Face Recognition using Eigenfaces, CVPR 1991

Representation and reconstruction

- Face **x** in "face space" coordinates:

$$\mathbf{x} \rightarrow [\mathbf{u}_1^{\mathrm{T}}(\mathbf{x} - \mu), \ldots, \mathbf{u}_k^{\mathrm{T}}(\mathbf{x} - \mu)]$$
$$= \quad w_1, \ldots, w_k$$

- Reconstruction:



$\hat{x} = \mu + w_1u_1+w_2u_2+w_3u_3+w_4u_4+ \ldots$

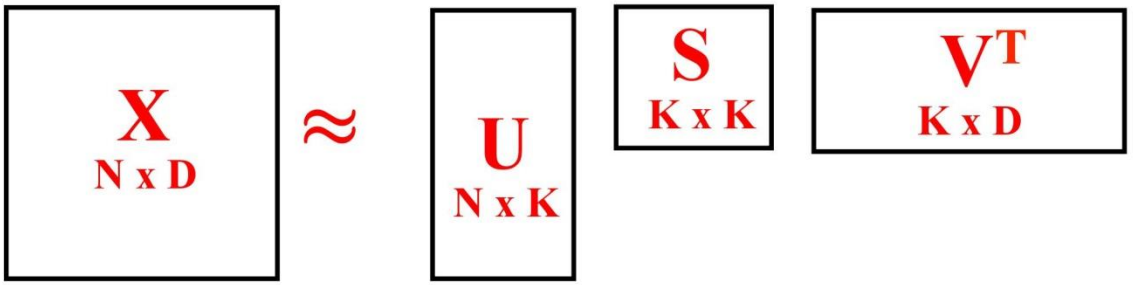
Slide credit: Derek Hoiem

Singular Value Decomposition

- Alternative method to calculate (still subtract mean 1st)
- Decompose $X = U\ S\ V^T$

- U*S matrix provides coefficients
  - Example $x_i = U_{i,1}\ S_{11}\ v_1 + U_{i,2}\ S_{22}\ v_2 + \ldots$

- Gives the least-squares approximation to X of this form



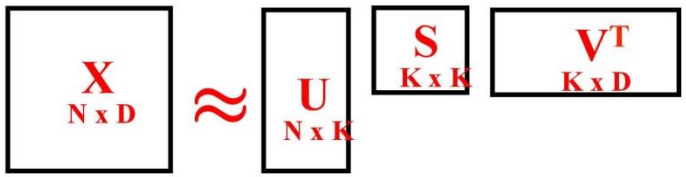
Slide credit: Alexander Ihler

Collaborative Filtering (Netflix)

From Y. Koren of BellKor team

users

movies	1	2	3	4	5	6	7	8	9	10	11	12
1	1		3		?	5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	



Slide credit: Alexander Ihler

From Y. Koren
of BellKor team

Latent Space Models

Model ratings matrix as "user" and "movie" positions

Infer values from known ratings

Extrapolate to unranked

users

items

1		3			5			5		4	
		5	4			4			2	1	3
2	4		1	2		3		4	3	5	
	2	4		5			4			2	
		4	3	4	2					2	5
1		3		3			2			4	

