Welcome!!

Unsupervised Learning

Dr. Sugata Ghosal
sugata.ghosal@pilani.bits-pilani.ac.in

BITS Pilani
Pilani Campus

Topics to be covered

Ref: Christopher Bishop: Chapter 9

- Unsupervised learning
- Clustering
- K-means Clustering
- Gaussian Mixture Models
- EM algorithm

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

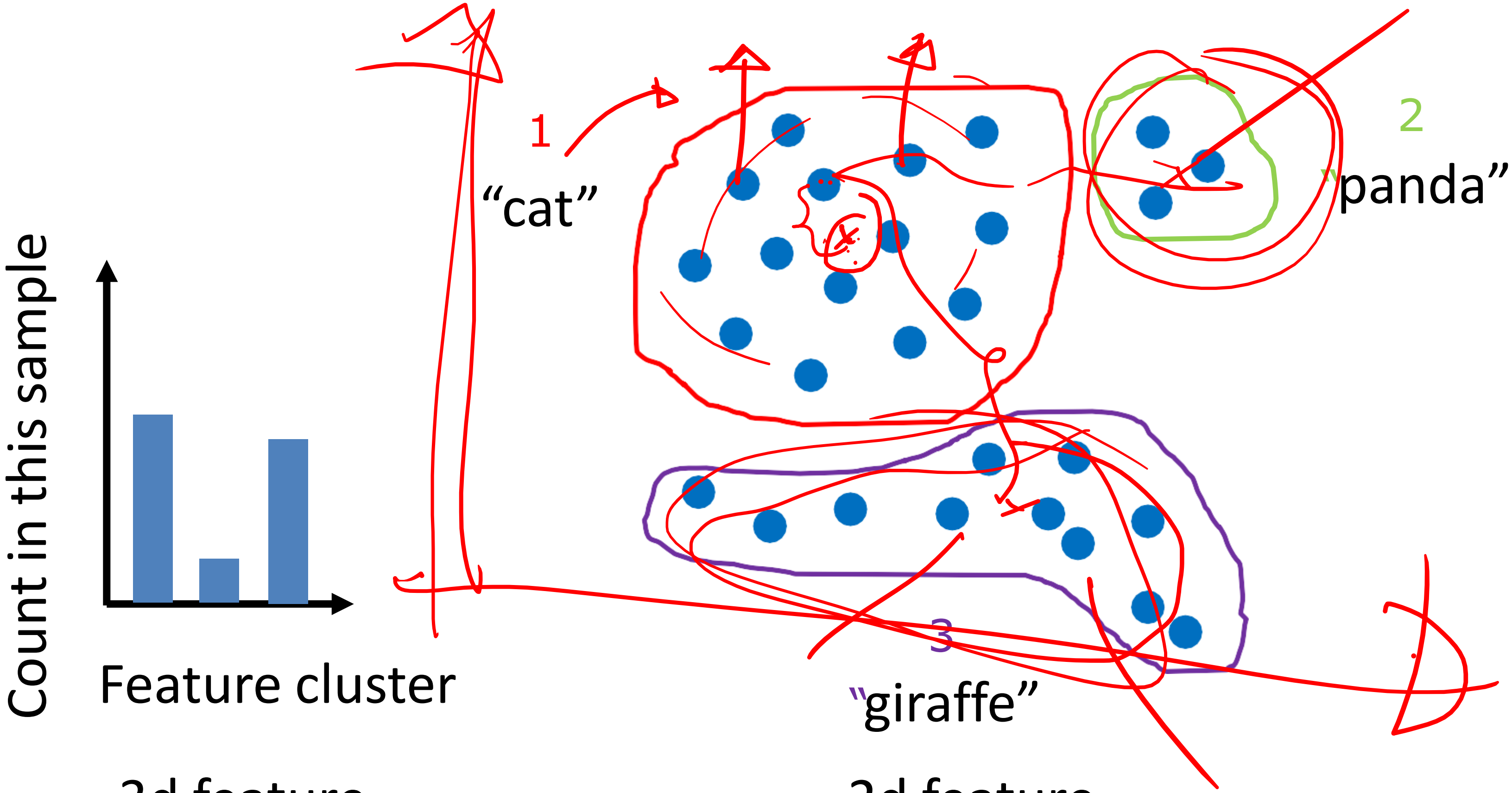
innovate achieve lead

What is clustering?

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

Why do we cluster?

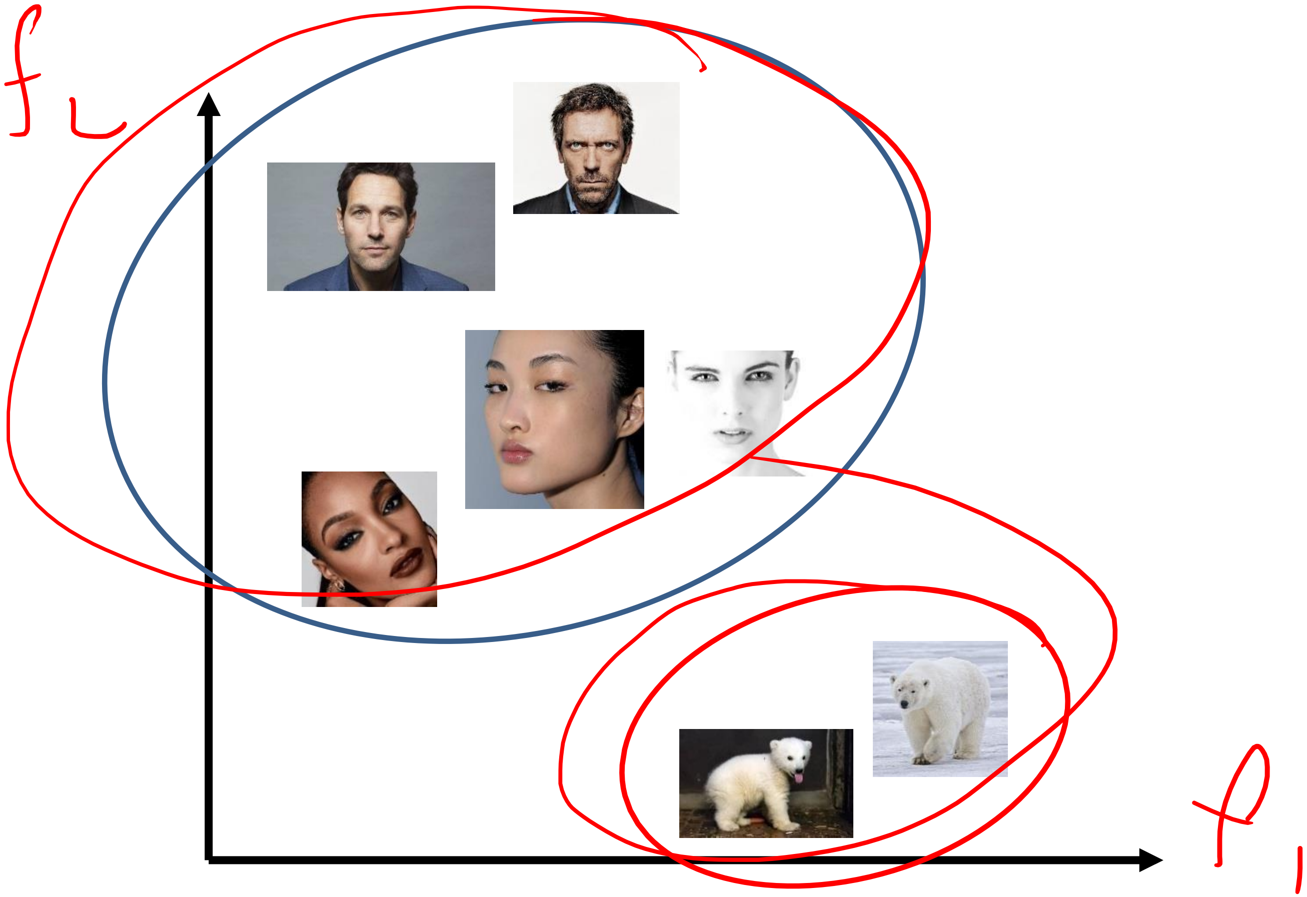
- Two uses of clustering in one application
- Cluster, then ask human to label groups
- Compute a histogram to summarize the data



3d feature

2d feature

Unsupervised discovery



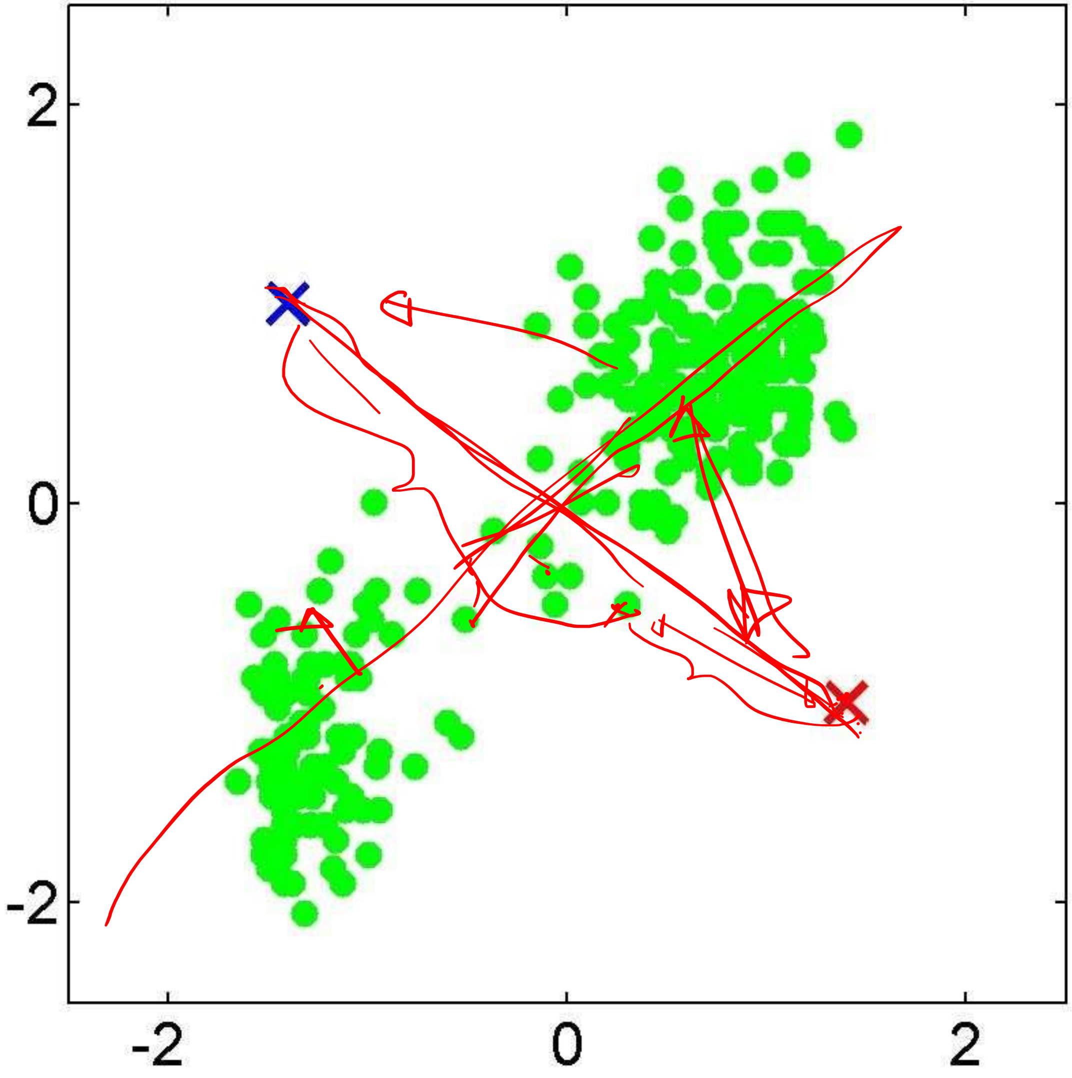
Clustering algorithms

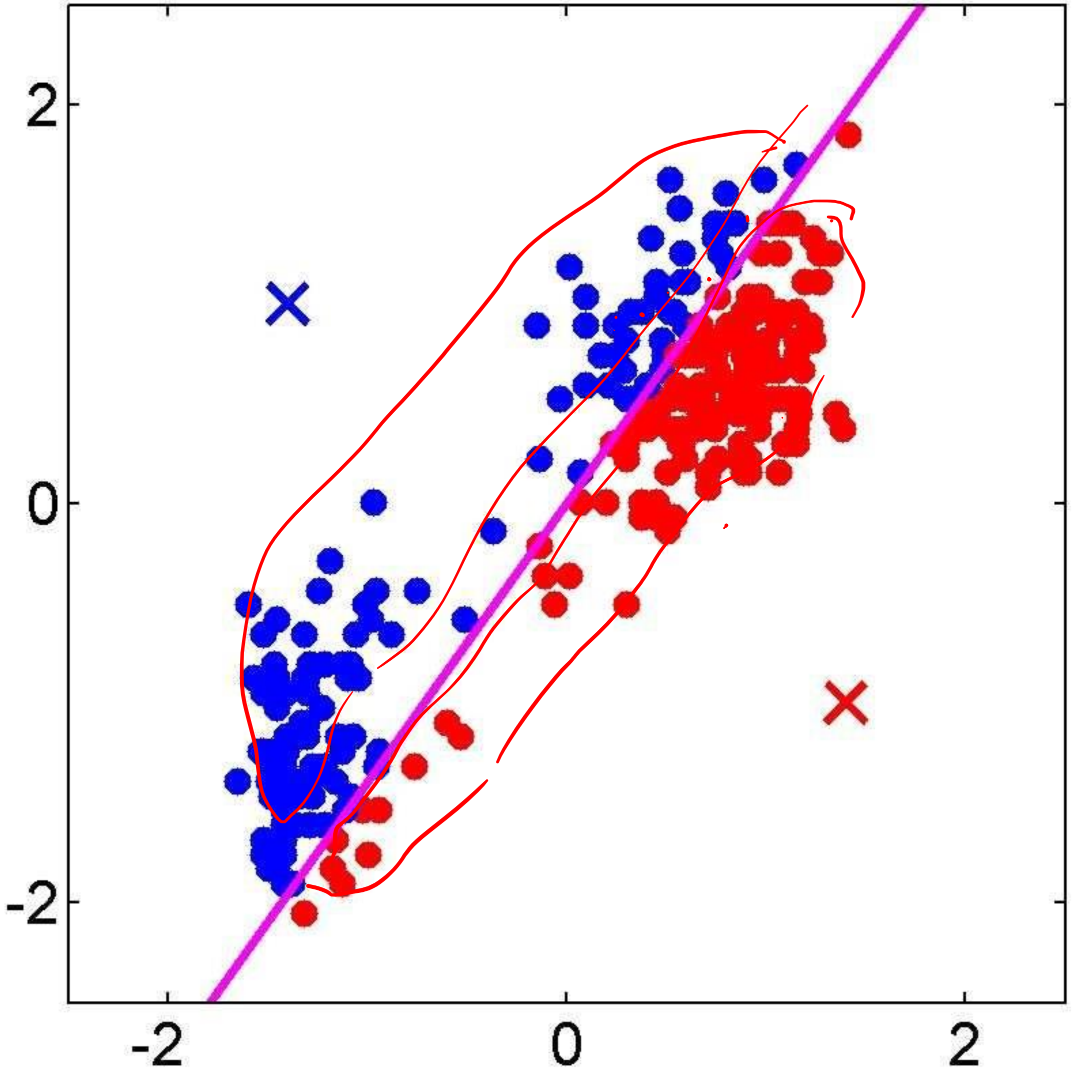
- In depth
  - K-means (iterate between finding centers and assigning points)
  - Gaussian Mixture Models (GMMs)

