CS9840a
Learning and Computer Vision Prof.
Olga Veksler

Lecture 11

Unsupervised Learning

EM

Today

- New Topic: *Unsupervised Learning*
  - supervised vs. unsupervised learning
  - unsupervised learning
    - nonparametric unsupervised learning = clustering
      - Proximity Measures
      - Criterion Functions
      - k-means
    - brief intro to Bayesian decision theory
      - need this for parametric supervised learning
    - parametric unsupervised learning
      - Expectation Maximization (EM)

Supervised vs. Unsupervised Learning

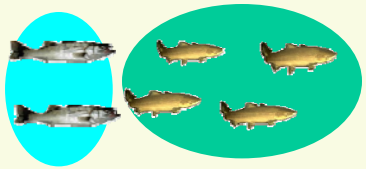
- Up to now considered ***supervised learning*** scenario, where we have
  1. samples $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$
  2. class label $\boldsymbol{y}_i$ for all samples $\boldsymbol{x}_i$
  - this is also called learning with teacher

- In this lecture we consider ***unsupervised learning*** scenario, where we are only given
  1. samples $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$
  - this is also called learning without teacher
  - do not split data into training and test sets
  - harder to say how good results are compared to the supervised case

Unsupervised Learning

- Data is *not labeled*

- Parametric Approach
  - assume parametric distribution of data
  - estimate parameters of this distribution
- Non-parametric Approach
  - group the data into ***clusters***
  - samples inside each cluster are more similar than samples across clusters
  - each cluster (hopefully) says something about categories (classes) present in the data

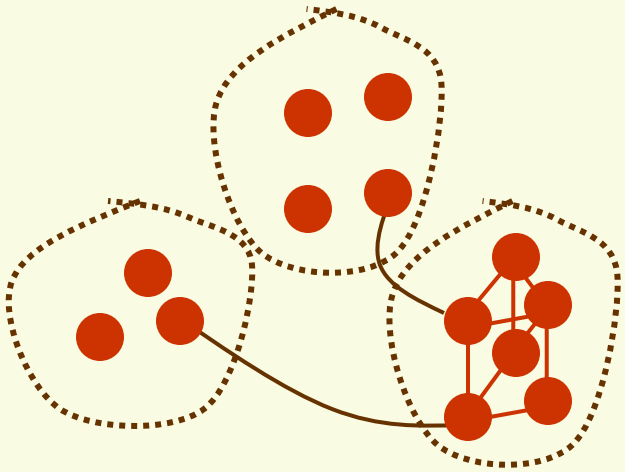


Why Unsupervised Learning?

- Unsupervised learning is harder
  - How do we know if results are meaningful? No answer labels are available.
    - Let the expert look at the results (external evaluation)
    - Define an objective function on clustering (internal evaluation)
- We nevertheless need it because
  1. Labeling large datasets is very costly (speech recognition)
     - sometimes can label only a few examples by hand
  2. May have no idea what/how many classes there are (data mining)
  3. May want to use clustering to gain some insight into the structure of the data before designing a classifier
     - Clustering as data description

Clustering

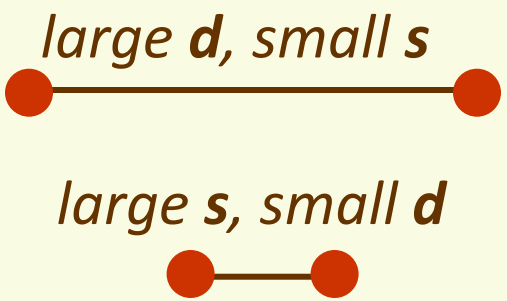
- Seek “natural” clusters in the data



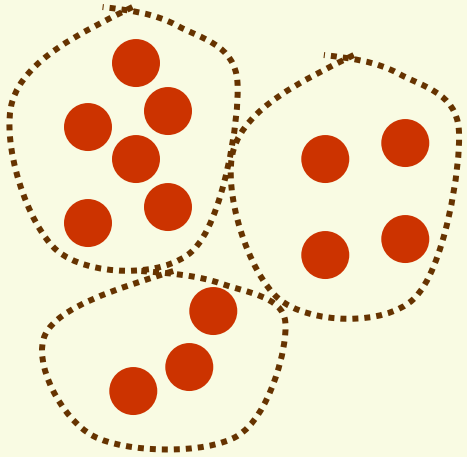
- What is a good clustering?
  - internal (within the cluster) distances should be small
  - external (intra-cluster) should be large
- Clustering is a way to discover new categories (classes)

What we Need for Clustering

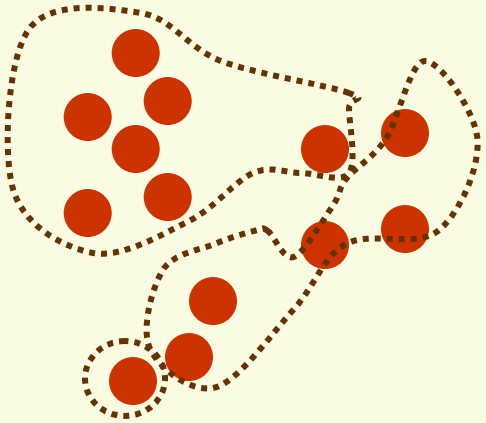
1. Proximity measure, *either*
   - similarity measure $s(x_i,x_k)$: large if x_i,x_k are similar
   - dissimilarity (or distance) measure $d(x_i,x_k)$: small if x_i,x_k are similar



2. Criterion function to evaluate a clustering



good clustering



bad clustering

3. Algorithm to compute clustering
   - usually by optimizing the criterion function

How Many Clusters?

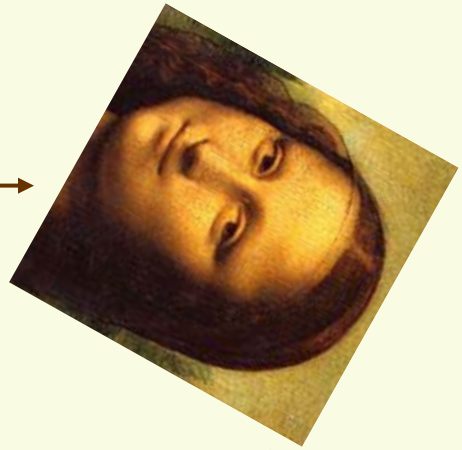
3 clusters or 2 clusters?

- Possible approaches
  - fix the number of clusters to ***k***
  - find the best clustering according to the criterion function (number of clusters may vary)

Proximity Measures

- good proximity measure is application dependent
  - clusters should be invariant under transformations “natural” to the problem
  - for object recognition, should have invariance to rotation

small distance



  - For character recognition, no invariance to rotation

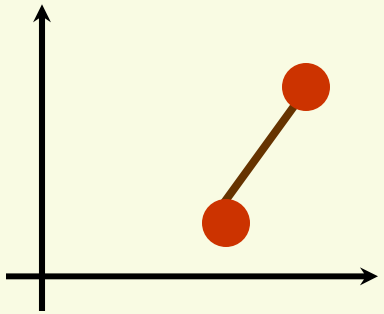
9 —*large distance*→ 6

Distance (dissimilarity) Measures

- Euclidean distance

$$d(x_i, x_j) = \sqrt{\sum_{k=1}^{d} \left(x_i^{(k)} - x_j^{(k)}\right)^2}$$

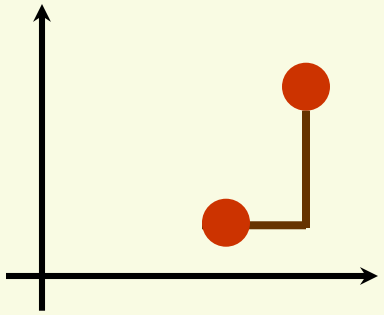
  - translation invariant



- Manhattan (city block) distance

$$d(x_i, x_j) = \sum_{k=1}^{d} \left| x_i^{(k)} - x_j^{(k)} \right|$$

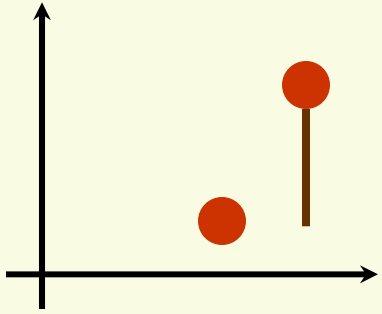
  - approximation to Euclidean distance, cheaper to compute



- Chebyshev distance

$$d(x_i, x_j) = \max_{1 \le k \le d} \left| x_i^{(k)} - x_j^{(k)} \right|$$

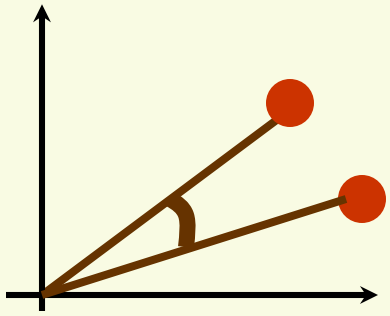
  - approximation to Euclidean distance, cheapest to compute



Similarity Measures

- Cosine similarity:

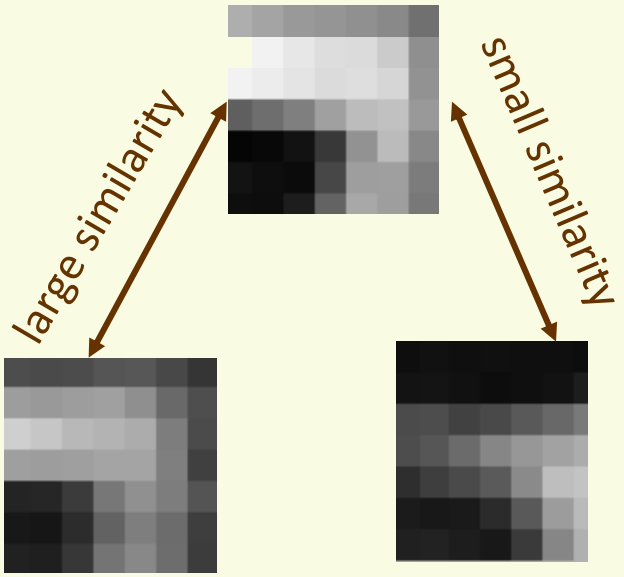
$$s(x_i, x_j) = \frac{x_i^T x_j}{\| x_i \| \, \| x_j \|}$$



  - smaller angle, gives larger similarity
  - scale invariant measure
  - popular in text retrieval

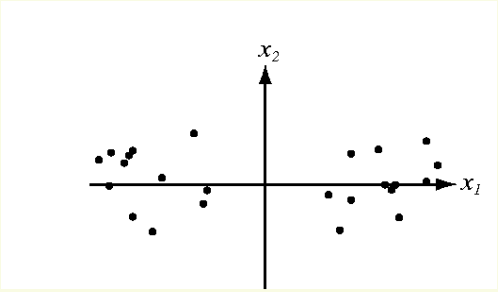
- Correlation coefficient
  - popular in image processing

$$s(x_i, x_j) = \frac{\sum_{k=1}^{d} \left(x_i^{(k)} - \bar{x}_i\right)\left(x_i^{(k)} - \bar{x}_i\right)}{\left[\sum_{k=1}^{d} \left(x_i^{(k)} - \bar{x}_i\right)^2 \sum_{k=1}^{d} \left(x_j^{(k)} - \bar{x}_j\right)^2\right]^{1/2}}$$

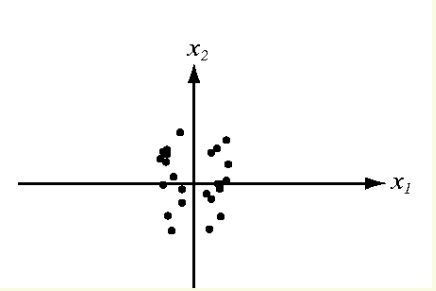


Feature Scale

- old problem: how to choose appropriate relative scale for features?
  - [length (in meters or cms?), weight(in in grams or kgs?)]
  - In supervised learning, can normalize to zero mean unit variance with no problems
  - in clustering this is more problematic, ***if variance in data is due to cluster presence, then normalizing features is not a good thing***



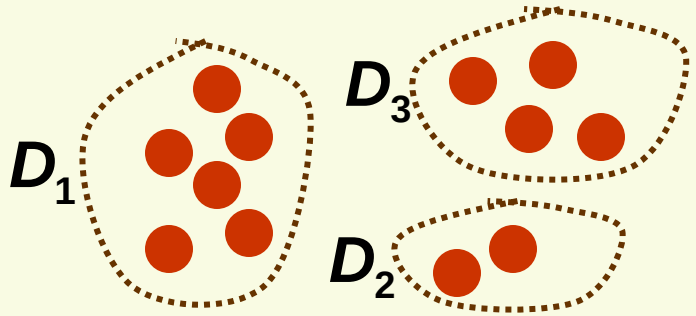
before normalization



after normalization

Criterion Functions for Clustering

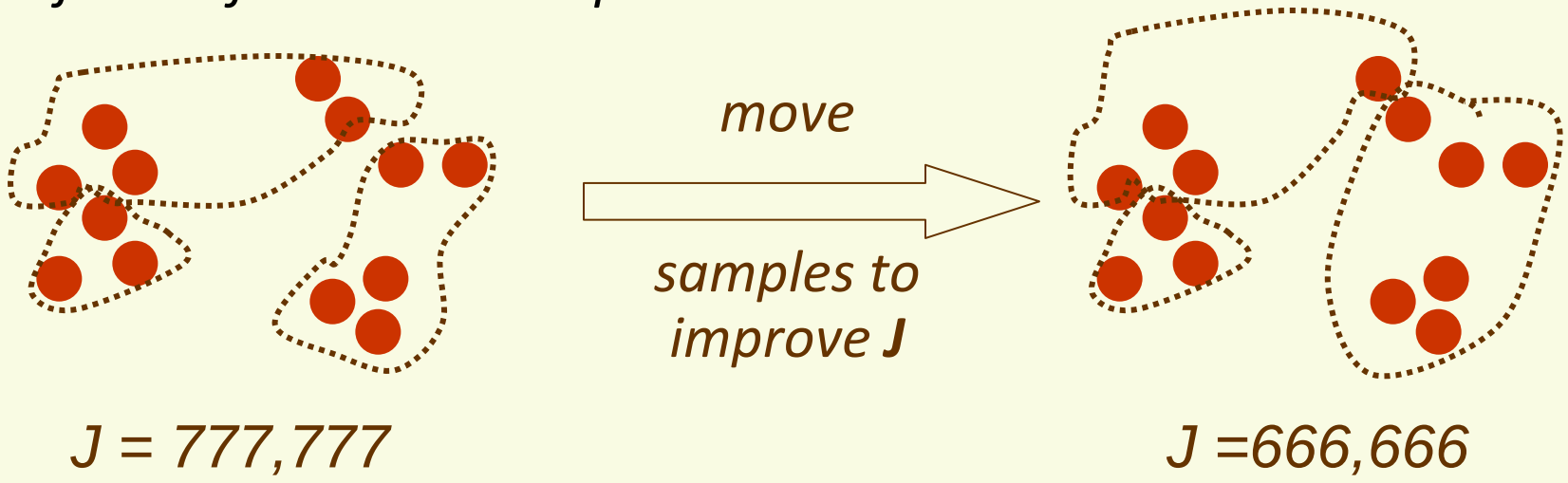
- Have samples $x_1,\ldots,x_n$
- Suppose partitioned samples into c subsets $D_1,\ldots,D_c$