~

~

items

.1	-.4	.2
-.5	.6	.5
-.2	.3	.5
1.1	2.1	.3
-.7	2.1	-2
-1	.7	.3

●

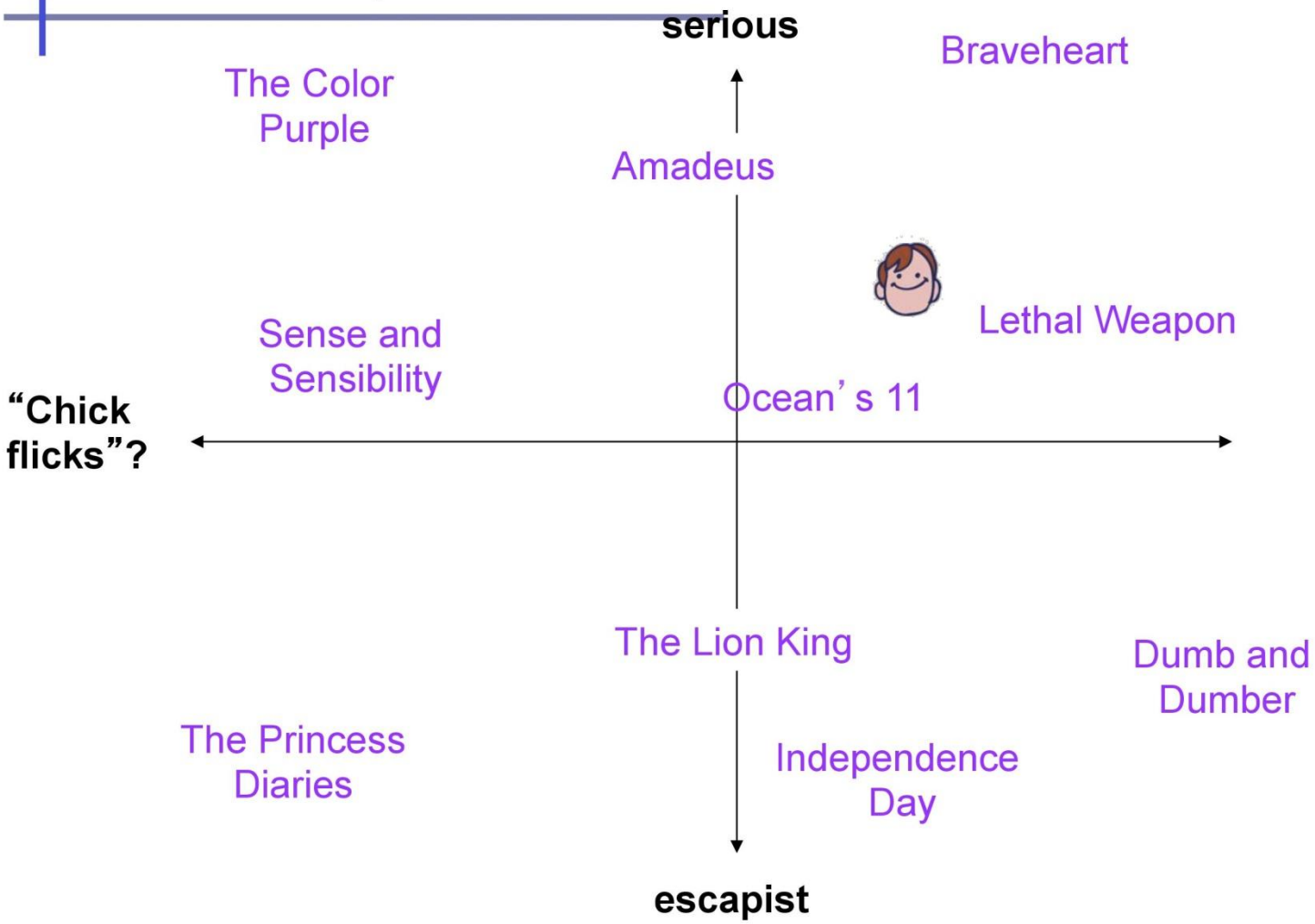
users

1.1	-.2	.3	.5	-2	-.5	.8	-.4	.3	1.4	2.4	-.9
-.8	.7	.5	1.4	.3	-1	1.4	2.9	-.7	1.2	-.1	1.3
2.1	-.4	.6	1.7	2.4	.9	-.3	.4	.8	.7	-.6	.1

Slide credit: Alexander Ihler

From Y. Koren
of BellKor team

Latent Space Models



Slide credit: Alexander Ihler

Other dimensionality reduction methods

- Non-linear:
  - Kernel PCA (Schölkopf et al., *Neural Computation* 1998)
  - Independent component analysis – Comon, *Signal Processing* 1994
  - LLE (locally linear embedding) – Roweis and Saul, *Science* 2000
  - ISOMAP (isometric feature mapping) – Tenenbaum et al., *Science* 2000
  - t-SNE (t-distributed stochastic neighbor embedding) – van der Maaten and Hinton, *JMLR* 2008