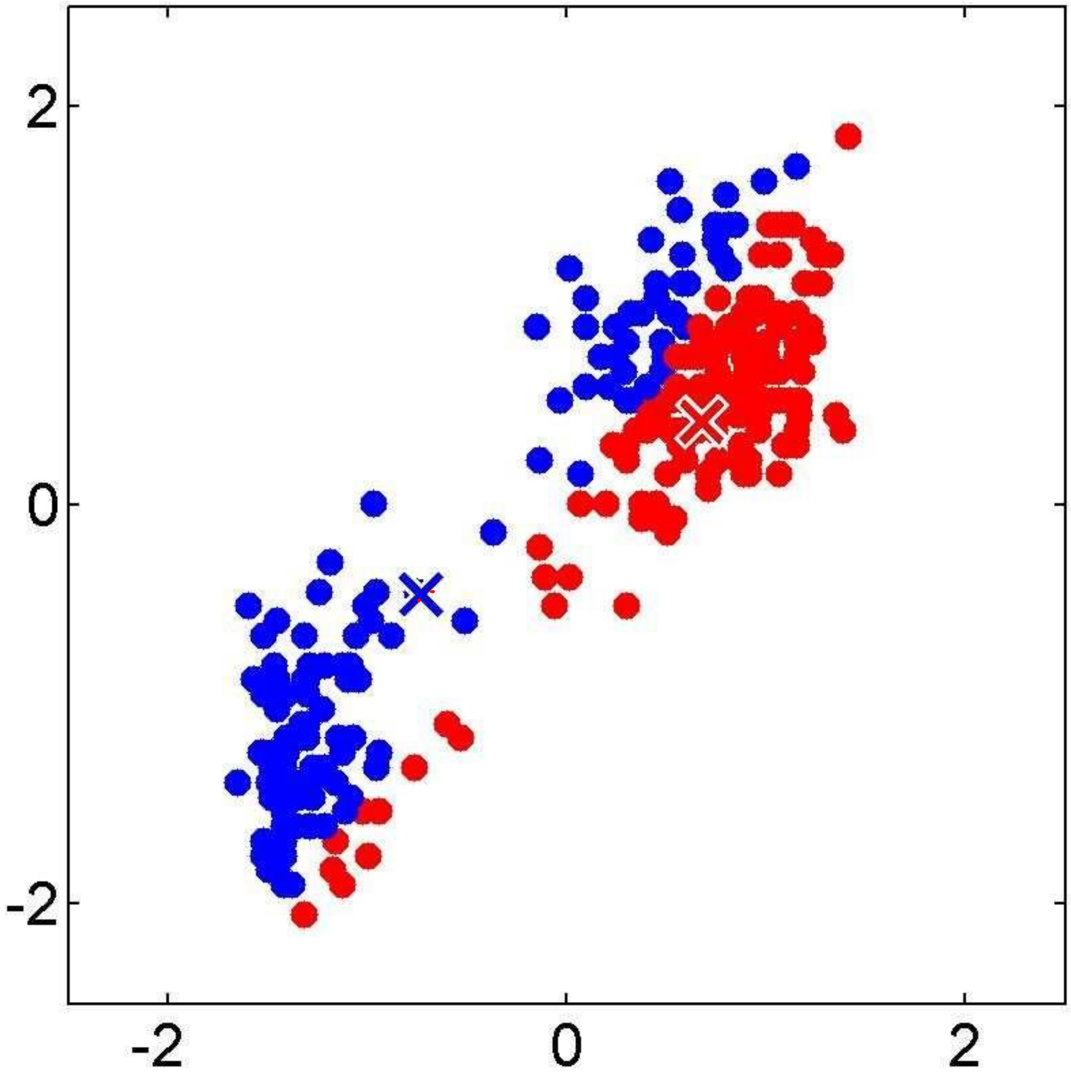
K-means Algorithm

Cluster centers

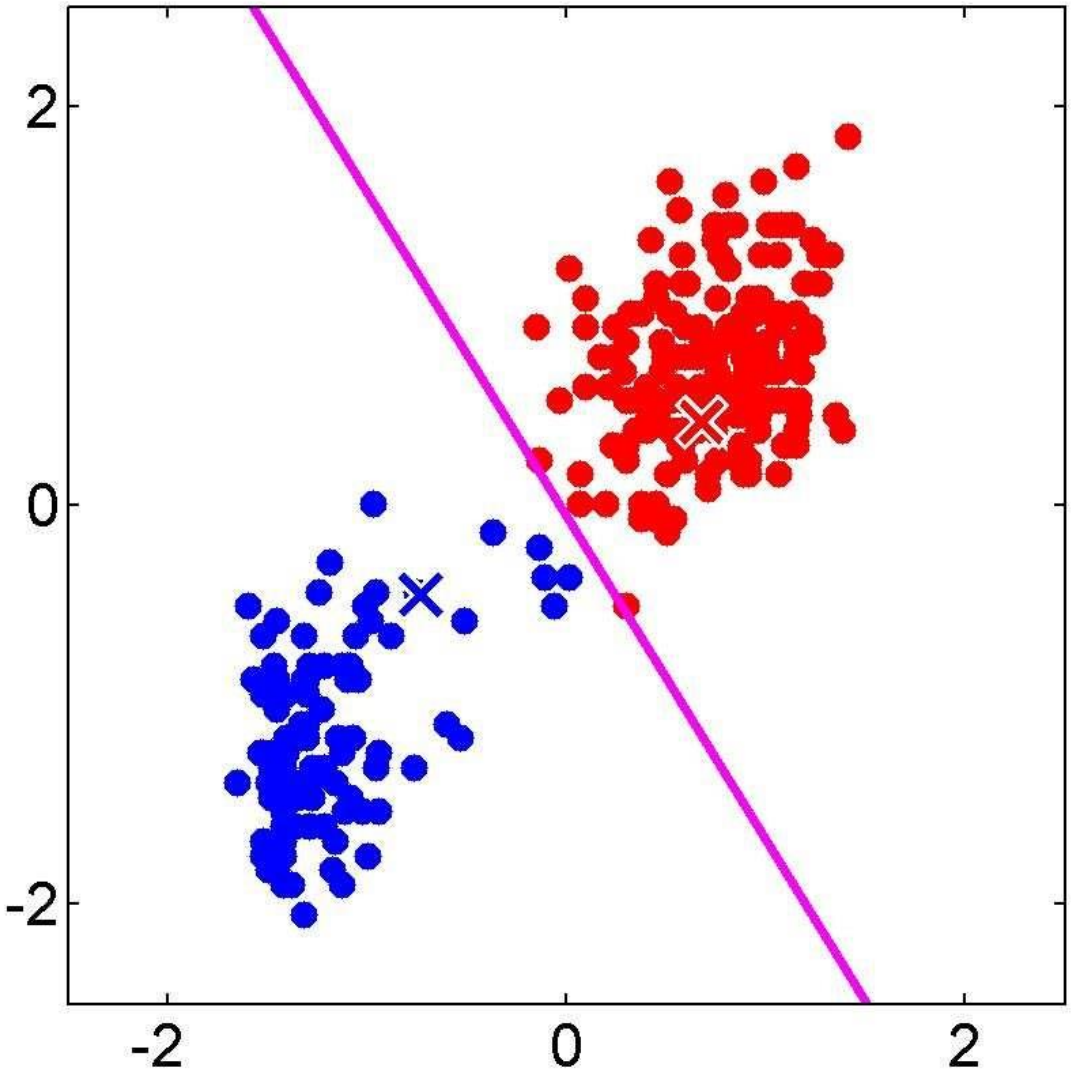
- Goal: represent a data set in terms of K clusters each of which is summarized by a prototype $\boldsymbol{\mu}_k$
- Initialize prototypes, then iterate between two phases:
  - E-step: assign each data point to nearest prototype
  - M-step: update prototypes to be the cluster means
- Simplest version is based on Euclidean distance
  - re-scale Old Faithful data