- There are approximately $c^n/c!$ distinct partitions
- Can define a criterion function $J(D_1,\ldots,D_c)$ which measures the quality of a partitioning $D_1,\ldots,D_c$
- Then the clustering problem is well defined
  - optimal clustering is partition which optimizes criterion function

Iterative Optimization Algorithms

- Now have both proximity measure and criterion function, need algorithm to find the optimal clustering
- Exhaustive search is impossible, since there are approximately $c^n/c!$ possible partitions
- Usually some iterative algorithm is used
  1. find a reasonable initial partition
  2. repeat: *move samples from one group to another s.t. the objective function **J** is improved*



K-means Clustering

- Probably the most famous clustering algorithm
- J_{SSE} **objective** function:

$$J_{SSE} = \sum_{i=1}^{k} \sum_{x \in D_i} \| x - \mu_i \|^2$$

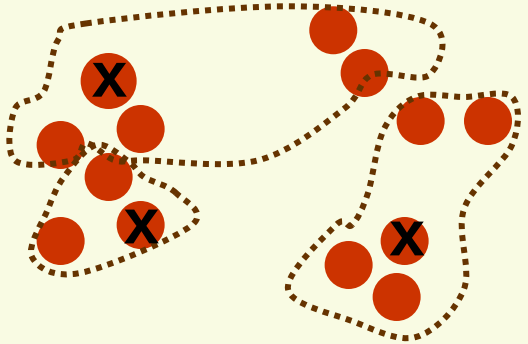
  - can work for some other objective functions, but not proven to work as well

- Fix number of clusters to k ($c = k$)
- Iterative clustering algorithm
- Moves many samples from one cluster to anther in a smart way

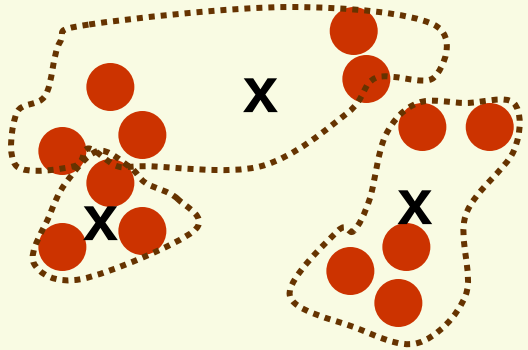
K-means Clustering

k = 3

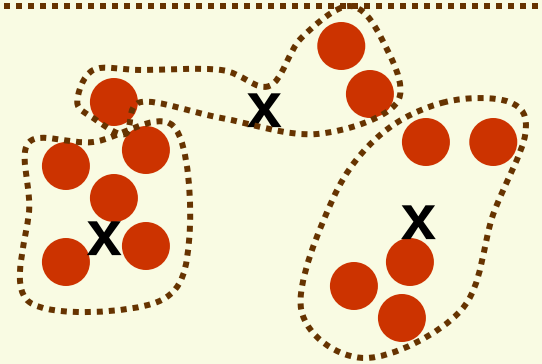
1. Initialize
   - pick ***k*** cluster centers arbitrary
   - assign each example to closest center



2. compute sample means for each cluster



3. reassign all samples to the closest mean

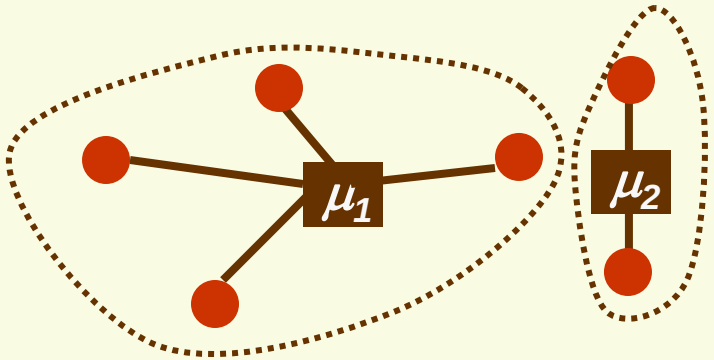


4. if clusters changed at step 3, go to step 2

K-means Clustering

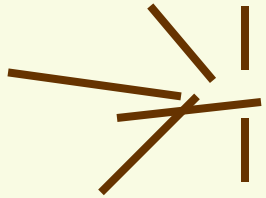
- Consider steps *2* and *3* of the algorithm

2. compute sample means for each cluster

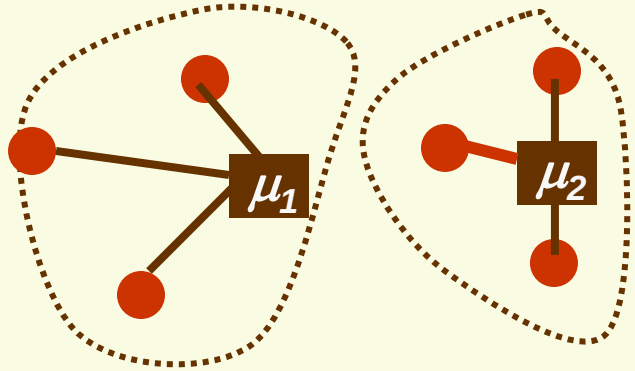


$$J_{SSE} = \sum_{i=1}^{k} \sum_{x \in D_i} \| x - \mu_i \|^2$$

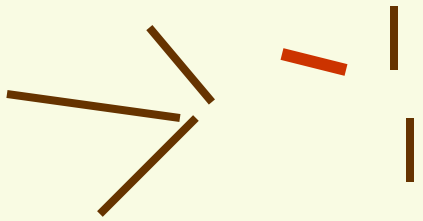
= sum of



3. reassign all samples to the closest mean

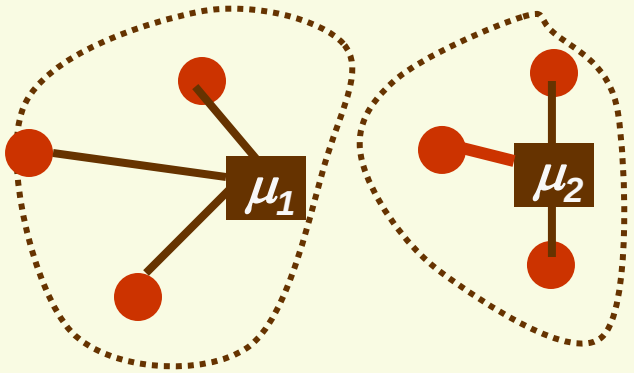


If we represent clusters by their old means, the error has gotten smaller

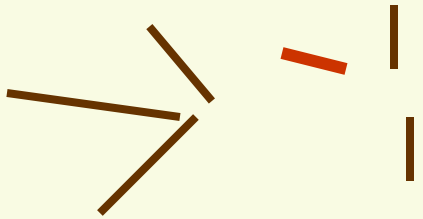


K-means Clustering

3. reassign all samples to the closest mean



If we represent clusters by their old means, the error has gotten smaller



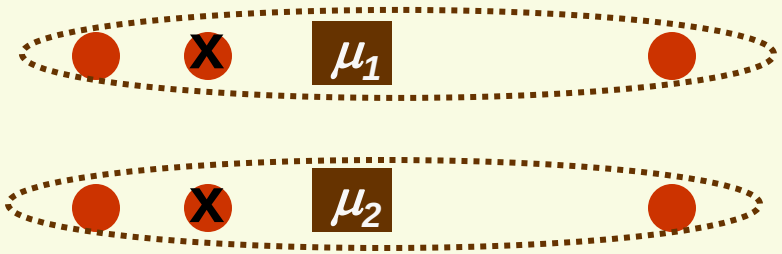
- However we represent clusters by their new means, and mean is always smallest representation of cluster

$$\frac{\partial}{\partial z}\sum_{x\in D_i}\frac{1}{2}\|x-z\|^2=\frac{\partial}{\partial z}\sum_{x\in D_i}\frac{1}{2}\left(\|x\|^2-2x^tz+\|z\|^2\right)=\sum_{x\in D_i}(-x+z)=0$$

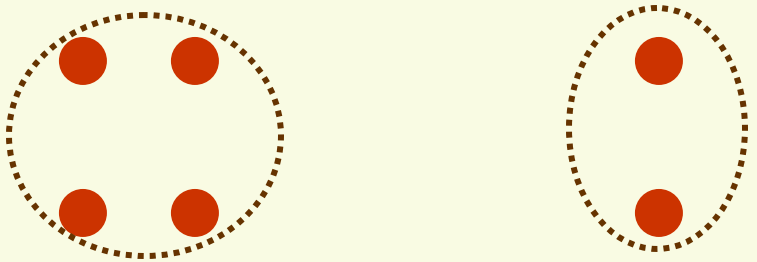
$$\Rightarrow z=\frac{1}{n_i}\sum_{x\in D_i}x$$

K-means Clustering

- We just proved that by doing steps *2* and *3*, the objective function goes down
  - found "smart " move which decreases objective function
- Algorithm converges after a finite number of iterations
- However not guaranteed to find a global minimum



2-means gets stuck here



global minimum of J_{SSE}

K-means Clustering

- Finding the optimum of J_{SSE} is NP-hard
- In practice, **k**-means usually performs well
- Very efficient
- Its solution can be used as a starting point for other clustering algorithms
- Many variants and improvements of **k**-means clustering exist

Bayesian Decision Theory

- Know probability distribution of the categories
  - almost never the case in real life!
  - nevertheless useful since other cases can be reduced to this one after some work
- Do not even need training data
- Can design optimal classifier

Example: Fish Sorting

- Respected fish expert says that
  - salmon length has distribution ***N***(5,1)
  - sea bass length has distribution ***N***(10,4)

- Recall N(μ,σ^2)

$$p(l) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(l-\mu)^2}{2\sigma^2}}$$

- Thus *class conditional* densities are

$$p(l \mid salmon) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(l-5)^2}{2}} \qquad p(l \mid bass) = \frac{1}{2\sqrt{2\pi}} e^{-\frac{(l-10)^2}{2*4}}$$

Likelihood function

- *Class conditional densities* are

$$p(l \mid salmon) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(l-5)^2}{2}} \qquad p(l \mid bass) = \frac{1}{2\sqrt{2\pi}} e^{-\frac{(l-10)^2}{2*4}}$$

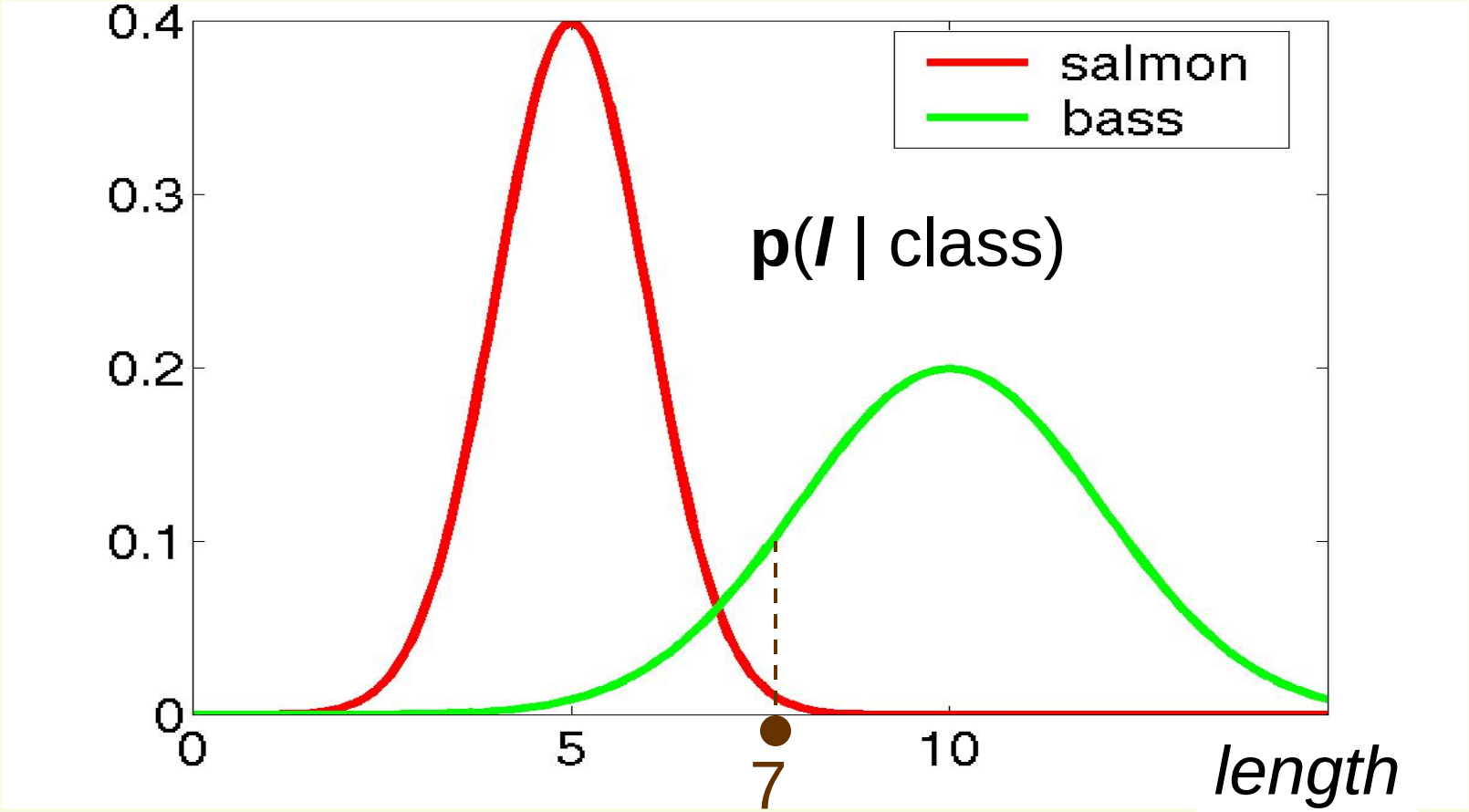
(l: fixed)

- Fix length, let fish class vary
- get *likelihood function*
  - it is ***not density*** and ***not probability mass***

$$p(l \mid class) = \begin{cases} \frac{1}{\sqrt{2\pi}} e^{-\frac{(l-5)^2}{2}} & \text{if } class = salmon \\ \frac{1}{2\sqrt{2\pi}} e^{-\frac{(l-10)^2}{8}} & \text{if } class = bass \end{cases}$$

(l: fixed)

Likelihood vs. Class Conditional Density



Suppose a fish has length 7. How do we classify it?

