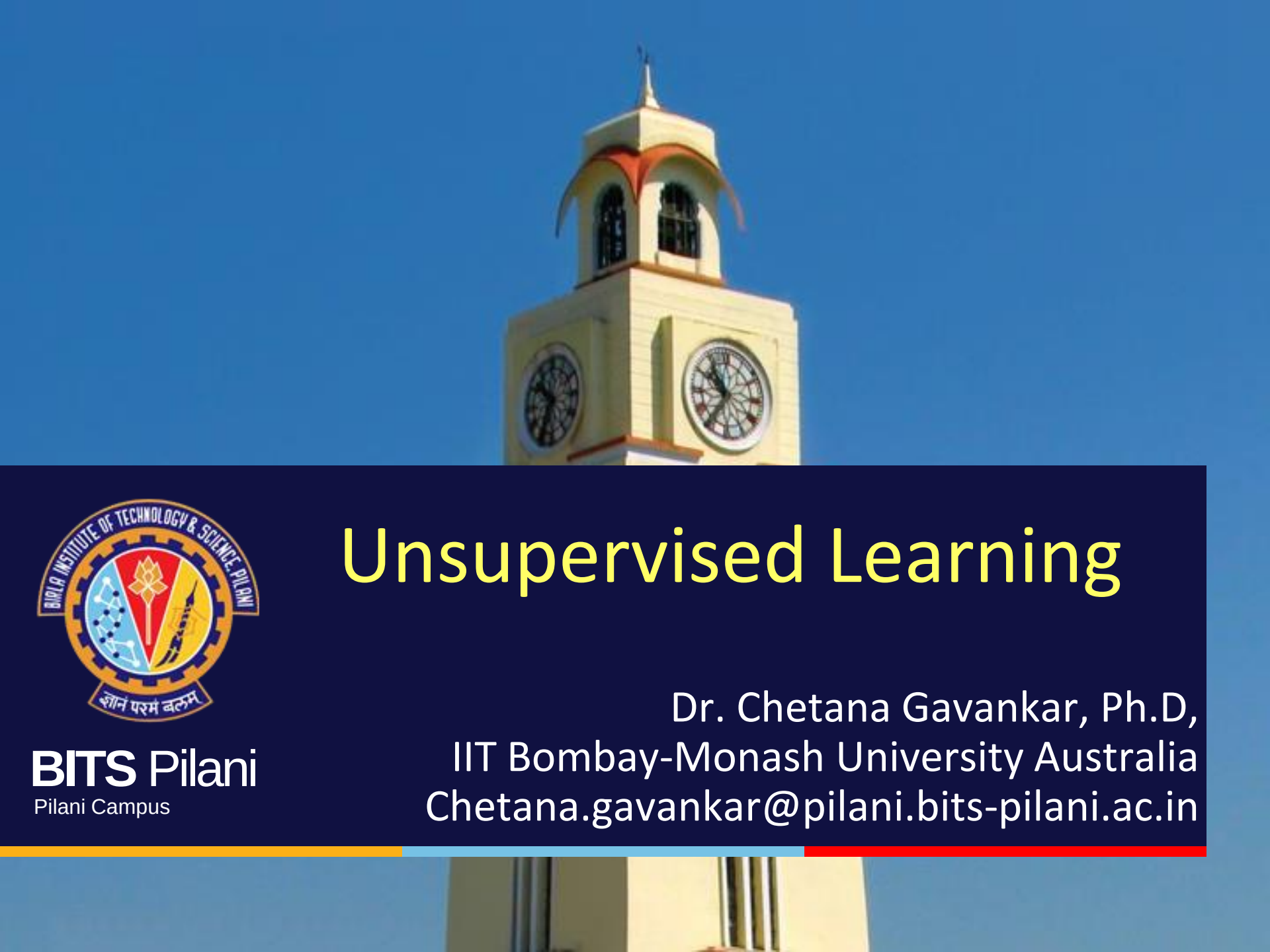


# Unsupervised Learning


Dr. Chetana Gavankar, Ph.D,
IIT Bombay-Monash University Australia
Chetana.gavankar@pilani.bits-pilani.ac.in

BITS Pilani
Pilani Campus

innovate achieve lead

BITS Pilani

Pilani Campus

## Text Book(s)

| | |
|---|---|
| T1 | Christopher Bishop: Pattern Recognition and Machine Learning, Springer International Edition |
| T2 | Tom M. Mitchell: Machine Learning, The McGraw-Hill Companies, Inc.. |

**These slides are prepared by the instructor, with grateful acknowledgement of Prof. Bishop and many others who made their course materials freely available online.**

# Topics to be covered

**Ref: Christopher Bishop: Chapter 9**

innovate achieve lead

- 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

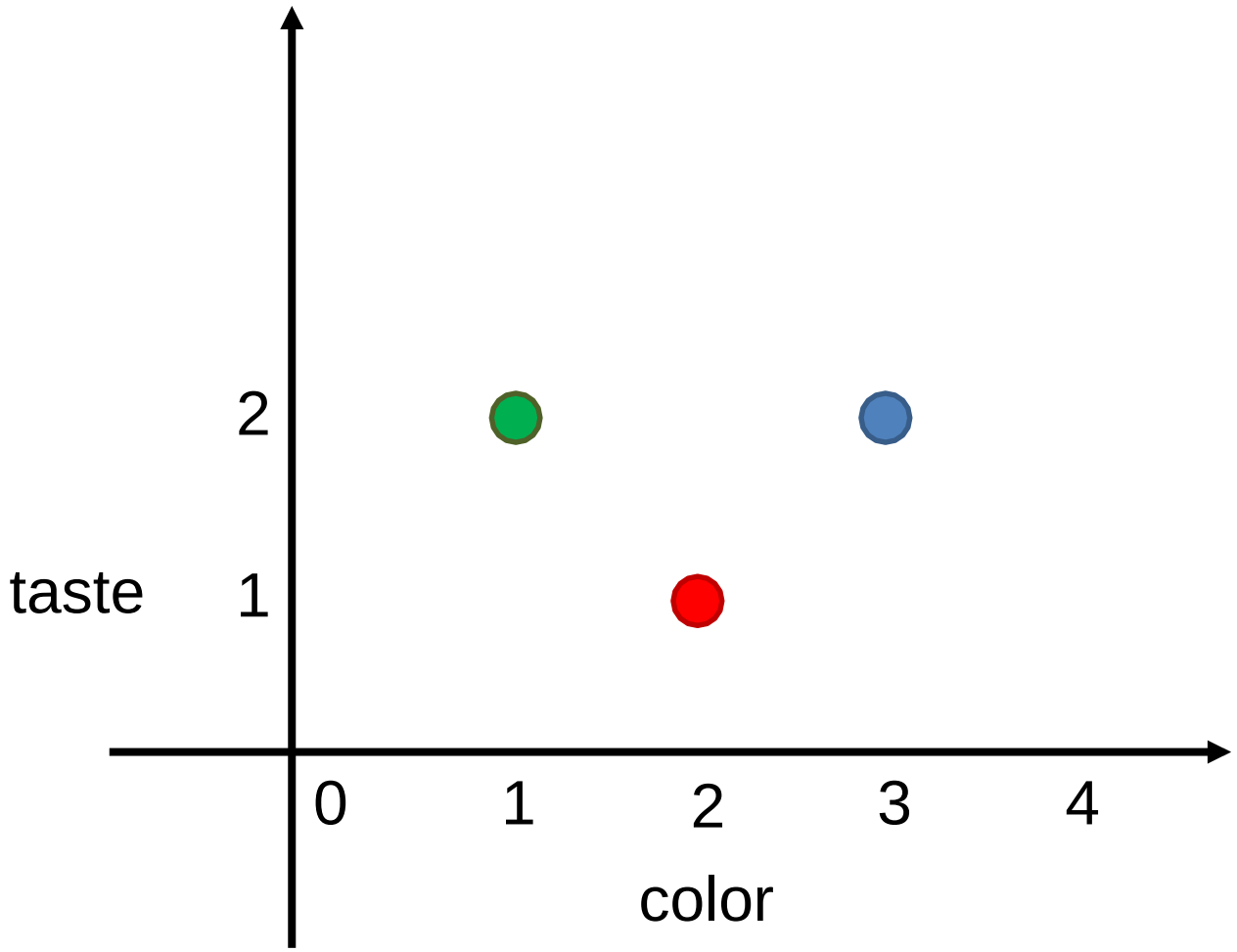
# What is clustering?

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

# Feature representation (x)

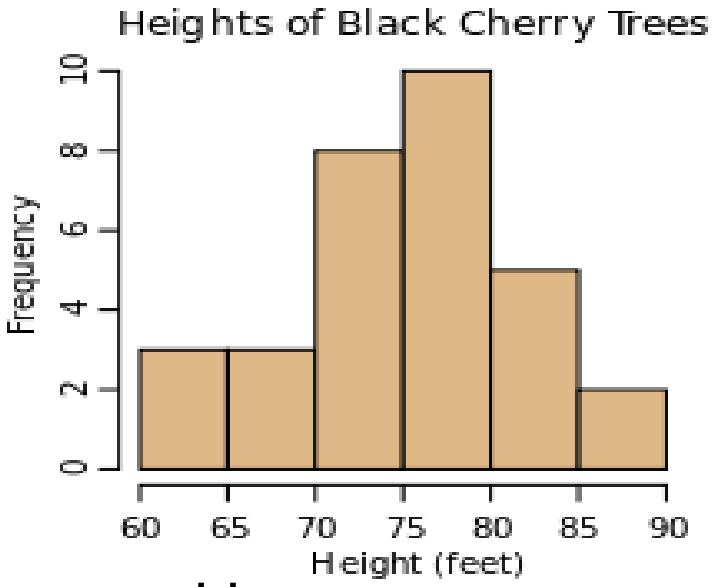
- A vector representing measurable characteristics of a data sample we have
- E.g. a glass of juice can be represented via its color = {yellow=1, red=2, green=3, purple=4} and taste = {sweet=1, sour=2}
- For a given glass i, this can be represented as a vector: xi = [3 2] represents sour green juice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

# Feature representation (x)



# Why do we cluster?

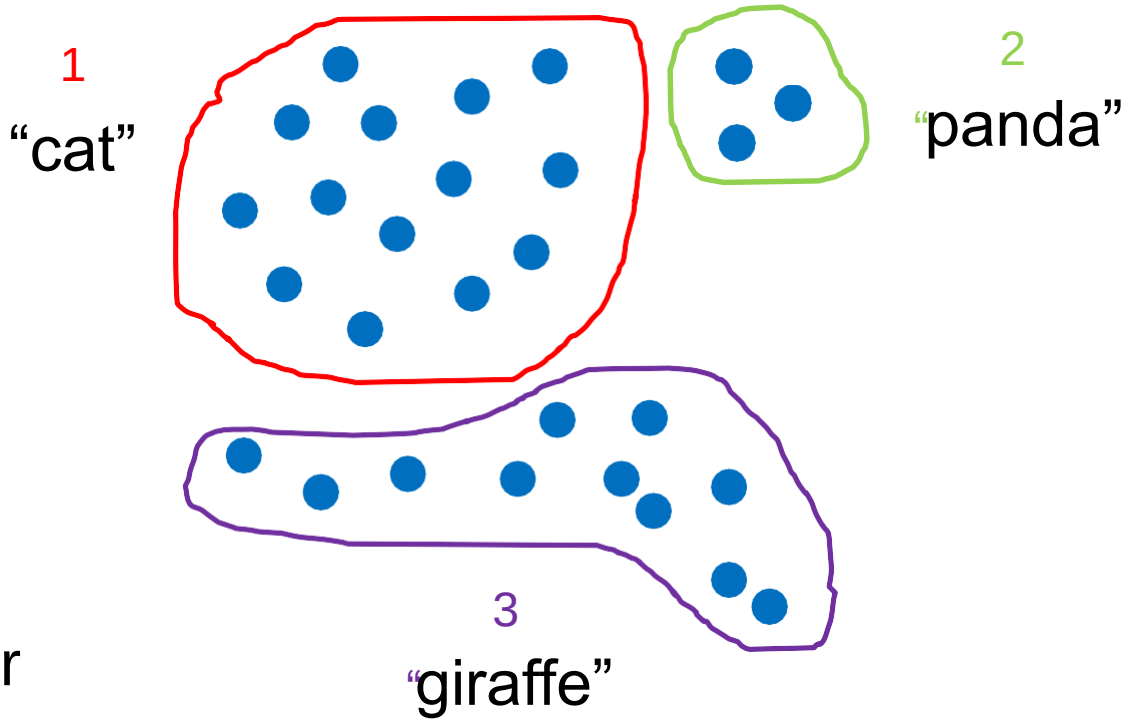
- Counting
  - Feature histograms: by grouping similar features and counting how many of each a data sample has
- Summarizing data
  - Look at large amounts of data
  - Represent a large continuous vector with the cluster number
- Prediction
  - Data points in the same cluster may have the same labels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



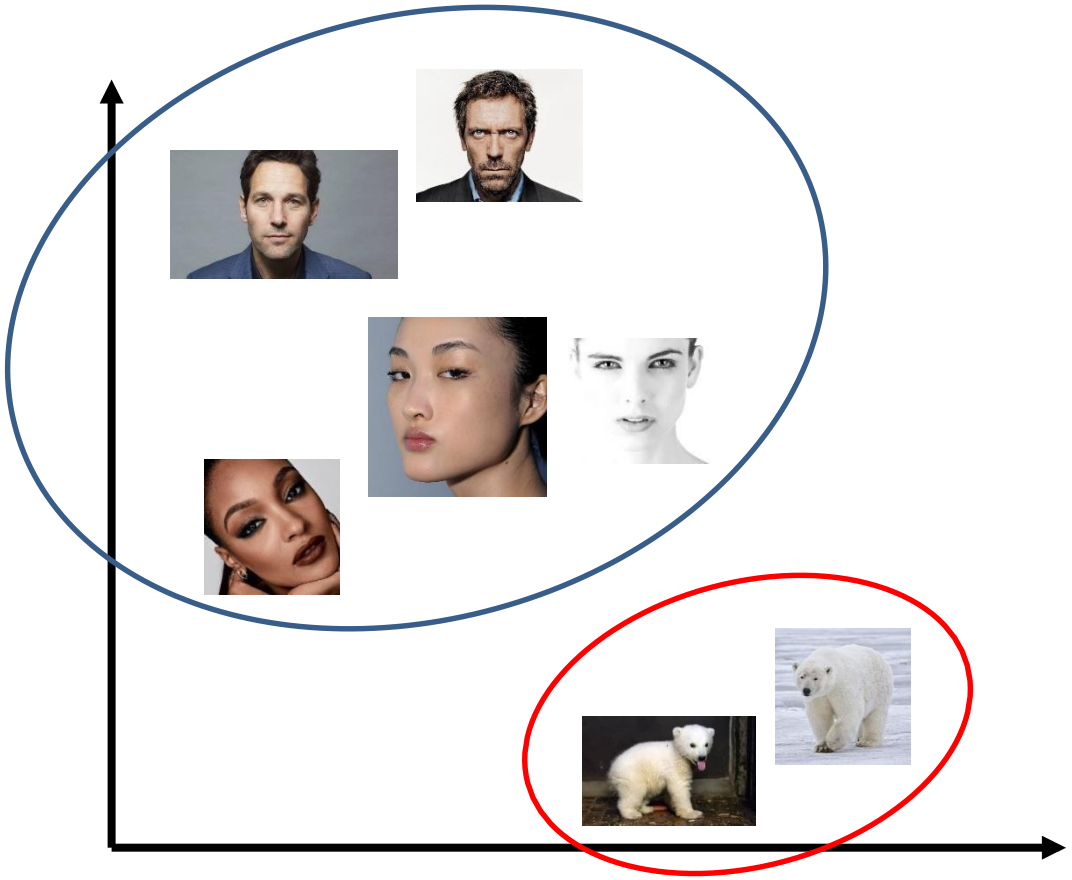

Slide credit: J. Hays, D. Hoiem

# 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

# Unsupervised discovery



# Clustering algorithms

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

# K-means Algorithm

- Goal: represent a data set in terms of $K$ clusters each of which is summarized by a prototype
- 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

# K-means Algorithm

Randomly initialize $K$ cluster centroids $\mu_1, \mu_2, \ldots, \mu_K \in \mathbb{R}^n$