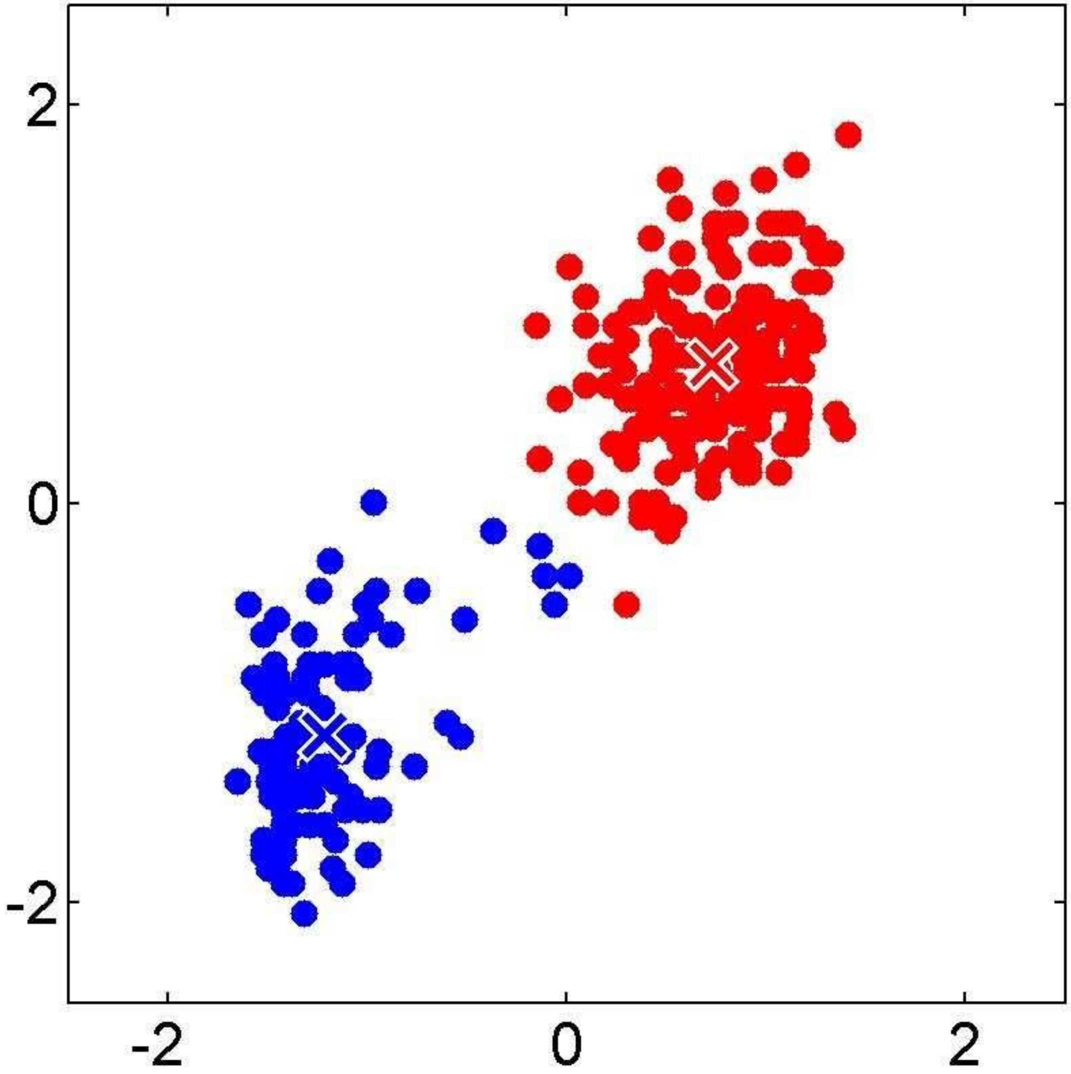

2
0
-2
-2
0
2

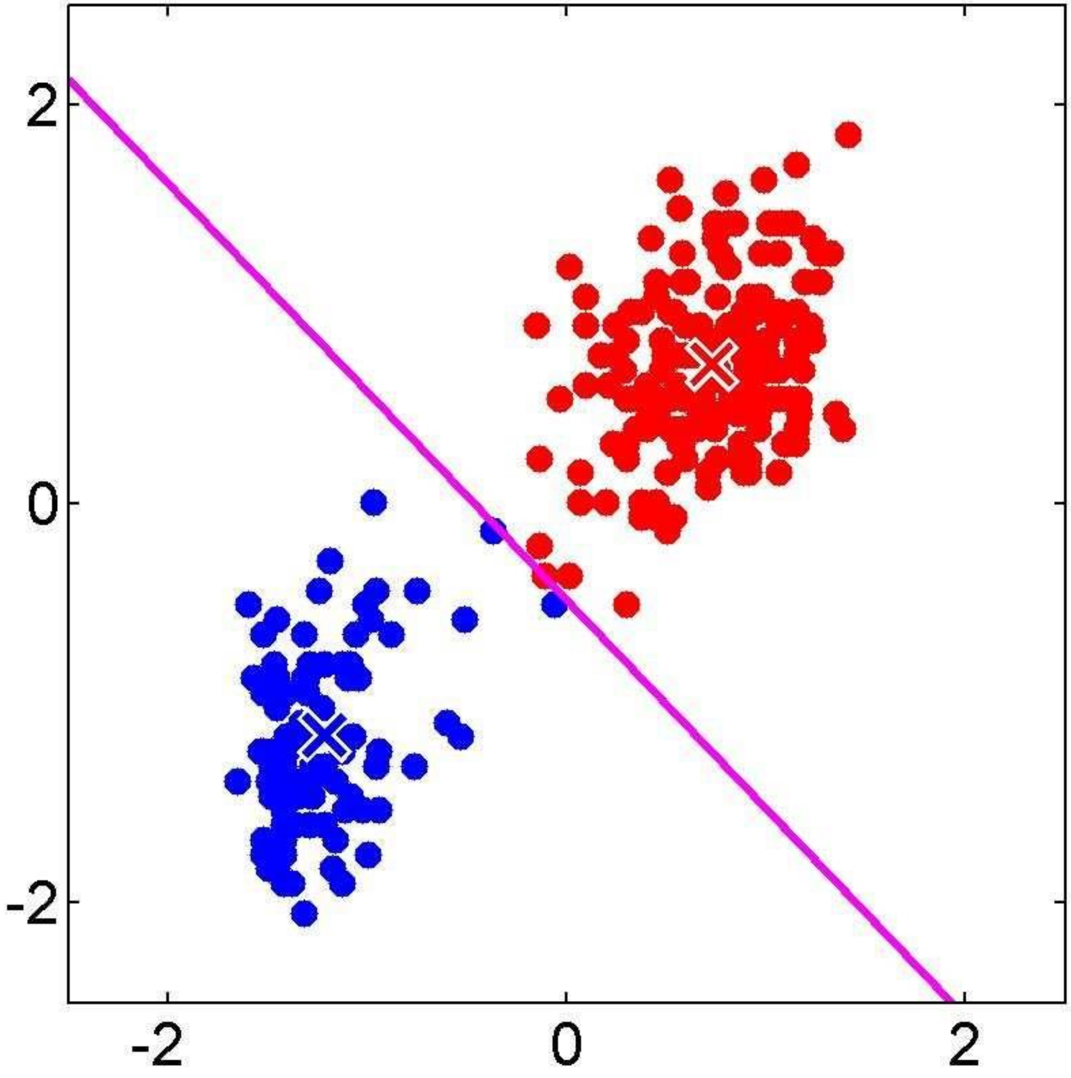

2
0
-2
-2
0
2

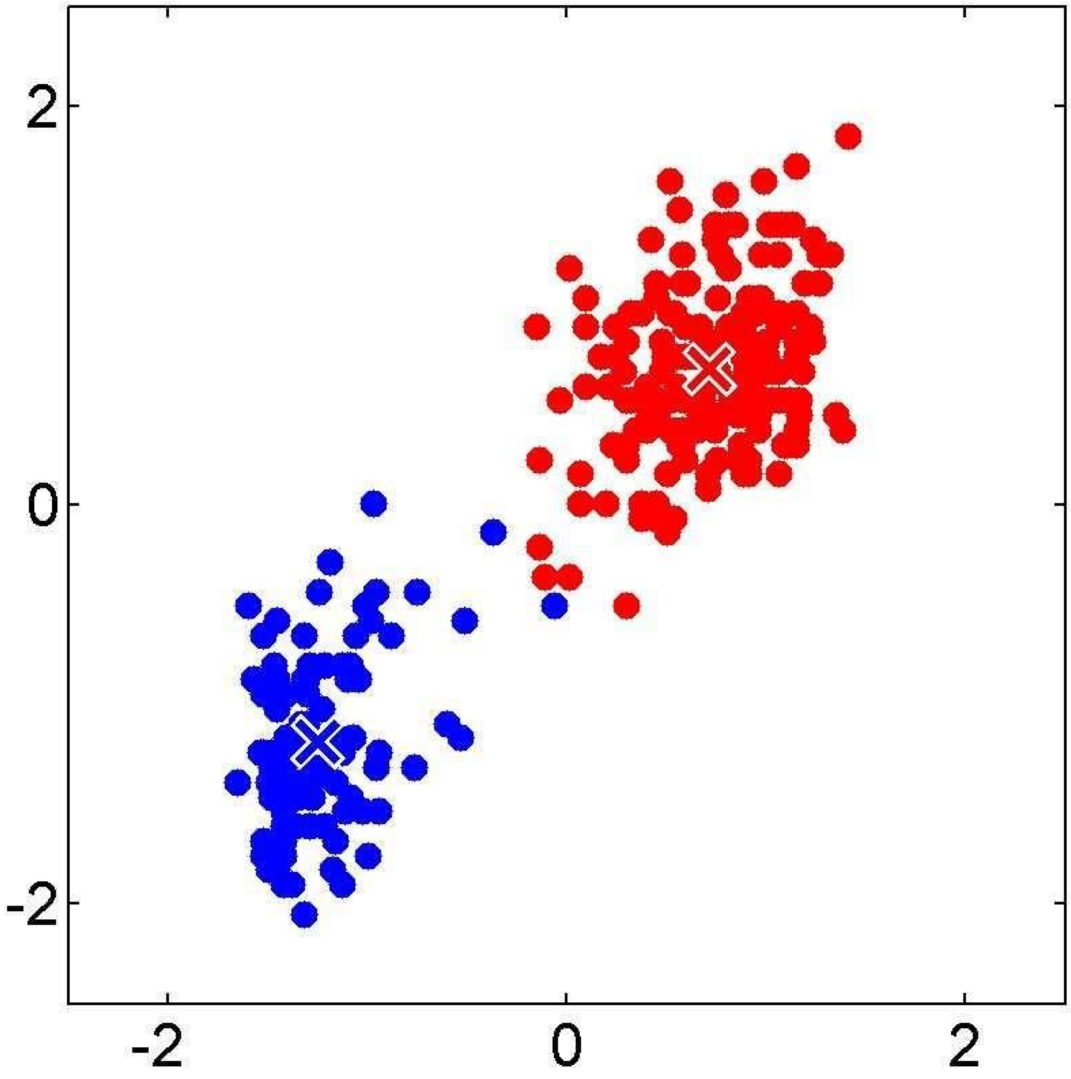


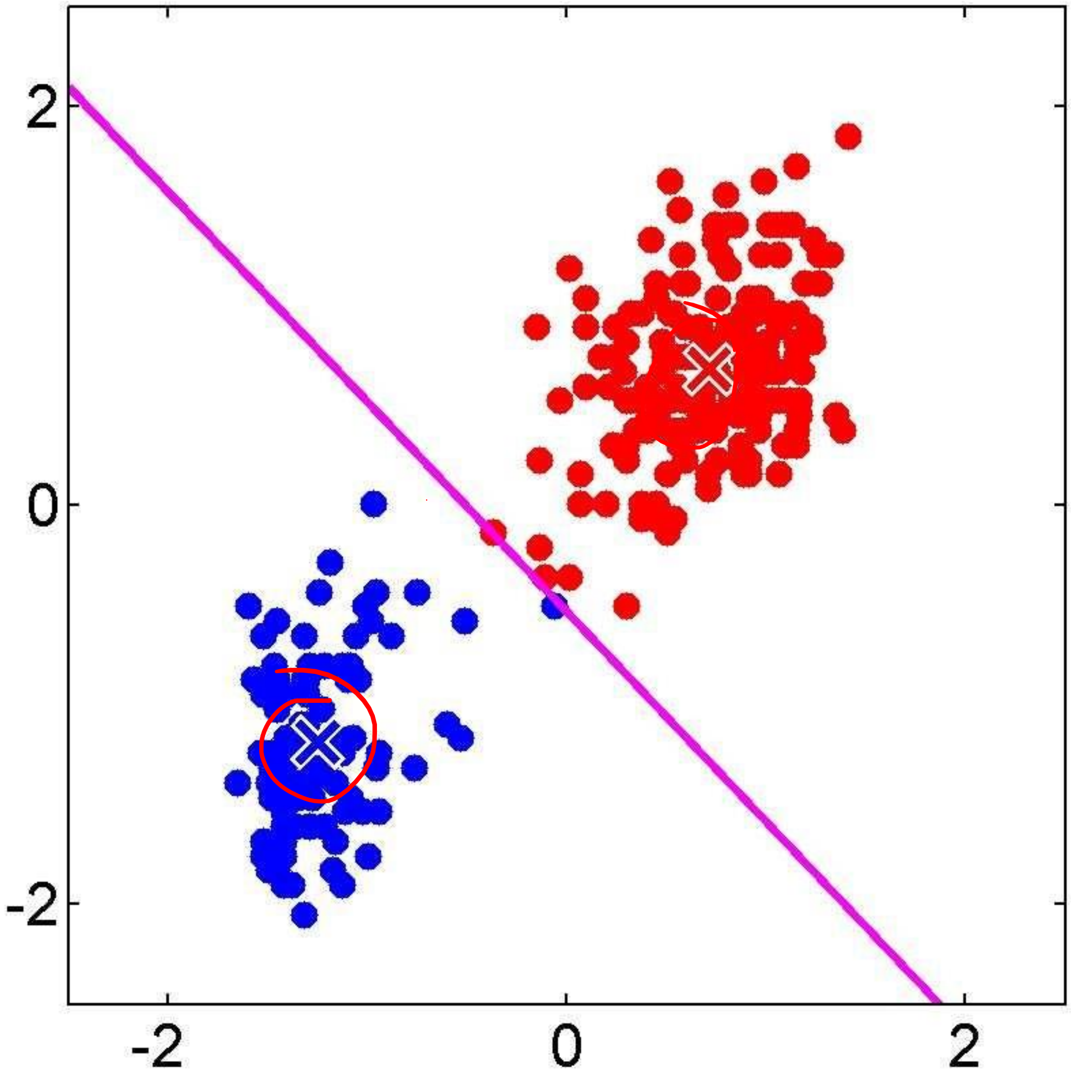
2
0
-2
-2
0
2

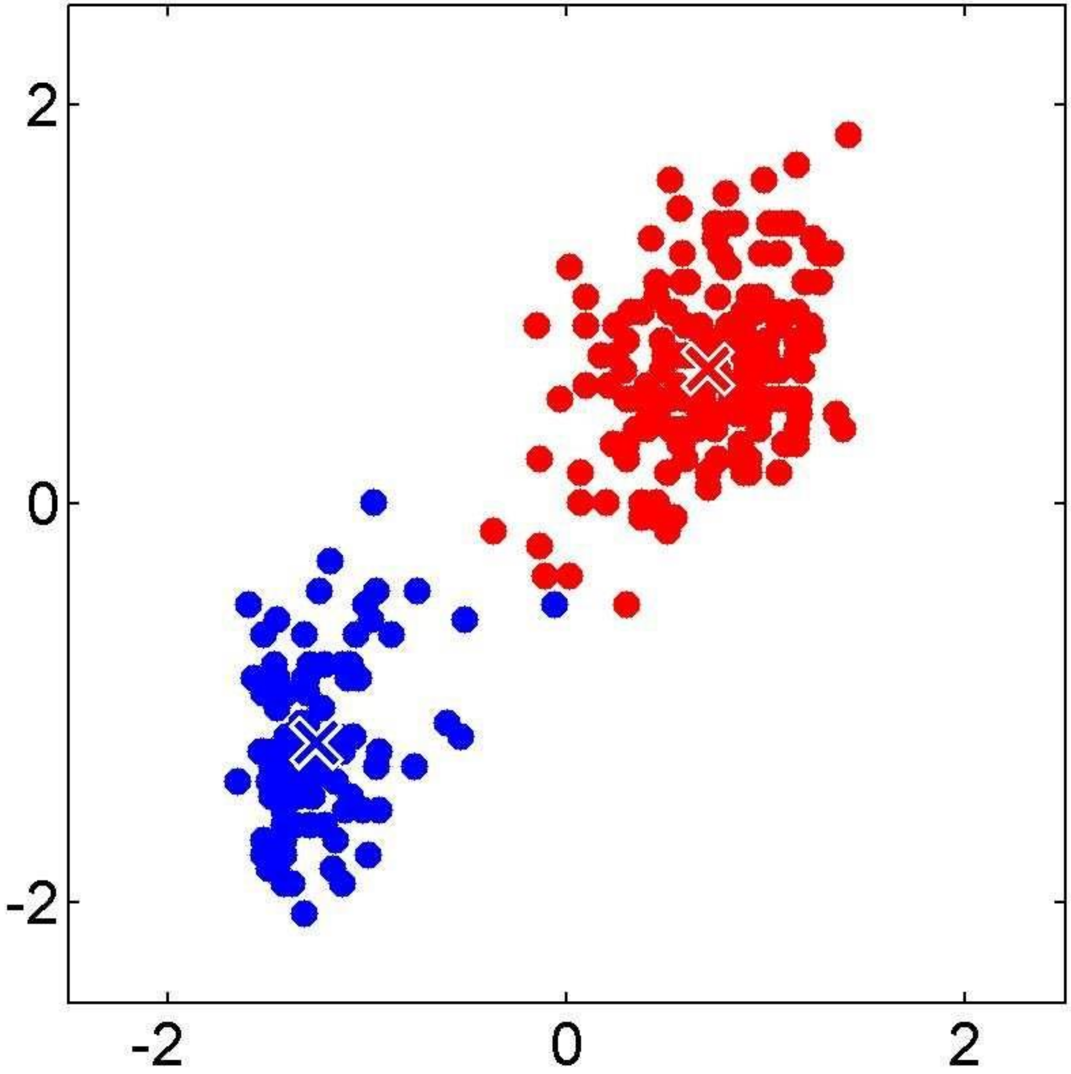

2
0
-2
-2
0
2


2
0
-2
-2
0
2


2
0
-2
-2
0
2


2
0
-2
-2
0
2


2
0
-2
-2
0
2


2
0
-2
-2
0
2

1 of K coding mechanism

k-th cluster

- *For each data point x_n, we introduce a set of binary indicator variables* $r_{nk} \in \{0, 1\}$

  n-th data point

  such that $\sum_k r_{nk} = 1$

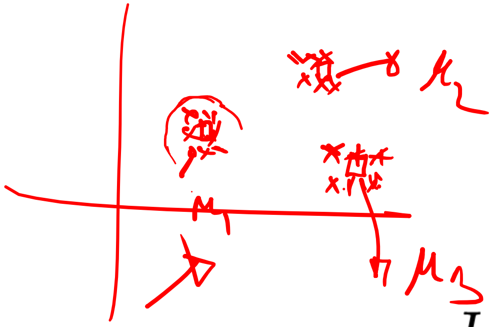
  responsibility

  where k =1,...,K describing which of the K clusters the data point x_n is assigned to, so that if data point x_n is assigned to cluster k then r_{nk} =1, and r_{nj} =0 for j not equal to k.

- Example: 5 data points and 3 clusters

$$(r_{nk}) = \begin{array}{ccc} & & 3 \\ \hline 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{array}$$

K-means Cost Function



$n \in$ k-th clusters

data

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

responsibilities

prototypes

Minimizing the Cost Function

$x_1 \quad x_2 \quad x_3 \in$

x_4 k-th cluster

$$\mu = \frac{x_1 + x_2 + x_3 + x_4}{4}$$

- E-step: minimize J w.r.t. r_{nk}

$$J = \sum_{n} \sum_{k} r_{nk} \| \underline{x}_n - \underline{\mu}_k \|^2$$

E→

- M-step: minimize J w.r.t $\boldsymbol{\mu}_k$

$$0 = 2 \sum_{n=1}^{N} r_{nj} (\underline{x}_n - \underline{\mu}_j)$$

$$\underline{\mu}_j = \frac{\sum_n r_{nj} x_n}{\sum_n r_{nj}}$$

- Convergence guaranteed since there is a finite number of possible settings for the responsibilities