ML (maximum likelihood) Classifier

- Intuitively, want to choose salmon if

$$Pr[length = 7 \mid salmon] > Pr[length = 7 \mid bass]$$

- However, since ***length*** is a continuous r.v.,

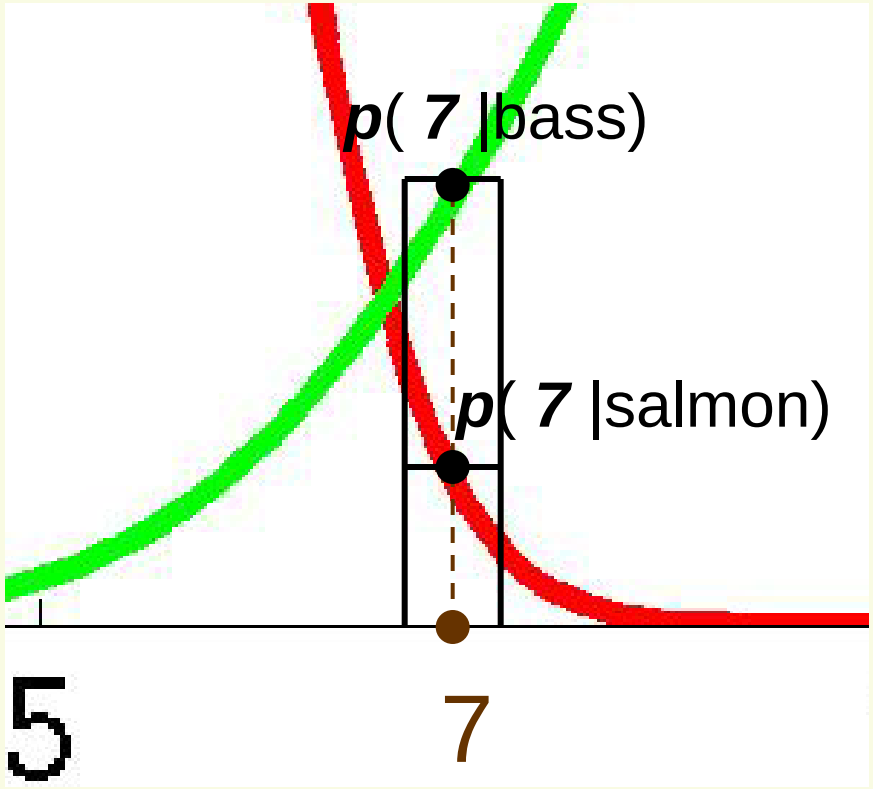
$$Pr[length = 7 \mid salmon] = Pr[length = 7 \mid bass] = 0$$

- Instead, we choose class which maximizes likelihood

$$p(l \mid salmon) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(l-5)^2}{2}} \qquad p(l \mid bass) = \frac{1}{2\sqrt{2\pi}} e^{-\frac{(l-10)^2}{2*4}}$$

- ML classifier: $p(l \mid salmon) \underset{salmon}{\overset{bass}{?}} p(l \mid bass)$ (bass if $<$, salmon if $>$)
  - if p(l | salmon) > p(l | bass), classify salmon, else classify bass

Interval Justification



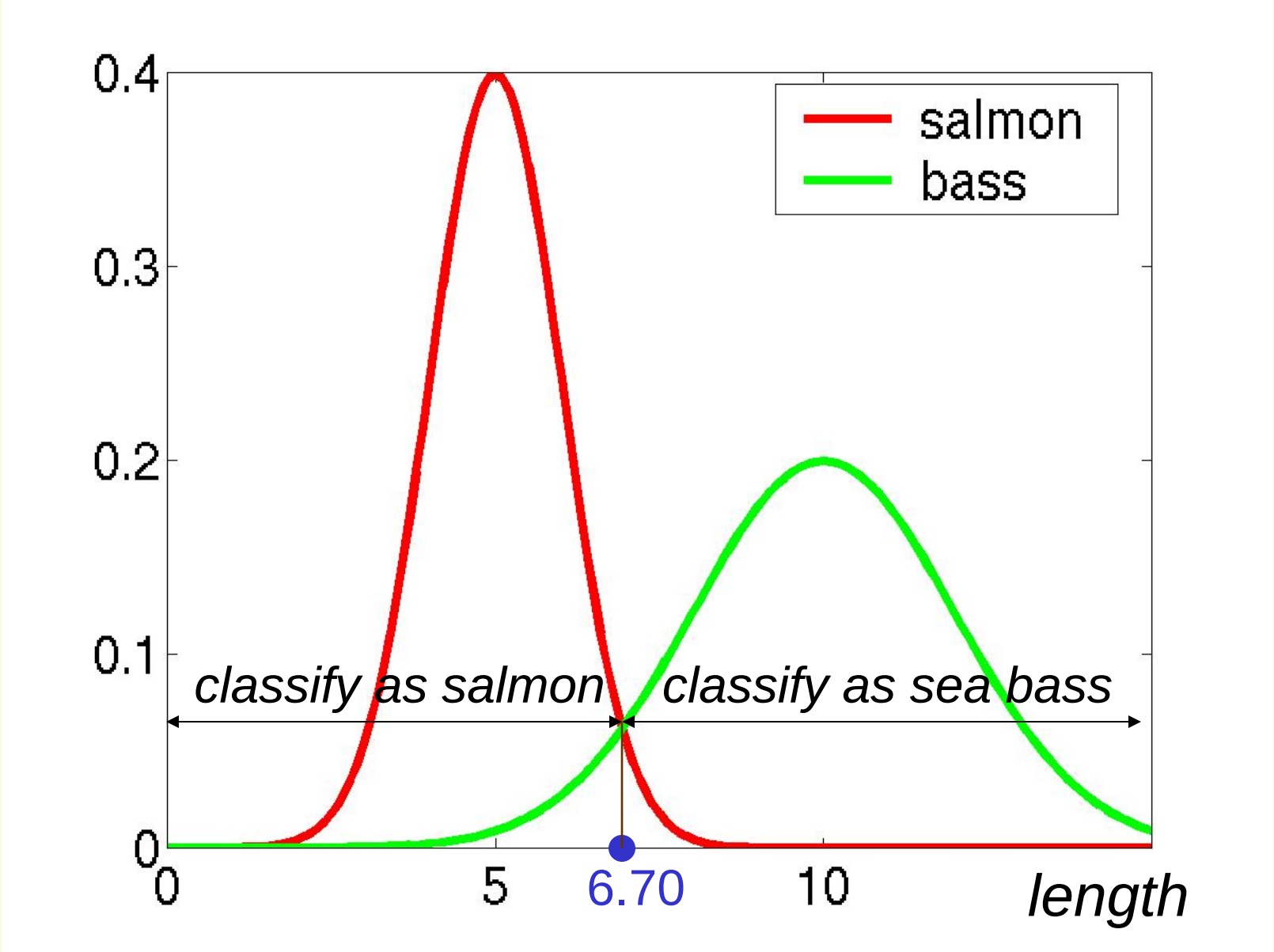
Thus we choose the class (bass) which is more likely to have given the observation

$$Pr[I \in B(7) \mid bass] \approx 2\varepsilon\, p(7 \mid bass)$$

$$\Uparrow \qquad \Leftarrow \qquad \Uparrow$$

$$Pr[I \in B(7) \mid salmon] \approx 2\varepsilon\, p(7 \mid salmon)$$

Decision Boundary

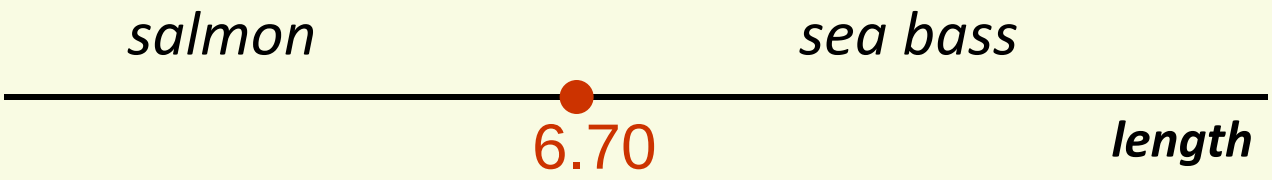


Priors

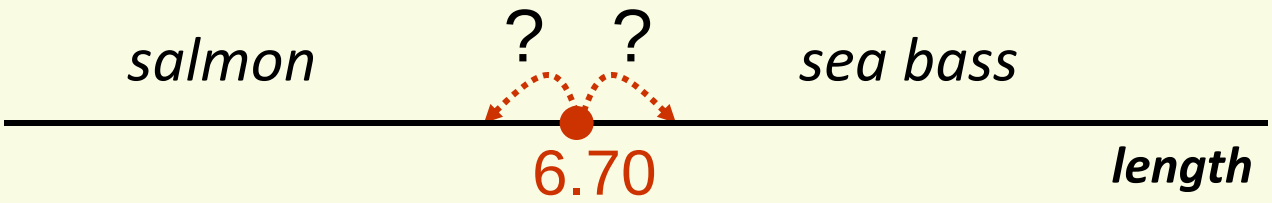
- Prior comes from prior knowledge, no data has been seen yet
- Suppose a fish expert says: in the fall, there are twice as many salmon as sea bass
- Prior for our fish sorting problem
  - ***P***(salmon) = 2/3
  - ***P***(bass) = 1/3
- With the addition of prior to our model, how should we classify a fish of length 7?

How Prior Changes Decision Boundary?

- Without priors



- How should this change with prior?
  - ***P***(salmon) = 2/3
  - ***P***(bass) = 1/3



Bayes Decision Rule

1. Have likelihood functions
   - **p**(length | salmon)
   - **p**(length | bass)
2. Have priors **P**(salmon) and **P**(bass)

- Question: Having observed fish of certain length, do we classify it as salmon or bass?
- Natural Idea:
  - salmon if **P**(salmon|length) > **P**(bass|length)
  - bass if **P**(bass|length) > **P**(salmon|length)

Posterior

- ***P***(salmon | length) and ***P***(bass | length) are called posterior distributions, because the data (length) was revealed (post data)
- How to compute posteriors?
- Bayes rule: $\mathbf{P(salmon \mid length)} = \frac{\mathbf{p(salmon, length)}}{\mathbf{p(length)}}$

$$= \frac{\mathbf{p(length \mid salmon)P(salmon)}}{\mathbf{p(length)}}$$

- Similarly: $\boldsymbol{P(bass \mid length)} = \frac{\boldsymbol{p(length \mid bass)P(bass)}}{\boldsymbol{p(length)}}$

MAP (maximum a posteriori) classifier

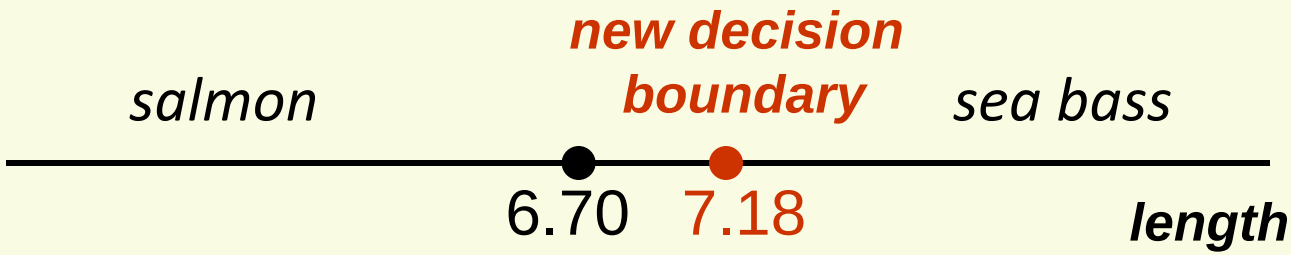
$$P(\text{salmon} \mid \text{length}) \overset{\text{salmon}}{\underset{\text{bass}}{\gtrless}} P(\text{bass} \mid \text{length})$$

$$\frac{p(\text{length} \mid \text{salmon})P(\text{salmon})}{p(\text{length})} \overset{\text{salmon}}{\underset{\text{bass}}{\gtrless}} \frac{p(\text{length} \mid \text{bass})P(\text{bass})}{p(\text{length})}$$

$$p(\text{length} \mid \text{salmon})P(\text{salmon}) \overset{\text{salmon}}{\underset{\text{bass}}{\gtrless}} p(\text{length} \mid \text{bass})P(\text{bass})$$

Back to Fish Sorting

- likelihood $p(l \mid salmon) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(l-5)^2}{2}}$ $p(l \mid bass) = \frac{1}{2\sqrt{2\pi}} e^{-\frac{(l-10)^2}{8}}$
- Priors: P(salmon) = 2/3, P(bass) = 1/3
- Solve inequality $\frac{1}{\sqrt{2\pi}} e^{-\frac{(l-5)^2}{2}} * \frac{2}{3} > \frac{1}{2\sqrt{2\pi}} e^{-\frac{(l-10)^2}{8}} * \frac{1}{3}$



- New boundary makes sense: expect to see more salmon

More on Posterior

posterior density (our goal)

likelihood (given)

Prior (given)

$$P(c \mid l) = \frac{P(l \mid c)\, P(c)}{P(l)}$$

normalizing factor, often do not even need it for classification since $P(l)$ *does not depend on class* c*. If we do need it, from the law of total probability:*

$$P(l) = p(l \mid salmon)\,p(salmon) + p(l \mid bass)\,p(bass)$$

Notice this formula consists of likelihoods and priors, which are given

More on Priors

- Prior comes from prior knowledge, no data has been seen yet
- If there is a reliable source prior knowledge, it should be used
- Some problems cannot even be solved reliably without a good prior
- However prior alone is not enough, we still need likelihood
  - P(salmon)=2/3, P(sea bass)=1/3
  - If I don't let you see the data, but ask you to guess, will you choose salmon or sea bass?