Repeat {

Cluster assignment step

for $i = 1$ to $m$

$c^{(i)}$ := index (from 1 to $K$) of cluster centroid closest to $x^{(i)}$

$\min_k \|x^{(i)} - \mu_k\|^2 \rightarrow c^{(i)}$

Move centroid

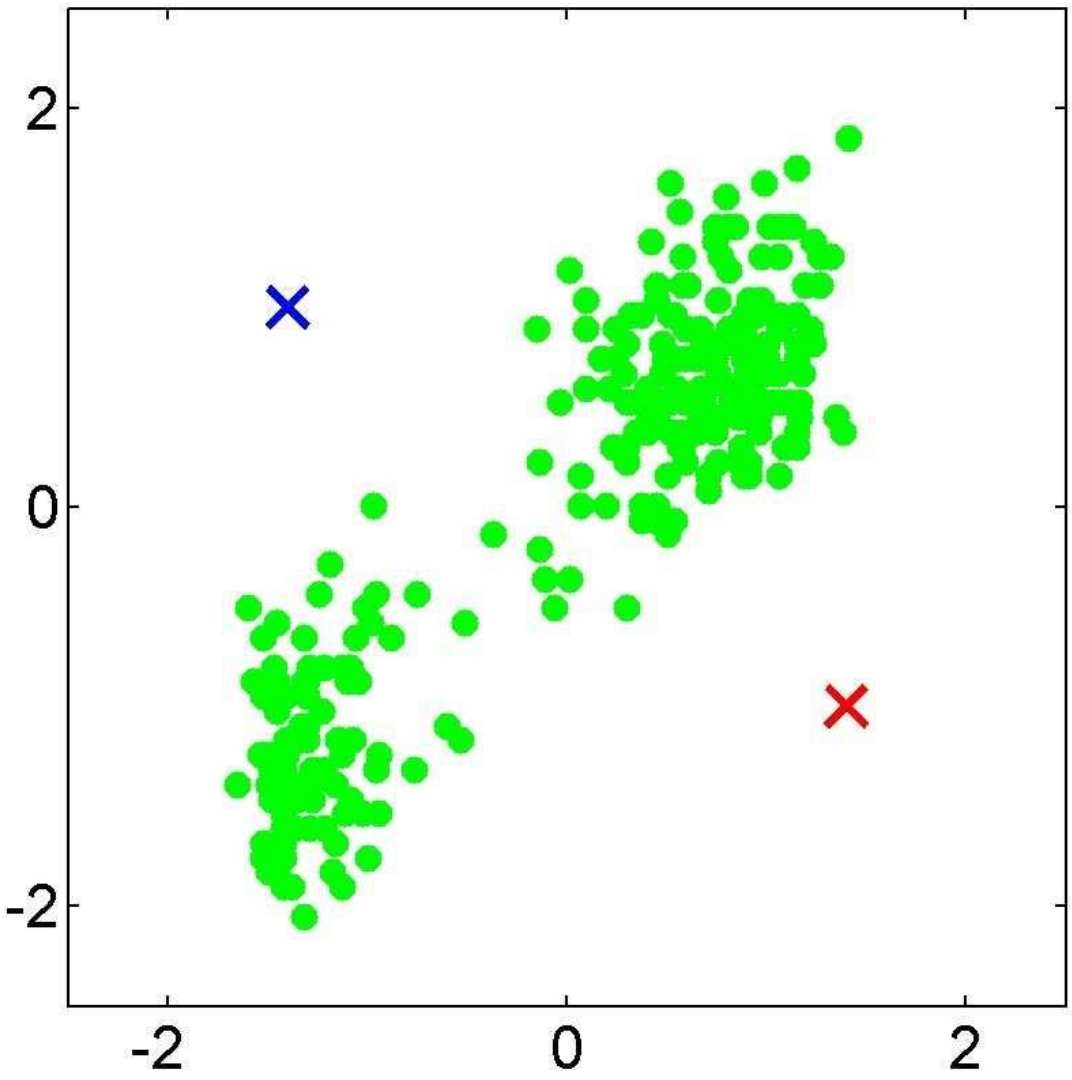
for $k = 1$ to $K$

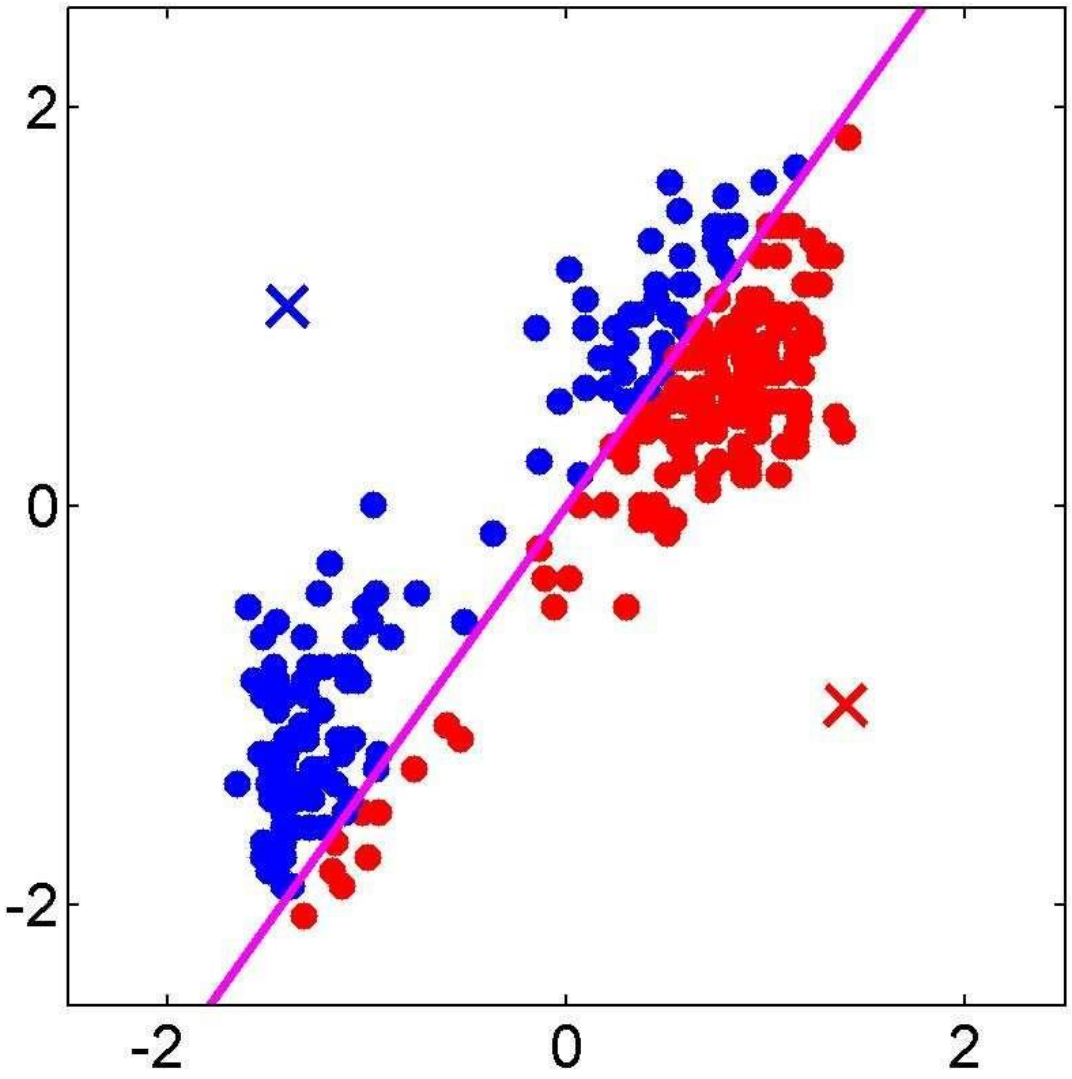
$\mu_k$ := average (mean) of points assigned to cluster $k$

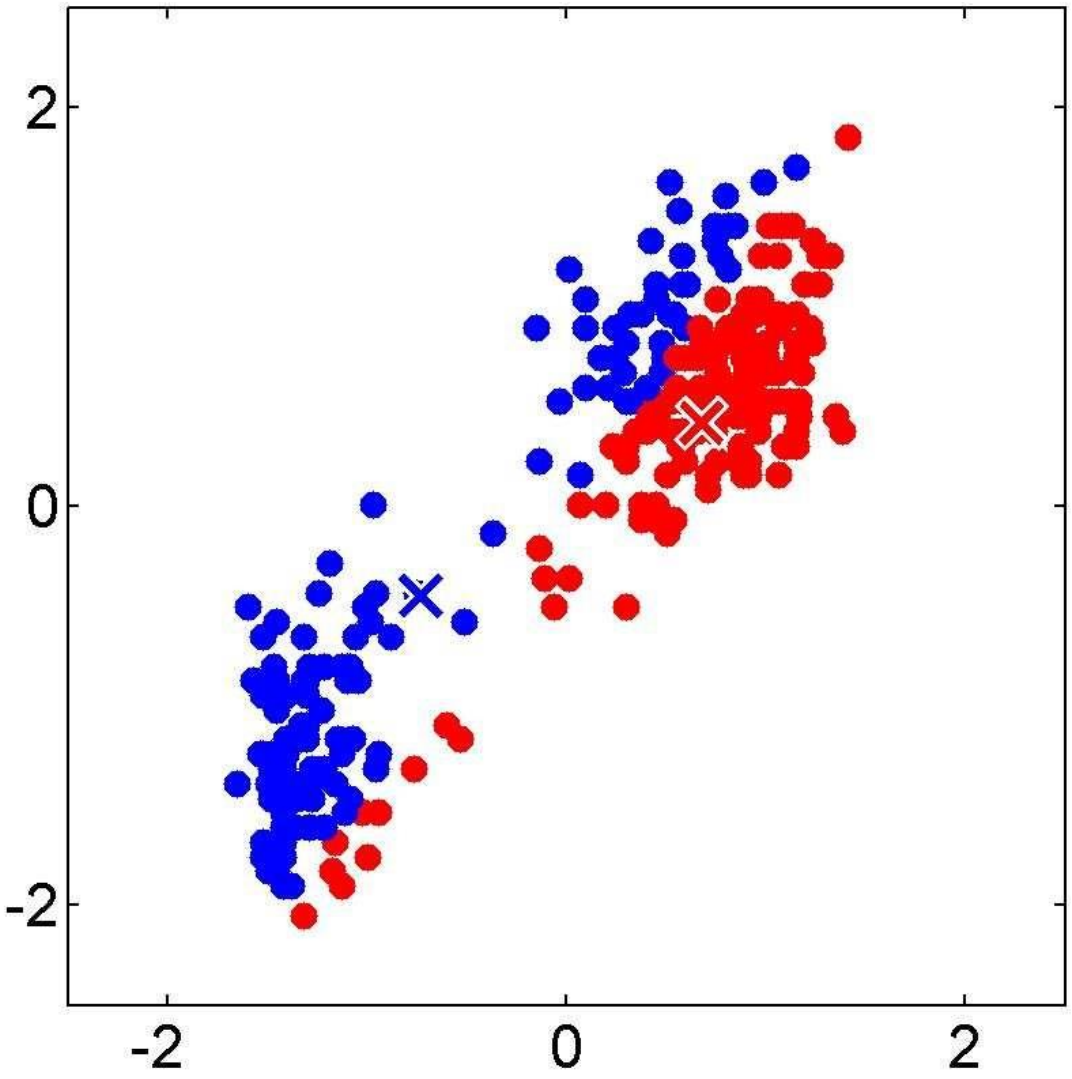
$x^{(1)}, x^{(5)}, x^{(6)}, x^{(10)} \Rightarrow c^{(1)}=2,\ c^{(5)}=2,\ c^{(6)}=2,\ c^{(10)}=2$

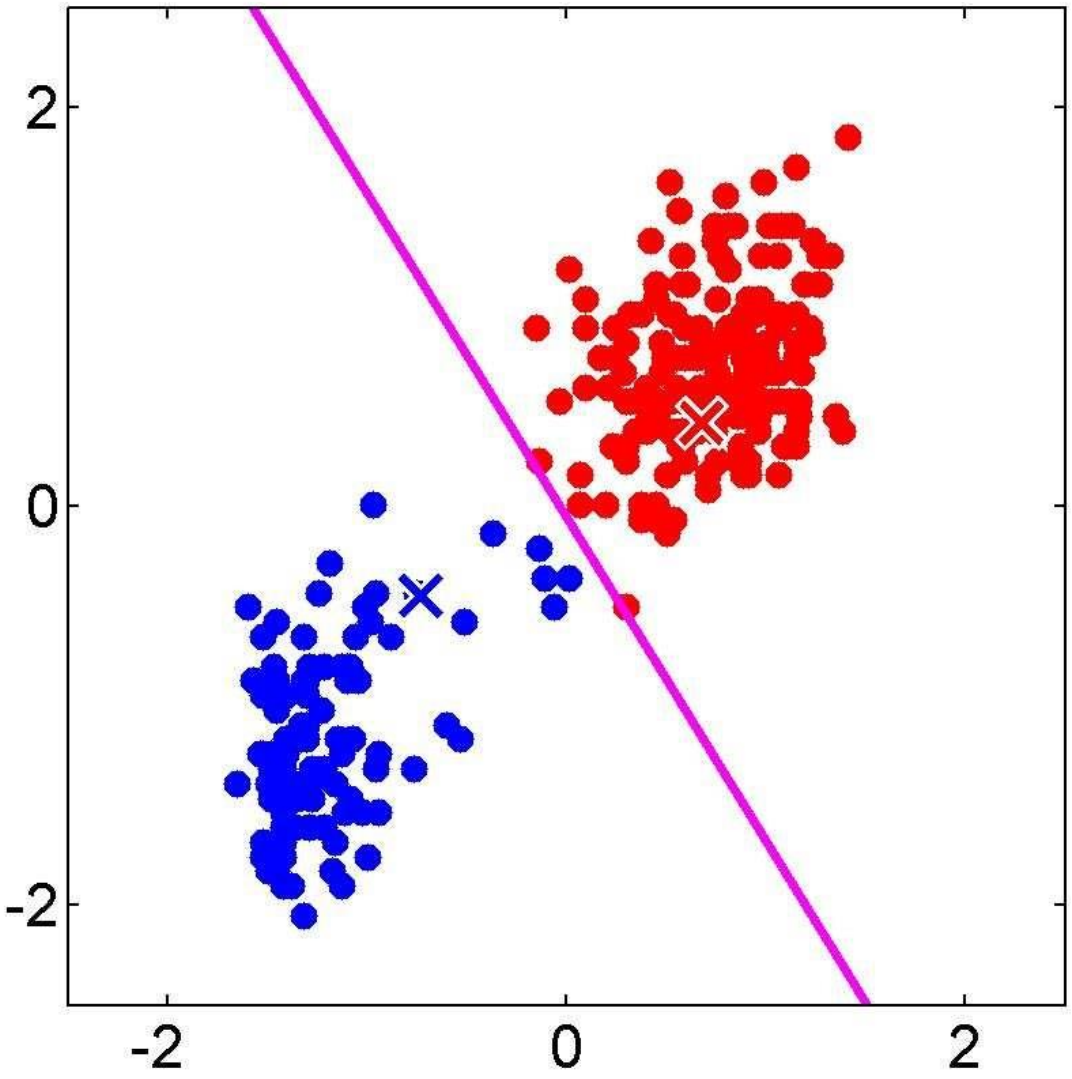
$$\mu_2 = \frac{1}{4}\left[x^{(1)} + x^{(5)} + x^{(6)} + x^{(10)}\right] \in \mathbb{R}^n$$