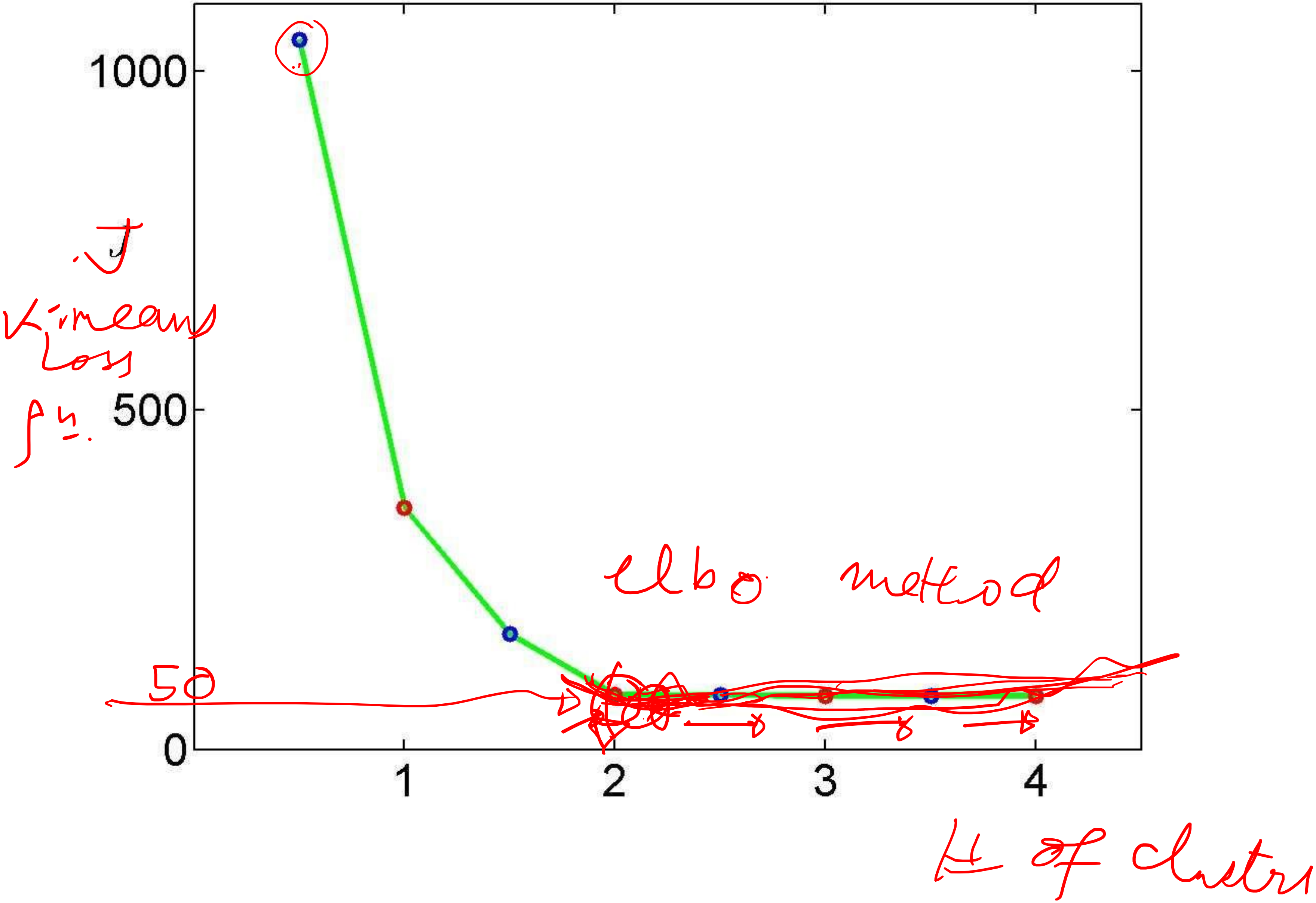
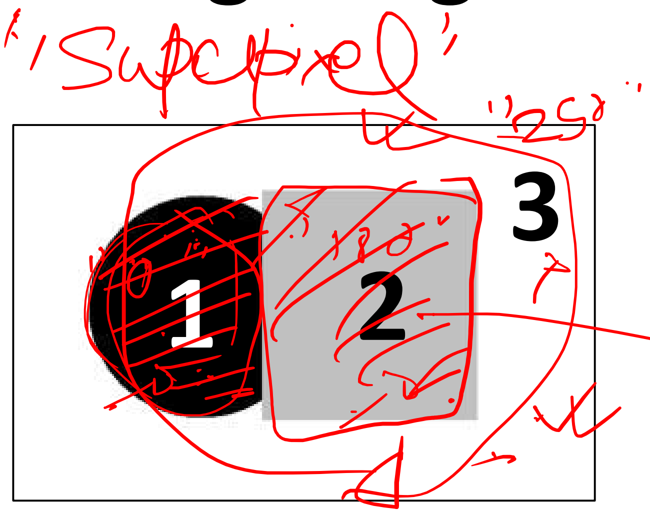
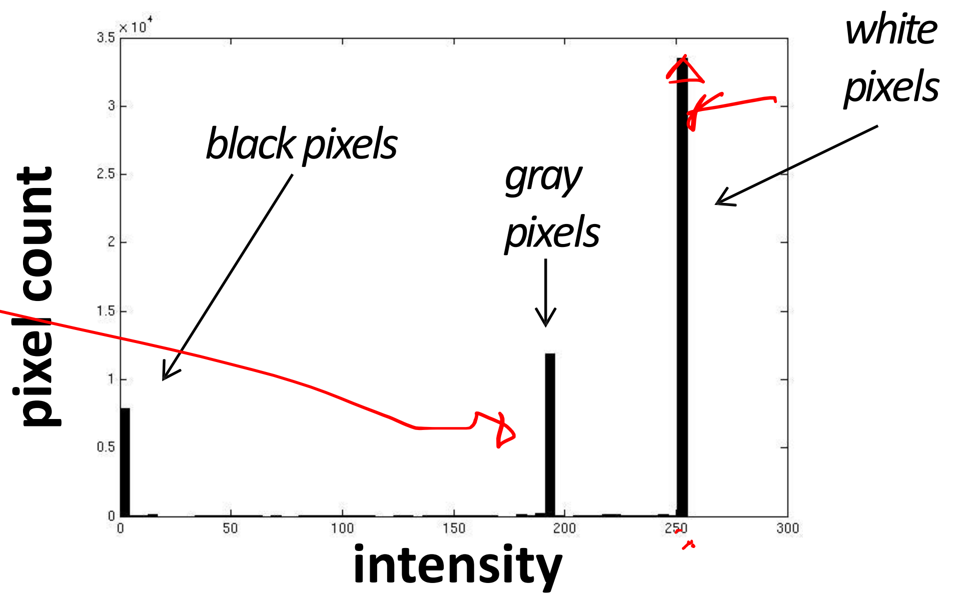

1000
500
0
1
2
3
4
J
K-means Loss fn.
50
elbo method
# of clusters

Image segmentation: toy example



input image

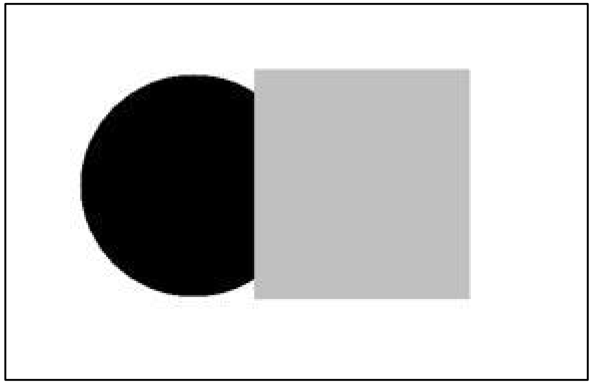


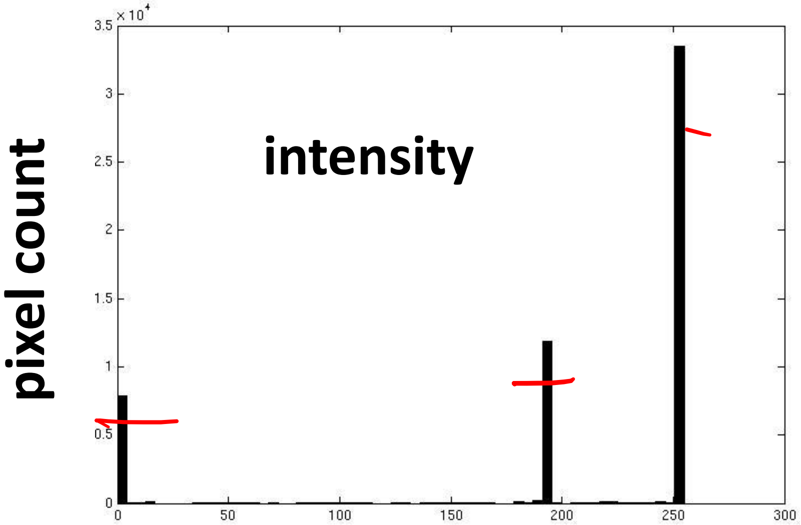
- These intensities define the three groups.
- We could label every pixel in the image according to which of these primary intensities it is.
  - i.e., *segment* the image based on the intensity feature.
- What if the image isn't quite so simple?

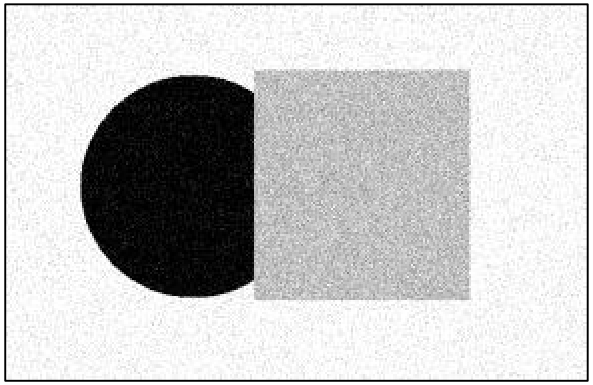
Source: K. Grauman

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

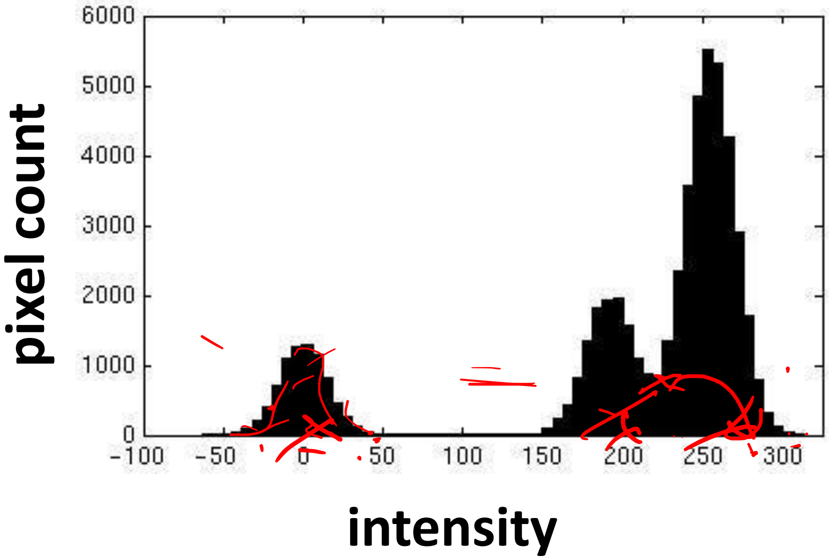
input image



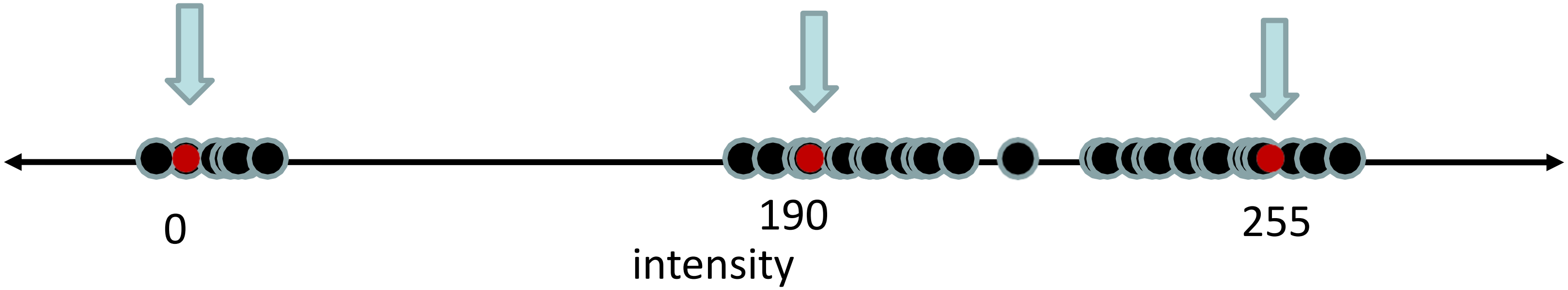


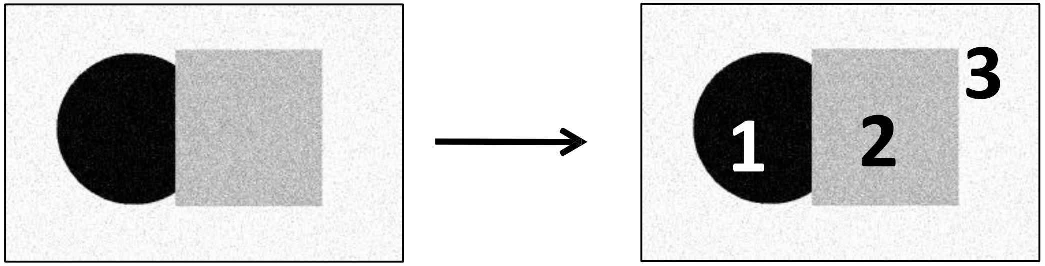


input image



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:

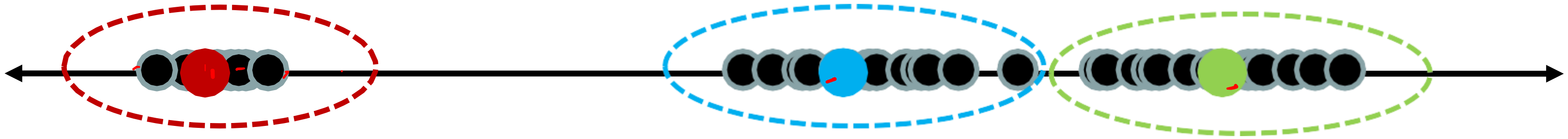
$$\sum_{\text{clusters } i} \sum_{\text{points p in cluster } i} ||p - c_i||^2$$