More on Map Classifier

$$P(c|I) = \frac{P(I|c)\ P(c)}{P(I)}$$

posterior *likelihood* *prior*

- Do not care about $P(I)$ when maximizing $P(c|I)$

$$P(c|I) \overset{\text{proportional}}{\propto} P(I|c)P(c)$$

- If $P(\text{salmon}) = P(\text{bass})$ (uniform prior) MAP classifier becomes ML classifier $P(c|I) \propto P(I|c)$

- If for some observation I, $P(I|\text{salmon}) = P(I|\text{bass})$, then this observation is uninformative and decision is based solely on the prior $P(c|I) \propto P(c)$

Justification for MAP Classifier

- Probability of error for MAP estimate:

$$P(\textit{salmon}\,|\,I) \underset{\textit{bass}}{\overset{\textit{salmon}}{\gtrless}} P(\textit{bass}\,|\,I)$$

- For any particular I, probability of error

$$Pr[\text{error}|\ I\ ] = \begin{cases} P(\text{bass}|I) & \text{if we decide salmon} \\ P(\text{salmon}|I) & \text{if we decide bass} \end{cases}$$

Thus MAP classifier is optimal for each individual I

Justification for MAP Classifier

- We are interested to minimize error not just for one ***I***, we really want to minimize the average error over all ***I***

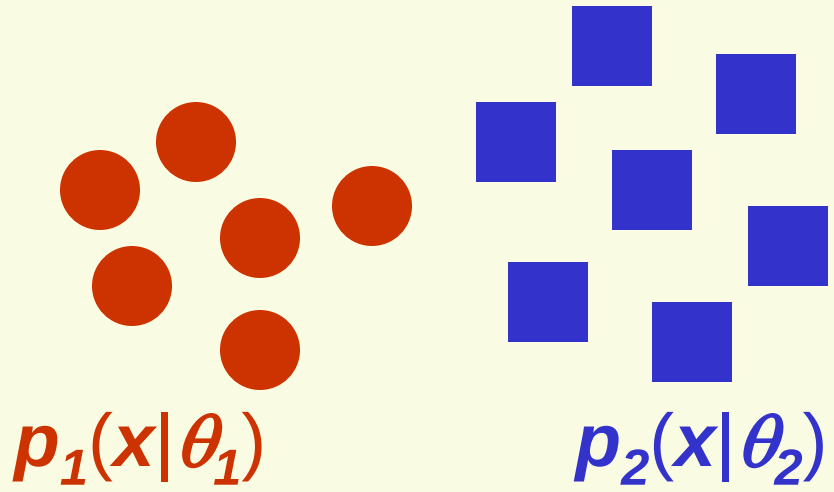
$$Pr[error] = \int_{-\infty}^{\infty} p(error, I)dI = \int_{-\infty}^{\infty} Pr[error \mid I]p(I)dI$$

- If ***Pr***[error| ***I***]is as small as possible, the integral is small as possible
- But Bayes rule makes ***Pr***[error| ***I***] as small as possible

Thus MAP classifier minimizes the probability of error

Parametric Supervised Learning

- Supervised parametric learning
  - have m classes
  - have samples $x_1, \ldots, x_n$ each of class $1, 2, \ldots, m$
  - suppose D_i holds samples from class i
  - probability distribution for class i is $p_i(x \mid \theta_i)$



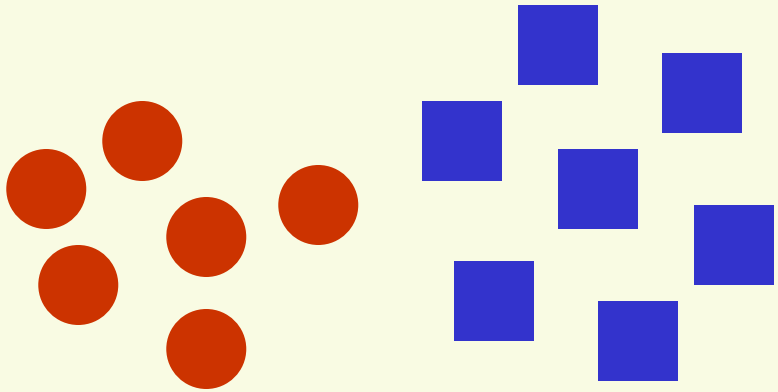
Parametric Supervised Learning

- Use the ML method to estimate parameters θ_i
  - find θ_i which maximizes the likelihood function $F(\theta_i)$

$$p(D_i \mid \theta_i) = \prod_{x \in D_i} p(x \mid \theta_i) = F(\theta_i)$$

  - or, equivalently, find q_i which maximizes the log likelihood $I(\theta_i)$

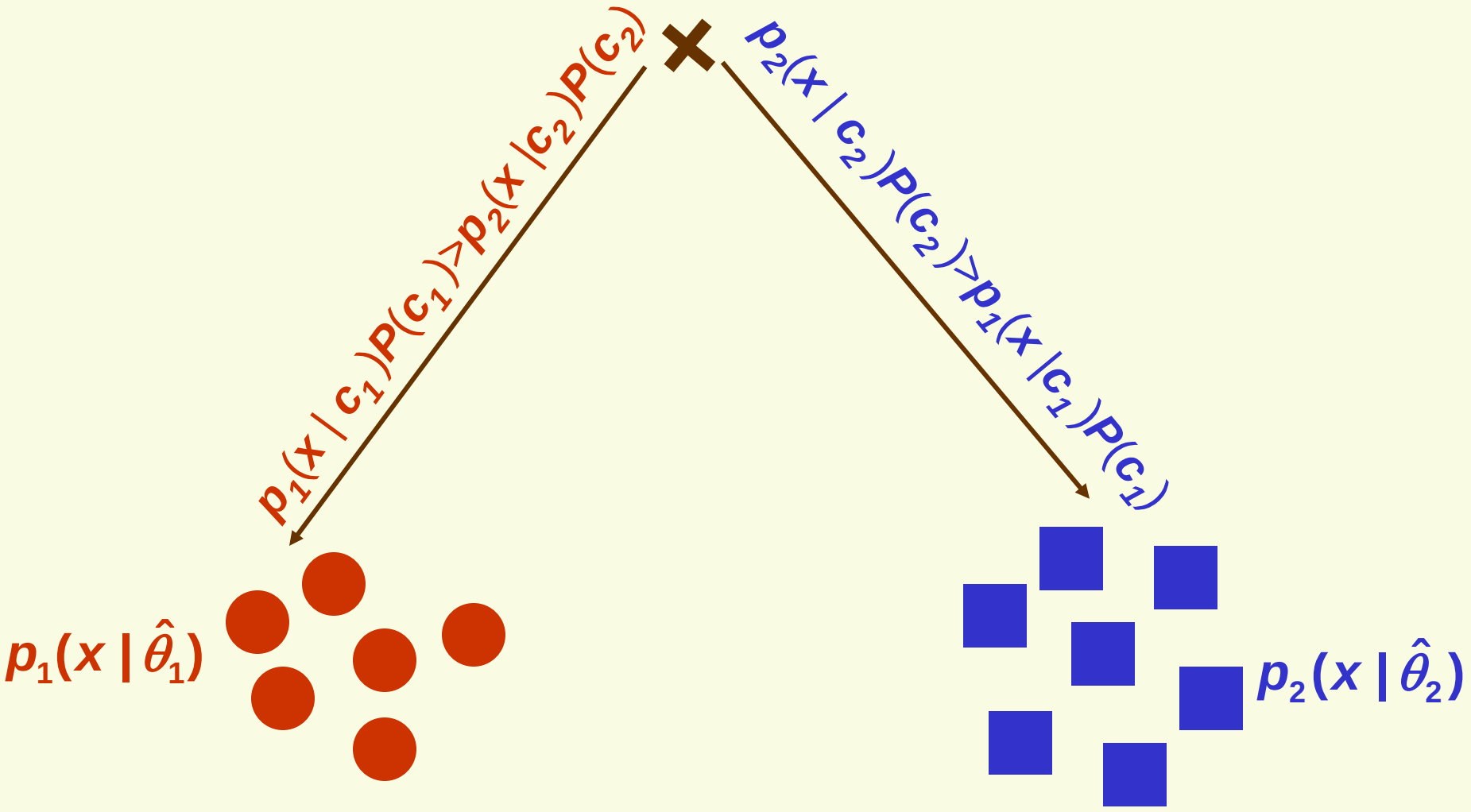
$$I(\theta_i) = \ln p(D_i \mid \theta_i) = \sum_{x \in D_i} \ln p(x \mid \theta_i)$$



$$\hat{\theta}_1 = \arg\max_{\theta_1} [\ln p(D_1 \mid \theta_1)] \qquad \hat{\theta}_2 = \arg\max_{\theta_2} [\ln p(D_2 \mid \theta_2)]$$

Parametric Supervised Learning

- now the distributions are fully specified
- can classify unknown sample using MAP (Maximum A Posteriori) classifier

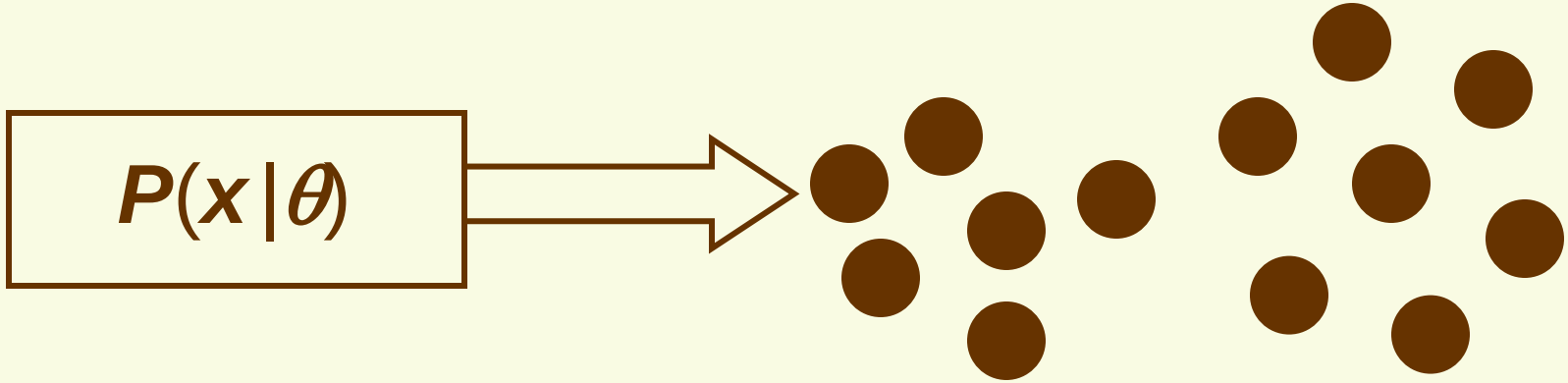


Parametric Unsupervised Learning

- Expectation Maximization (EM)
  - one of the most useful statistical methods
  - oldest version in 1958 (*Hartley*)
  - seminal paper in 1977 (*Dempster et al*.)
  - can also be used when some samples are missing features

Parametric Unsupervised Learning

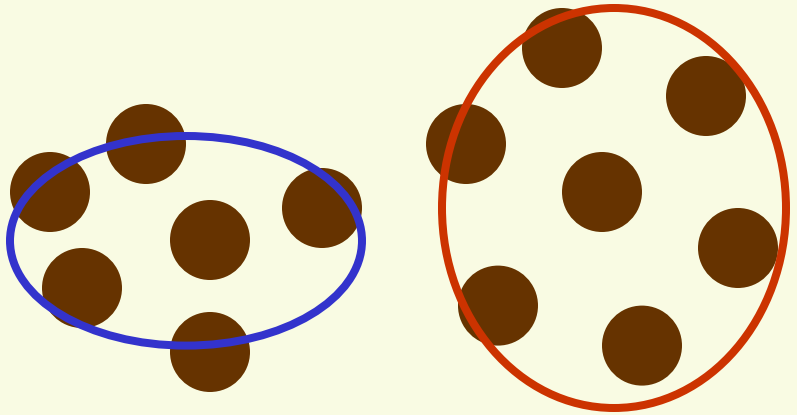
- Assume the data was generated by a model with known shape but unknown parameters



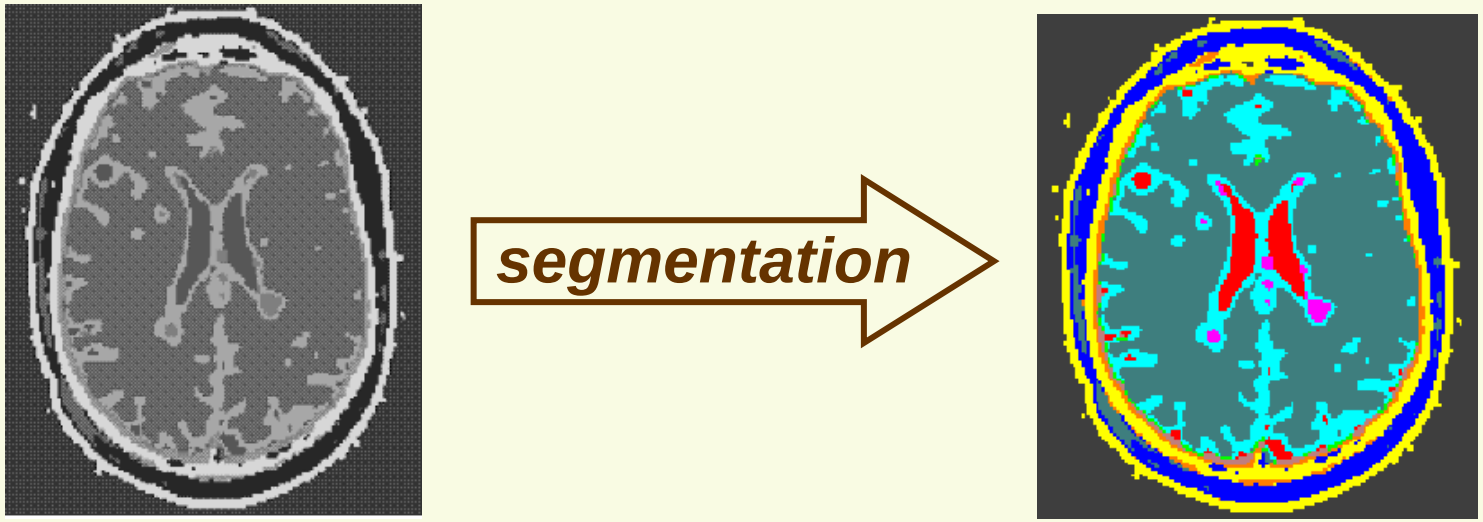
- Advantages of having a model
  - Gives a meaningful way to cluster data
    - adjust the parameters of the model to maximize the probability that the model produced the observed data
  - Can sensibly measure if a clustering is good
    - compute the likelihood of data induced by clustering
  - Can compare 2 clustering algorithms
    - which one gives the higher likelihood of the observed data?

Parametric Unsupervised Learning

- In unsupervised learning, no one tells us the true classes for samples. We still know
  - have m classes
  - have samples $x_1, \ldots, x_n$ each of ***unknown*** class
  - probability distribution for class i is $p_i(x \mid \theta_i)$
- Can we determine the classes and parameters simultaneously?