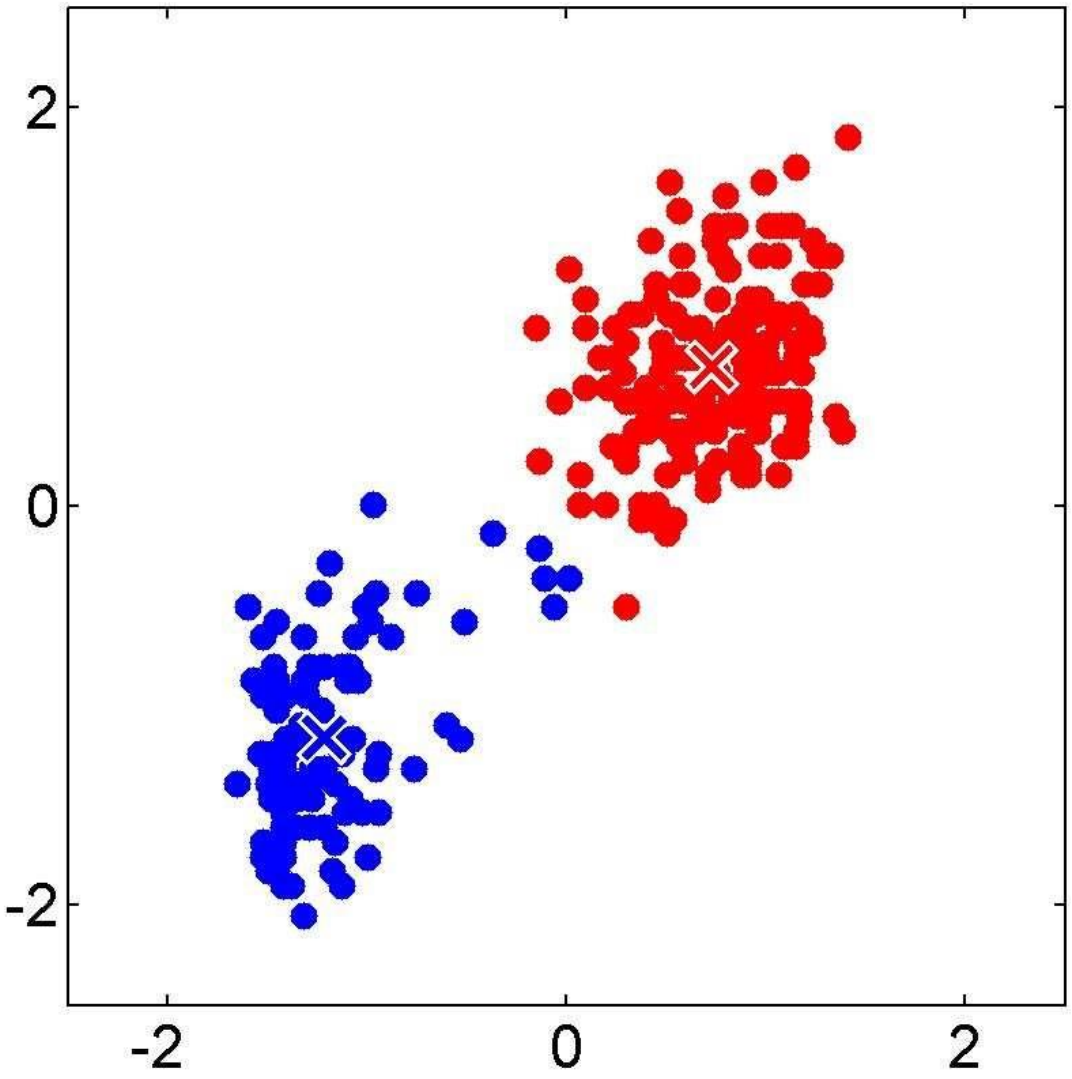
}

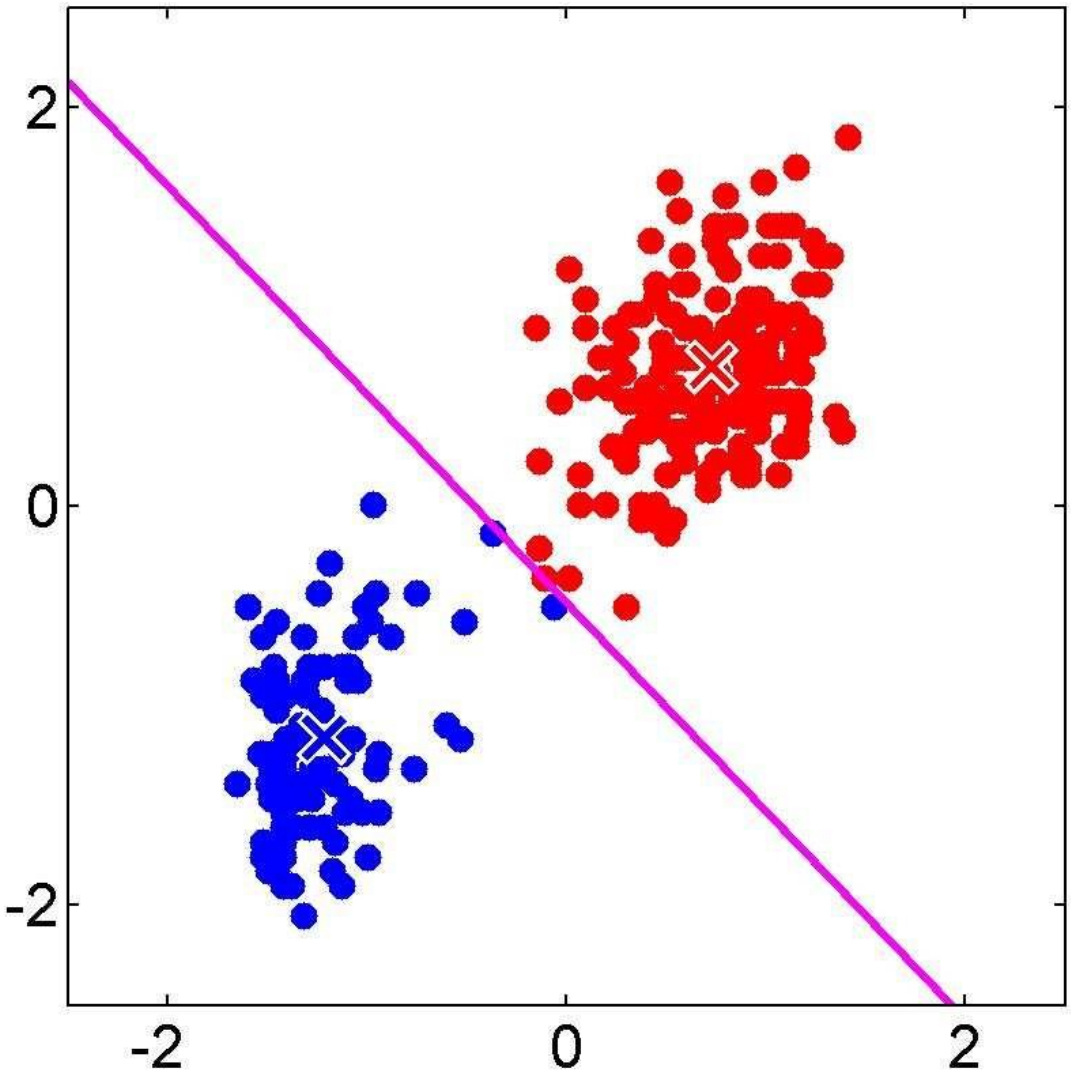

2
0
-2
-2
0
2

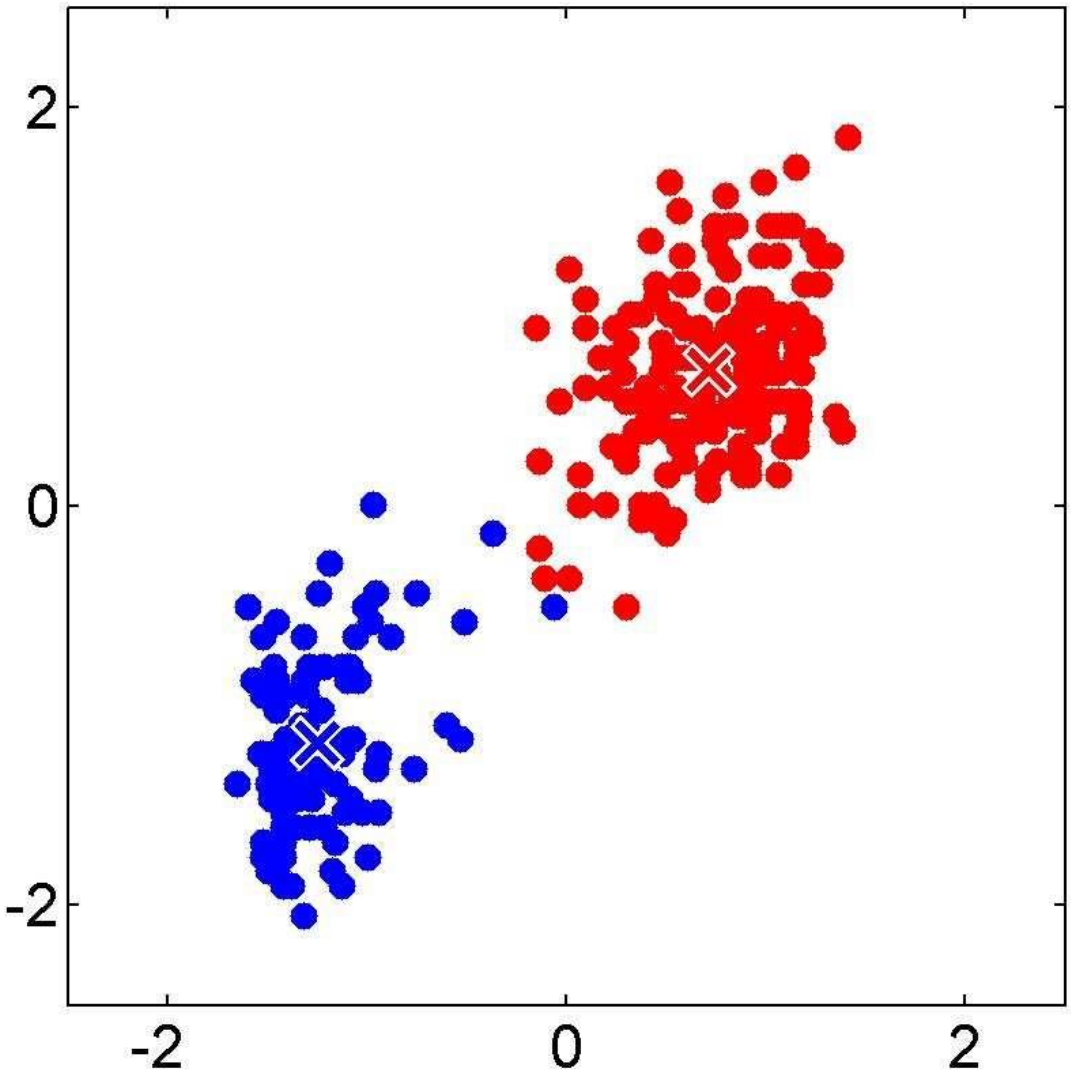

2
0
-2
-2
0
2

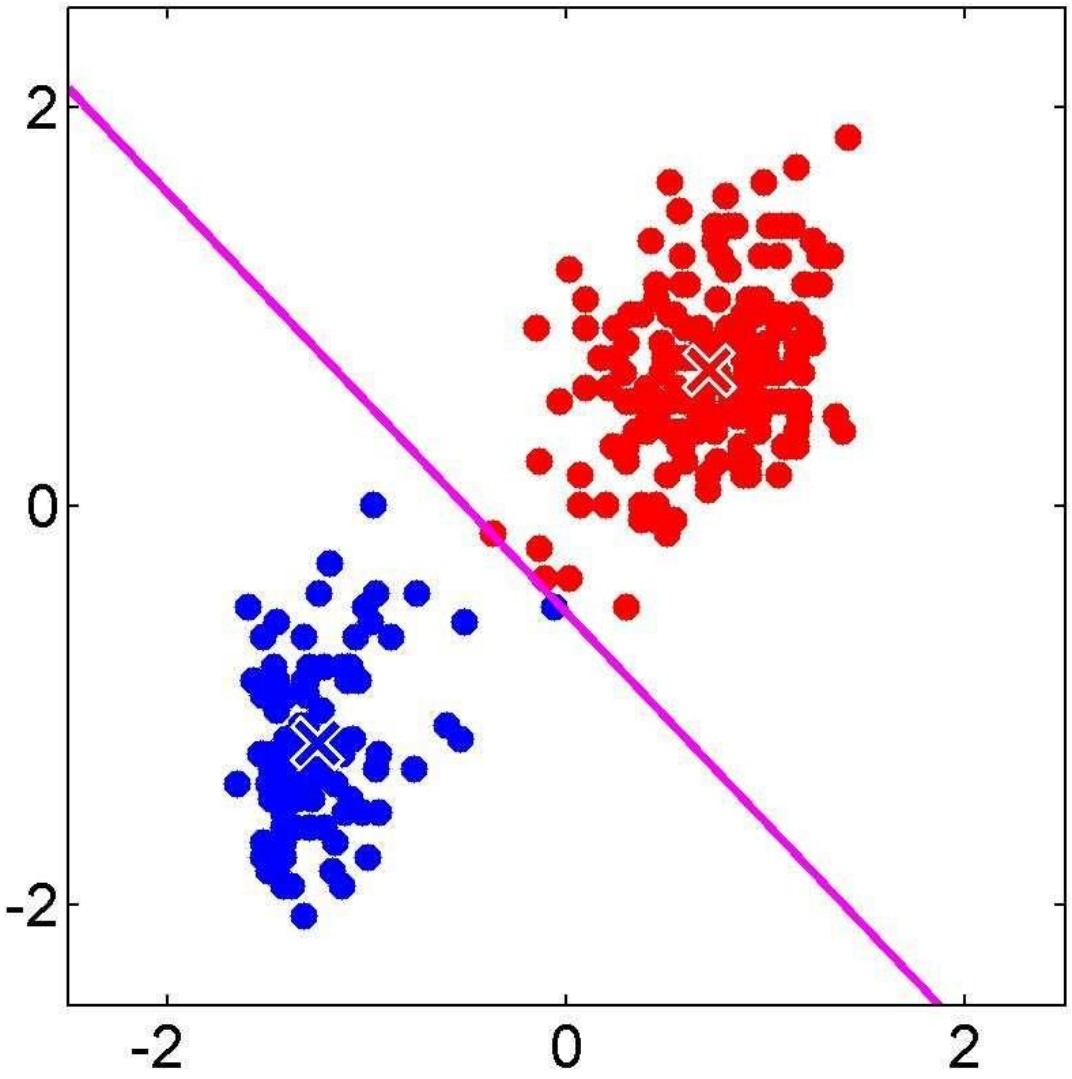

2
0
-2
-2
0
2

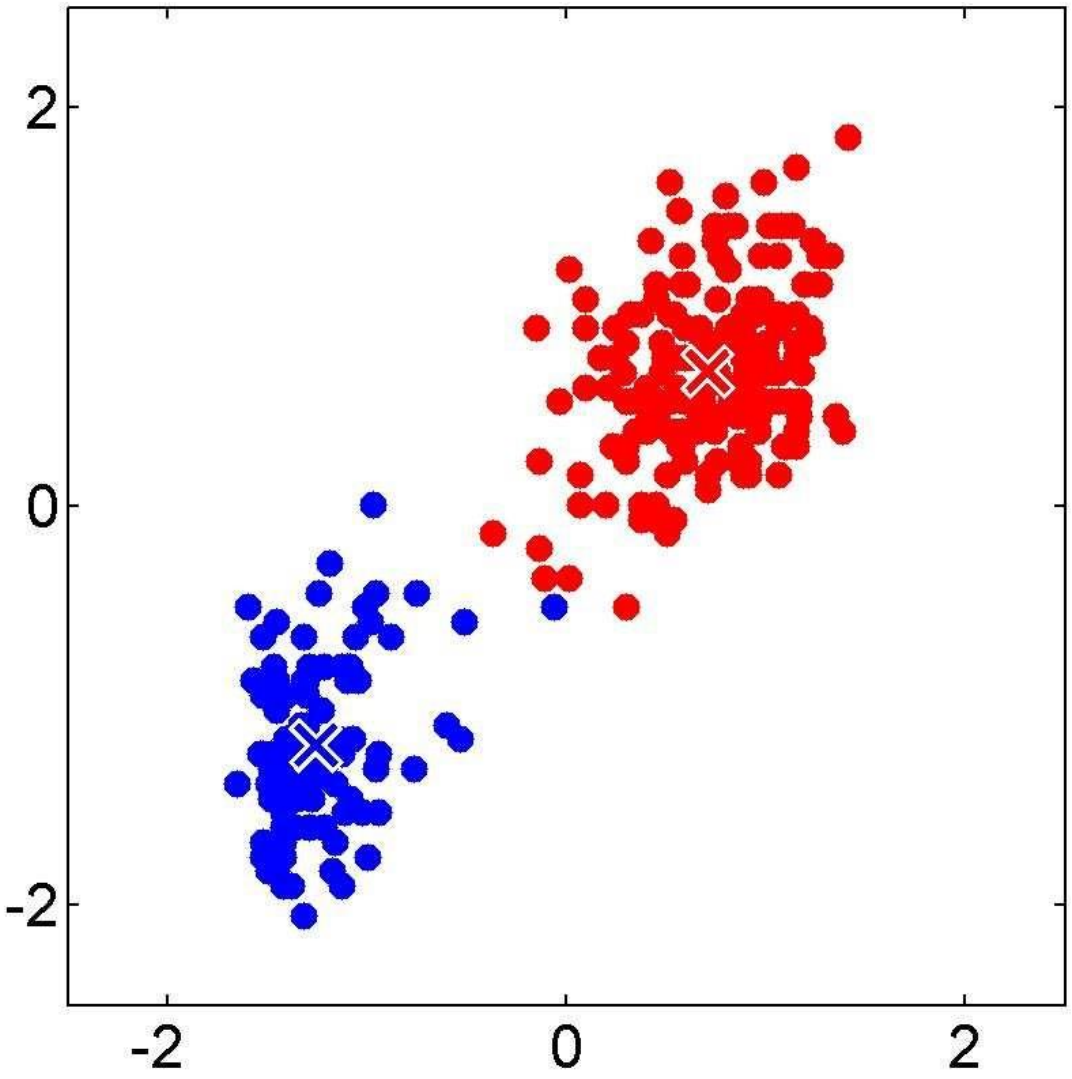

2
0
-2
-2
0
2


2
0
-2
-2
0
2


2
0
-2
-2
0
2


2
0
-2
-2
0
2

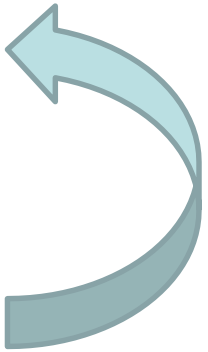

2
0
-2
-2
0
2


2
0
-2
-2
0
2

# 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



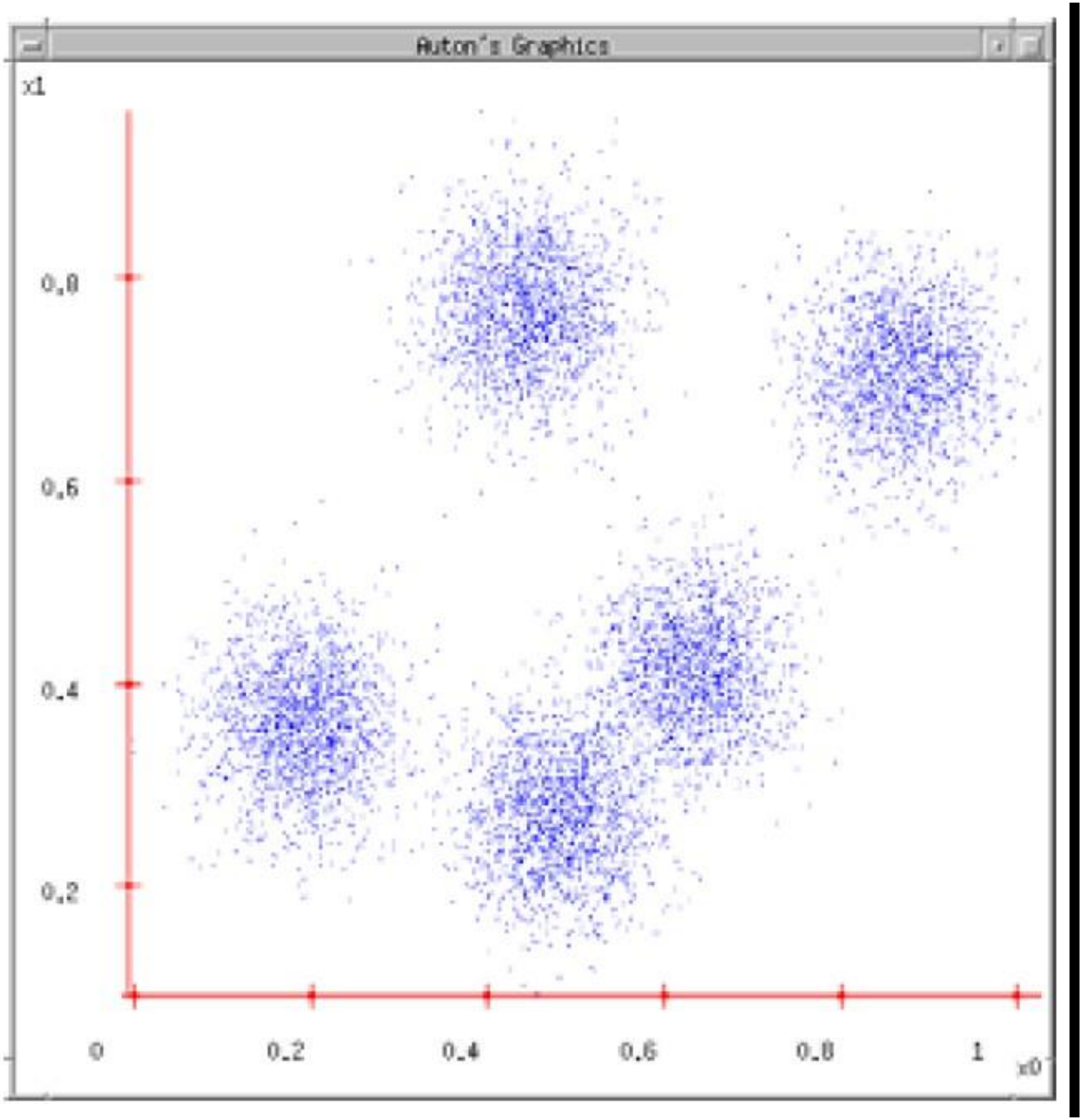
Properties

- Will always converge to some solution
- Can be a “local minimum”

Slide: Steve Seitz, image: Wikipedia

# K-means

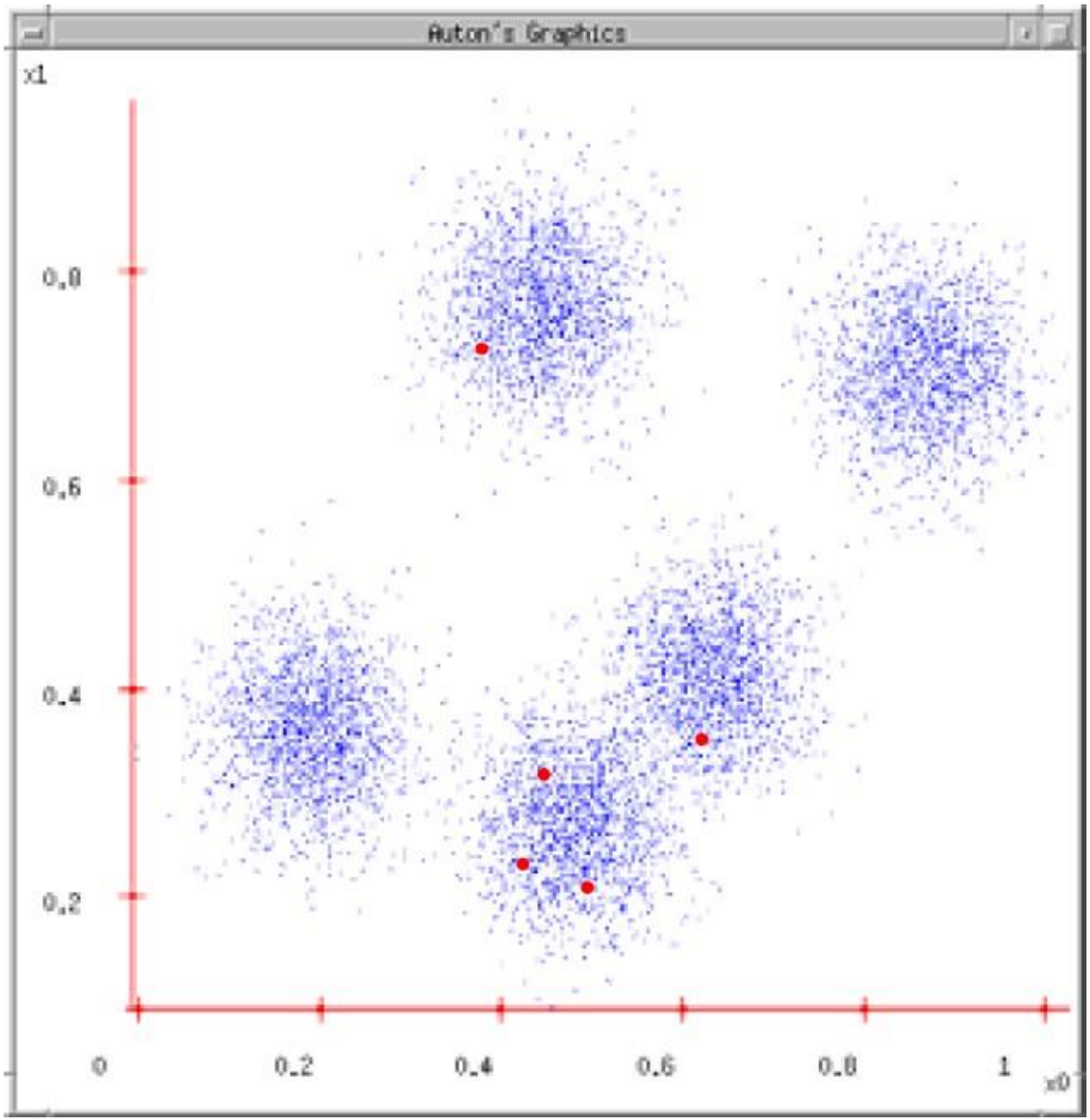
1. Ask user how many clusters they'd like. *(e.g. k=5)*