Euclidean

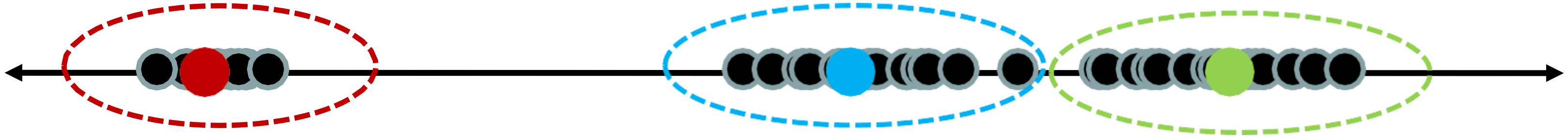
Source: K. Grauman

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

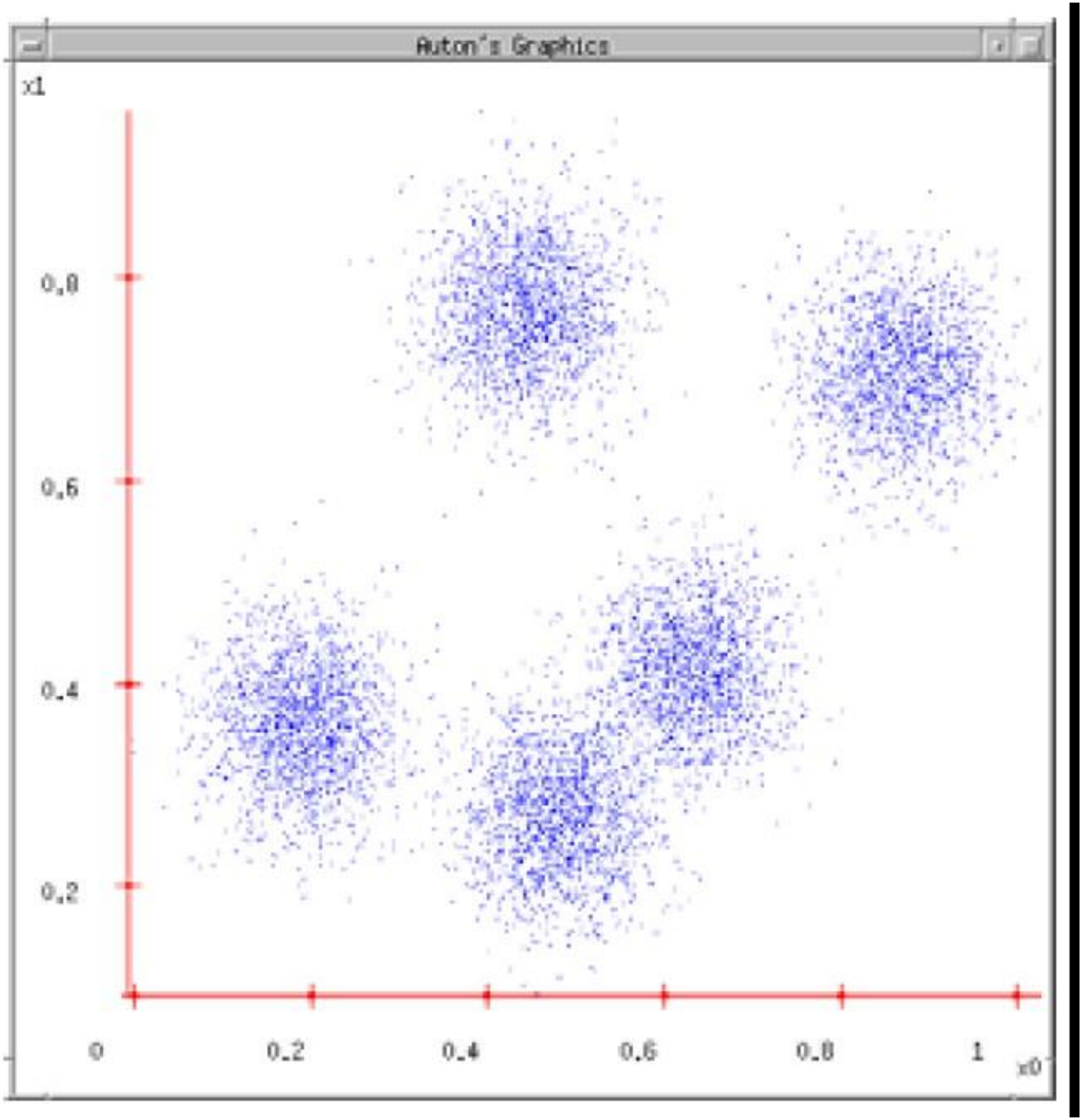
- Will always converge to some solution
- Can be a "local minimum" of objective:

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

Slide: Steve Seitz, image: Wikipedia

K-means

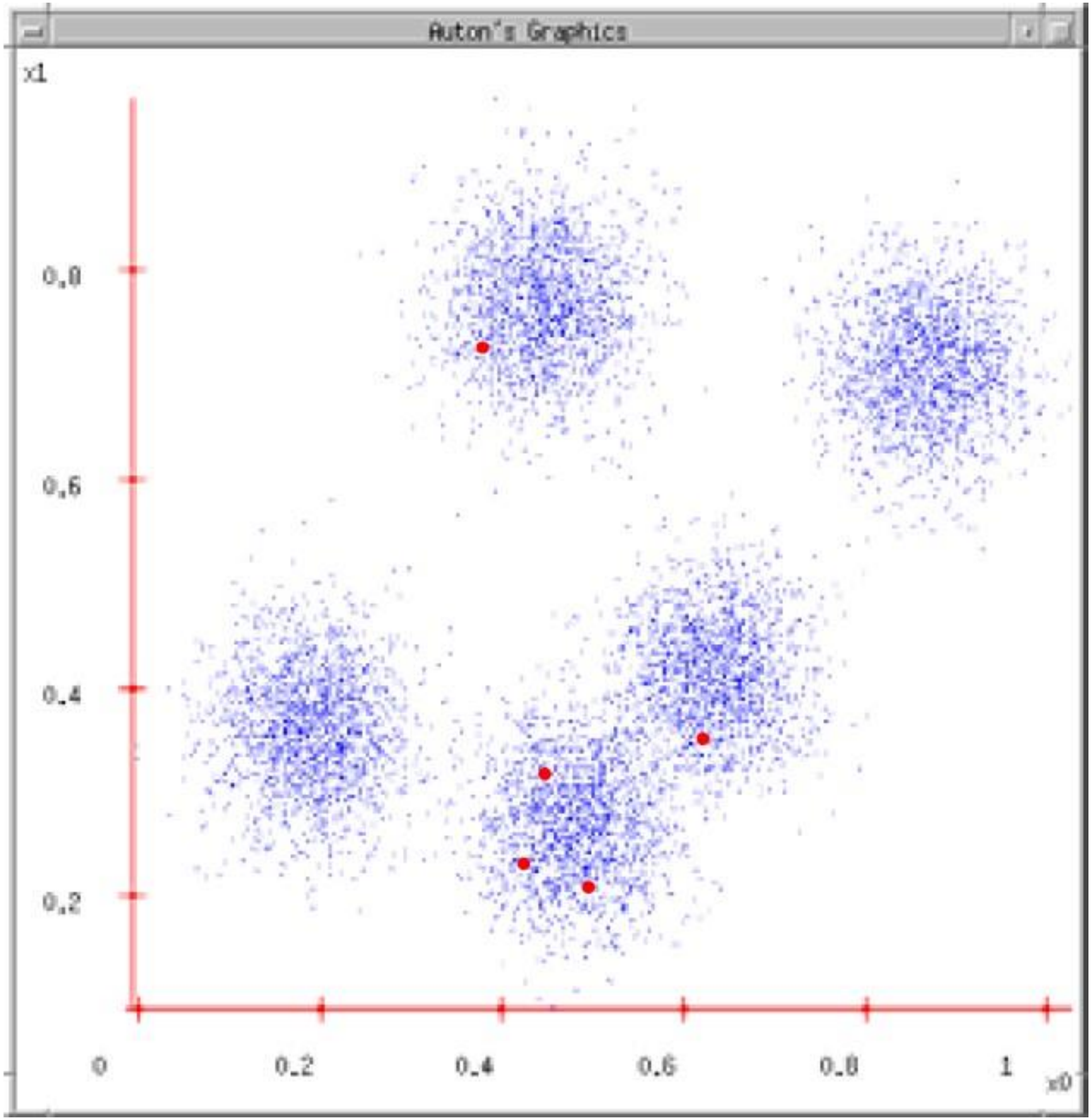
1. Ask user how many clusters they'd like. *(e.g. k=5)*



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



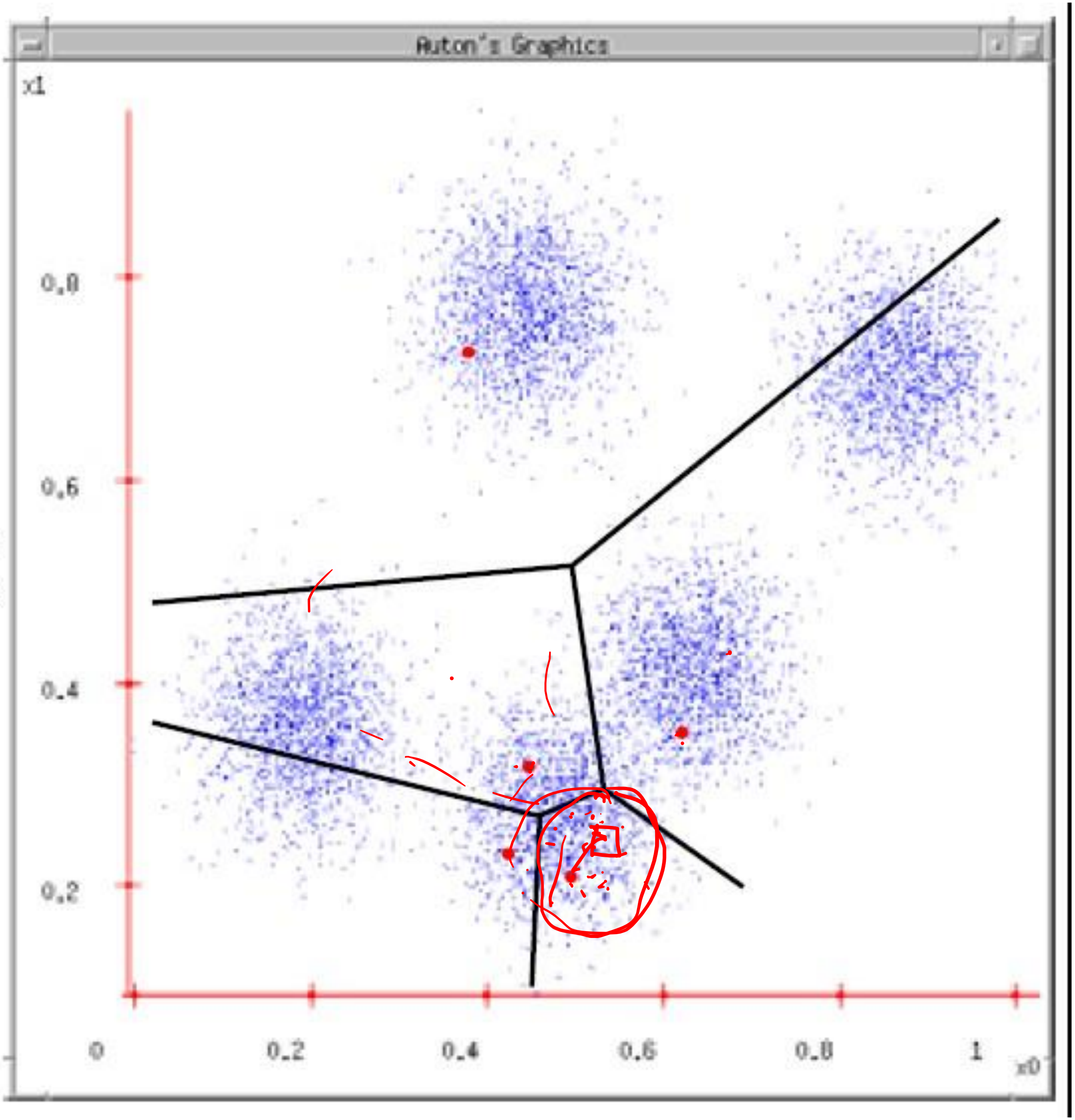
Source: A. Moore

K-means

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

2. Randomly guess k cluster Center locations

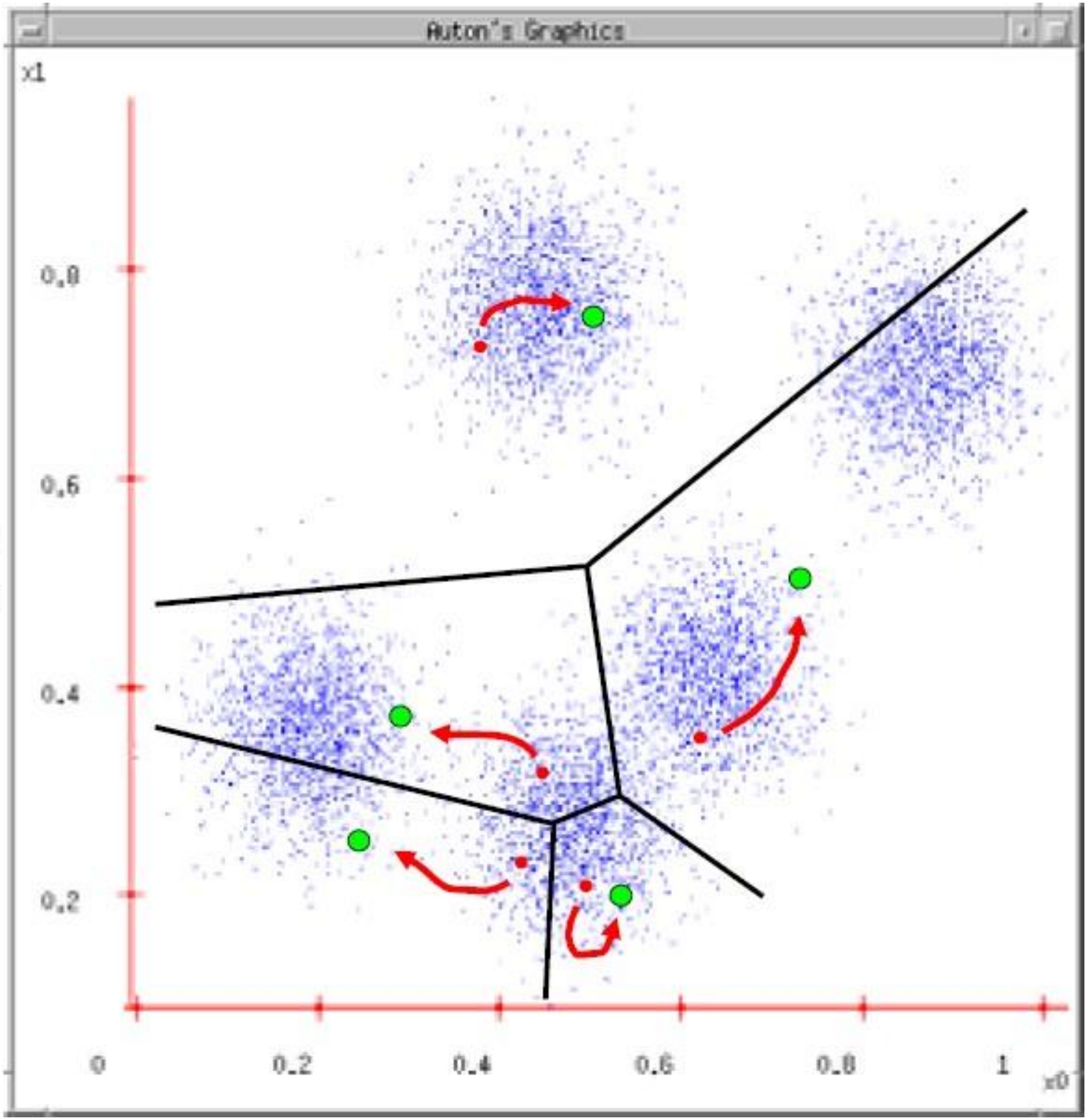
3. Each datapoint finds out which Center it's closest to. (Thus each Center "owns" a set of datapoints)