Example: MRI Brain Segmentation



Picture from M. Leventon

- In MRI brain image, different brain tissues have different intensities
- Know that brain has 6 major types of tissues
- Each type of tissue can be modeled by a Gaussian $\boldsymbol{N}(\mu_i, \sigma_i^2)$ reasonably well, parameters μ_i, σ_i^2 are unknown
- Segmenting (classifying) the brain image into different tissue classes is very useful
  - don't know which image pixel corresponds to which tissue (class)
  - don't know parameters for each $\boldsymbol{N}(\mu_i, \sigma_i^2)$

Mixture Density Model

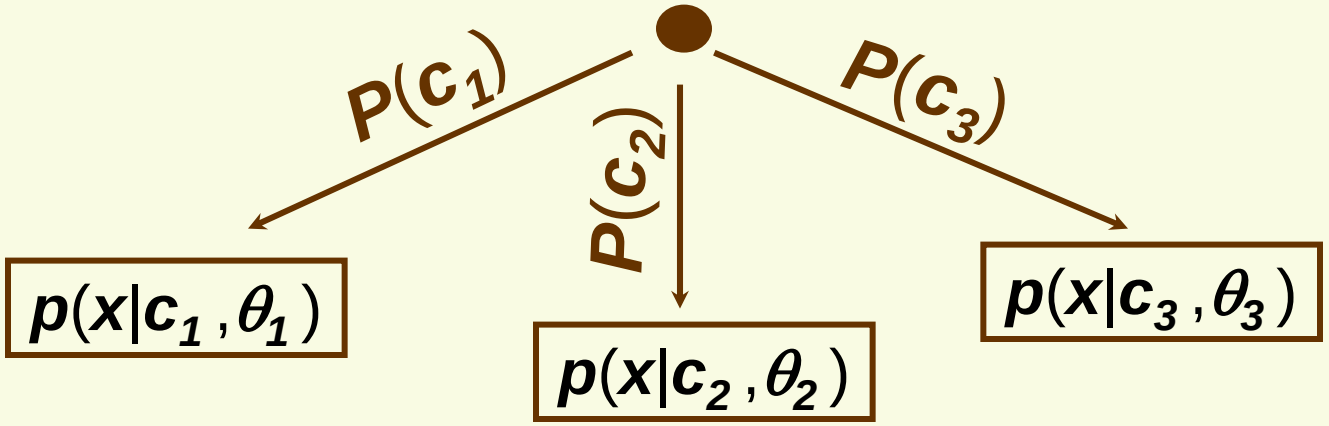
- Model data with ***mixture density***

$$p(x \mid \theta) = \sum_{j=1}^{m} p(x \mid c_j, \theta_j) P(c_j)$$

component densities — *mixing parameters*

  - where $\theta = \{\theta_1, \ldots, \theta_m\}$
  - $P(c_1) + P(c_2) + \ldots + P(c_m) = 1$

- To generate a sample from distribution $p(x|\theta)$
  - first select class j with probability $P(c_j)$
  - then generate x according to probability law $p(x|c_j, \theta_j)$



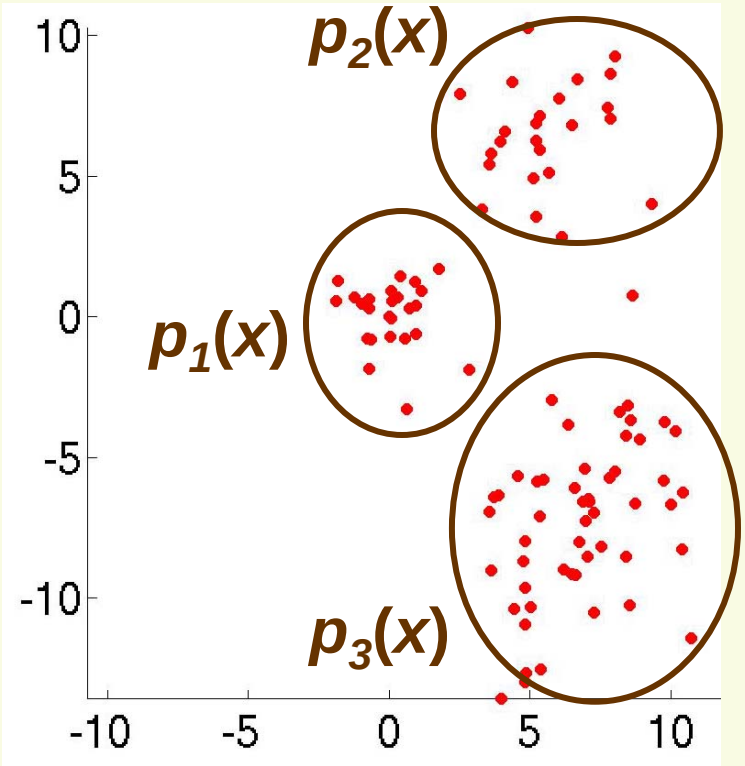
Example: Gaussian Mixture Density

- Mixture of 3 Gaussians

$$p_1(x) \cong N\left(\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right)$$

$$p_2(x) \cong N\left([6,6], \begin{bmatrix}4 & 0\\0 & 4\end{bmatrix}\right)$$

$$p_3(x) \cong N\left([7,-7], \begin{bmatrix}6 & 0\\0 & 6\end{bmatrix}\right)$$



$$p(x) = 0.2p_1(x) + 0.3p_2(x) + 0.5p_3(x)$$

Mixture Density

$$p(x \mid \theta) = \sum_{j=1}^{m} p\left(x \mid c_j, \theta_j\right) P\left(c_j\right)$$

- $P(c_1), \ldots, P(c_m)$ can be known or unknown
- Suppose we know how to estimate $\theta_1, \ldots, \theta_m$ and $P(c_1), \ldots, P(c_m)$
- Can "break apart" mixture $p(x|\theta)$ for classification
- To classify sample x, use MAP estimation, that is choose class i which maximizes

$$P(c_i \mid x, \theta_i) \propto \underbrace{p(x \mid c_i, \theta_i)}_{\text{probability of component } i \text{ to generate } x} \underbrace{P\left(c_i\right)}_{\text{probability of component } i}$$

ML Estimation for Mixture Density

$$p(x \mid \theta, \rho) = \sum_{j=1}^{m} p(x \mid c_j, \theta_j) P(c_j) = \sum_{j=1}^{m} p(x \mid c_j, \theta_j) \rho_i$$

- Use Maximum Likelihood estimation for a mixture density
- Need to estimate
  - $\theta_1, \ldots, \theta_m$
  - $\rho_1 = P(c_1), \ldots, \rho_m = P(c_m)$, and $\rho = \{\rho_1, \ldots, \rho_m\}$
- As in supervised case, form the log likelihood function

$$l(\theta, \rho) = \ln p(D \mid \theta, \rho) = \sum_{k=1}^{n} \ln p(x_k \mid \theta, \rho) = \sum_{k=1}^{n} \ln \left[\sum_{j=1}^{m} p(x \mid c_j, \theta_j) \rho_i \right]$$

ML Estimation for Mixture Density

$$l(\theta,\rho)=\sum_{k=1}^{n}\ln\left[\sum_{j=1}^{m}p\left(x\mid c_j,\theta_j\right)\rho_i\right]$$

- need to maximize $l(\theta,\rho)$ with respect to θ and ρ
- $l(\theta, \rho)$ is not hard to maximize directly
  - partial derivatives with respect to θ, ρ usually give a "coupled" nonlinear system of equation
  - usually closed form solution cannot be found
- We could use the gradient ascent method
  - in general, it is not the greatest method to use, should only be used as last resort
- There is a better algorithm, called ***EM***

Mixture Density

- Before EM, let's look at the mixture density again

$$p(x \mid \theta, \rho) = \sum_{j=1}^{m} p\left(x \mid c_j, \theta_j\right)\rho_j$$

- Say we know how to estimate $\theta_1, \ldots, \theta_m$ and $\rho_1, \ldots, \rho_m$
  - estimating the class of x is easy with MAP, maximize

$$p(x \mid c_i, \theta_i)P\left(c_i\right) = p(x \mid c_i, \theta_i)\rho_i$$

- Suppose know class of samples $x_1, \ldots, x_n$
  - just as supervised case, so estimating $\theta_1, \ldots, \theta_m$ and $\rho_1, \ldots, \rho_m$ is easy

$$\hat{\theta}_i = \arg\max_{\theta_i}\left[\ln p(D_i \mid \theta_i)\right] \qquad \hat{\rho}_i = \frac{|D_i|}{n}$$

- This is an example of chicken-and-egg problem
  - *EM* algorithm solution is adding "hidden" variables

Expectation Maximization Algorithm

- ***EM*** is an algorithm for ML parameter estimation when the data has missing values. It is used when
  1. data is incomplete (has missing values)
     - some features are missing for some samples due to data corruption, partial survey responses, etc.
  2. Suppose data $\boldsymbol{X}$ is complete, but $\boldsymbol{p}(\boldsymbol{X}|\boldsymbol{q})$ is hard to optimize. Suppose further that introducing certain hidden variables $\boldsymbol{U}$ whose values are missing, and suppose it is easier to optimize the "complete" likelihood function $\boldsymbol{p}(\boldsymbol{X},\boldsymbol{U}|\boldsymbol{q})$. Then EM is useful.
     - useful for the mixture density estimation, and is subject of our lecture today
  - Notice after we introduce artificial (hidden) variables $\boldsymbol{U}$ with missing values, case ***2*** is completely equivalent to case ***1***

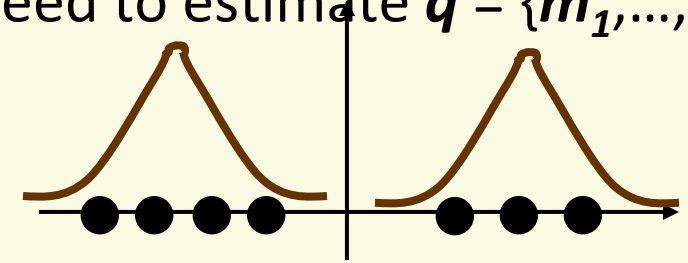
EM: Hidden Variables for Mixture Density

$$p(x \mid \theta) = \sum_{j=1}^{m} p\left(x \mid c_j, \theta_j\right) \rho_j$$

- For simplicity, assume component densities are

$$p\left(x \mid c_j, \theta_j\right) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{\left(x - \mu_j\right)^2}{2\sigma^2}\right)$$

  - assume for now that the variance is known
  - need to estimate $q = \{m_1, \ldots, m_m\}$



- If we knew which sample came from which component the ML parameter estimation would be easy
- To get this easier problem, introduce hidden variables which indicate which component each sample belongs to

EM: Hidden Variables for Mixture Density

- For $1 \le i \le n,\ 1 \le k \le m$, define hidden variables $z_i^{(k)}$

$$z_i^{(k)} = \begin{cases} 1 & \text{if sample } i \text{ was generated by component } k \\ 0 & \text{otherwise} \end{cases}$$

$$x_i \rightarrow \left\{ x_i, z_i^{(1)}, \ldots, z_i^{(m)} \right\}$$

- $z_i^{(k)}$ are indicator ***random*** variables, they indicate which Gaussian component generated sample x_i
- Let $z_i = \{z_i^{(1)}, \ldots, z_i^{(m)}\}$, indicator r.v. corresponding to x_i
- Conditioned on z_i, distribution of x_i is Gaussian

$$p(x_i \mid z_i, \theta) \sim N(\mu_k, \sigma^2)$$

  - where k is s.t. $z_i^{(k)} = 1$

EM: Joint Likelihood

- Let $z_i = \{z_i^{(1)}, \ldots, z_i^{(m)}\}$, and $Z = \{z_1, \ldots, z_n\}$
- The complete likelihood is

$$p(X, Z \mid \theta) = p(x_1, \ldots, x_n, z_1, \ldots, z_n \mid \theta) = \prod_{i=1}^{n} p(x_i, z_i \mid \theta)$$

$$= \prod_{i=1}^{n} \underbrace{p(x_i \mid z_i, \theta)}_{\text{gaussian}} \underbrace{p(z_i \mid \theta)}_{\text{part of } \rho_c}$$

- If we observed Z, the log likelihood $\ln[p(X,Z \mid q)]$ would be trivial to maximize with respect to q and r_i
- The problem, is that the values of Z are missing, since we made it up (that is Z is hidden)

EM Derivation

- Instead of maximizing $\ln[p(X,Z|q)]$ idea behind *EM* is to maximize some function of $\ln[p(X,Z|q)]$, usually its expected value (conditioned on X)

$$E_{Z|X}\left[\ln p(X,Z|\theta)\right]$$

  - common trick: integrate out unknown variables
  - the expectation is with respect to the missing data Z
  - that is with respect to density $p(Z|X,q)$

- however q is our ultimate goal, we don't know q !

EM Algorithm

- *EM* solution is to iterate

  1. start with initial parameters $\boldsymbol{q}^{(0)}$

  iterate the following 2 step until convergence

  E. compute the expectation of log likelihood with respect to current estimate $\boldsymbol{q}^{(t)}$ and $\boldsymbol{X}$.
  Let's call it $\boldsymbol{Q}(\boldsymbol{q} \mid \boldsymbol{q}^{(t)})$

$$\boldsymbol{Q}\left(\theta \mid \theta^{(t)}\right) = \boldsymbol{E}_{Z|X}\left[\boldsymbol{ln}\, \boldsymbol{p}(\boldsymbol{X}, \boldsymbol{Z} \mid \theta) \mid \boldsymbol{X}, \theta^{(t)}\right]$$

  M. maximize $\boldsymbol{Q}(\boldsymbol{q} \mid \boldsymbol{q}^{(t)})$

$$\theta^{(t+1)} = \arg\max_{\theta} \boldsymbol{Q}\left(\theta \mid \theta^{(t)}\right)$$

