Machine Learning

CSCI 5622 Fall 2020

Prof. Claire Monteleoni

Today

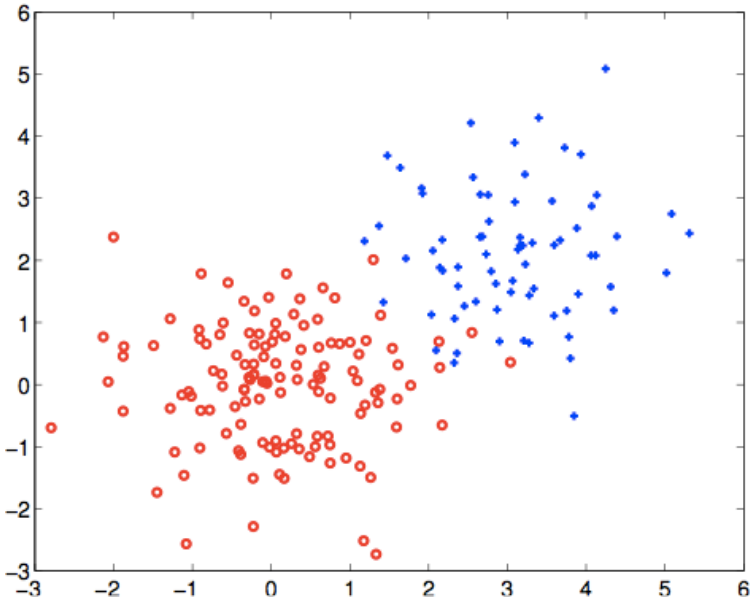
- Intro. to Generative Learning / probabilistic models
  - Maximum likelihood parameter estimation
  - Mixture models
  - Expectation maximization (EM)

with much credit to S. Dasgupta and T. Jaakkola

Training/test data generation

- We assume that the training (and test) examples are drawn as samples (generated) from some unknown distribution

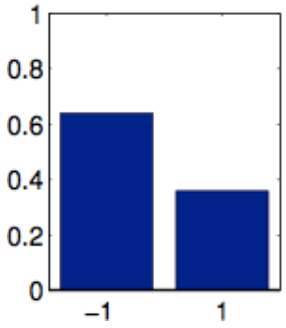
$$(\underline{x}, y) \sim P(\underline{x}, y) = P(\underline{x})P(y|\underline{x}) = P(\underline{x}|y)P(y)$$



- We can always think of these samples $(\underline{x}, y)$ as having been generated in two steps: first y, then $\underline{x}$ given y

$$y \sim P(y), \quad \underline{x} \sim P(\underline{x}|y)$$

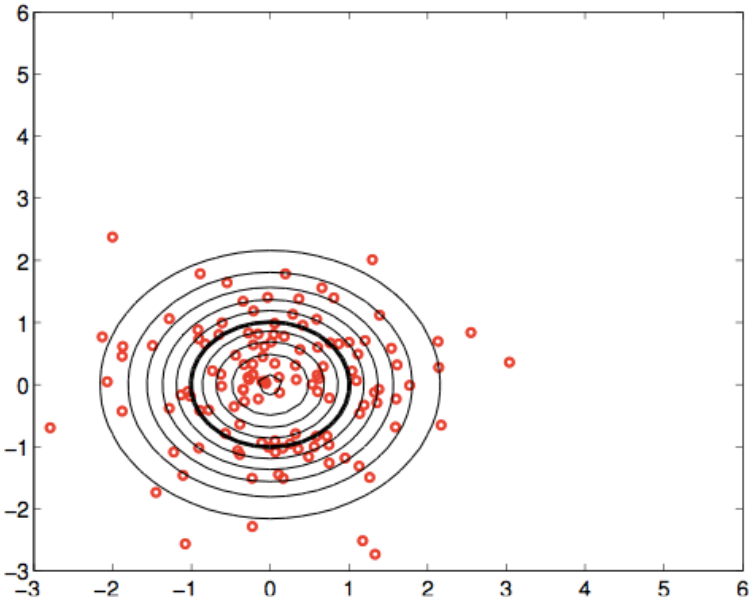
Generative modeling: estimation

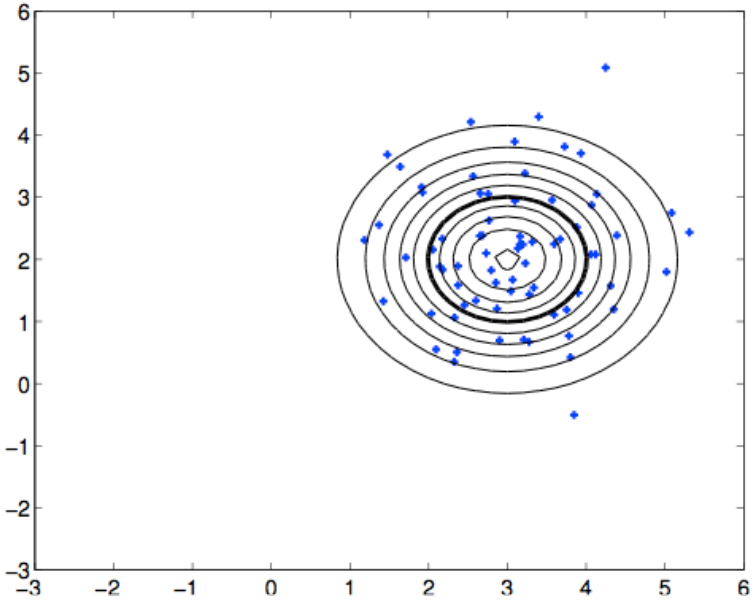


$P(y;\hat{\theta})$

$P(\underline{x}|y=-1;\hat{\theta})$

$P(\underline{x}|y=1;\hat{\theta})$





Maximum likelihood estimation

- Our parameterized Gaussian model is

$$P(\underline{x}, y; \theta) = P(\underline{x}|y; \theta)P(y; \theta) = N(\underline{x};\, \mu_y, \sigma_y^2 I)\, P(y; \theta)$$