Source: A. Moore

# K-means

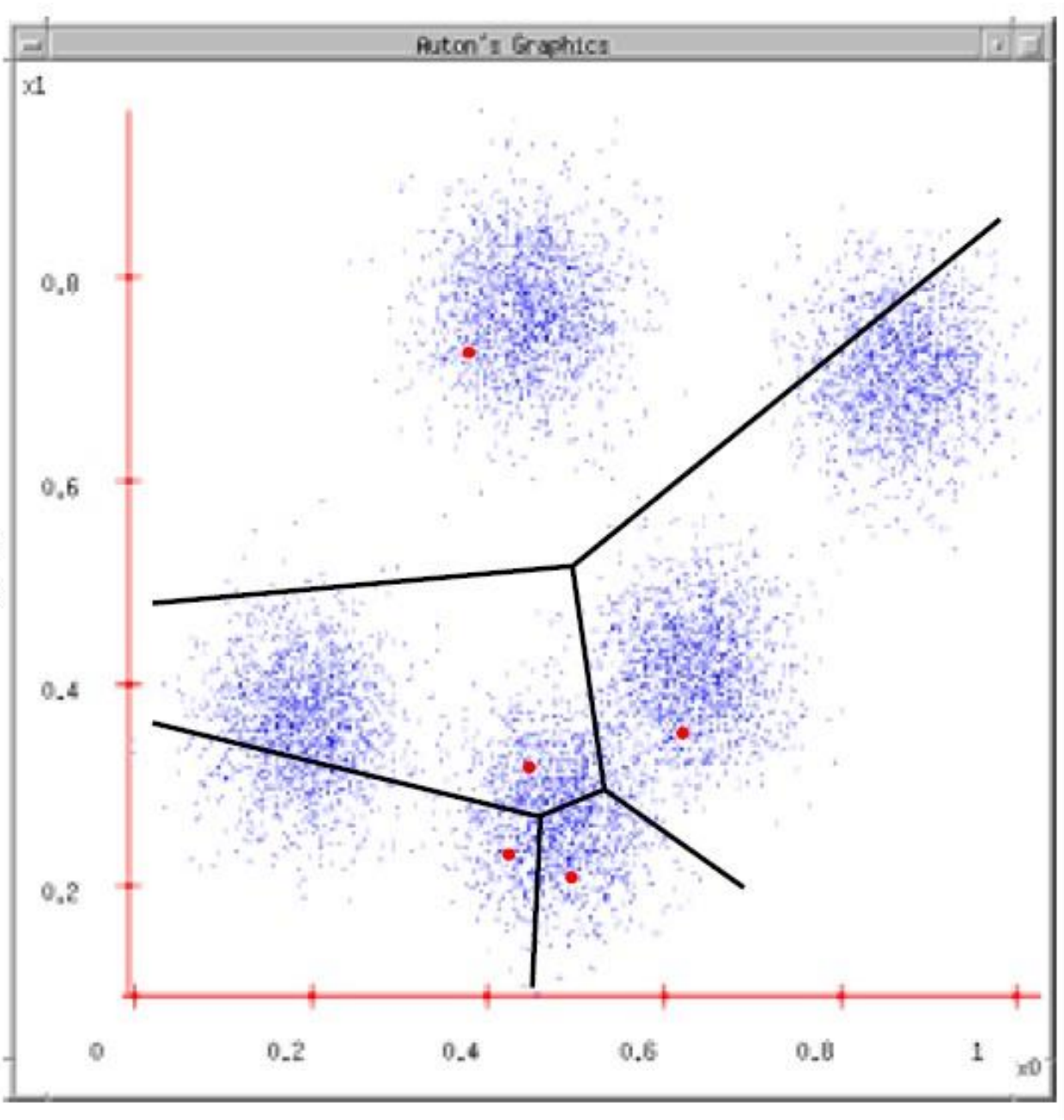
1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations




Source: A. Moore

# K-means

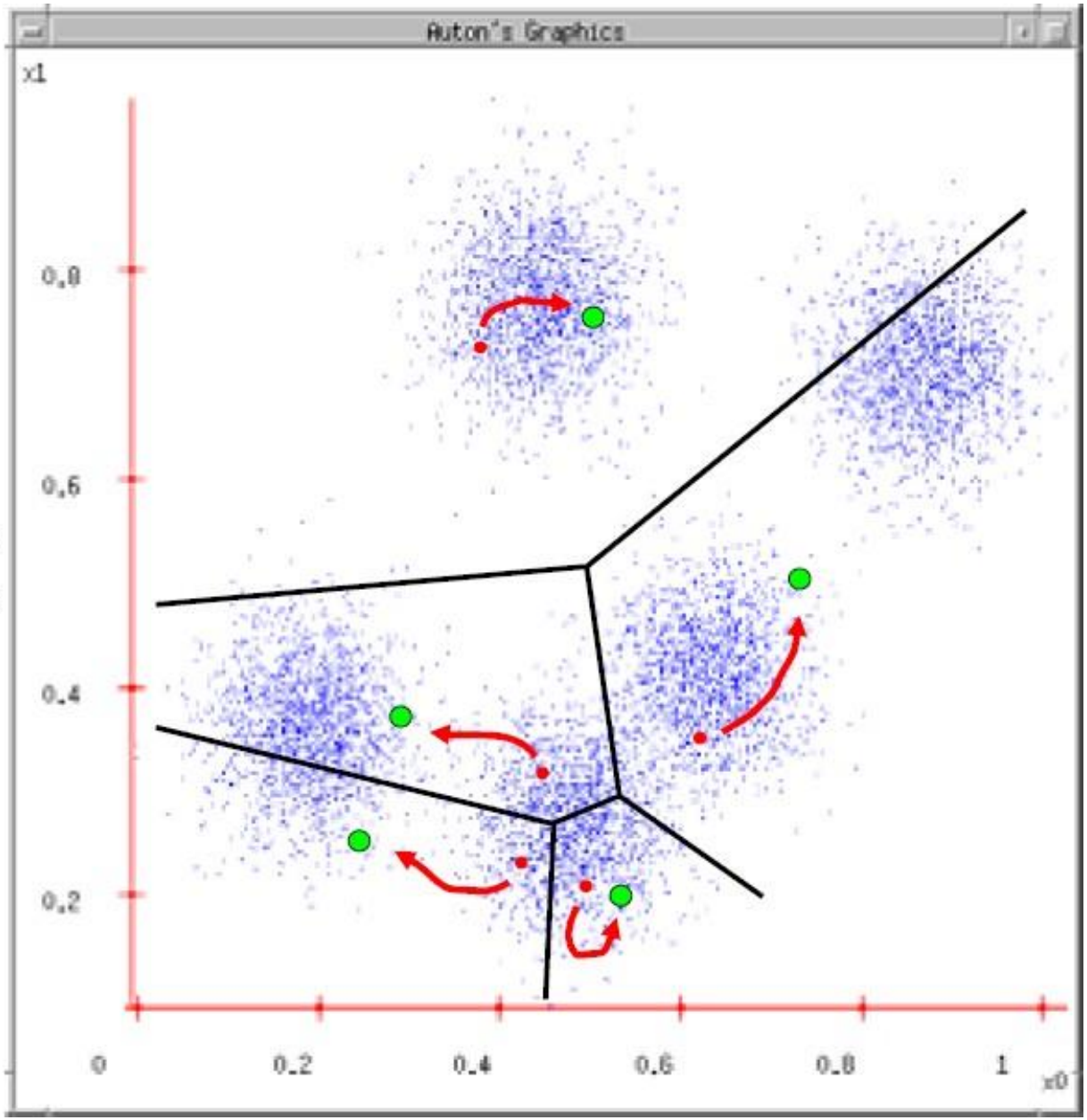
1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to. (Thus each Center "owns" a set of datapoints)




Source: A. Moore

# K-means

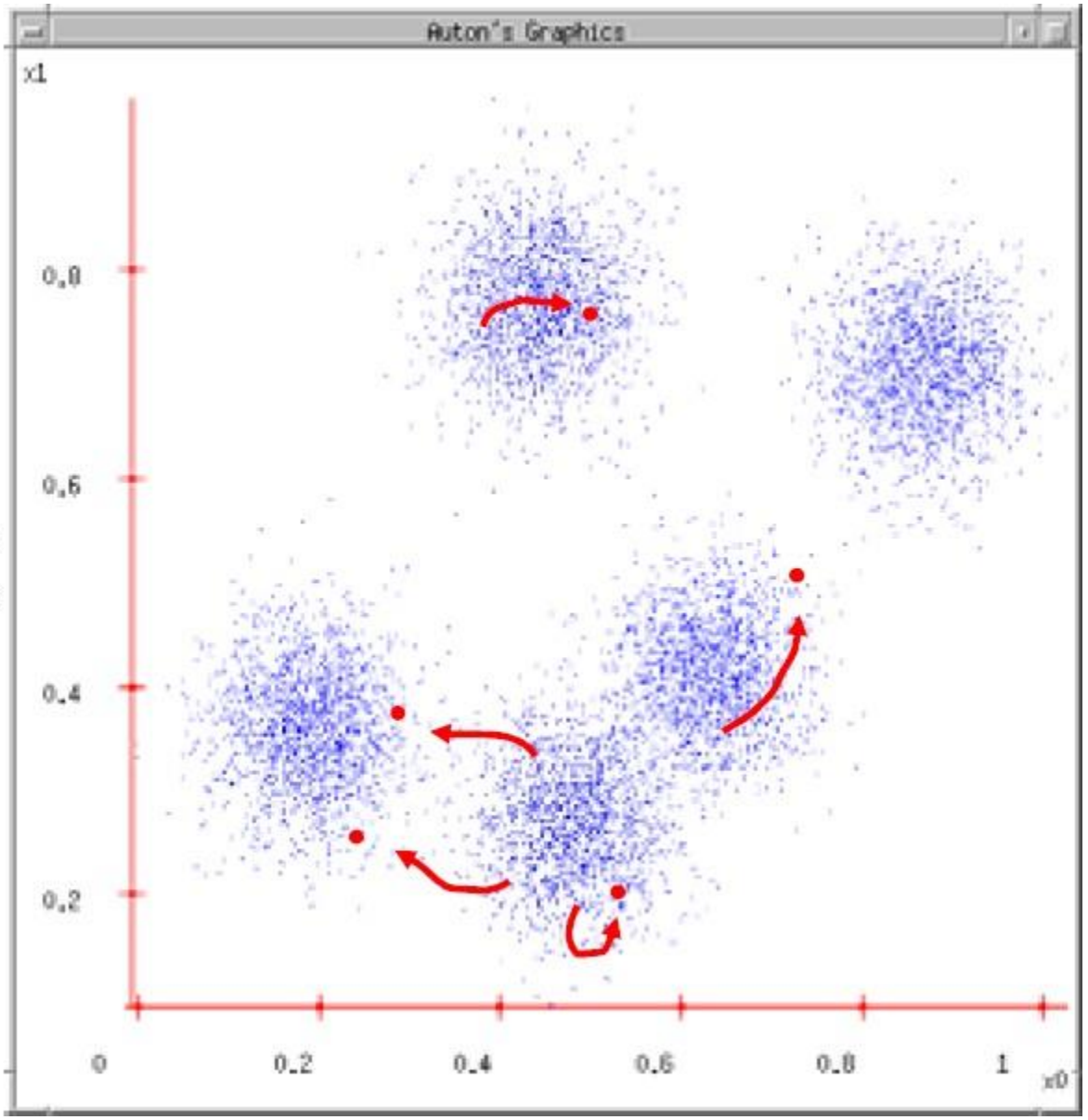
1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds the centroid of the points it owns




Source: A. Moore

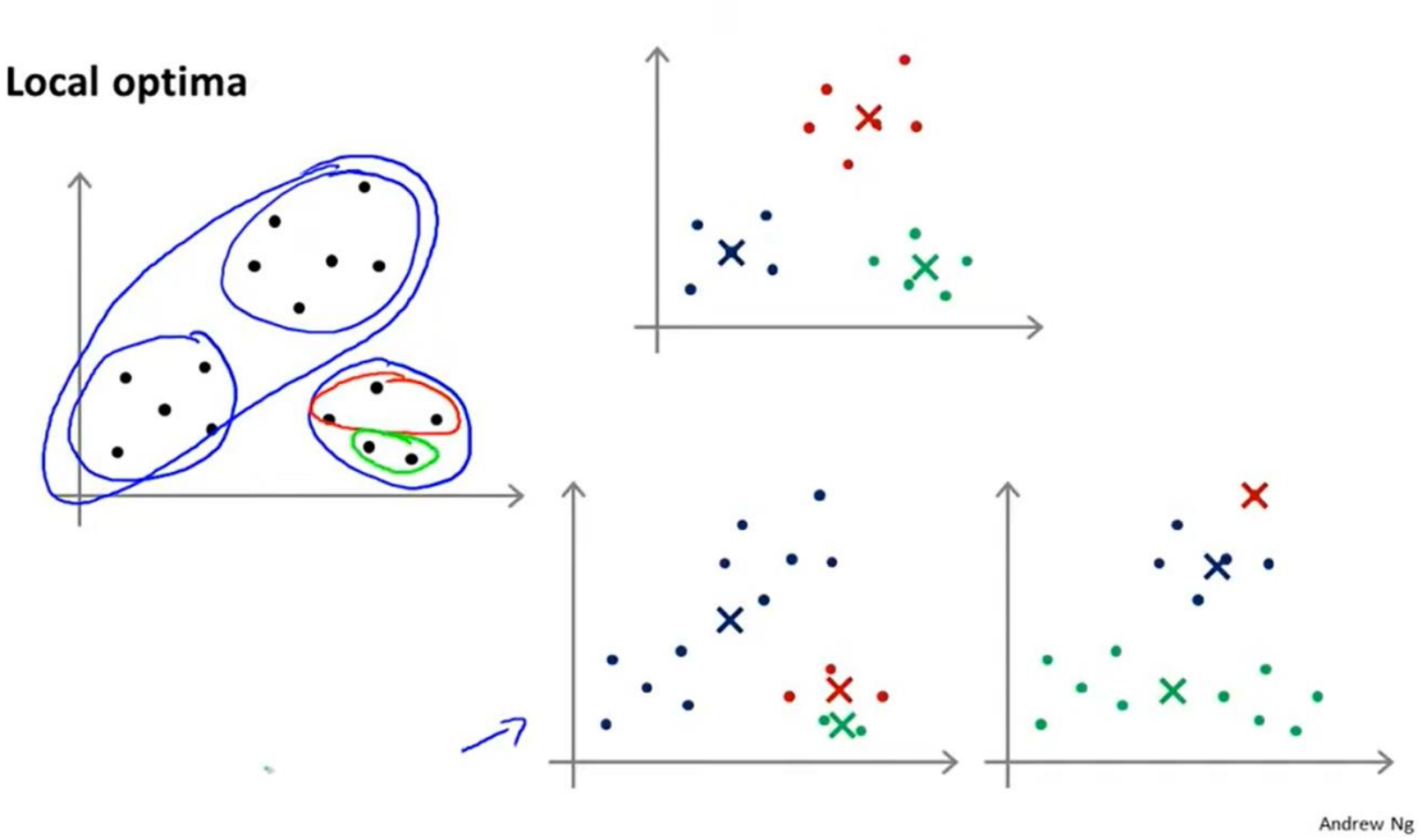
# K-means

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

# Local Optima



# Random Initialization

For i = 1 to 100 {

50 - 1000

→ Randomly initialize K-means.

Run K-means. Get $c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K$.

Compute cost function (distortion)

→ $J(c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K)$

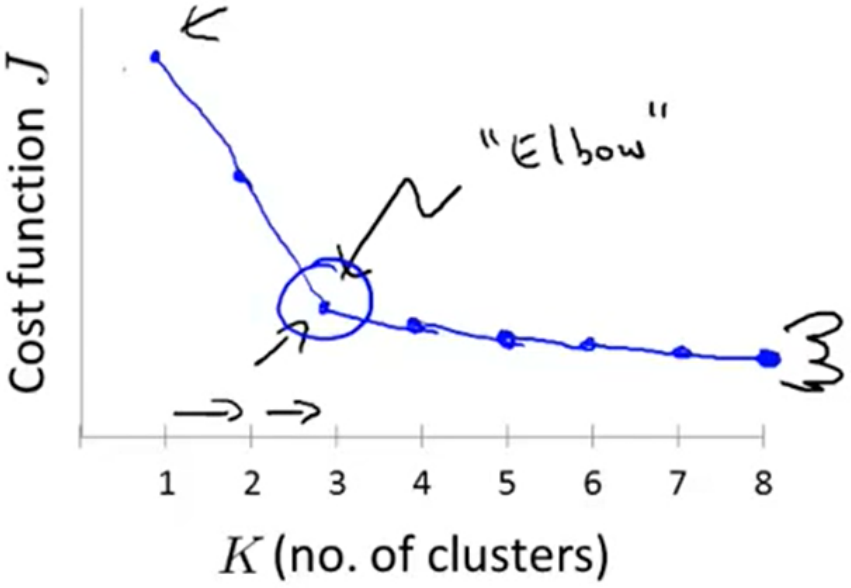
}

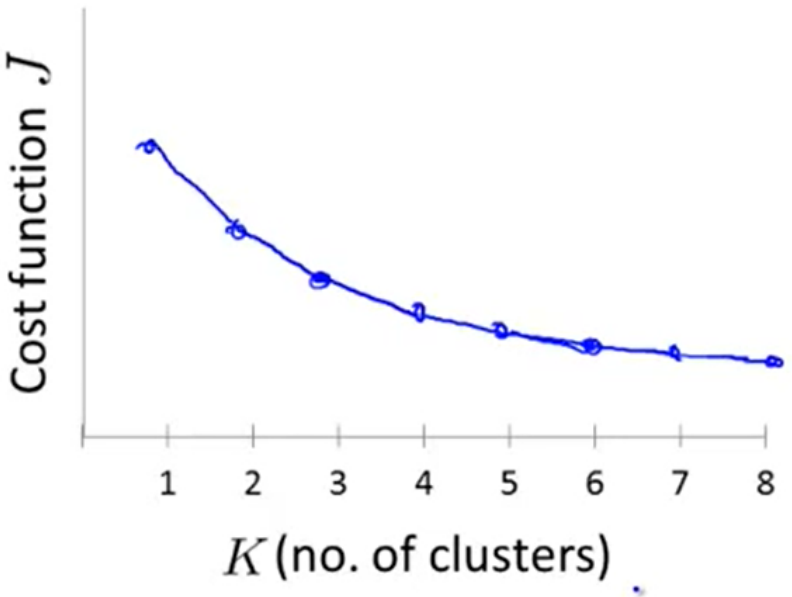
Pick clustering that gave lowest cost $J(c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K)$

k = 2 - 10

# Choosing K

Elbow method:



# Choosing K

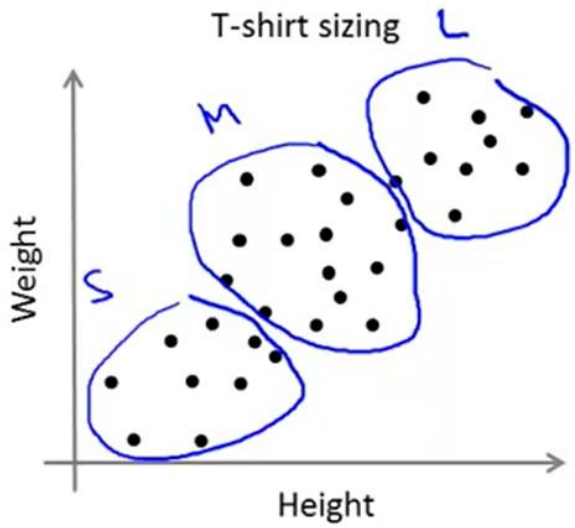
**Choosing the value of K**

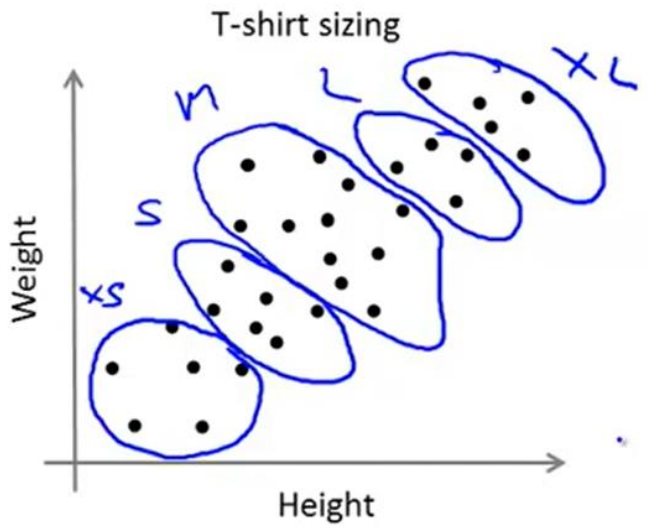
Sometimes, you're running K-means to get clusters to use for some later/downstream purpose. Evaluate K-means based on a metric for how well it performs for that later purpose.