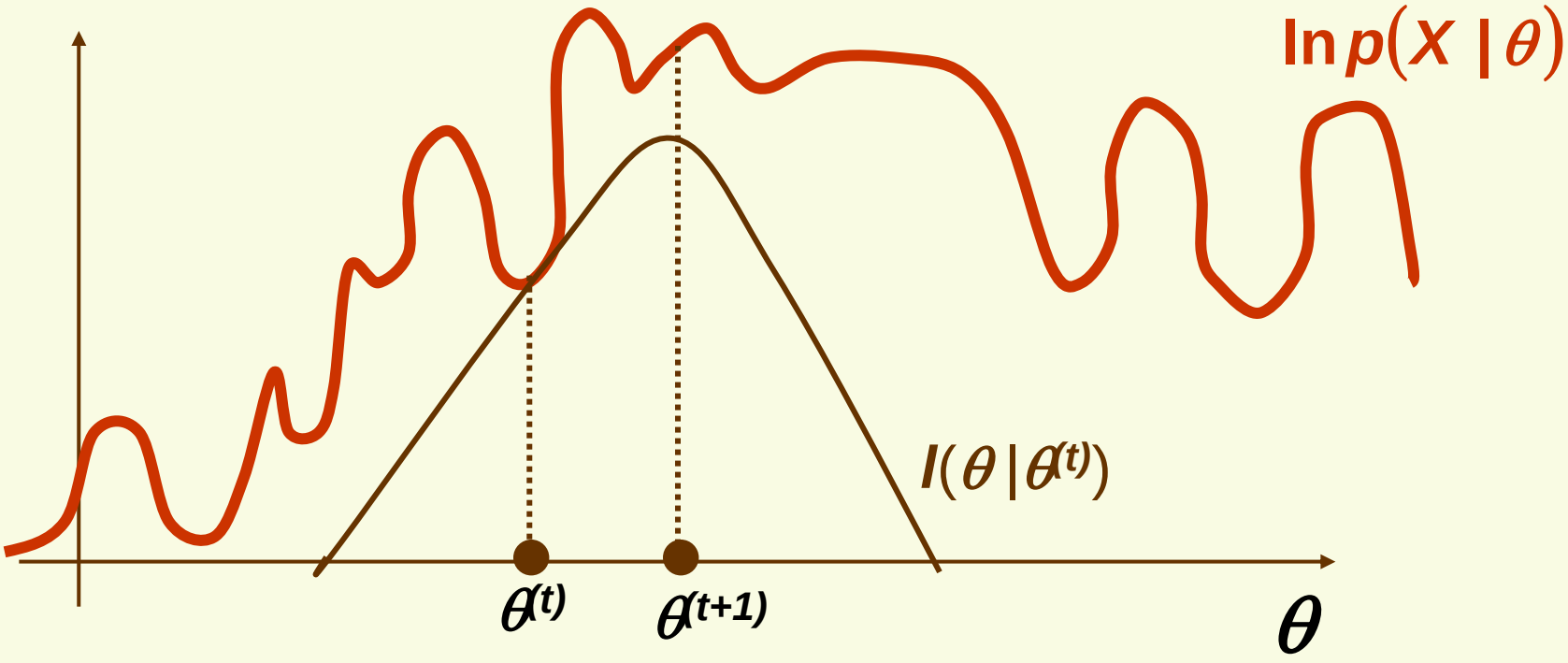
EM Algorithm

- It can be proven that EM algorithm converges to the local maximum of the log-likelihood

$$\ln p(X \mid \theta)$$

- Why is it better than gradient ascent?
  - Convergence of EM is usually ***significantly*** faster, in the beginning, very large steps are made (that is likelihood function increases rapidly), as opposed to gradient ascent which usually takes tiny steps
  - gradient descent is not guaranteed to converge
  - recall the difficulties of choosing the appropriate learning rate

EM: Lower Bound Maximization



- It can be shown that at time step ***t*** EM algorithm
  - constructs function $l(\theta \mid \theta^{(t)})$ which is bounded above by $\ln \mathbf{p}(X \mid \theta)$ and touches $l(\theta \mid \theta^{(t)})$ at $\theta = \theta^{(t)}$
  - finds $\theta^{(t+1)}$ that maximizes $l(\theta \mid \theta^{(t)})$
- Therefore, log likelihood $\ln \mathbf{p}(X \mid \theta)$ can only go up

EM for Mixture of Gaussians: E step

- Let's come back to our example $p(x \mid \theta) = \sum_{j=1}^{m} p\left(x \mid c_j, \theta_j\right)\rho_j$

$$p\left(x \mid c_j, \theta_j\right) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{\left(x-\mu_j\right)^2}{2\sigma^2}\right)$$

  - need to estimate $\theta = \{\mu_1, \ldots, \mu_m\}$ and $\rho_1, \ldots, \rho_m$

- For $1 \le i \le n$, $1 \le k \le m$, define $z_i^{(k)}$

$$z_i^{(k)} = \begin{cases} 1 & \text{if sample } i \text{ was generated by component } k \\ 0 & \text{otherwise} \end{cases}$$

- as before, $z_i = \{z_i^{(1)}, \ldots, z_i^{(m)}\}$, and $Z = \{z_1, \ldots, z_n\}$

- need log-likelihood of observed X and hidden Z

$$\ln p(X, Z \mid \theta) = \ln \prod_{i=1}^{n} p(x_i, z_i \mid \theta) = \sum_{i=1}^{n} \ln p(x_i \mid z_i, \theta) P(z_i)$$

EM for Mixture of Gaussians: E step

- We need log-likelihood of observed $\boldsymbol{X}$ and hidden $\boldsymbol{Z}$

$$\ln p(X, Z \mid \theta) = \ln \prod_{i=1}^{n} p(x_i, z_i \mid \theta) = \sum_{i=1}^{n} \ln p(x_i \mid z_i, \theta) P(z_i)$$

- First let's rewrite $p(x_i \mid z_i, \theta) P(z_i)$

$$p(x_i \mid z_i, \theta) P(z_i) = \begin{cases} p(x_i \mid z_i^{(1)} = 1, \theta) P(z_i^{(1)} = 1) & \text{if } z_i^{(1)} = 1 \\ \vdots & \vdots \\ p(x_i \mid z_i^{(m)} = 1, \theta) P(z_i^{(m)} = 1) & \text{if } z_i^{(m)} = 1 \end{cases}$$

$$= \prod_{k=1}^{m} \left[p(x_i \mid z_i^{(k)} = 1, \theta) P(z_i^{(k)} = 1) \right]^{z_i^{(k)}}$$

$$= \prod_{k=1}^{m} \left[\frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x_i - \mu_k)^2}{2\sigma^2} \right) P(z_i^{(k)} = 1) \right]^{z_i^{(k)}}$$

EM for Mixture of Gaussians: E step

- log-likelihood of observed $\boldsymbol{X}$ and hidden $\boldsymbol{Z}$ is

$$\ln p(X, Z \mid \theta) = \sum_{i=1}^{n} \ln p(x_i \mid z_i, \theta) P(z_i)$$

$$= \sum_{i=1}^{n} \ln \prod_{k=1}^{m} \left[\frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x_i - \mu_k)^2}{2\sigma^2} \right) P\left(z_i^{(k)} = 1\right) \right]^{z_i^{(k)}}$$

$$= \sum_{i=1}^{n} \sum_{k=1}^{m} \ln \left[\frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x_i - \mu_k)^2}{2\sigma^2} \right) P\left(z_i^{(k)} = 1\right) \right]^{z_i^{(k)}}$$

$$= \sum_{i=1}^{n} \sum_{k=1}^{m} z_i^{(k)} \left[\ln \frac{1}{\sigma\sqrt{2\pi}} - \frac{(x_i - \mu_k)^2}{2\sigma^2} + \ln \underbrace{P\left(z_i^{(k)} = 1\right)}_{P(\text{sample } x_i \text{ from class } k) = P(c_k) = \rho_k} \right]$$

$$= \sum_{i=1}^{n} \sum_{k=1}^{m} z_i^{(k)} \left[\ln \frac{1}{\sigma\sqrt{2\pi}} - \frac{(x_i - \mu_k)^2}{2\sigma^2} + \ln \rho_k \right]$$

EM for Mixture of Gaussians: E step

- log-likelihood of observed ***X*** and hidden ***Z*** is

$$\ln p(X,Z\mid\theta)=\sum_{i=1}^{n}\sum_{k=1}^{m}z_i^{(k)}\left[\ln\frac{1}{\sigma\sqrt{2\pi}}-\frac{(x_i-\mu_k)^2}{2\sigma^2}+\ln\rho_k\right]$$

- For the E step, we must compute

$$Q(\theta\mid\theta^{(t)})=Q(\theta\mid\mu_1^{(t)},\ldots,\mu_m^{(t)},\rho_1^{(t)},\ldots,\rho_m^{(t)})=E_Z\left[\ln p(X,Z\mid\theta)\mid X,\theta^{(t)}\right]$$

$$=E_Z\left(\sum_{i=1}^{n}\sum_{k=1}^{m}z_i^{(k)}\left[\ln\frac{1}{\sigma\sqrt{2\pi}}-\frac{(x_i-\mu_k)^2}{2\sigma^2}+\ln\rho_k^{(t)}\right]\right)$$

$$E_X\left[\sum_i a_i x_i+b\right]=\sum_i a_i E_X[x_i]+b$$

$$=\sum_{i=1}^{n}\sum_{k=1}^{m}E_Z\left[z_i^{(k)}\right]\left(\ln\frac{1}{\sigma\sqrt{2\pi}}-\frac{(x_i-\mu_k)^2}{2\sigma^2}+\ln\rho_k\right)$$

EM for Mixture of Gaussians: E step

$$Q\left(\theta \mid \theta^{(t)}\right) = \sum_{i=1}^{n}\sum_{k=1}^{m} E_Z\left[z_i^{(k)}\right]\left(\ln\frac{1}{\sigma\sqrt{2\pi}} - \frac{\left(x_i-\mu_k\right)^2}{2\sigma^2} + \ln\rho_k\right)$$

- need to compute $E_Z[z_i^{(k)}]$ in the above expression

$$E_Z\left[z_i^{(k)}\right] = 0 * P\left(z_i^{(k)} = 0 \mid \theta^{(t)}, x_i\right) + 1 * P\left(z_i^{(k)} = 1 \mid \theta^{(t)}, x_i\right)$$

$$= P\left(z_i^{(k)} = 1 \mid \theta^{(t)}, x_i\right) = \frac{p\left(x_i \mid \theta^{(t)}, z_i^{(k)} = 1\right) P\left(z_i^{(k)} = 1 \mid \theta^{(t)}\right)}{p\left(x_i \mid \theta^{(t)}\right)}$$

$$= \frac{\rho_k^{(t)} \exp\left(-\frac{1}{2}\left(x_i-\mu_k^{(t)}\right)^2\right)}{\sum_{j=1}^{m} P\left(x_i \mid \theta^{(t)}, z_i^{(j)} = 1\right) P\left(z_i^{(j)} = 1 \mid \theta^{(t)}\right)} = \frac{\rho_k^{(t)} \exp\left(-\frac{1}{2\sigma^2}\left(x_i-\mu_k^{(t)}\right)^2\right)}{\sum_{j=1}^{m} \rho_j^{(t)} \exp\left(-\frac{1}{2\sigma^2}\left(x_i-\mu_j^{(t)}\right)^2\right)}$$

- Done with the ***E*** step
  - for implementation, need to compute $E_Z[z_i^{(k)}]$'s don't need ***Q***

EM for Mixture of Gaussians: M step

$$Q\left(\theta \mid \theta^{(t)}\right) = \sum_{i=1}^{n}\sum_{k=1}^{m} E_z\left[z_i^{(k)}\right]\left(\ln\frac{1}{\sigma\sqrt{2\pi}} - \frac{(x_i-\mu_k)^2}{2\sigma^2} + \ln\rho_k\right)$$

- Need to maximize ***Q*** with respect to all parameters
- First differentiate with respect to m_k

$$\frac{\partial}{\partial\mu_k}Q\left(\theta \mid \theta^{(t)}\right) = \sum_{i=1}^{n} E_z\left[z_i^{(k)}\right]\frac{(x_i-\mu_k)}{\sigma^2} = 0$$

$$\Rightarrow new\ \mu_k = \mu_k^{(t+1)} = \boxed{\frac{1}{n}\sum_{i=1}^{n} E_z\left[z_i^{(k)}\right]x_i}$$

the mean for class **k** is weighted average of all samples, and this weight is proportional to the current estimate of probability that the sample belongs to class **k**

EM for Mixture of Gaussians: M step

$$Q\left(\theta \mid \theta^{(t)}\right) = \sum_{i=1}^{n}\sum_{k=1}^{m} E_Z\left[z_i^{(k)}\right]\left(\ln\frac{1}{\sigma\sqrt{2\pi}} - \frac{\left(x_i-\mu_k\right)^2}{2\sigma^2} + \ln\rho_k\right)$$

- For ρ_k use Lagrange multipliers to preserve constraint $\sum_{j=1}^{m}\rho_j = 1$
- Need to differentiate $F(\lambda,\rho) = Q\left(\theta \mid \theta^{(t)}\right) - \lambda\left(\sum_{j=1}^{m}\rho_j - 1\right)$

$$\frac{\partial}{\partial\rho_k}F(\lambda,\rho) = \sum_{i=1}^{n}\frac{1}{\rho_k}E_Z\left[z_i^{(k)}\right] - \lambda = 0 \quad \Rightarrow \sum_{i=1}^{n}E_Z\left[z_i^{(k)}\right] - \lambda\rho_k = 0$$

- Summing up over all components: $\sum_{k=1}^{m}\sum_{i=1}^{n}E_Z\left[z_i^{(k)}\right] = \sum_{k=1}^{m}\lambda\rho_k$
- Since $\sum_{k=1}^{m}\sum_{i=1}^{n}E_Z\left[z_i^{(k)}\right] = n$ and $\sum_{k=1}^{m}\rho_k = 1$ we get $\lambda = n$

$$\boxed{\rho_k^{(t+1)} = \frac{1}{n}\sum_{i=1}^{n}E_Z\left[z_i^{(k)}\right]}$$

EM Algorithm

This algorithm applies to univariate Gaussian with known variances

1. Randomly initialize $\mu_1, \ldots, \mu_m$, $\rho_1, \ldots, \rho_m$,
   - with constraint $\Sigma \rho_i = 1$

iterate until no change in $\mu_1, \ldots, \mu_m$, $\rho_1, \ldots, \rho_m$

E. for all ***i***, ***k***, compute

$$E_z\left[z_i^{(k)}\right] = \frac{\rho_k \exp\left(-\frac{1}{2\sigma^2}\left(x_i - \mu_k\right)^2\right)}{\sum_{j=1}^{m} \rho_j \exp\left(-\frac{1}{2\sigma^2}\left(x_i - \mu_j\right)^2\right)}$$

M. for all ***k***, do parameter update

$$\mu_k = \frac{1}{n}\sum_{i=1}^{n} E_z\left[z_i^{(k)}\right] x_i \qquad \rho_k = \frac{1}{n}\sum_{i=1}^{n} E_z\left[z_i^{(k)}\right]$$

EM Algorithm

- For more general case of multivariate Gaussians with unknown means and variances
  - ***E*** step: $E_z\left[z_i^{(k)}\right] = \dfrac{\rho_k\, p(x \mid \mu_k, \Sigma_k)}{\sum_{j=1}^{m} \rho_j p(x \mid \mu_j, \Sigma_j)}$

    where $p(x \mid \mu_k, \Sigma_k) = \dfrac{1}{(2\pi)^{d/2}\left|\Sigma_k^{-1}\right|^{1/2}} exp\left[-\dfrac{1}{2}(x-\mu_k)^t \Sigma_k^{-1}(x-\mu_k)\right]$
  - ***M*** step:

$$\rho_k = \frac{1}{n}\sum_{i=1}^{n} E_z\left[z_i^{(k)}\right]$$

$$\mu_k = \frac{\sum_{i=1}^{n} E_z\left[z_i^{(k)}\right] x_i}{\sum_{i=1}^{n} E_z\left[z_i^{(k)}\right]}$$

$$\Sigma_k = \frac{\sum_{i=1}^{n} E_z\left[z_i^{(k)}\right](x_i-\mu_k)(x_i-\mu_k)^T}{\sum_{i=1}^{n} E_z\left[z_i^{(k)}\right]}$$