t-SNE example

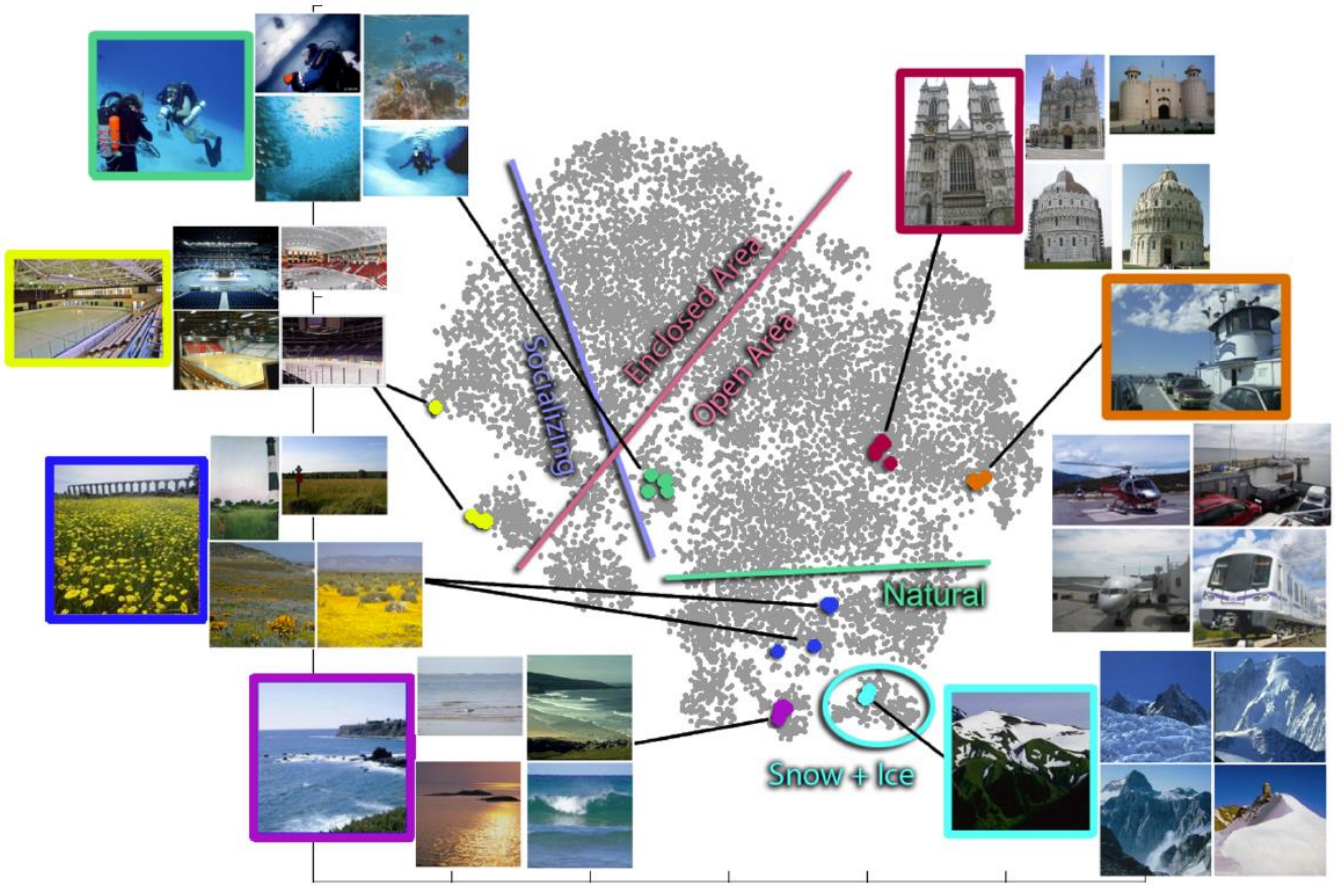
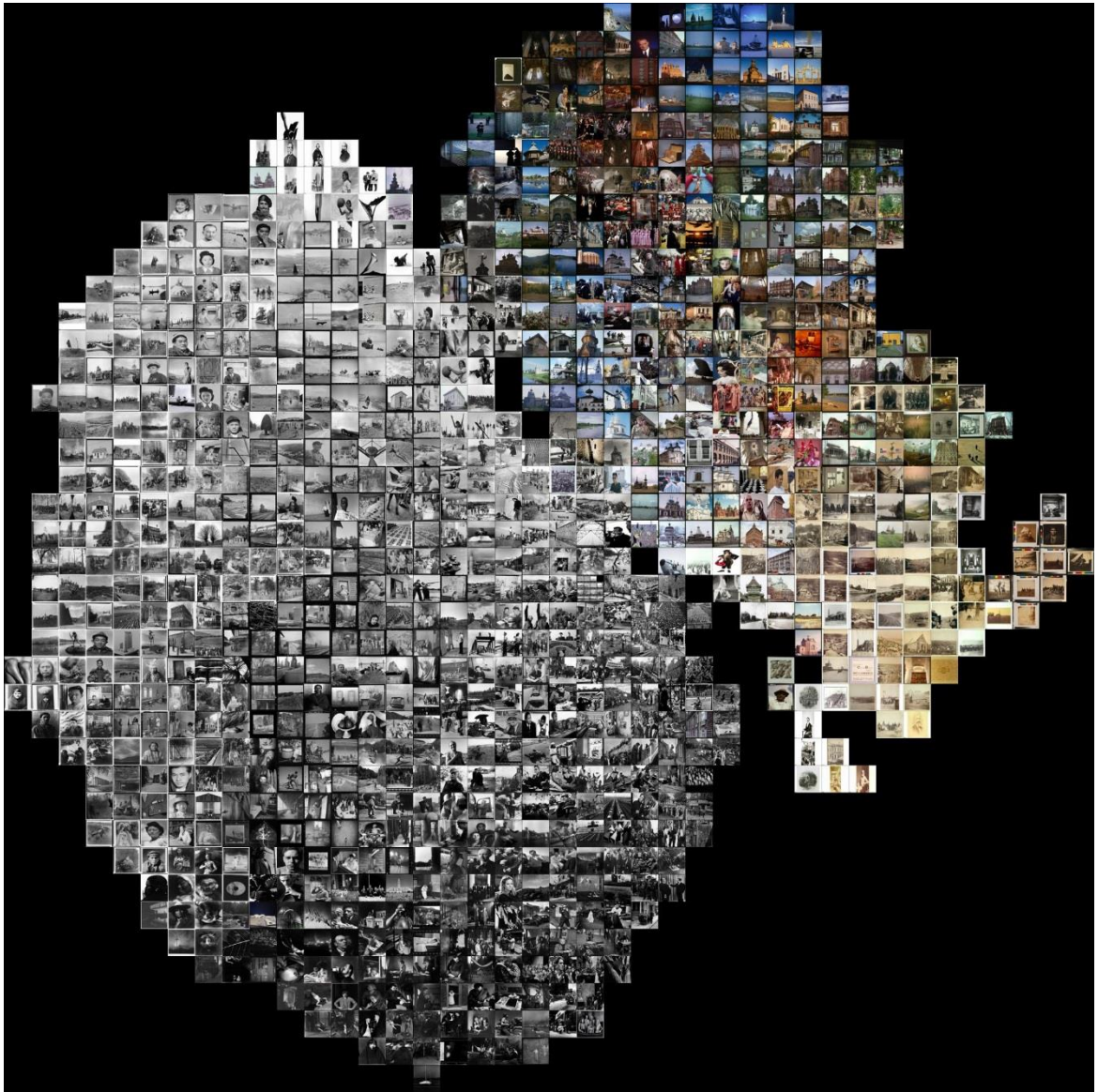


Fig. 8 2D visualization of the SUN Attribute dataset. Each image in the dataset is represented by the projection of its 102-dimensional attribute feature vector onto two dimensions using t-Distributed Stochastic Neighbor Embedding (Van der Maaten and Hinton 2008). There are groups of nearest neighbors, each designated by a *color*. Interestingly, while the nearest-neighbor scenes in attribute space are semantically very similar, for most of these examples (underwater_ocean, abbey, coast, ice skating rink, field_wild, bistro, office) none of the nearest neighbors actually fall in the same SUN database category. The *colored border lines* delineate the approximate separation of images with and without the attribute associated with the border. Figure best viewed in color (Color figure online)

Figure from Genevieve Patterson, IJCV 2014

t-SNE example



Baseline from Thomas and Kovashka, CVPR 2016