Source: A. Moore

K-means

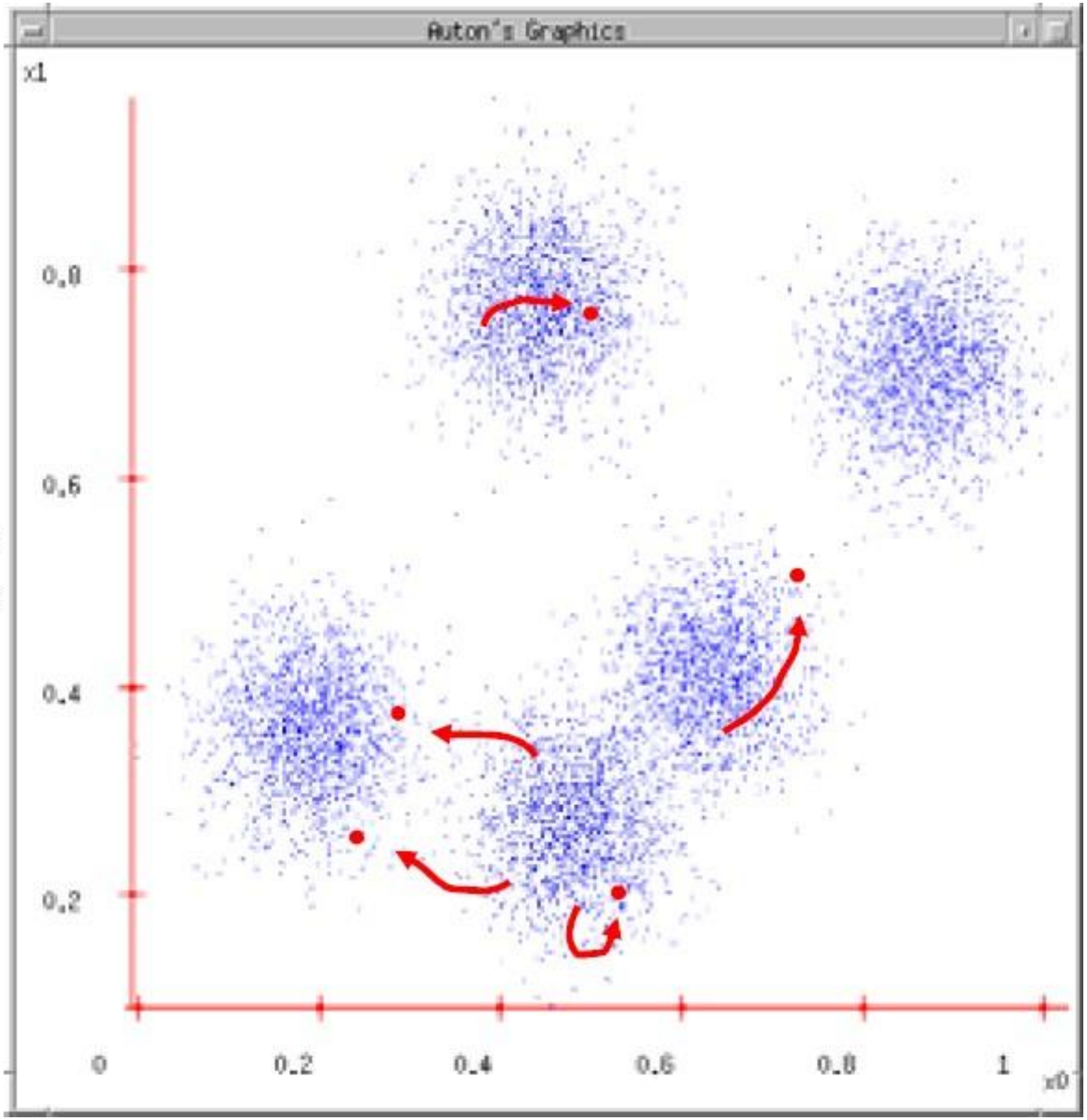
1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds the centroid of the points it owns



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 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-means the centre of a cluster is not necessarily one of the input data points (it is the average between the points in the cluster).

- **K-medoids** (more general distances):

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

- k-medoids chooses data points as centers (medoids or exemplars) and can be used with arbitrary distances
- k-medoids more robust to noise and outliers as compared to *k*-means because it minimizes a sum of pairwise dissimilarities instead of a sum of squared Euclidean distances.

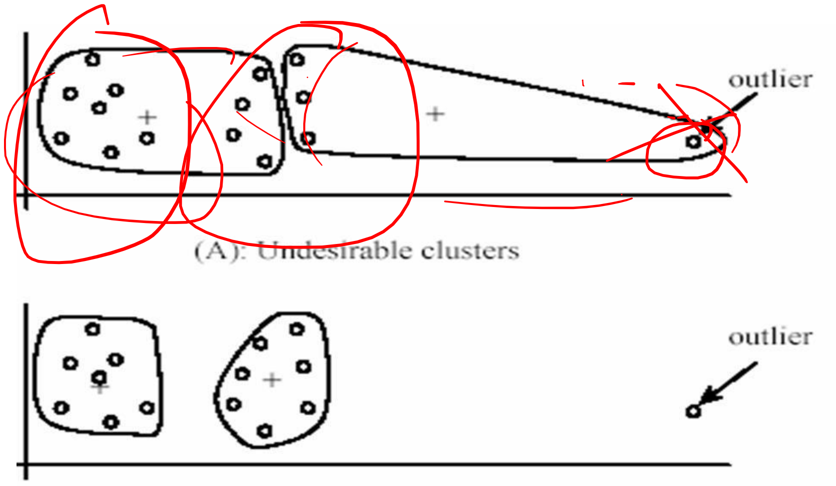
K-means: pros and cons

Pros

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

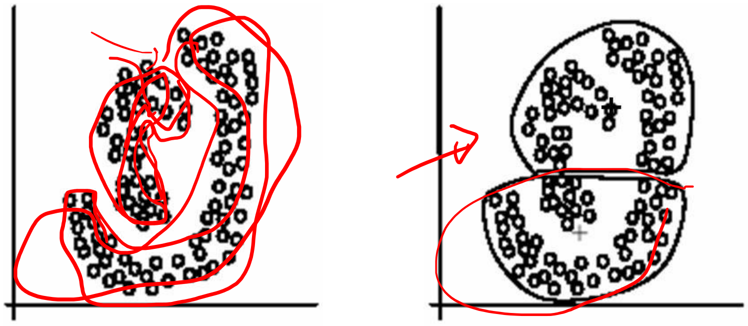
Cons/issues

- Setting k?
- Sensitive to initial centers
  - Use heuristics or output of another method
- Sensitive to outliers
- Detects spherical clusters



(A): Undesirable clusters

(B): Ideal clusters



(A): Two natural clusters

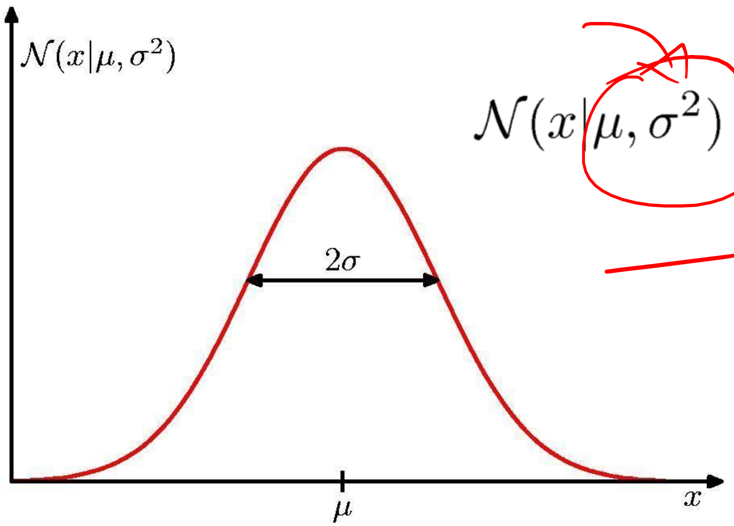
(B): *k*-means clusters

Adapted from K. Grauman

Probabilistic Clustering

- Represent the probability distribution of the data as a *mixture model*
  - captures uncertainty in cluster assignments
  - gives model for data distribution
  - *Bayesian* mixture model allows us to determine K
- Consider mixtures of *Gaussians*

Review: Gaussian Distribution

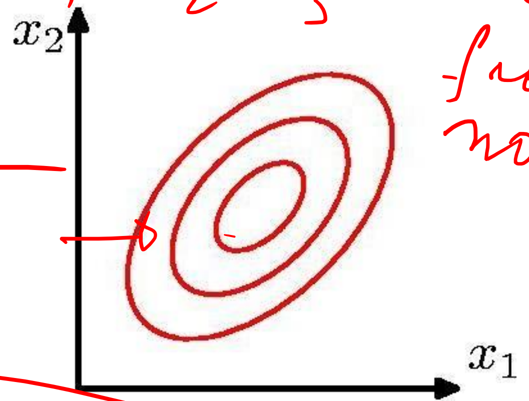


$$\mathcal{N}(x|\mu,\sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\}$$

$[d \times d]$

$\frac{\pi_1}{\pi_1+\pi_2+\pi_3}$

function of data from model,



μ_k $\frac{\pi_k}{\sum \pi}$

π_1 π_2 π_3

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

Chris Bishop

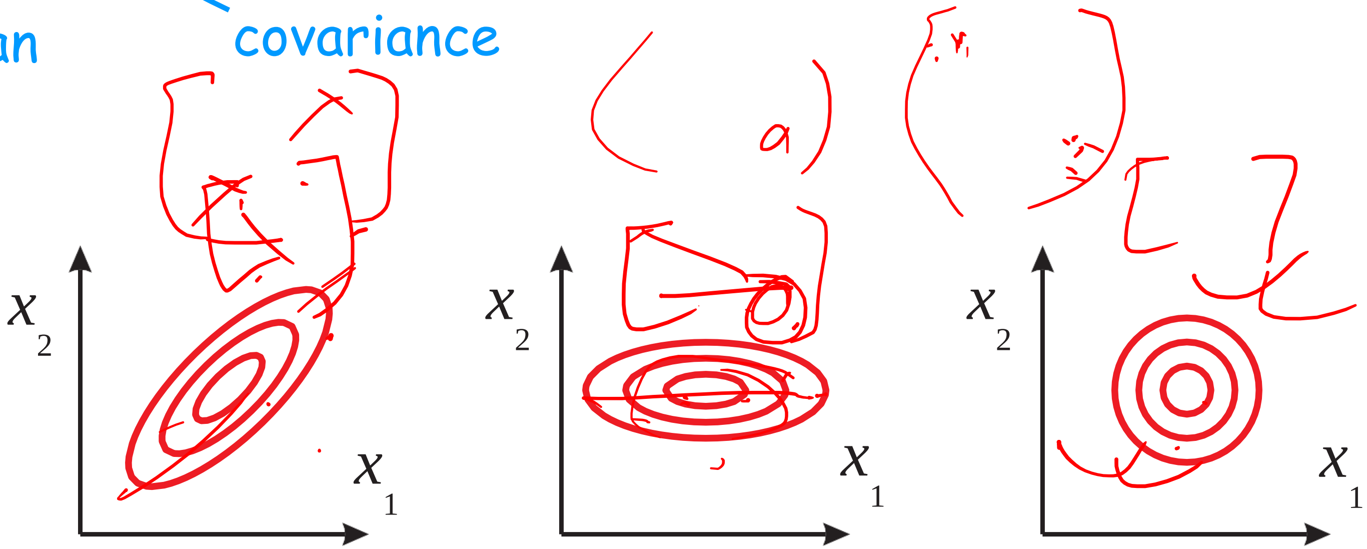
The Gaussian Distribution

- Multivariate Gaussian

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

mean

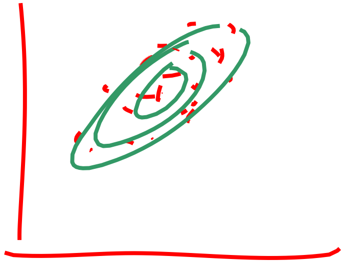
covariance



(a) (b) (c)

- Data set

$$D = \{\mathbf{x}_n\} \qquad n = 1, \ldots, N$$



- Consider first a single Gaussian
- Assume observed data points generated independently

Like

$$p(D|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \prod_{n=1}^{N} \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

- Viewed as a function of the parameters, this is known as the *likelihood function*

Log likelihood

Mixtures of Gaussians

K # of

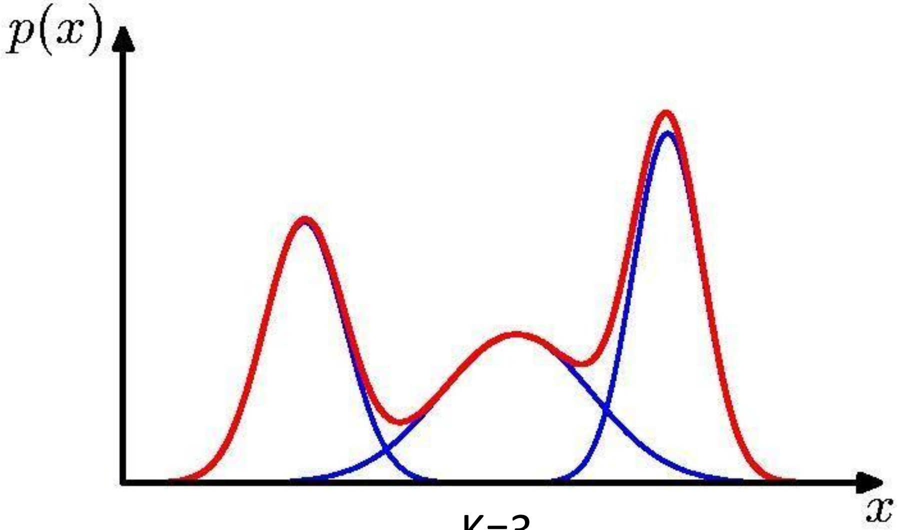
- Combine simple models into a complex model:

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

Component

Mixing coefficient

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



K=3

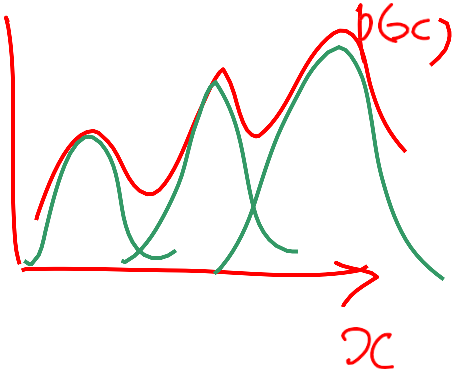
- Find parameters through EM (Expectation Maximization) algorithm

Adapted from Chris Bishop

Gaussian Mixture Model

- Linear super-position of Gaussians

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



- Normalization and positivity require

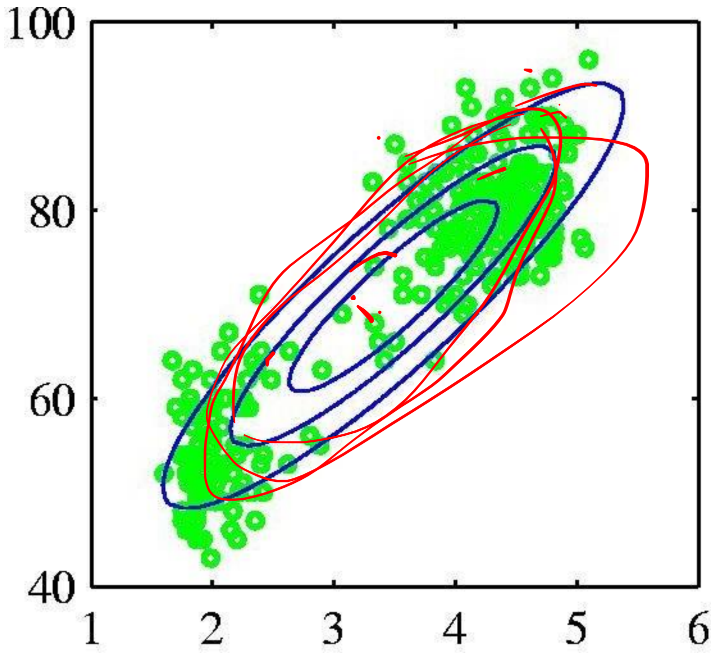
$$\sum_{k=1}^{K} \pi_k = 1 \qquad 0 \leqslant \pi_k \leqslant 1$$