EM Algorithm and K-means

- ***k***-means can be derived from EM algorithm
- Setting mixing parameters equal for all classes,

$$E_z\left[z_i^{(k)}\right] = \frac{\rho_k \exp\left(-\frac{1}{2\sigma^2}(x_i - \mu_k)^2\right)}{\sum_{j=1}^{m} \rho_j \exp\left(-\frac{1}{2\sigma^2}(x_i - \mu_j)^2\right)} = \frac{\exp\left(-\frac{1}{2\sigma^2}(x_i - \mu_k)^2\right)}{\sum_{j=1}^{m} \exp\left(-\frac{1}{2\sigma^2}(x_i - \mu_j)^2\right)}$$

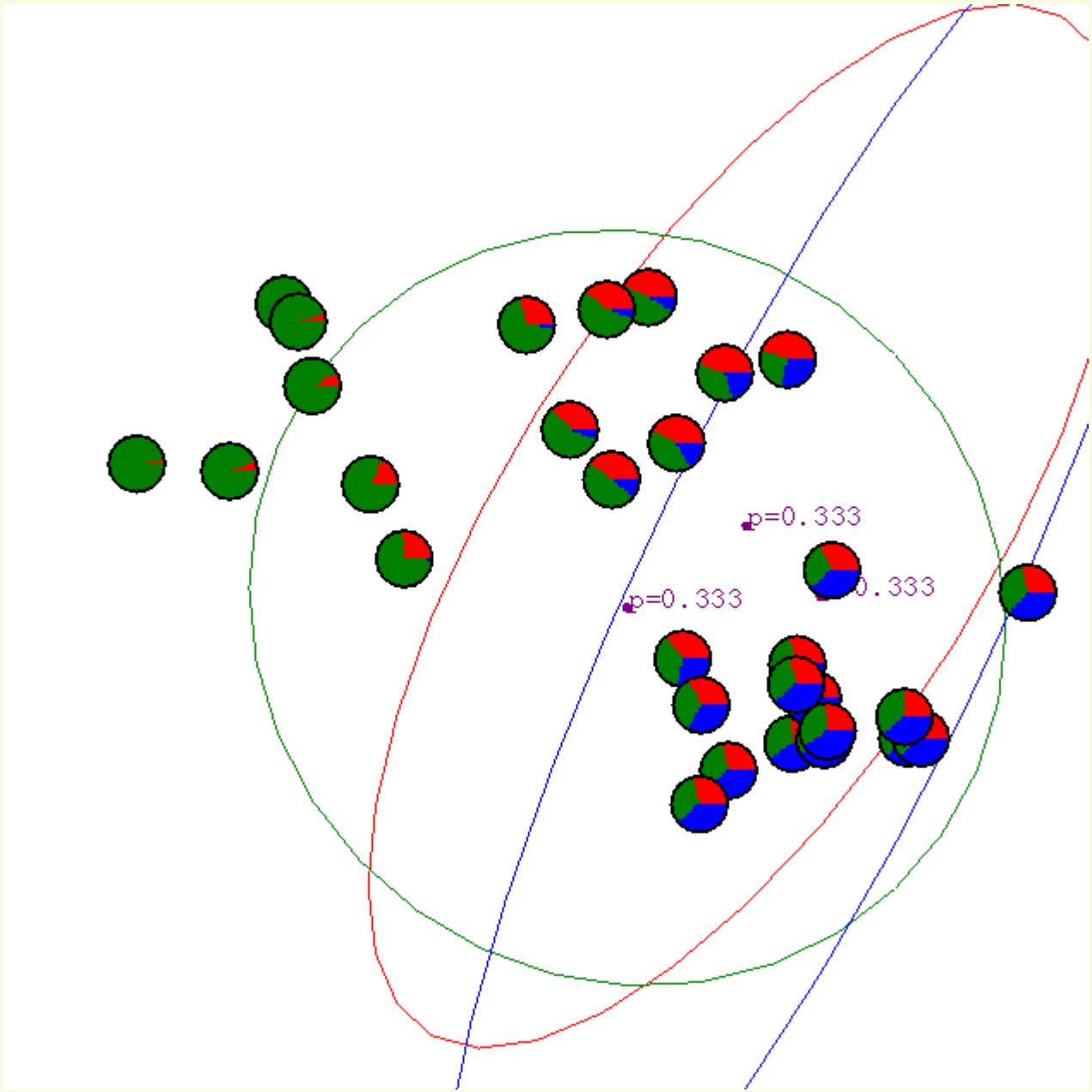
- If we let $\sigma \to 0$, then

$$E_z\left[z_i^{(k)}\right] = \begin{cases} 1 & \text{if } \forall j,\ \| x_i - \mu_k \| > \| x_i - \mu_j \| \\ 0 & \text{otherwise} \end{cases}$$

  - *E* step, for each current mean, find all points closest to it and form new clusters
  - *M* step, compute new means inside current clusters

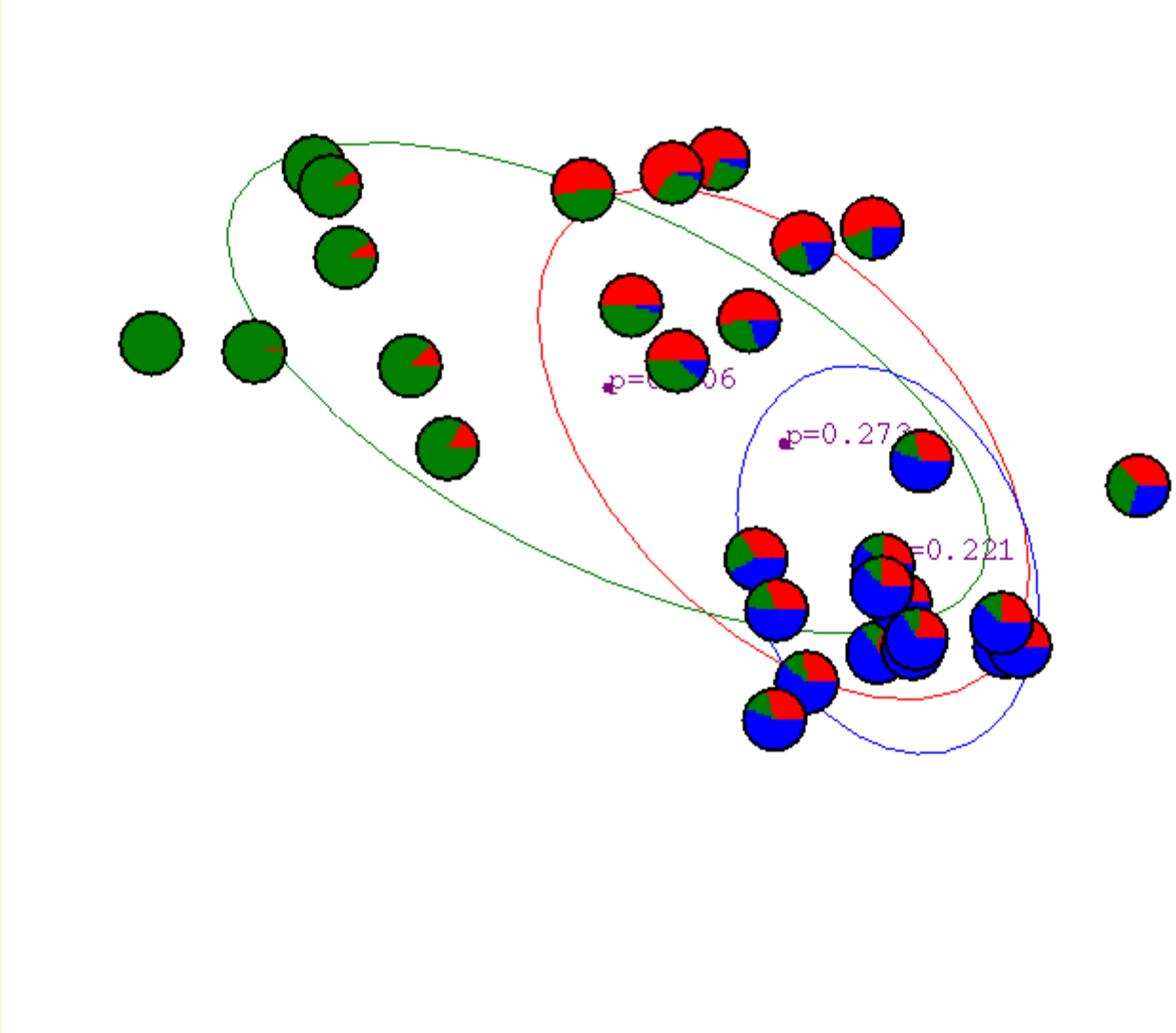
$$\mu_k = \frac{1}{n} \sum_{i=1}^{n} E_z\left[z_i^{(k)}\right] x_i$$