K=3 S, M, L

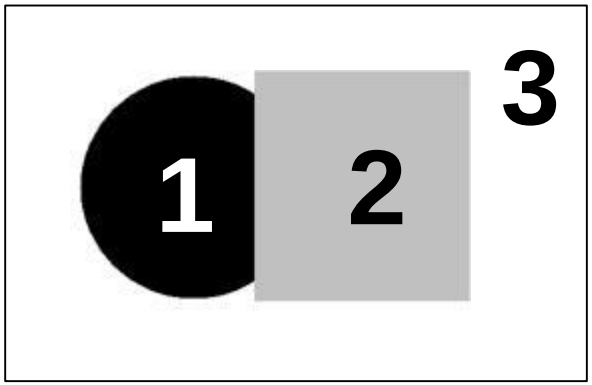
K=5 XS, S, M, L, XL

E.g.

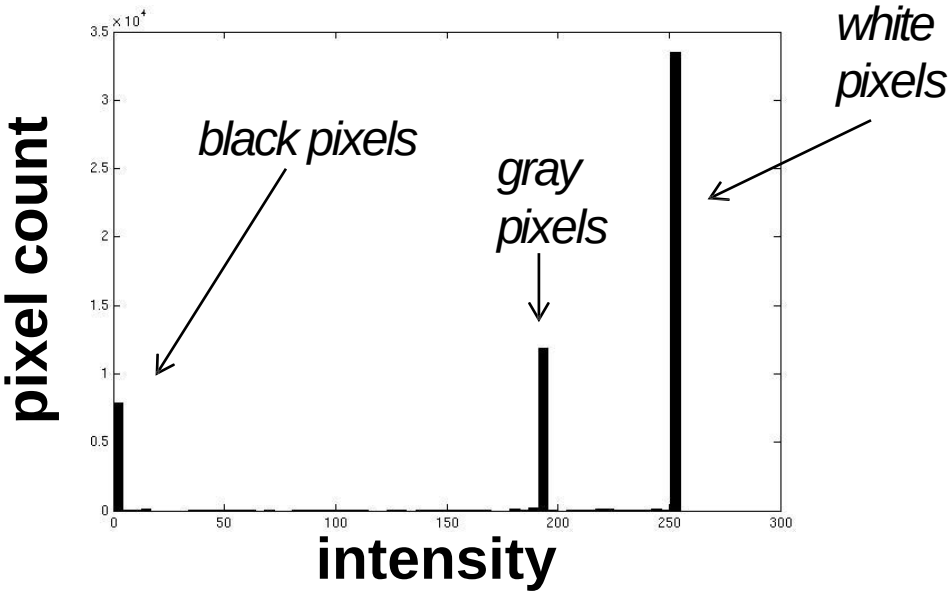


# Image segmentation: toy example




**input image**

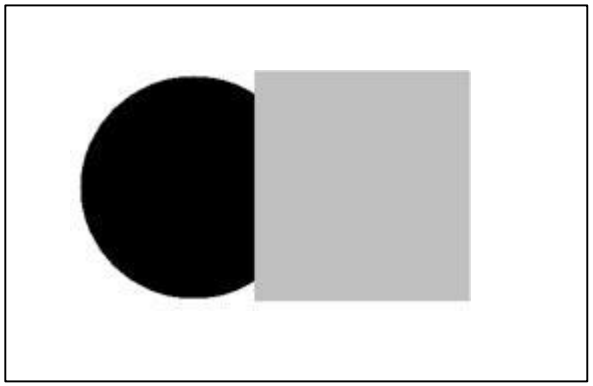


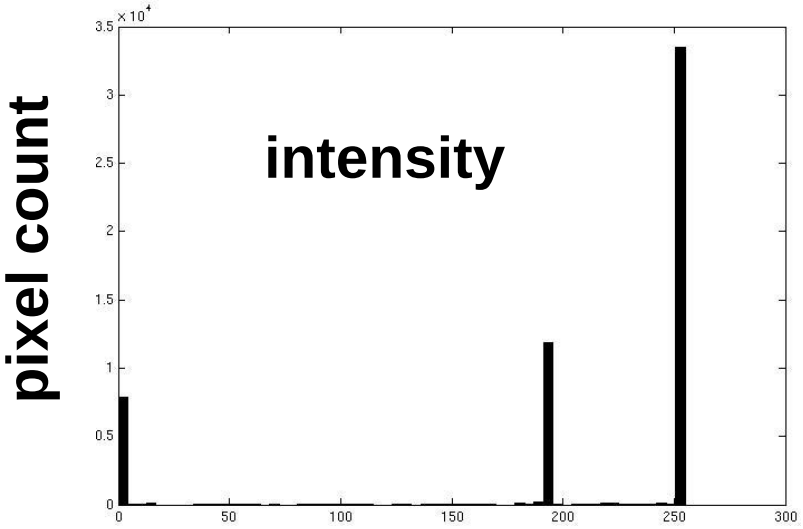

- These intensities define the three groups.
- We could label every pixel in the image according to which of these primary intensities it is.
  - i.e., *segment* the image based on the intensity feature.
- What if the image isn't quite so simple?

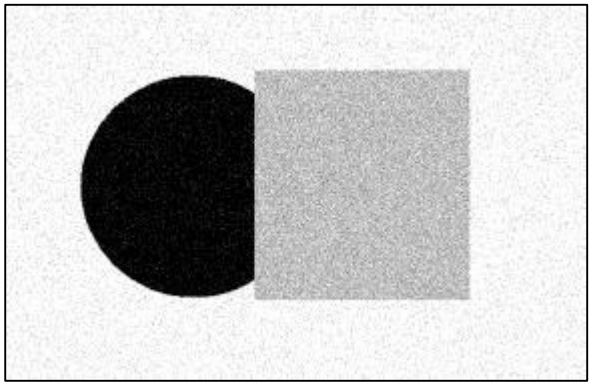
Source: K. Grauman

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

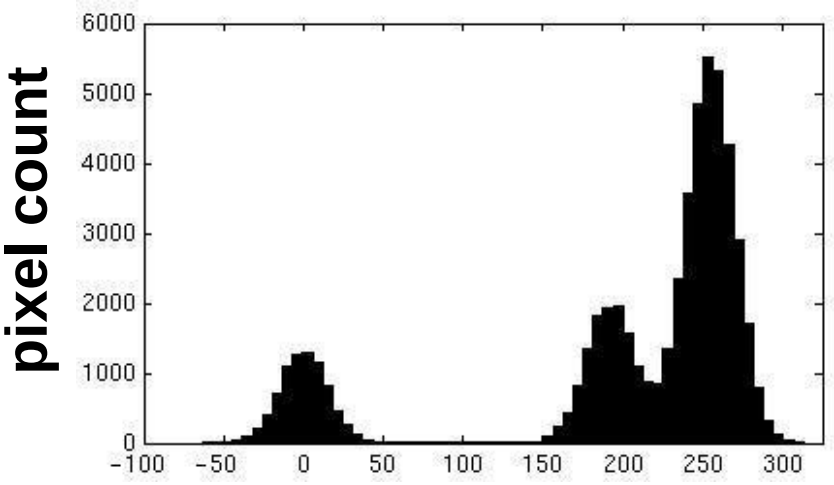
**input image**



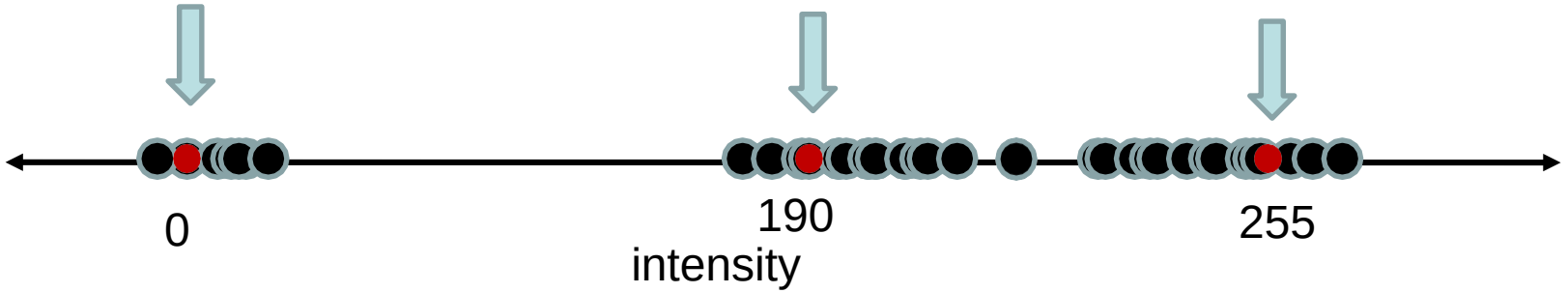


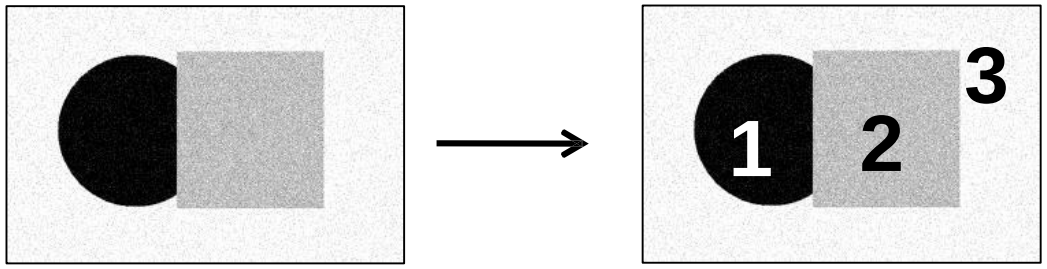



**input image**




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 $c_i$:

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

Source: K. Grauman

# 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):

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

- 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.

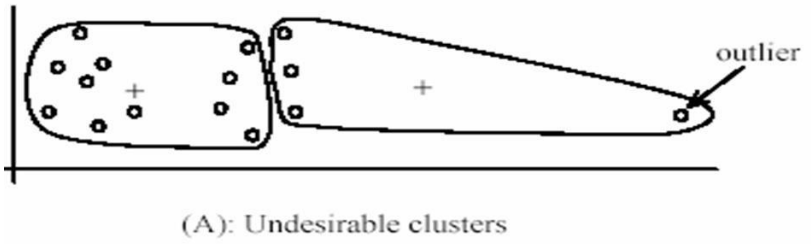
# K-means: pros and cons

## Pros

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

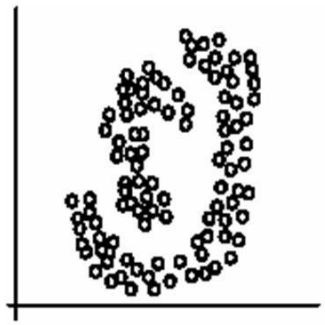
## Cons/issues

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

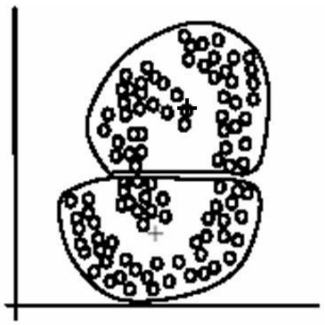



(A): Undesirable clusters


(B): Ideal clusters



(A): Two natural clusters



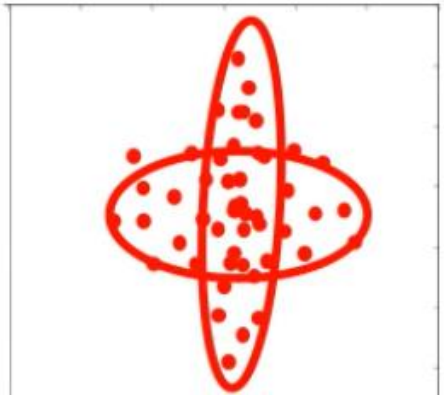
(B): *k*-means clusters

Adapted from K. Grauman

# Gaussian Mixture Models (GMM)

- K-means algorithm
  - Assigned each example to exactly one cluster
  - What if clusters are overlapping?
    - Hard to tell which cluster is right
    - Maybe we should try to remain uncertain
  - Used Euclidean distance
  - What if cluster has a non-circular shape?

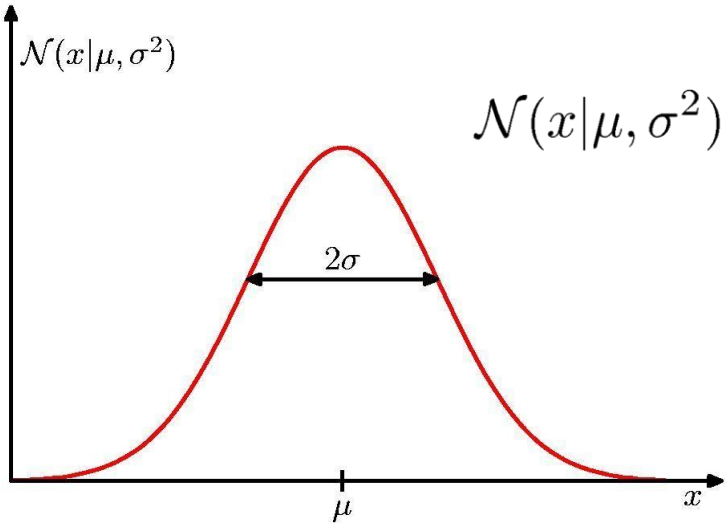
- Gaussian mixture models
  - Clusters modeled as Gaussians
    - Not just by their mean
  - EM algorithm: assign data to cluster with some *probability*
  - Gives probability model of x! ("generative")



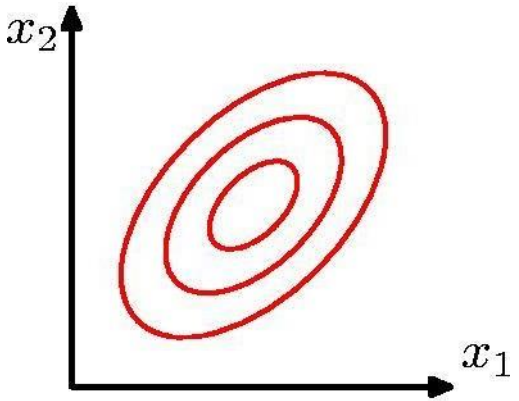
# Probabilistic Clustering

- 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




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




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

# Multivariate Gaussian Distribution

## Density estimation

Training set: $\{x^{(1)}, \ldots, x^{(m)}\}$

Each example is $x \in \mathbb{R}^n$

$x_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$

$x_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$

$x_3 \sim \mathcal{N}(\mu_3, \sigma_3^2)$

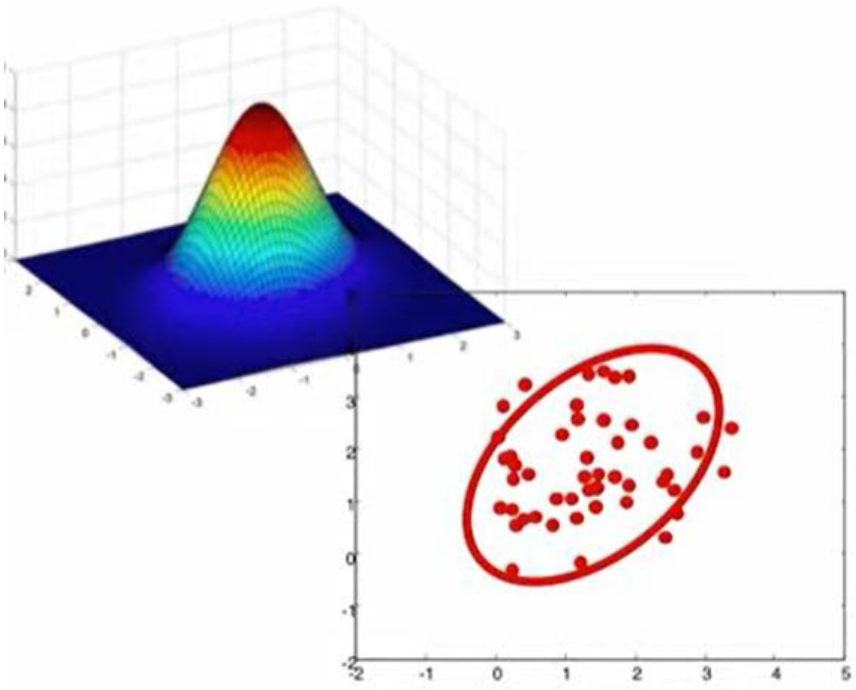
$$p(x) = p(x_1; \mu_1, \sigma_1^2)\, p(x_2; \mu_2, \sigma_2^2)\, p(x_3; \mu_3, \sigma_3^2) \cdots p(x_n; \mu_n, \sigma_n^2)$$

$$= \prod_{j=1}^{n} p(x_j; \mu_j, \sigma_j^2)$$

# Multivariate Gaussian Distribution

- Similar to univariate case

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



$\mu$ = length-d row vector
$\Sigma$ = d x d matrix

$|\Sigma|$ = matrix determinant

**Maximum likelihood estimate:**

$$\hat{\mu} = \frac{1}{m}\sum_j \underline{x}^{(j)}$$

$$\hat{\Sigma} = \frac{1}{m}\sum_j (\underline{x}^{(j)}-\underline{\hat{\mu}})^T(\underline{x}^{(j)}-\underline{\hat{\mu}})$$

(average of dxd matrices)

# Multivariate Gaussian Distribution

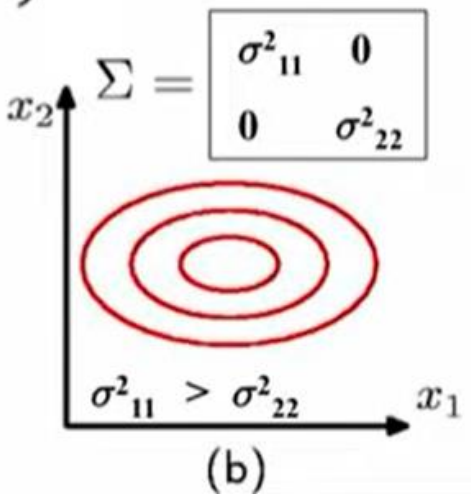
$$p(x_1) = \frac{1}{Z}\exp\left\{-\frac{1}{2\sigma_1^2}(x_1-\mu_1)^2\right\} \qquad p(x_2) = \frac{1}{Z_2}\exp\left\{-\frac{1}{2\sigma_2^2}(x_2-\mu_2)^2\right\}$$

$$\underline{x} = [x_1 \; x_2]$$

$$p(x_1)p(x_2) = \frac{1}{Z_1 Z_2}\exp\left\{-\frac{1}{2}(\underline{x}-\underline{\mu})^T\Sigma^{-1}(\underline{x}-\underline{\mu})\right\}$$

$$\underline{\mu} = [\mu_1 \; \mu_2]$$

$$\Sigma = \mathrm{diag}(\sigma_1^2 \; , \; \sigma_2^2)$$



# Mixtures of Gaussians

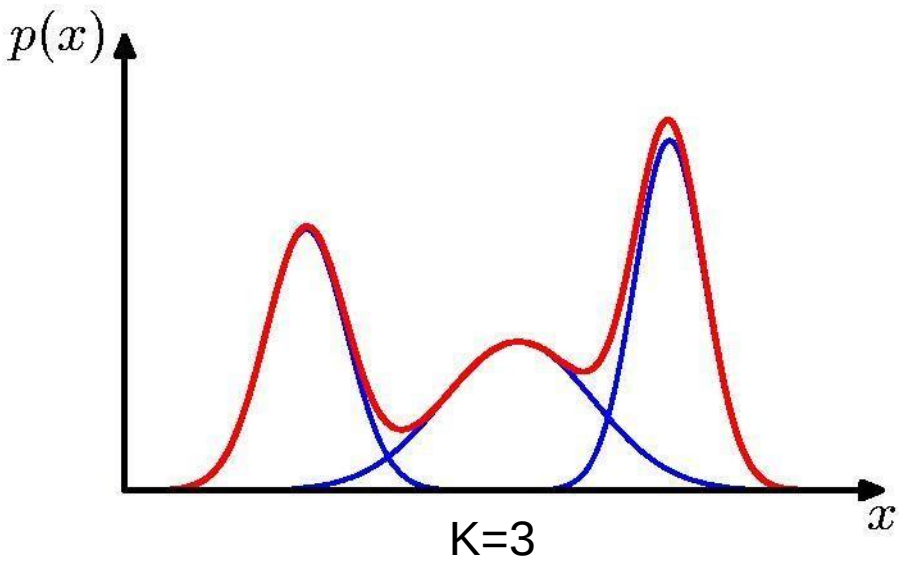
- Combine simple models into a complex model:

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

Component

Mixing coefficient

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



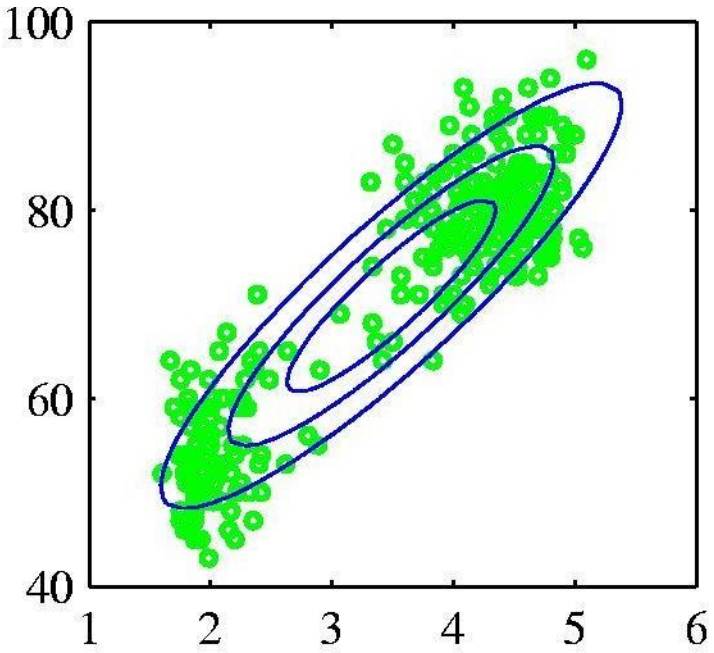

K=3

- Find parameters through EM (Expectation Maximization) algorithm

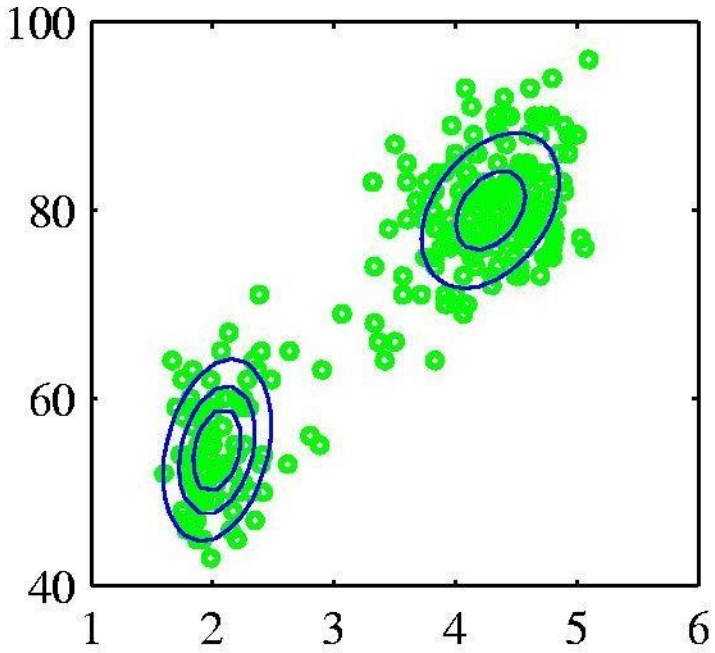
Adapted from Chris Bishop

# Probabilistic version: Mixtures of Gaussians

- Old Faithful data set




Single Gaussian

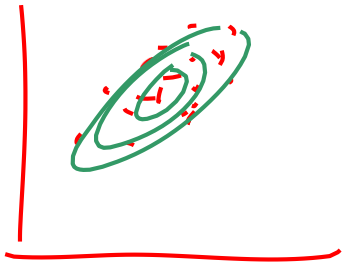



Mixture of two Gaussians

# Gaussian Mixture Model

- Data set

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



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

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

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

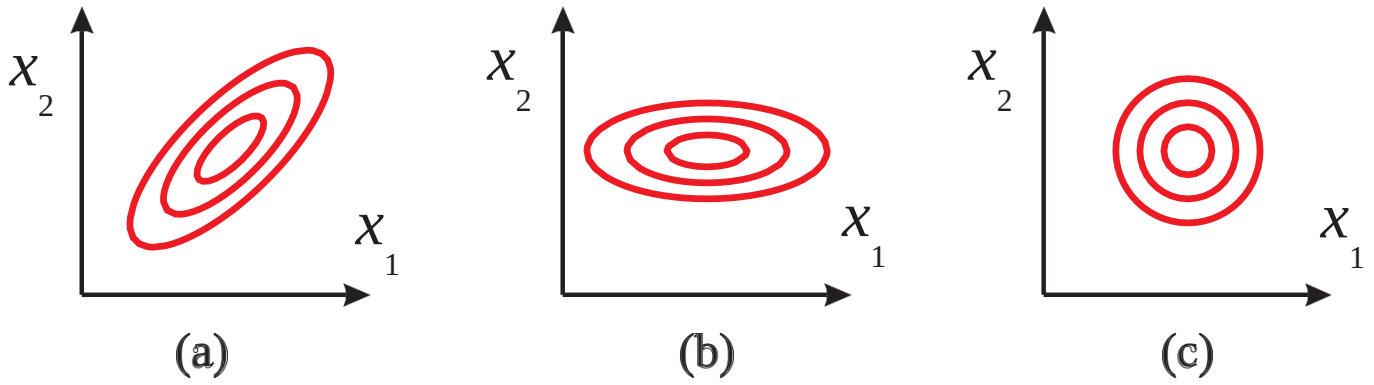
# The Gaussian Distribution

- Multivariate Gaussian

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

mean

covariance



# Gaussian Mixture 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 $\pi_k$, such that $p(z_k = 1) = \pi_k$

# Gaussian Mixture Model

- Start with parameters describing each cluster
- Mean $\mu_c$, variance $\sigma_c$, "size" $\pi_c$
- Probability distribution: $p(x) = \sum_c \pi_c \mathcal{N}(x\ ;\ \mu_c, \sigma_c)$

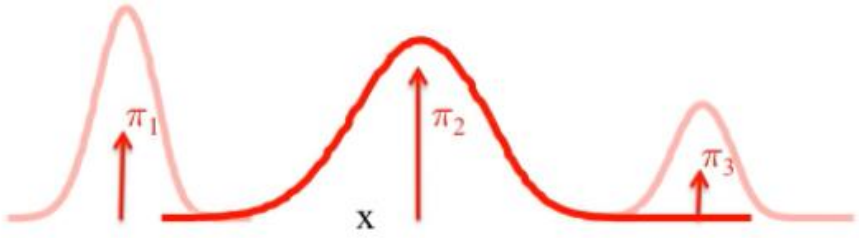
- Equivalent "latent variable" form:

$p(z = c) = \pi_c$ Select a mixture component with probability $\pi$

$p(x|z = c) = \mathcal{N}(x\ ;\ \mu_c, \sigma_c)$ Sample from that component's Gaussian

"Latent assignment" z:
we observe x, but z is hidden

p(x) = marginal over x



# Sampling from the Gaussian

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

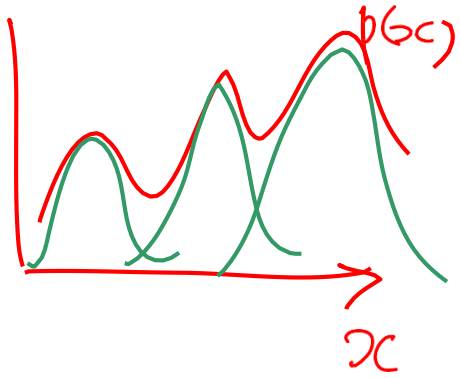
# Fitting the Gaussian Mixture

- We wish to invert this process – given the data set, find the corresponding parameters:
  - mixing coefficients
  - means
  - covariances
- 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

- Linear super-position of Gaussians

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



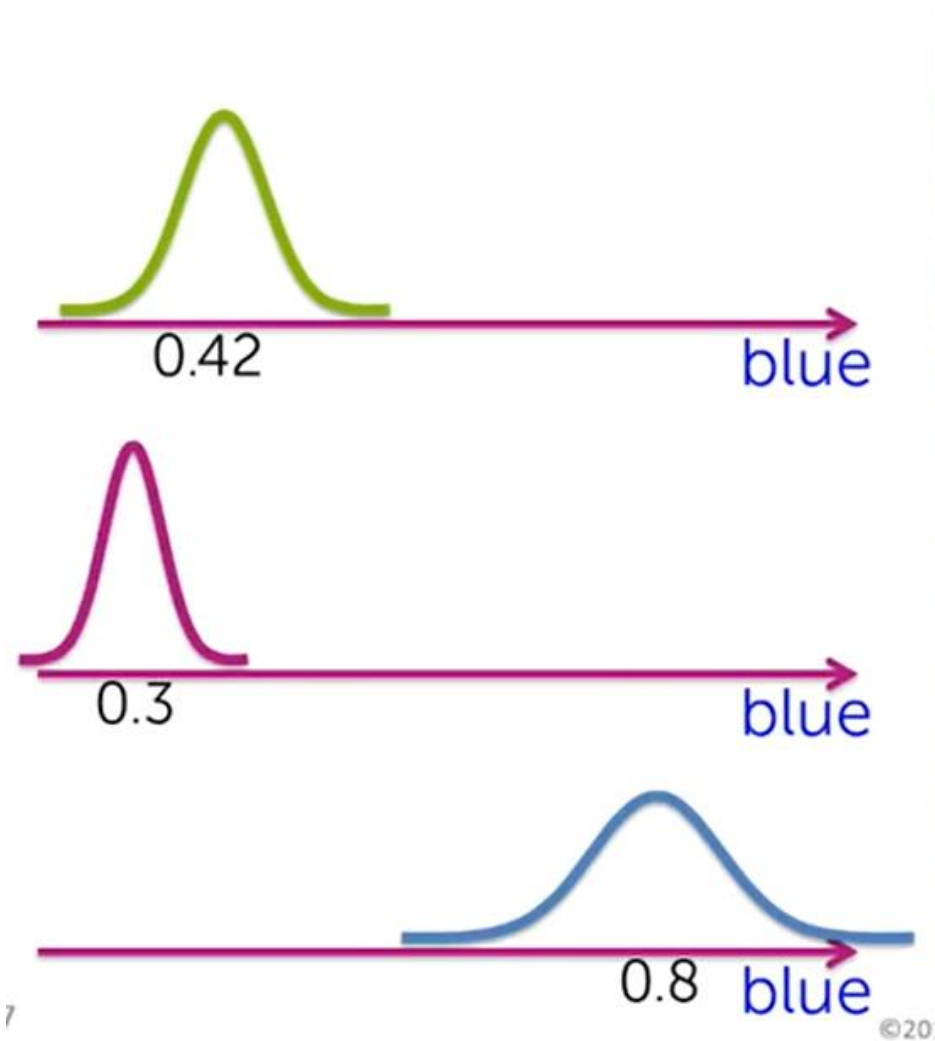

- Normalization and positivity require

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

- Can interpret the mixing coefficients as prior probabilities

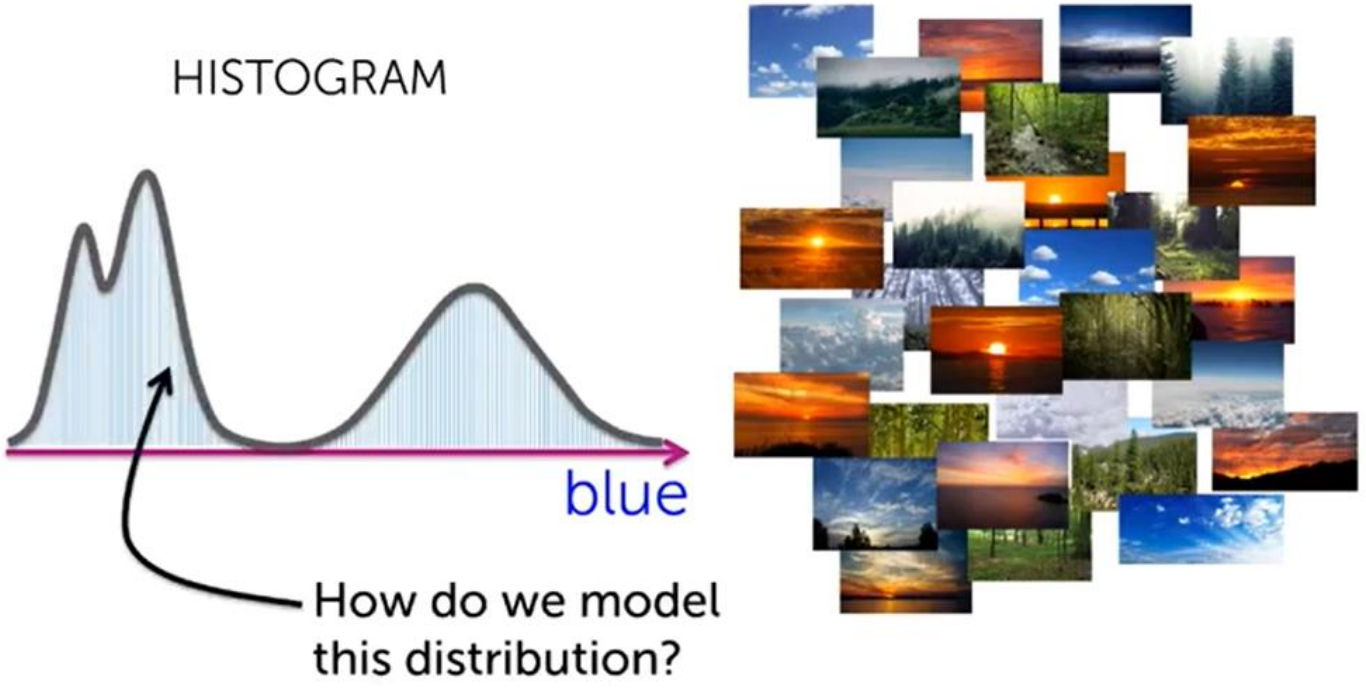
$$p(\mathbf{x}) = \sum_{k=1}^{K} p(k)p(\mathbf{x}|k)$$

# Example

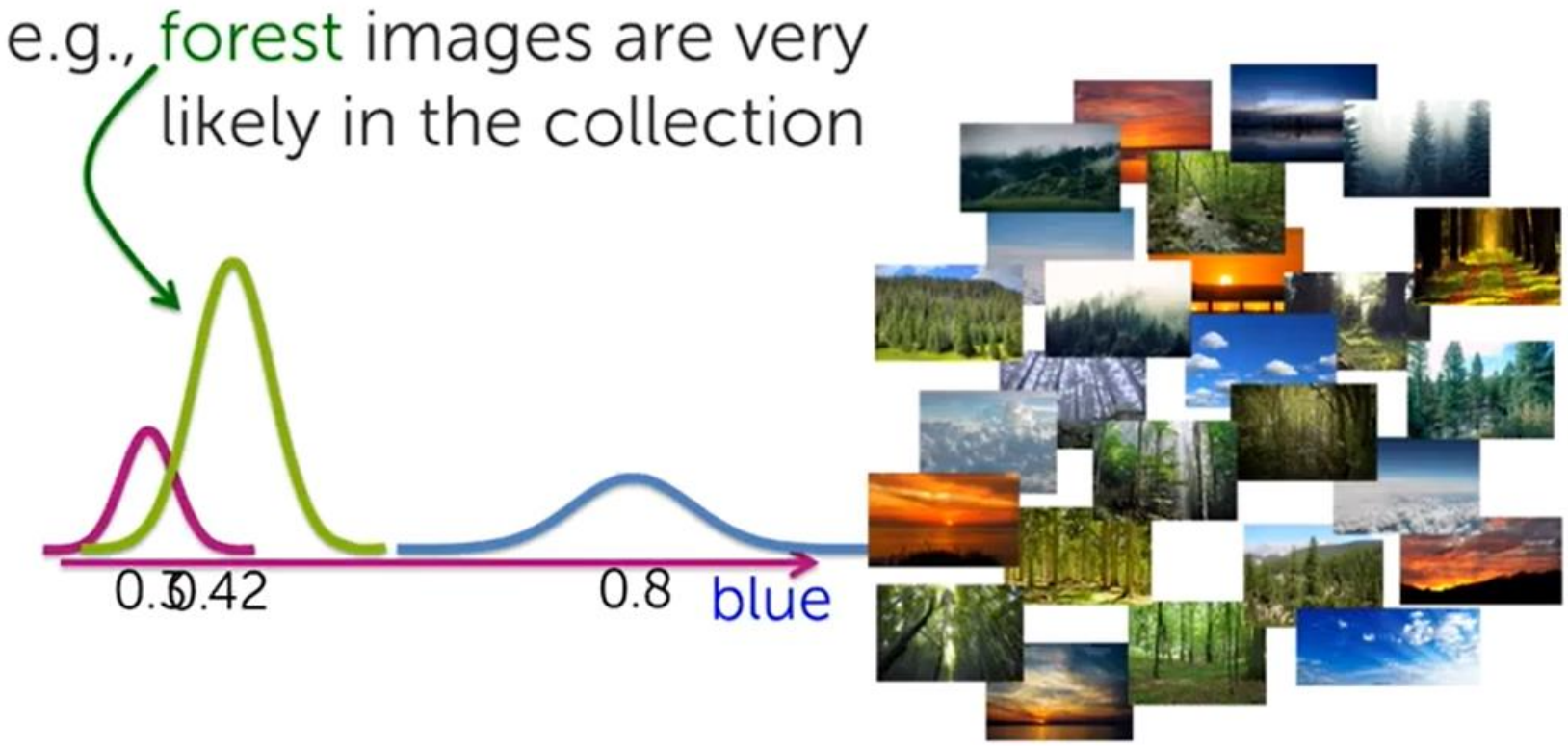



# Example

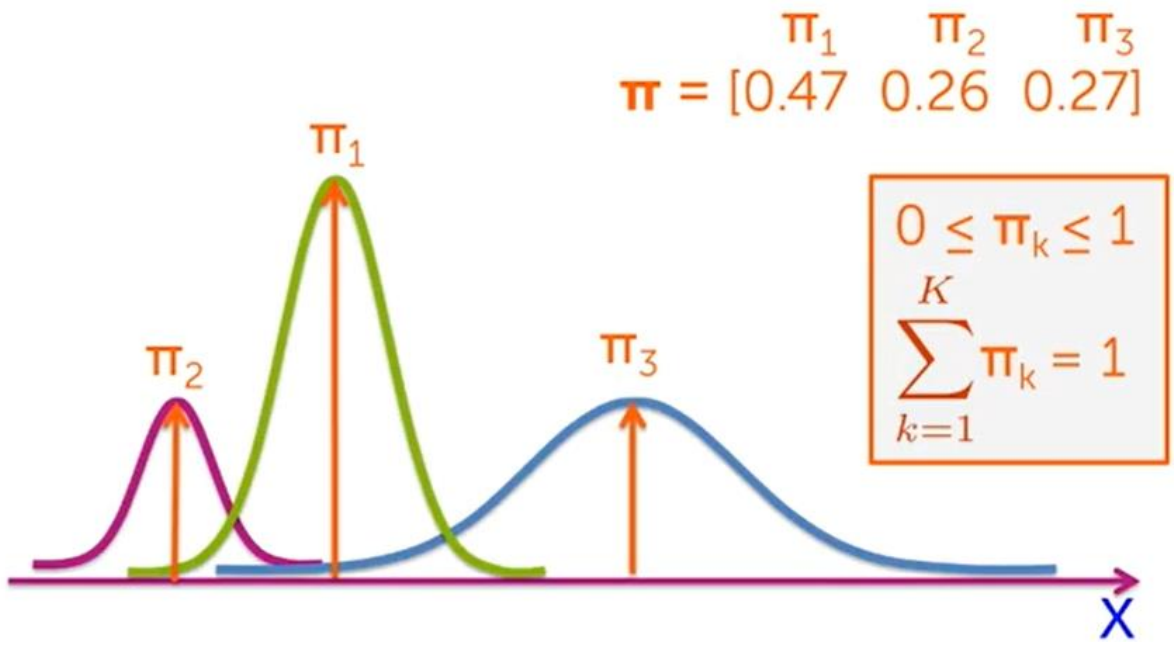
Jumble of unlabeled images



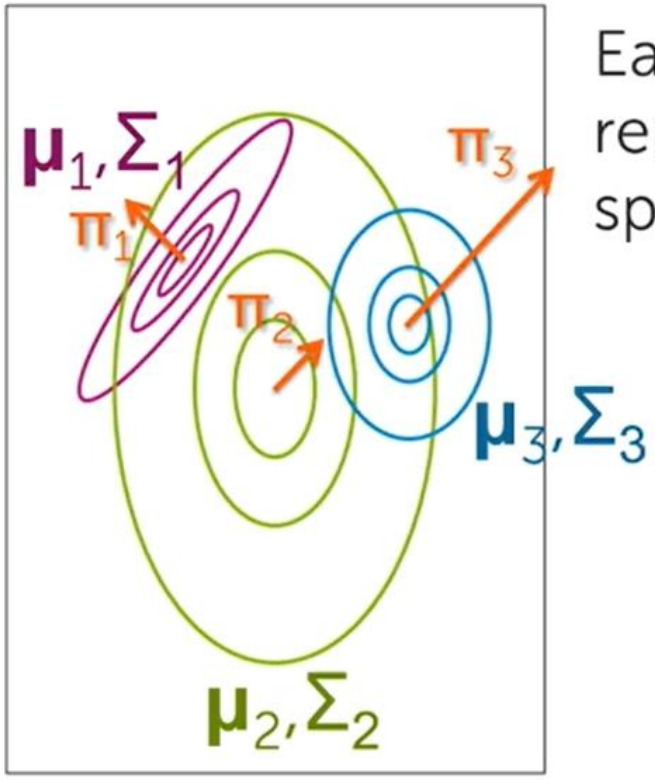
# Example


e.g., forest images are very likely in the collection
0.42
0.8
blue

# Associate weight with each Gaussian



# Mixture of Gaussian




Each mixture component represents a unique cluster specified by:

$$\{\pi_k , \mu_k , \Sigma_k\}$$

# Example

Without observing the image content, what's the probability it's from cluster k? (e.g., prob. of seeing "clouds" image)

cluster assignment for obs. $x_i$

$$p(z_i = k) = \pi_k$$

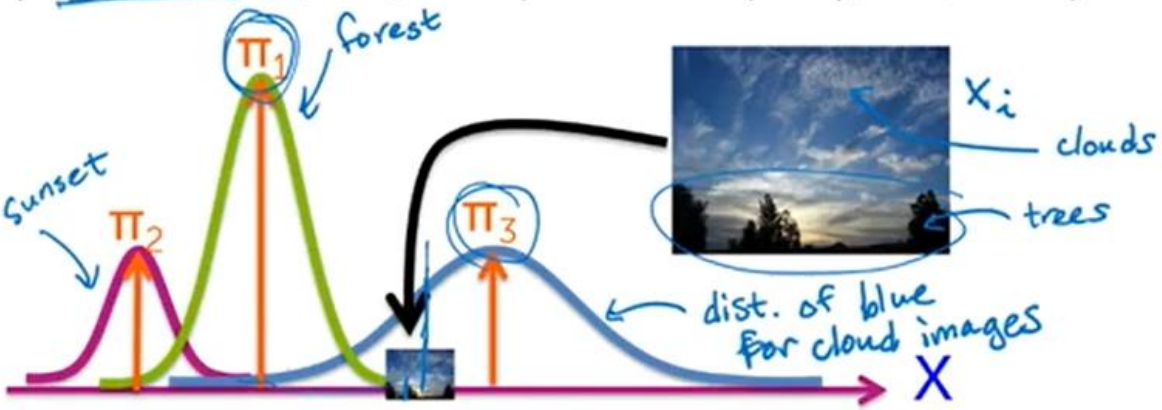
prior

Given observation $\mathbf{x}_i$ is from cluster k, what's the likelihood of seeing $\mathbf{x}_i$? (e.g., just look at distribution for "clouds")

$$p(x_i \mid z_i = k, \mu_k, \Sigma_k) = N(x_i | \mu_k, \Sigma_k)$$

likelihood

$[RGB]_i$



# Gaussian Mixture Model

- z uses a 1-of-K representation, we can also write this distribution in the form

$$p(\mathbf{z}) = \prod_{k=1}^{K} \pi_k^{z_k}$$

- Conditional distribution of x given a particular value for z is a Gaussian

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

- 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

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

# Gaussian Mixture Model

- Conditional probability of z given x
- use $\gamma(z_k)$ to denote $p(z_k = 1 \mid x)$, whose value can be found using Bayes' theorem

$$\gamma(z_k) \equiv p(z_k = 1|\mathbf{x}) = \frac{p(z_k = 1)p(\mathbf{x}|z_k = 1)}{\sum_{j=1}^{K} p(z_j = 1)p(\mathbf{x}|z_j = 1)}$$

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

- $\pi_k$ as the prior probability of $z_k = 1$, and the quantity $\gamma(z_k)$ 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.

# GMM Problems and Solutions

- How to maximize the log likelihood
  - solved by expectation-maximization (EM) algorithm
- How to avoid singularities in the likelihood function
  - solved by a Bayesian treatment
- How to choose number $K$ of components
  - also solved by a Bayesian treatment

# Expectation Maximization (EM) Algorithm



- We first choose some initial values for the means, covariances, and mixing coefficients.
- 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,
- We then use these probabilities in the maximization step, or M step, to re-estimate the means, covariances, and mixing
- 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

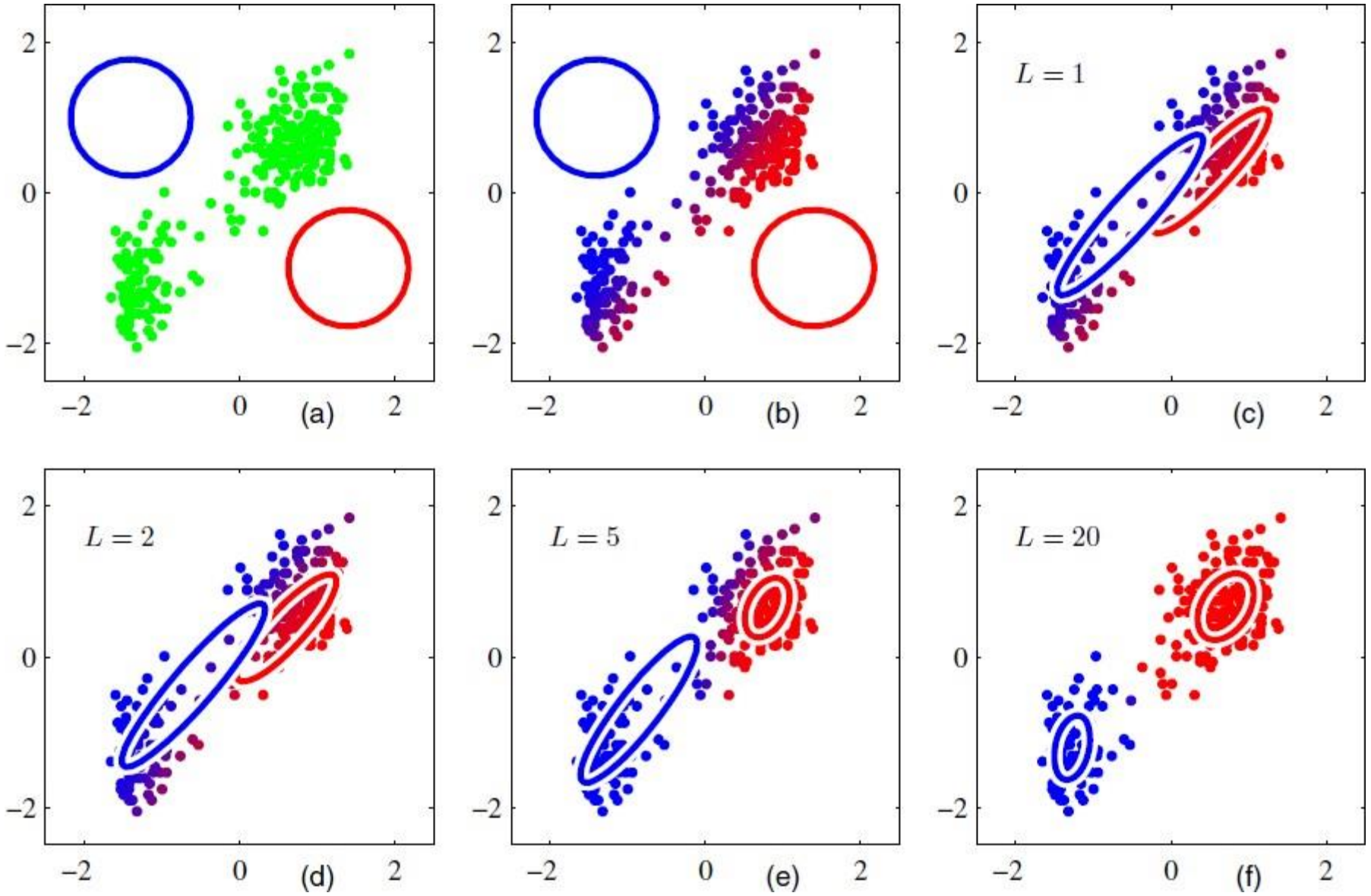
# EM Algorithm – Informal Derivation

- 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
- Each EM cycle guaranteed not to decrease the likelihood