- Can interpret the mixing coefficients as prior probabilities

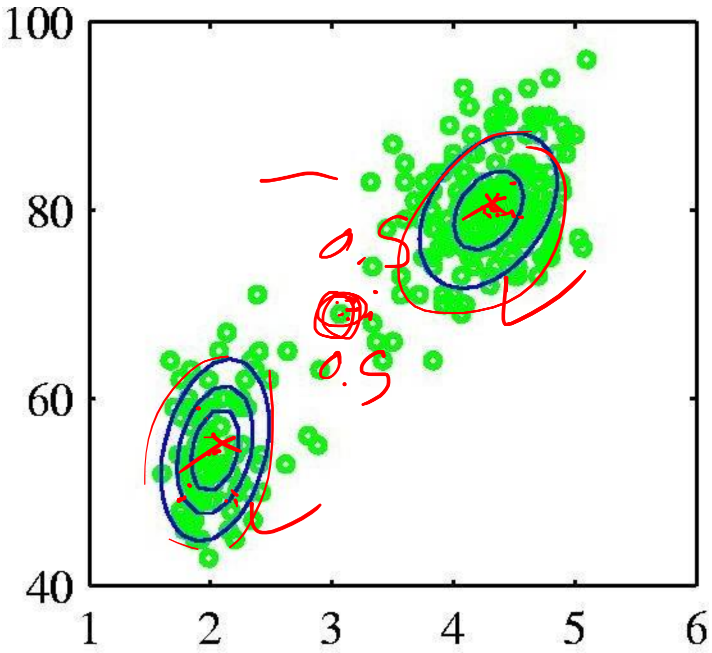
$$p(\mathbf{x}) = \sum_{k=1}^{K} p(k) p(\mathbf{x}|k)$$

Probabilistic version: Mixtures of Gaussians

- Old Faithful data set



Single Gaussian



Mixture of two Gaussians

Chris Bishop

Sampling from the Mixture of Gaussian

- To generate a data point:
  - first pick one of the components with probability π_k
  - then draw a sample $\underline{\mathbf{x}}_n$ from that component
- Repeat these two steps for each new data point

Fitting the Gaussian Mixture

- We wish to invert this process – given the data set, find the corresponding parameters:
  - mixing coefficients π_k
  - means ✓ μ_k
  - covariances ✓ Σ_k
- If we knew which component generated each data point, the maximum likelihood solution would involve fitting each component to the corresponding cluster
- Problem: the data set is unlabelled
- We shall refer to the labels as *latent* (= hidden) variables

Gaussian Mixture Model

model model model model, model,

- K-dimensional binary random variable z having a 1-of-K representation in which a particular element z_k is equal to 1 and all other elements are equal to 0.
- The values of z_k therefore satisfy $z_k \in \{0,1\}$
- K possible states for the vector z according to which element is nonzero.
- Joint distribution p(x,z) in terms of a marginal distribution p(z) and a conditional distribution p(x|z),
- Marginal distribution over z is specified in terms of the mixing coefficients π_k, such that $p(z_k = 1) = \pi_k$

z_1

Gaussian Mixture Model

- Joint distribution is given by p(z)p(x|z), and the marginal distribution of x is then obtained by summing the joint distribution over all possible states of z to given

μ_k, Σ_k

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

$Z_k = 1 \longrightarrow \theta$ model

Gaussian Mixture Model

- Conditional probability of z given x
- use γ(z_k) to denote p(z_k =1 | x), whose value can be found using Bayes’ theorem

$$\gamma(z_k) \equiv p(z_k = 1|\mathbf{x}) = \frac{p(z_k = 1)p(\mathbf{x}|z_k = 1)}{\underbrace{\sum_{j=1}^{K} p(z_j = 1)p(\mathbf{x}|z_j = 1)}_{p(x)}} = \frac{\pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}.$$

$\gamma(z_1)$

$\gamma(z_2)$ $\gamma(z_k)$

$\gamma(z_K)$

- π_k as the prior probability of z_k =1, and the quantity γ(z_k) as the corresponding posterior probability once we have observed x.

Maximum Likelihood

Log of likelihood function:

$$\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

- Maximizing the log likelihood function for a Gaussian mixture model turns out to be a more complex problem than for the case of a single Gaussian.
- The difficulty arises from the presence of the summation over k that appears inside the logarithm, so that the logarithm function no longer acts directly on the Gaussian.

EM Algorithm – Informal Derivation

- The solutions are not closed form since they are coupled
- Suggests an iterative scheme for solving them:
  - make initial guesses for the parameters
  - alternate between the following two stages:
    1. E-step: evaluate responsibilities
    2. M-step: update parameters using ML results

$\gamma(z_k)$ ∀ all k, x_n, n ..

$\rightarrow \mu, \Sigma, \pi$

- Each EM cycle guaranteed not to decrease the likelihood

Expectation Maximization (EM) Algorithm

'$\lambda_k, \mu_k, \Sigma_k$''

- Conditions for MLE: Setting the derivatives of ln p(X|π,μ,Σ)

in with respect to the means μk of the Gaussian components to zero, we obtain

$$0 = -\sum_{n=1}^{N} \underbrace{\frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}}_{\gamma(z_{nk})} \boldsymbol{\Sigma}_k(\mathbf{x}_n - \boldsymbol{\mu}_k)$$

+

rearranging we obtain:

$$\boxed{\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n}$$

where we have defined $$\boxed{\boxed{N_k} = \sum_{n=1}^{N} \gamma(z_{nk})}$$

Expectation Maximization (EM) Algorithm

- μ_k for the kth Gaussian component is obtained by taking a weighted mean of all of the points in the data set
- Weighting factor for data point x_n is given by the posterior probability $\gamma(z_{nk})$ that component k was responsible for generating xn
- If we set the derivative of ln p(X|π,μ,Σ) with respect to Σk to 0, and follow a similar line of reasoning, making use of the result for the maximum likelihood solution for the covariance matrix of a single Gaussian

$$\boxed{\Sigma_k = \frac{1}{N_k}\sum_{n=1}^{N}\gamma(z_{nk})(\mathbf{x}_n-\boldsymbol{\mu}_k)(\mathbf{x}_n-\boldsymbol{\mu}_k)^{\mathrm{T}}}$$

Expectation Maximization (EM) Algorithm

- Maximize ln p(X|π,μ,Σ) with respect to the mixing coefficients $\underline{\underline{\pi_k}}$ with constraint $\sum_{k=1}^{K} \pi_k = 1 \qquad 0 \leqslant \pi_k \leqslant 1$

- Using a Lagrange multiplier and maximizing the following quantity

$$\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) + \lambda \left(\sum_{k=1}^{K} \pi_k - 1 \right)$$

which gives

$$\boxed{0 = \sum_{n=1}^{N} \frac{\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} + \lambda}$$

- If we now multiply both sides by πk and sum over k, we find λ = −N.
- Rearranging we obtain

$$\boxed{\pi_k = \frac{N_k}{N}}$$

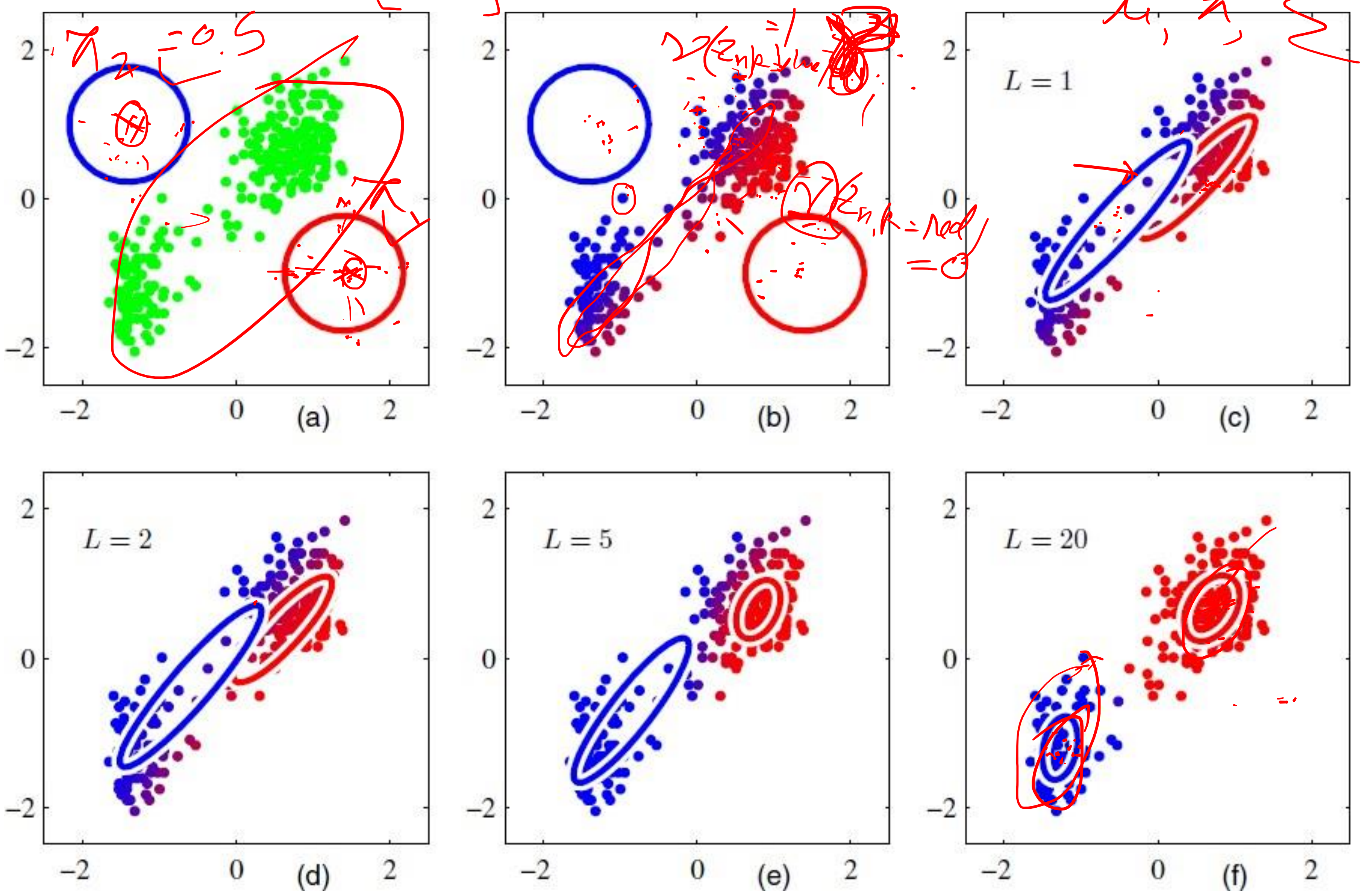
Expectation Maximization (EM) Algorithm

μ

- We first choose some initial values for the means, covariances, and mixing coefficients. Σ π
- Then we alternate between the following two updates that we shall call the E step and the M step
- In the expectation step, or E step, we use the current values for the parameters to evaluate the posterior probabilities, $\boxed{\gamma(z_{nk})}$
- We then use these probabilities in the maximization step, or M step, to re-estimate the means, covariances, and mixing π μ Σ $\sum_{k} \{ \}$
- In practice, the algorithm is deemed to have converged when the change in the log likelihood function, or alternatively in the parameters, falls below some threshold

$e^{-\frac{(x_n-\mu)^2}{2\sigma^2}}$ Initialization $\begin{bmatrix} 1 & \\ & 1 \end{bmatrix}$ E step M step

μ, π, Σ



Figures from Chris Bishop

EM algorithm for GMM

1. Initialize the means μ_k, covariances Σ_k and mixing coefficients π_k, and evaluate the initial value of the log likelihood.
2. **E step:** Evaluate the responsibilities using the current parameter values

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

EM algorithm for GMM

3. **M step:** Re-estimate the parameters using the current responsibilities

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n$$

$$\boldsymbol{\Sigma}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \left(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}}\right) \left(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}}\right)^{\mathrm{T}}$$

$$\pi_k^{\text{new}} = \frac{N_k}{N}$$

where $N_k = \sum_{n=1}^{N} \gamma(z_{nk})$

4. Evaluate the log likelihood

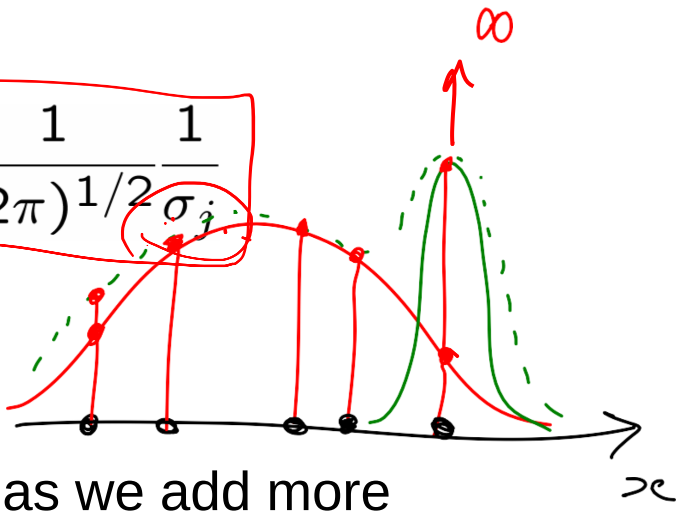
$$\ln p(\mathbf{X}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

Over-fitting in Gaussian Mixture Models

- Singularities in likelihood function when a component ‘collapses’ onto a data point:

$$\boxed{\mathcal{N}(\mathbf{x}_n|\mathbf{x}_n,\sigma_j^2\mathbf{I}) = \frac{1}{(2\pi)^{1/2}}\frac{1}{\sigma_j}}$$

then consider $\boxed{\sigma_j \to 0}$



- Likelihood function gets larger as we add more components (and hence parameters) to the model
  - – not clear how to choose the number K of components

Let $(x_1, x_2, x_3) = (2, 4, 7)$ be our three datapoints, presumed to have each been generated from one of two Gaussians.

The stdev of both Gaussians are given: $\sigma_1 = \sigma_2 = 1/\sqrt{2}$.
The prior over the two Gaussians in also given: $\Pi_1 = \Pi_2 = 0.5$

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

Let us initialize the Gaussian means to some reasonable values (inside the data range, and integer valued, to make calculation easy): $\mu_1^{[0]} = 3$, $\mu_2^{[0]} = 6$.