EM Gaussian Mixture Example



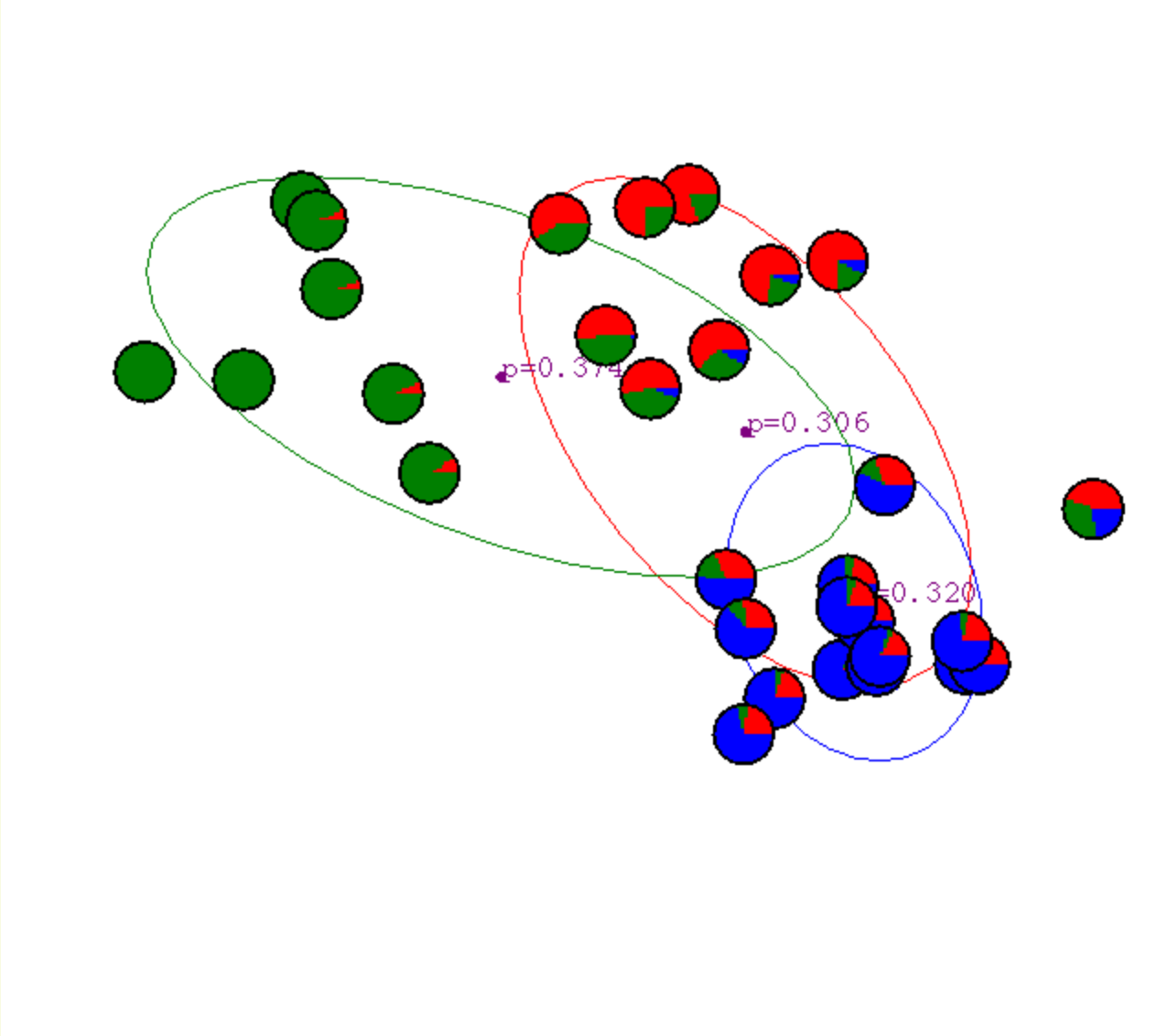
EM Gaussian Mixture Example

After first iteration



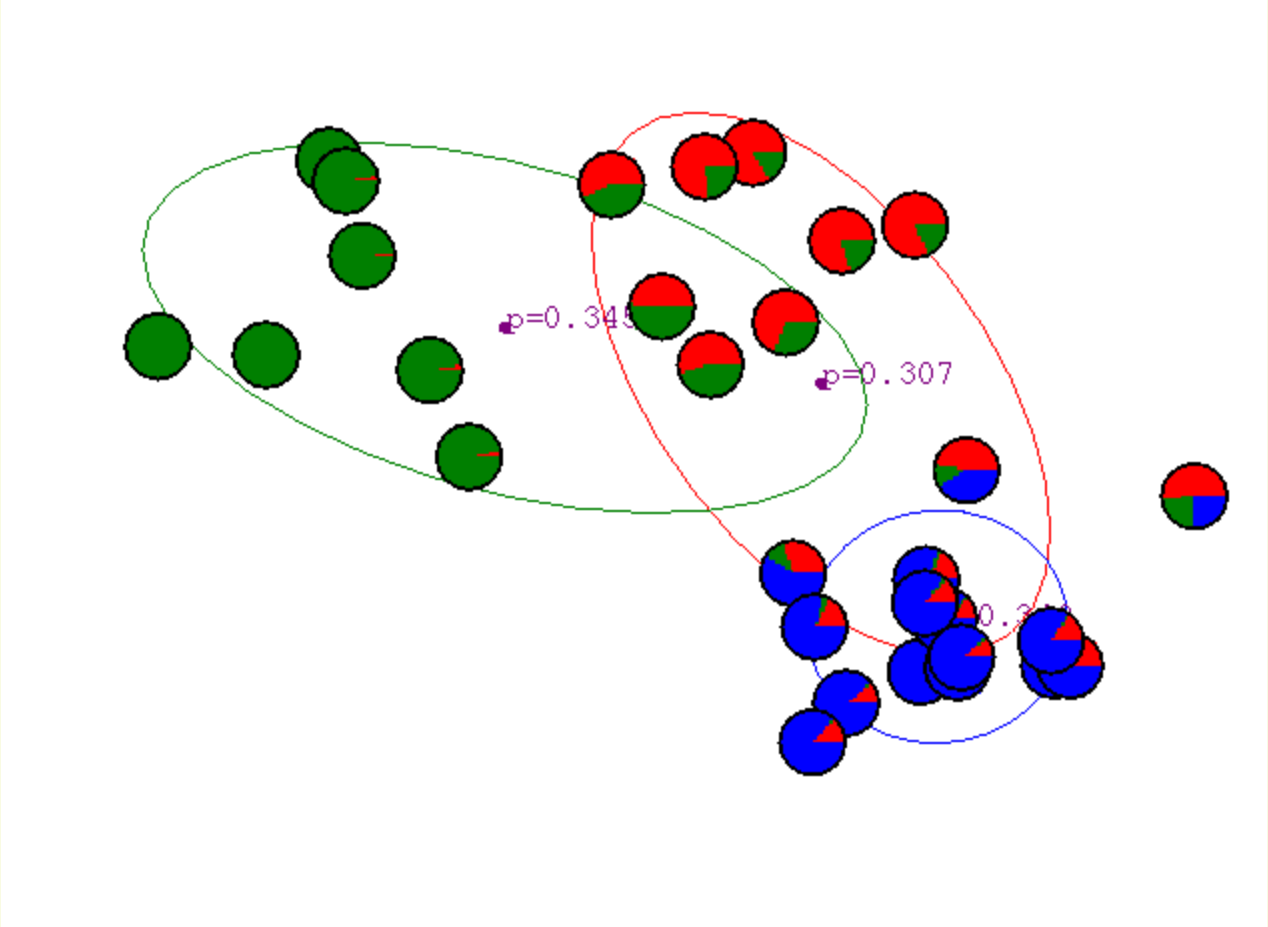
EM Gaussian Mixture Example

After second iteration



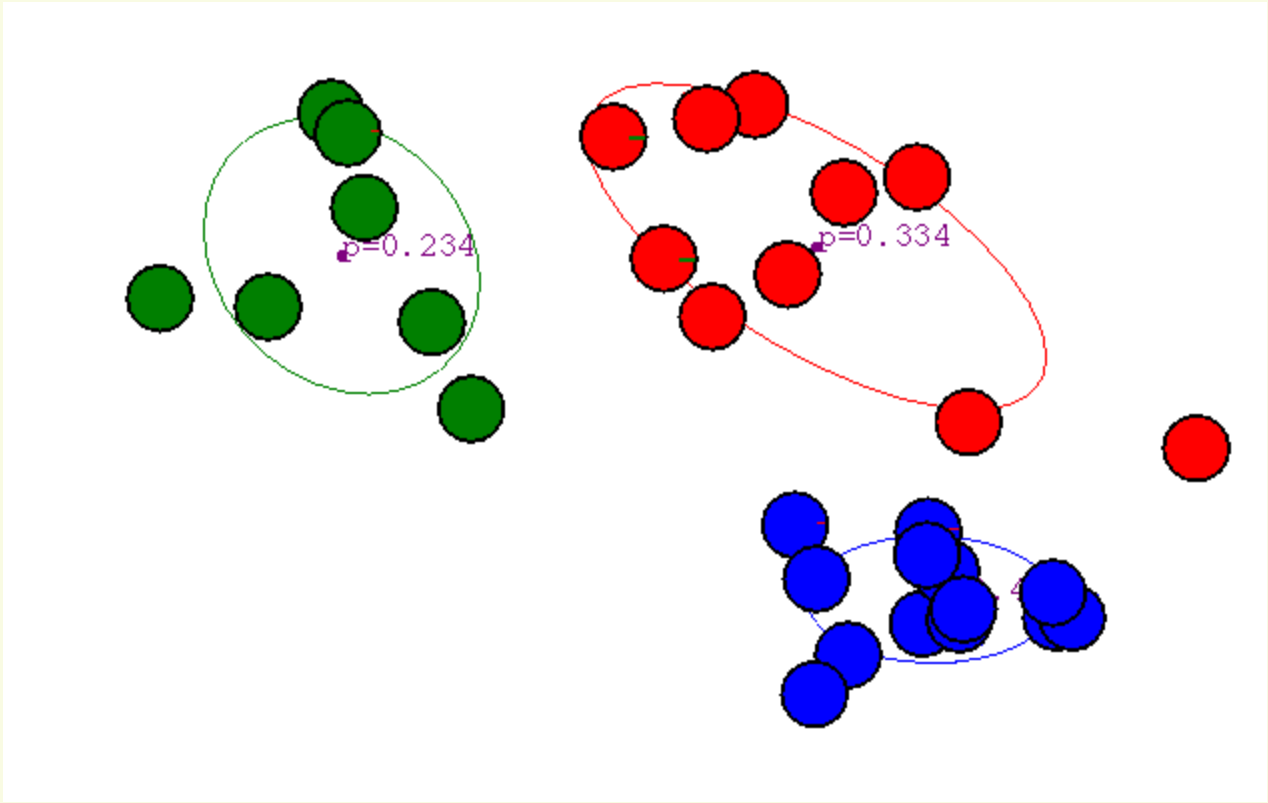
EM Gaussian Mixture Example

After third iteration



EM Gaussian Mixture Example

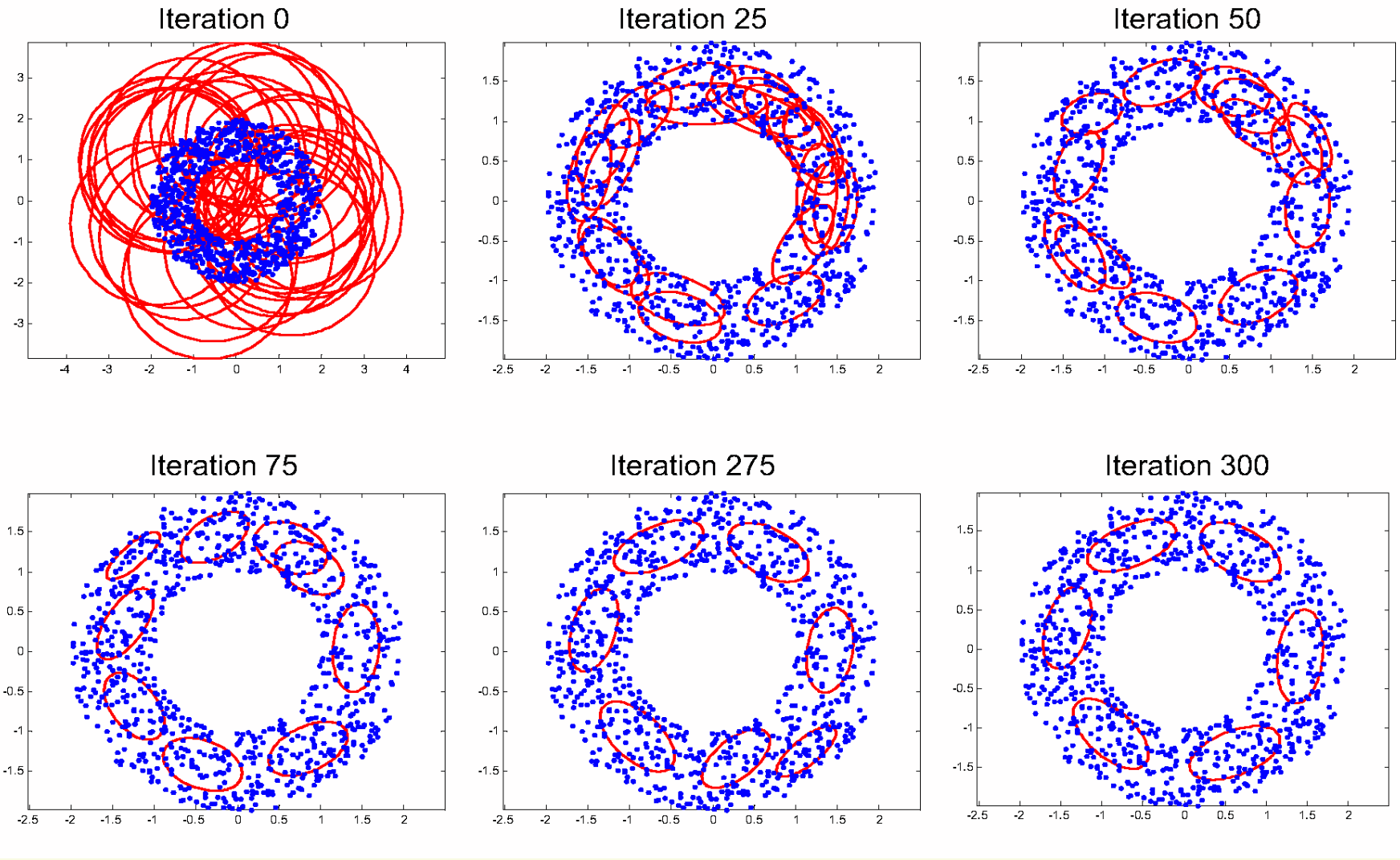
After 20th iteration



EM Example

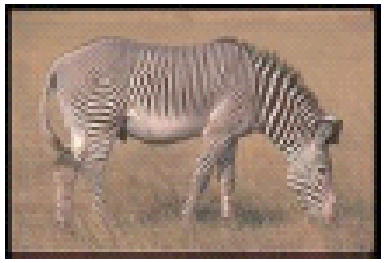
- Example from R. Gutierrez-Osuna
- Training set of 900 examples forming an annulus
- Mixture model with ***m*** = 30 Gaussian components of unknown mean and variance is used
- Training:
  - Initialization:
    - means to 30 random examples
    - covaraince matrices initialized to be diagonal, with large variances on the diagonal, compared to training data variance
  - During EM training, components with small mixing coefficients were trimmed
    - This is a trick to get in a more compact model, with fewer than 30 Gaussian components

EM Example



from R. Gutierrez-Osuna

EM Texture Segmentation Example



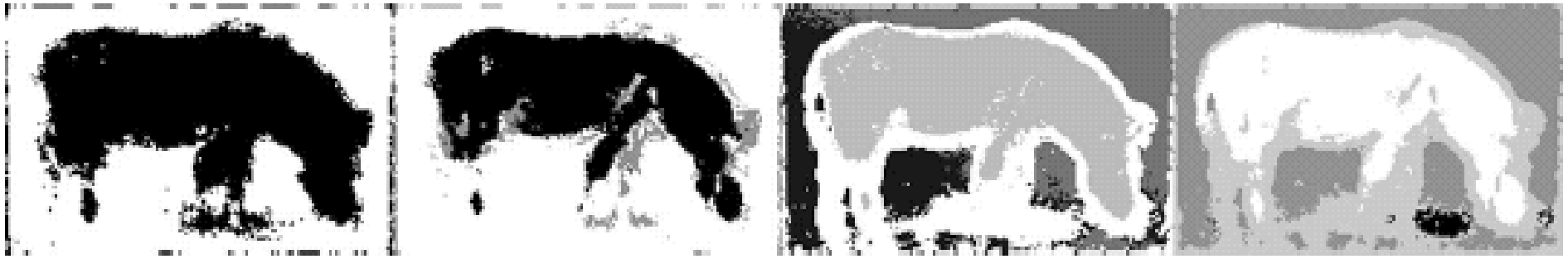


Figure from "Color and Texture Based Image Segmentation Using EM and Its Application to Content Based Image Retrieval",S.J. Belongie et al., ICCV 1998

EM Motion Segmentation Example

Three frames from the MPEG "flower garden" sequence

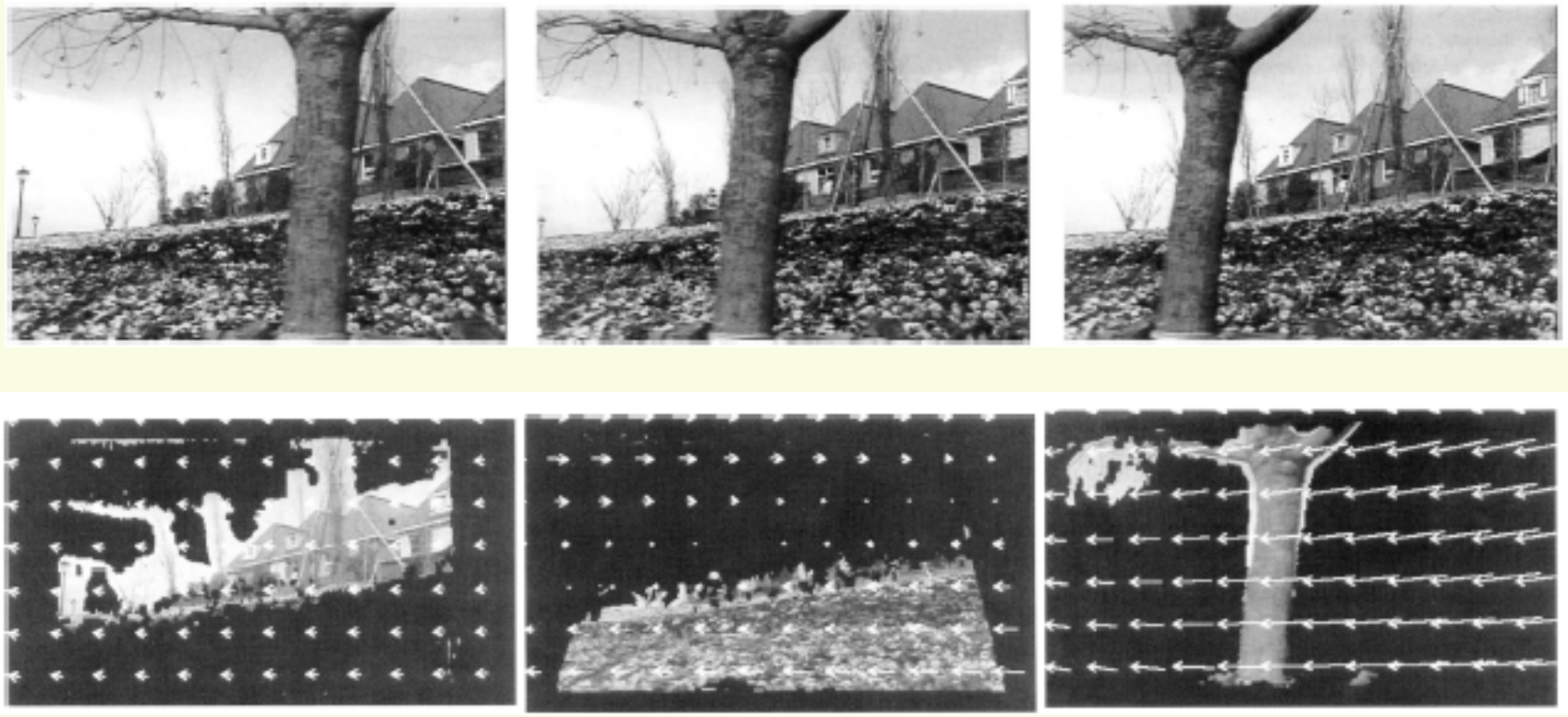


Figure from "Representing Images with layers,", by J. Wang and E.H. Adelson, IEEE Transactions on Image Processing, 1994, c 1994, IEEE

EM Algorithm Summary

- Advantages
  - Guaranteed to converge (to a local max)
  - If the assumed data distribution is correct, the algorithm works well
- Disadvantages
  - If assumed data distribution is wrong, results can be quite bad.
    - In particular, bad results if use incorrect number of classes (i.e. the number of mixture components)