Initialization

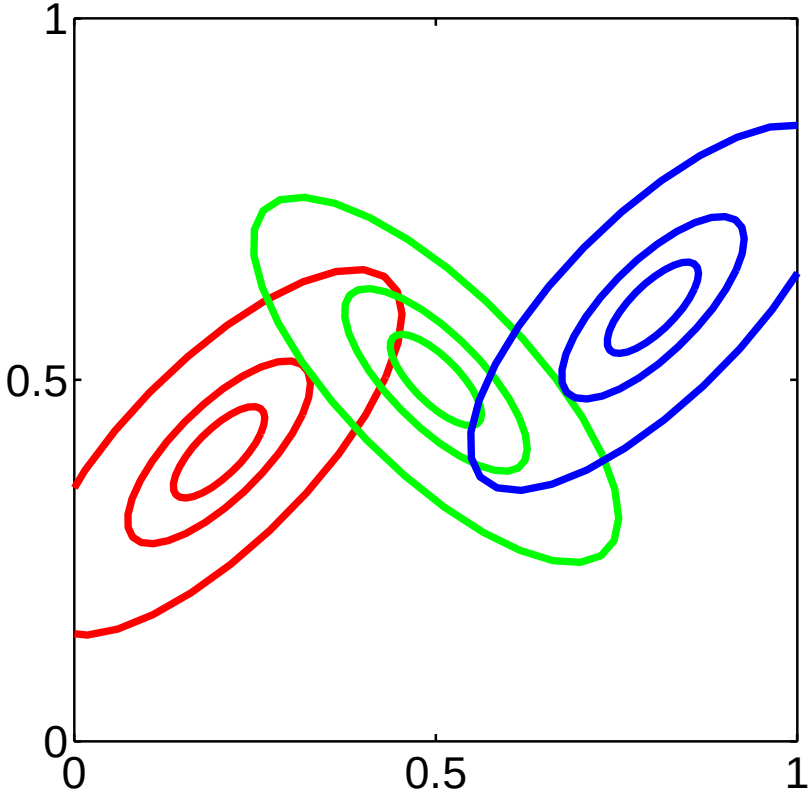
E step

M step

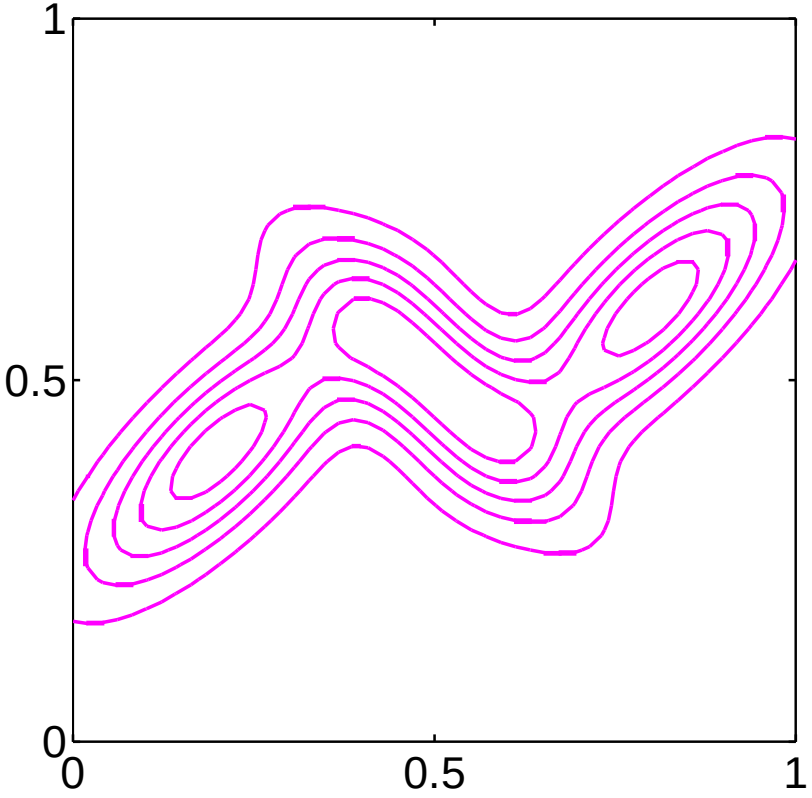



Figures from Chris Bishop

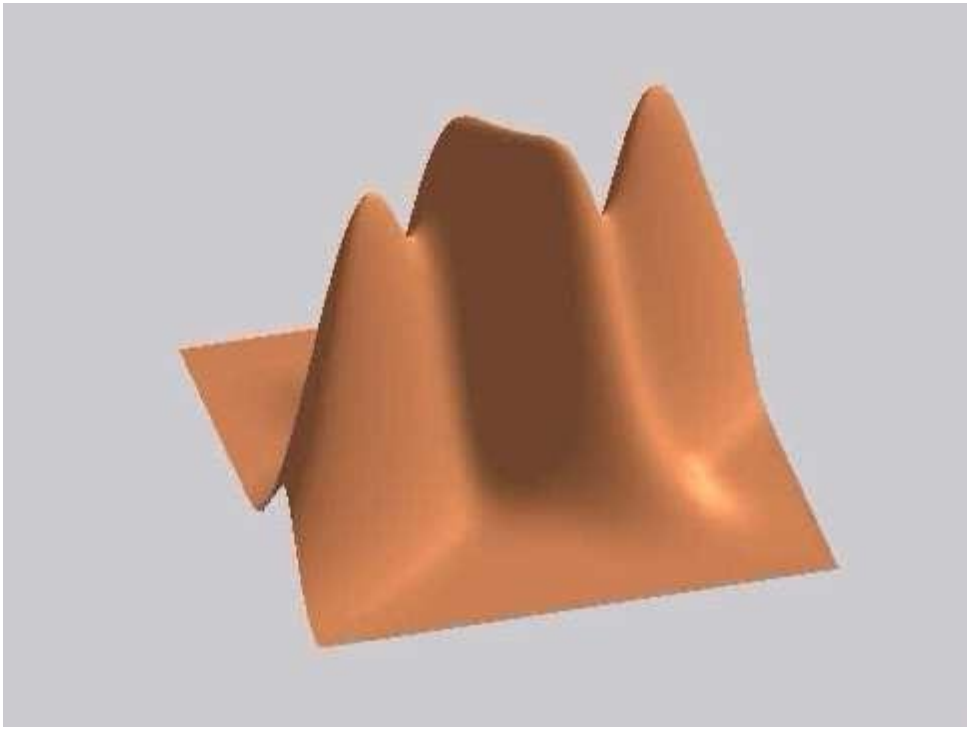
# Example: Mixture of 3 Gaussians



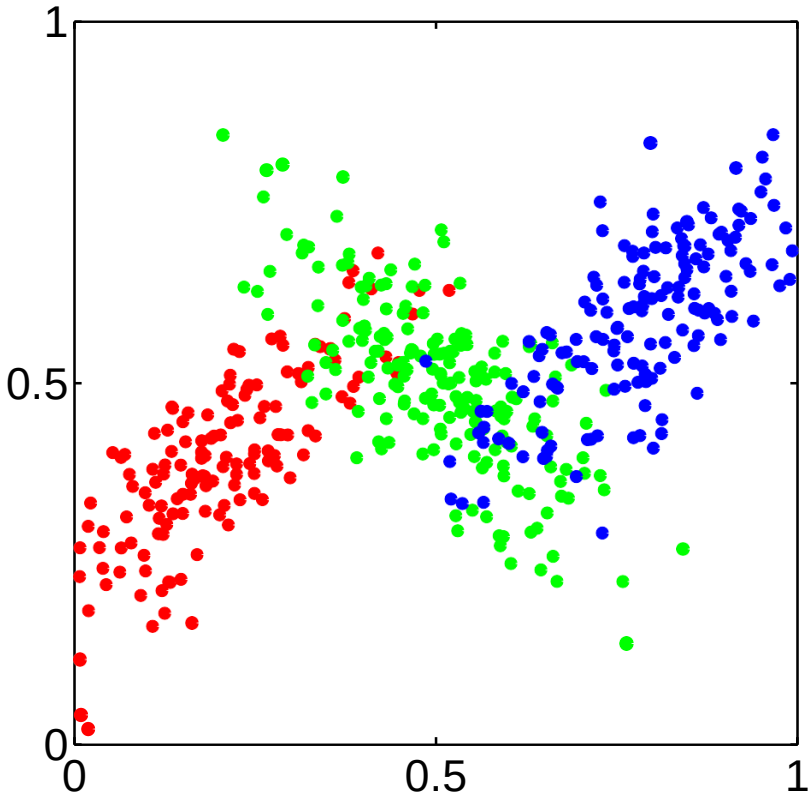
# Contours of Probability Distribution



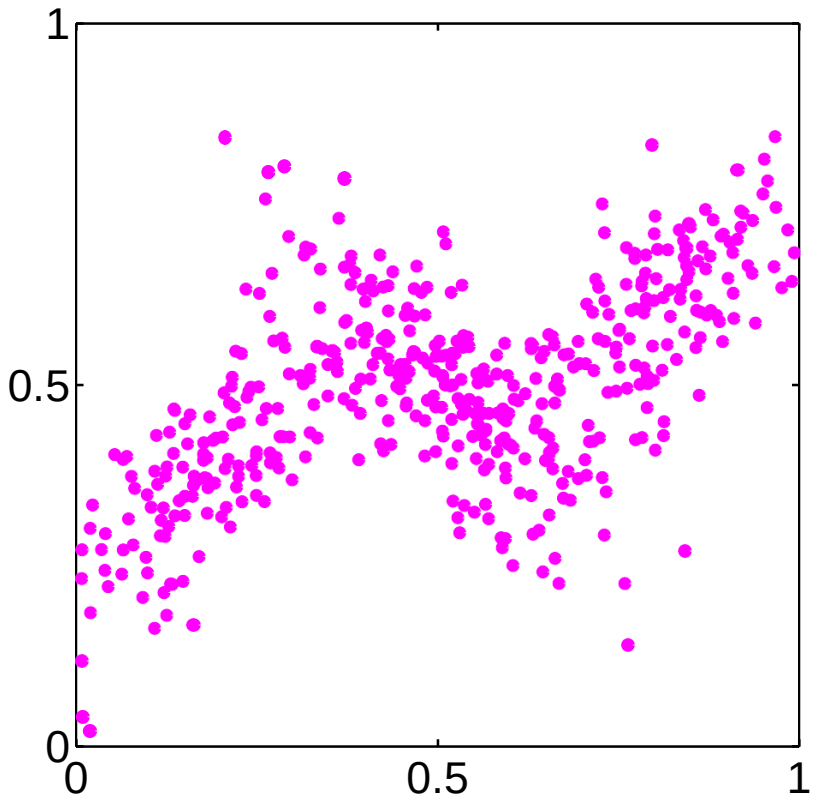
# Surface Plot



# Synthetic Data Set

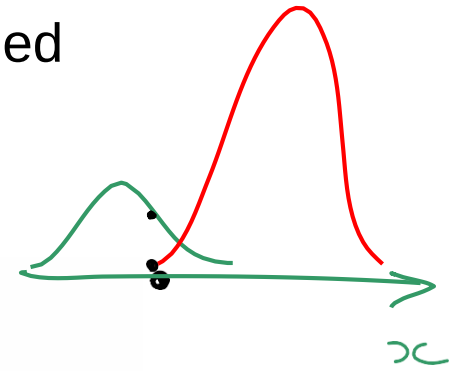


# Synthetic Data Set Without Labels



# Posterior Probabilities

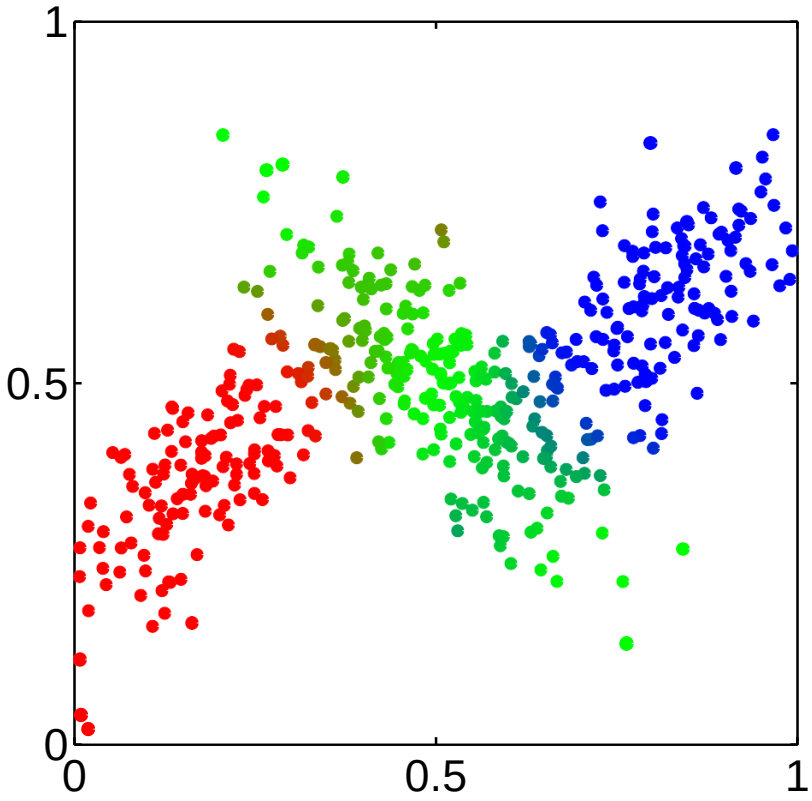
- We can think of the mixing coefficients as prior probabilities for the components
- For a given value of $\mathbf{x}$ we can evaluate the corresponding posterior probabilities, called *responsibilities*
- These are given from Bayes' theorem by



$$\gamma_k(\mathbf{x}) \equiv p(k|\mathbf{x}) = \frac{p(k)p(\mathbf{x}|k)}{p(\mathbf{x})}$$

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

# Posterior Probabilities (colour coded)



# EM algorithm for GMM



1. Initialize the means μk, covariances Σk and mixing coefficients πk, 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

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

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

$$\pi_k^{\text{new}} = \frac{N_k}{N} \qquad \text{where} \quad N_k=\sum_{n=1}^{N}\gamma(z_{nk})$$

4. Evaluate the log likelihood

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

# Segmentation as clustering

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

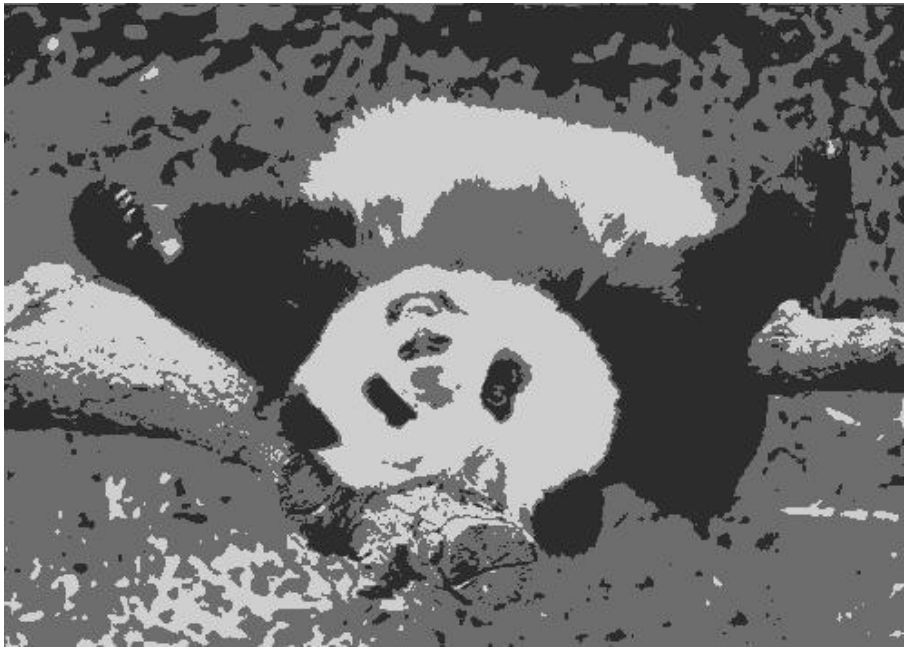
Grouping pixels based on **intensity** similarity

Feature space: intensity value (1-d)

Source: K. Grauman

K=2

K=3



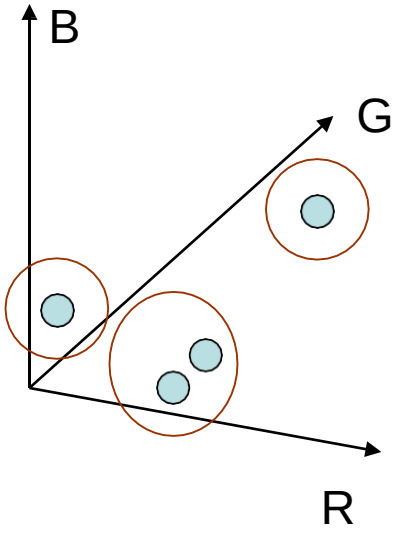
*quantization* of the feature space; segmentation label map

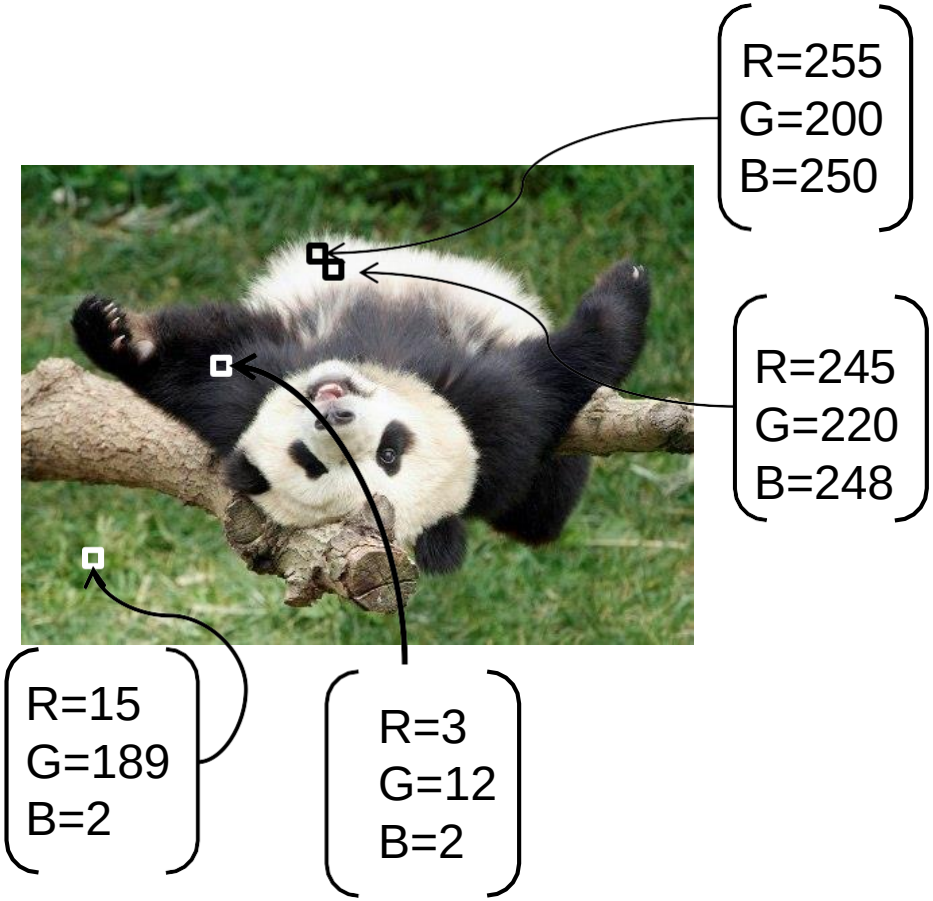
Source: K. Grauman

# 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

# Application: Face Recognition

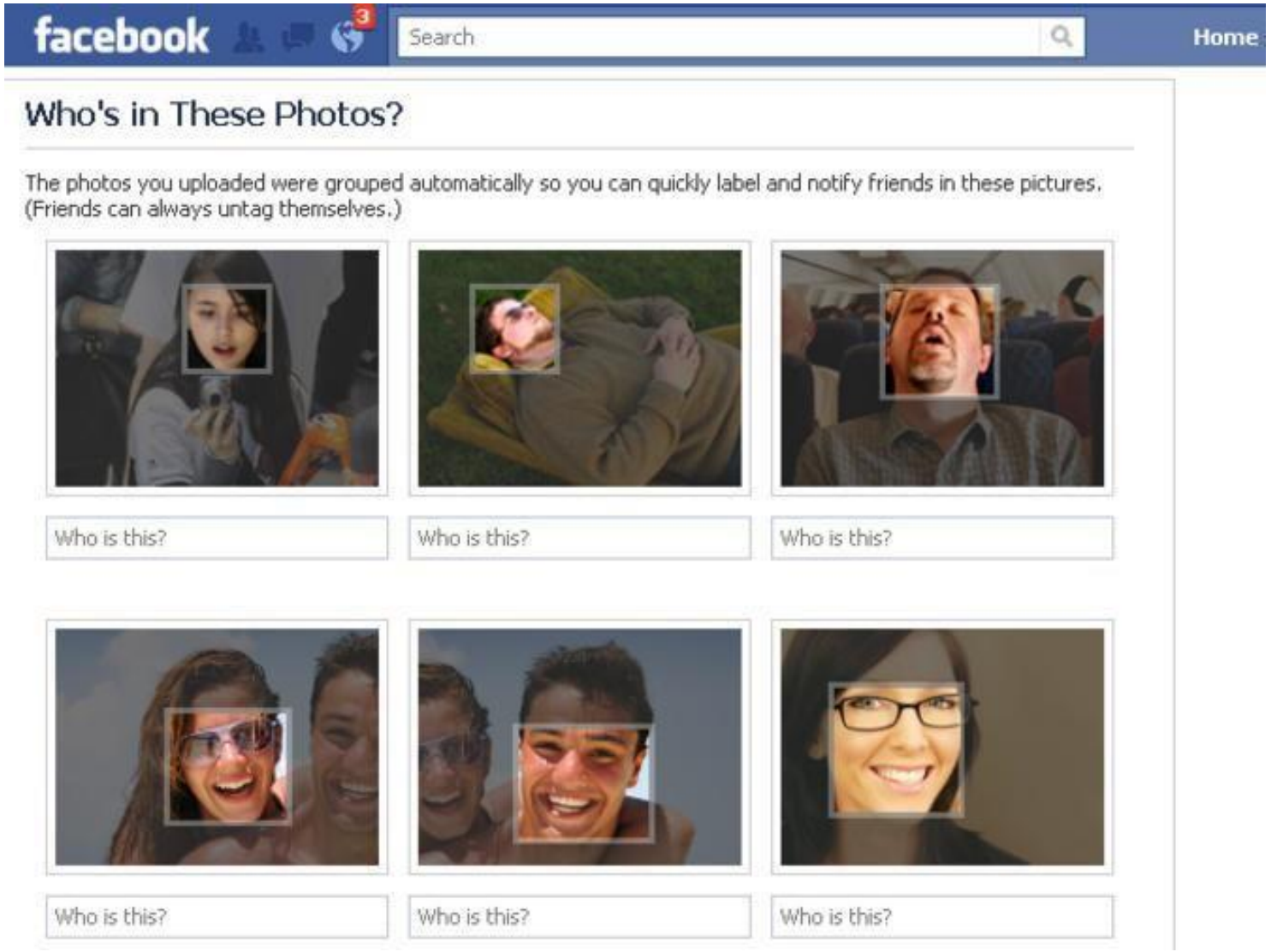



Image from cnet.com

# References

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

Gaussian Mixture Models for Clustering – YouTube

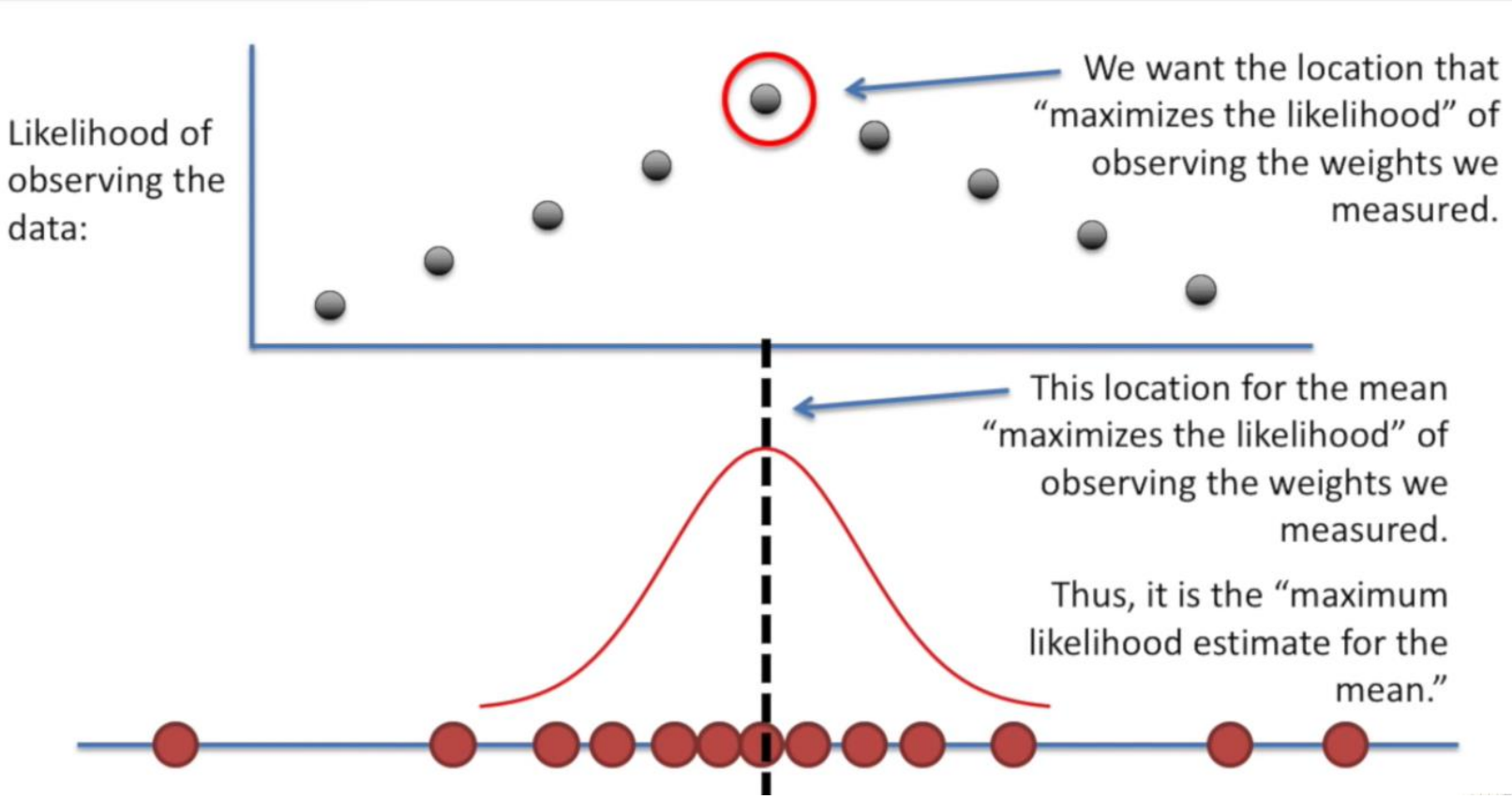
Clustering (4): Gaussian Mixture Models and EM – YouTube

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

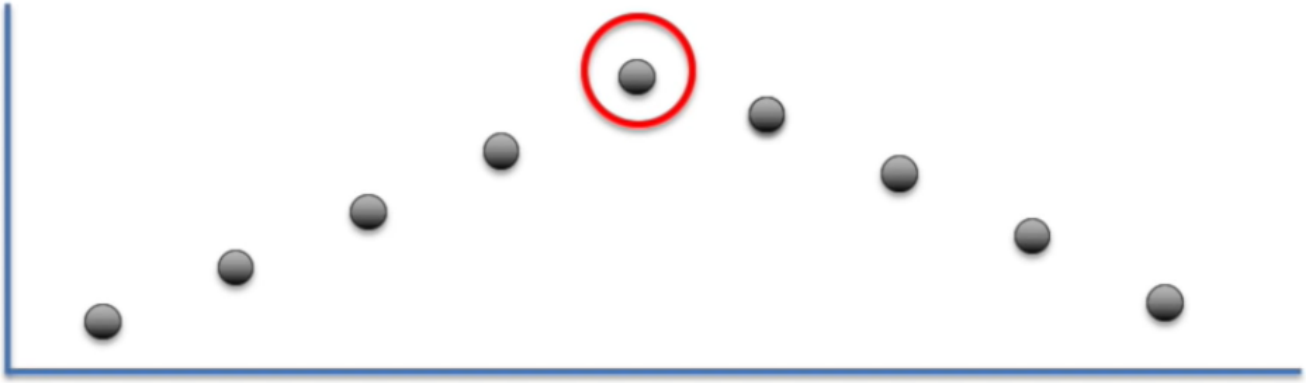
Multivariate Gaussian

https://www.youtube.com/watch?v=eho8xH3E6mE

# Maximum likelihood Estimate



Likelihood of observing the data:



Now when someone says that they have the maximum likelihood estimates for the mean or the standard deviation, or for something else...

... you know that they found the value for the mean or the standard deviation (or for whatever) that maximizes the likelihood that you observed the things you observed.