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n$$

$$\boldsymbol{\Sigma}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \left(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}}\right) \left(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}}\right)^{\mathrm{T}}$$

$$\pi_k^{\text{new}} = \frac{N_k}{N}$$

$\alpha_1 = \alpha_2 = \frac{1}{\sqrt{2}}$ $\mu_1 = 3$ $\mu_2 = 6$

$\frac{1}{\sqrt{2\pi \frac{1}{\sqrt{2}}}} e^{-\frac{(2-3)^2}{2\left(\frac{1}{\sqrt{2}}\right)^2}}$

i	1	2	3
x_i	2.0	4.0	7.0
N(xi|μ1)	$\frac{1}{\sqrt{\pi}}e^{-1}$	$\frac{1}{\sqrt{\pi}}e^{-1}$	$\frac{1}{\sqrt{\pi}}e^{-16}$
N(xi|μ2)	$\frac{1}{\sqrt{\pi}}e^{-16}$	$\frac{1}{\sqrt{\pi}}e^{-4}$	$\frac{1}{\sqrt{\pi}}e^{-1}$
N(xi|μ1)+**N**(xi|μ2)	$\frac{1}{2\sqrt{\pi}}(e^{-1}+e^{-16})$	$\frac{1}{2\sqrt{\pi}}(e^{-1}+e^{-4})$	$\frac{1}{2\sqrt{\pi}}(e^{-16}+e^{-1})$
γ(zi,1)	$\frac{e^{-1}}{e^{-1}+e^{-16}} \approx 1$	$\frac{e^{-1}}{e^{-1}+e^{-4}} \approx 0.953$	$\frac{e^{-16}}{e^{-1}+e^{-16}} \approx 0$
γ(zi,2)	$\frac{e^{-16}}{e^{-1}+e^{-16}} \approx 0$	$\frac{e^{-4}}{e^{-1}+e^{-4}} \approx 0.047$	$\frac{e^{-1}}{e^{-1}+e^{-16}} \approx 1$

And therefore:

$$\mu_1^{[1]} \approx \frac{1 * 2.0 + 0.953 * 4.0 + 0 * 7.0}{1 + 0.953} \approx 2.976$$

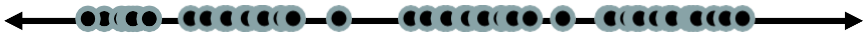
and

$$\mu_2^{[1]} \approx \frac{0 * 2.0 + 0.047 * 4.0 + 1 * 7.0}{1 + 0.047} \approx 6.865$$

Segmentation as clustering

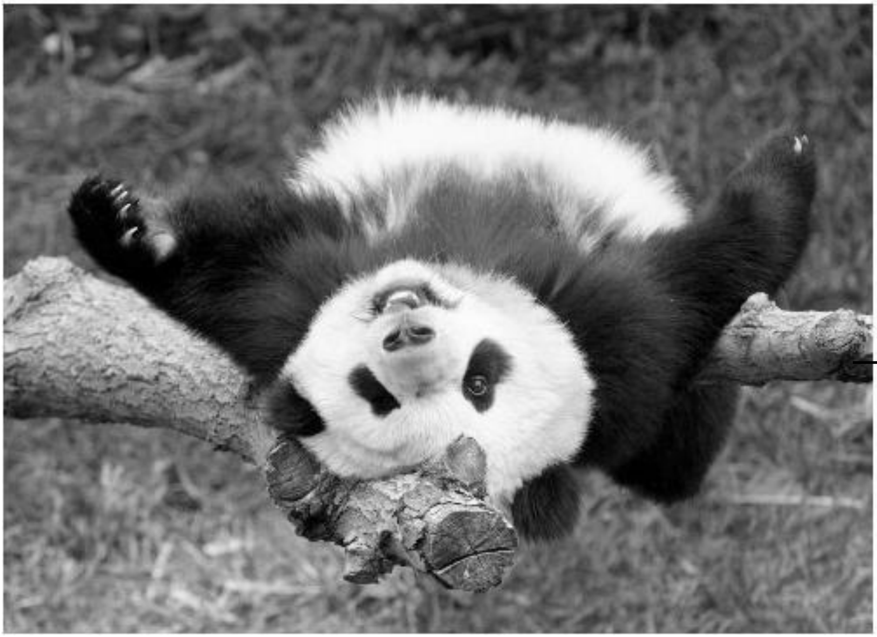
Depending on what we choose as the *feature space*, we can group pixels in different ways.

Grouping pixels based on **intensity** similarity

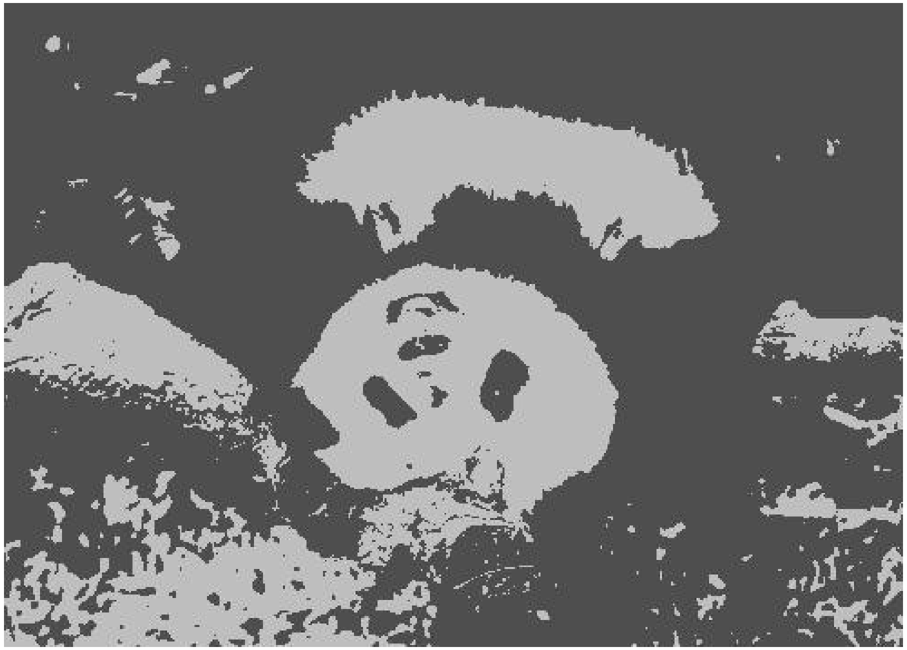


Feature space: intensity value (1-d)

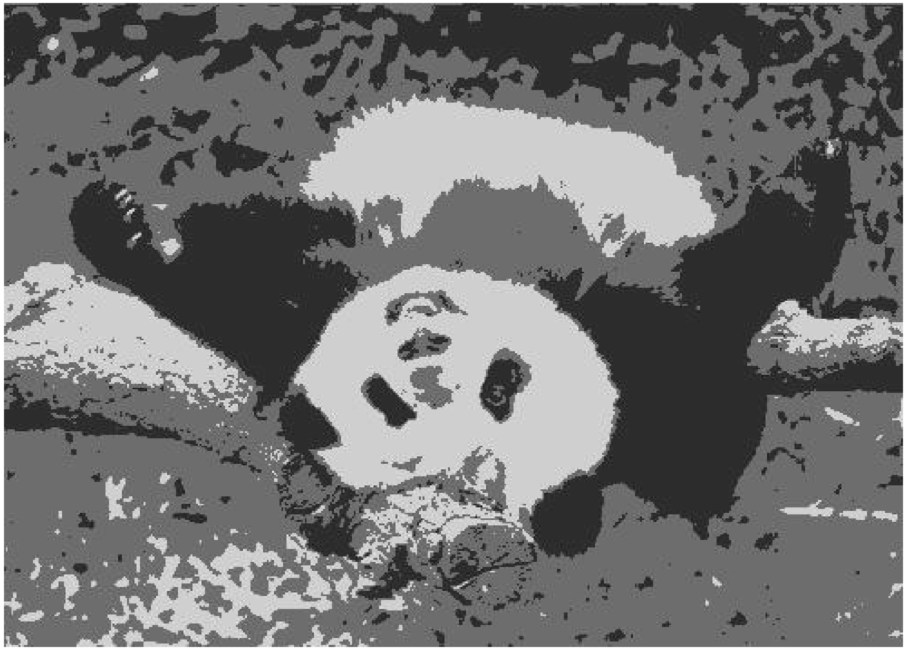
Source: K. Grauman



K=2



K=3



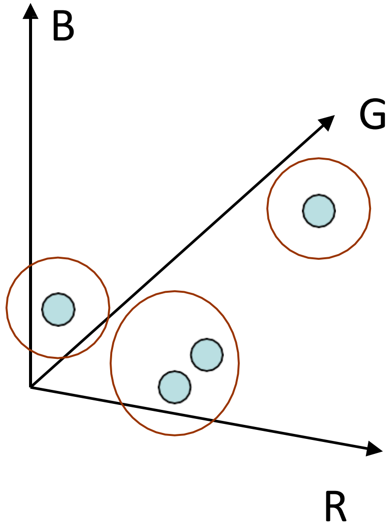
quantization of the feature space; segmentation label map

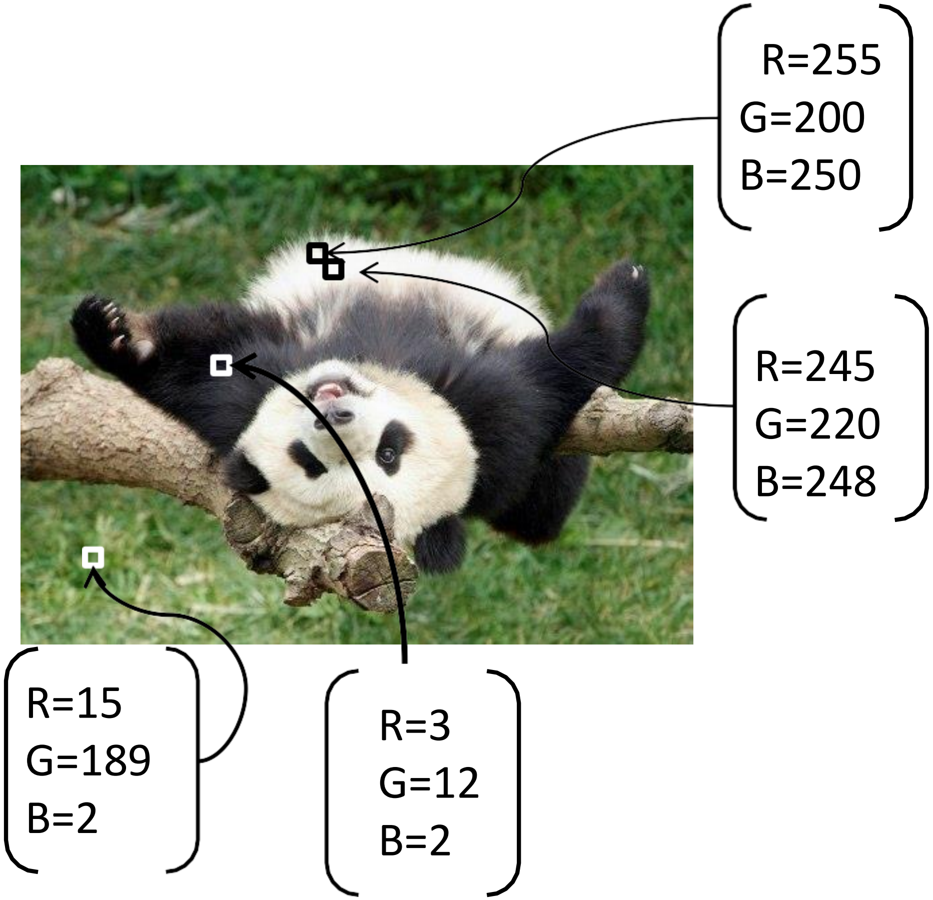
Source: K. Grauman

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





R=255
G=200
B=250

R=245
G=220
B=248

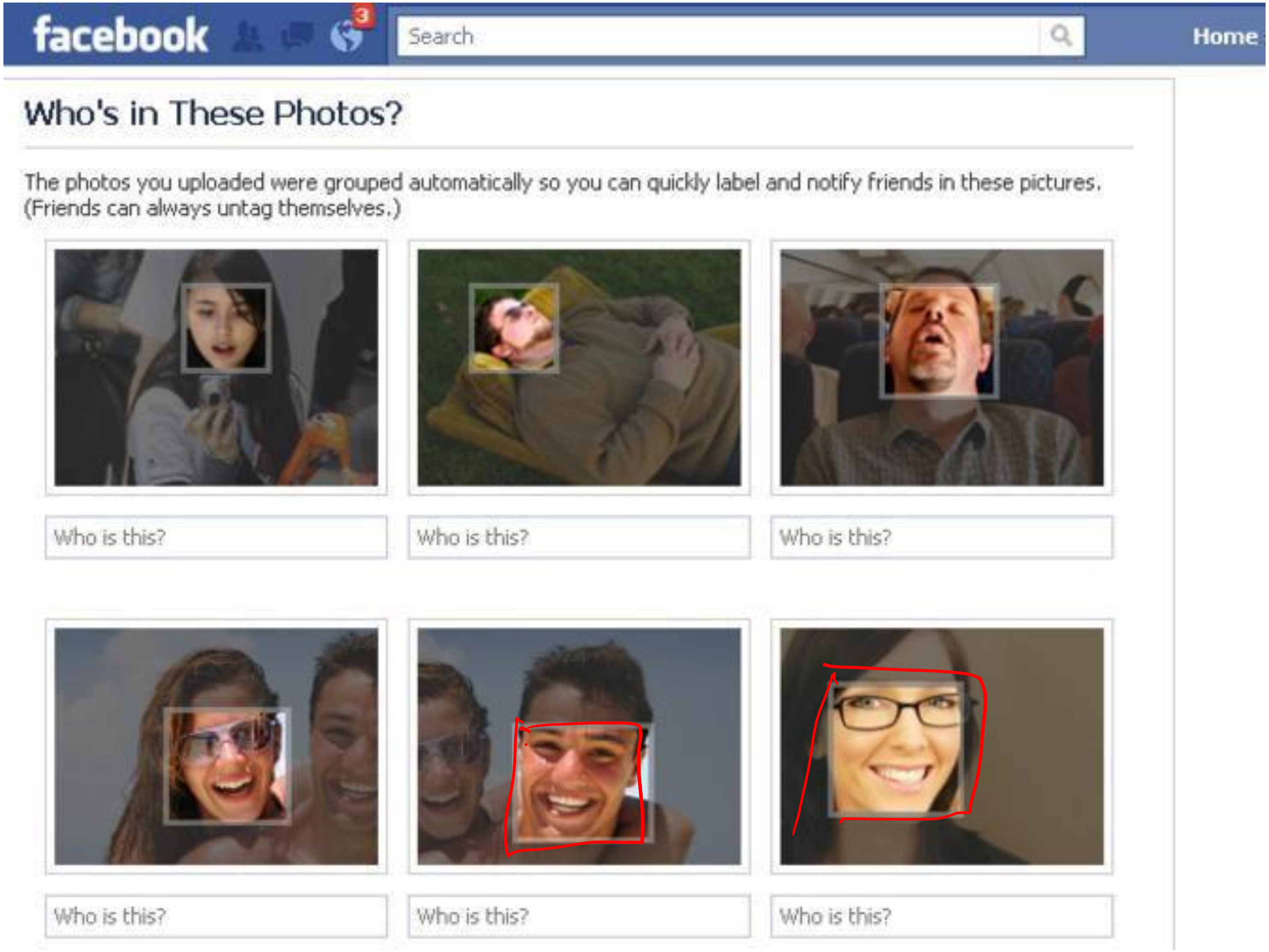
R=15
G=189
B=2

R=3
G=12
B=2

Feature space: color value (3-d)

Source: K. Grauman

Application: Face Recognition



Who's in These Photos?

The photos you uploaded were grouped automatically so you can quickly label and notify friends in these pictures. (Friends can always untag themselves.)

Image from cnet.com

References

Christopher Bishop: Pattern Recognition and Machine Learning, Springer International Edition

https://www.youtube.com/watch?v=TG6Bh-NFhA0

https://www.youtube.com/watch?v=qMTuMa86NzU