- We find parameters $\theta = (\underline{\mu}_+, \underline{\mu}_-, \sigma_+^2, \sigma_-^2, q)$ that maximize the log-likelihood of the training data (examples and labels)

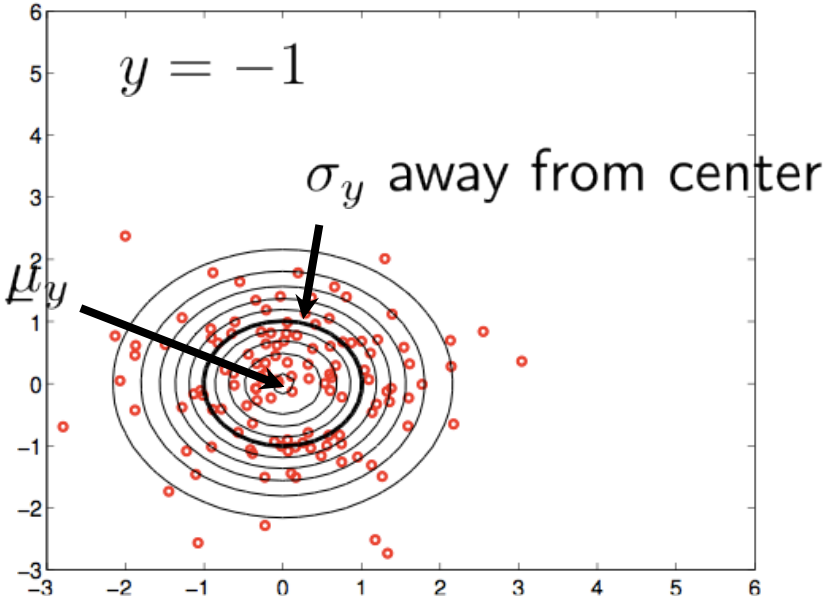
$$l(D; \theta) = \sum_{i=1}^{n} \log P(\underline{x}_i, y_i; \theta) = \sum_{i=1}^{n} \left[\log P(\underline{x}_i|y_i; \theta) + \log P(y_i; \theta) \right]$$

Generative modeling

- We can use simple spherical Gaussian models for the class-conditional distributions

$$P(\underline{x}|y;\theta) = N(\underline{x};\, \underline{\mu}_y, \sigma_y^2 I)$$

$$= \frac{1}{(2\pi\sigma_y^2)^{d/2}} \exp\left\{ -\frac{1}{2\sigma_y^2}\|\underline{x} - \underline{\mu}_y\|^2 \right\}$$



Maximum likelihood estimation

- Our parameterized Gaussian model is

$$P(\underline{x}, y; \theta) = P(\underline{x}|y; \theta)P(y; \theta) = N(\underline{x};\, \mu_y, \sigma_y^2 I)\, P(y; \theta)$$

- We find parameters $\theta = (\underline{\mu}_+, \underline{\mu}_-, \sigma_+^2, \sigma_-^2, q)$ that maximize the log-likelihood of the training data (examples and labels)

$$l(D; \theta) = \sum_{i=1}^{n} \log P(\underline{x}_i, y_i; \theta) = \sum_{i=1}^{n} \left[\log P(\underline{x}_i|y_i; \theta) + \log P(y_i; \theta) \right]$$

$$= \sum_{i=1}^{n} \left[-\frac{d}{2} \log(2\pi\sigma_{y_i}^2) - \frac{1}{2\sigma_{y_i}^2} \|\underline{x}_i - \mu_{y_i}\|^2 \right] +$$

Maximum likelihood estimation

- Our parameterized Gaussian model is

$$P(\underline{x}, y; \theta) = P(\underline{x}|y; \theta)P(y; \theta) = N(\underline{x};\, \mu_y, \sigma_y^2 I)\, P(y; \theta)$$

- We find parameters $\theta = (\underline{\mu}_+, \underline{\mu}_-, \sigma_+^2, \sigma_-^2, q)$ that maximize the log-likelihood of the training data (examples and labels)

$$l(D; \theta) = \sum_{i=1}^{n} \log P(\underline{x}_i, y_i; \theta) = \sum_{i=1}^{n} \left[\log P(\underline{x}_i|y_i; \theta) + \log P(y_i; \theta) \right]$$

$$= \sum_{i=1}^{n} \left[-\frac{d}{2} \log(2\pi\sigma_{y_i}^2) - \frac{1}{2\sigma_{y_i}^2} \|\underline{x}_i - \mu_{y_i}\|^2 \right] +$$

$$+ \sum_{i=1}^{n} \left[\delta(y_i, 1) \log q + \delta(y_i, -1) \log(1 - q) \right]$$

indicator function

Maximum likelihood estimation

$$l(D;\theta) = \sum_{i=1}^{n} \left[-\frac{d}{2}\log(2\pi\sigma_{y_i}^2) - \frac{1}{2\sigma_{y_i}^2}\|\underline{x}_i - \mu_{y_i}\|^2 \right]$$

$$+ \sum_{i=1}^{n} \left[\delta(y_i, 1)\log q + \delta(y_i, -1)\log(1-q) \right]$$

$$\frac{\partial}{\partial q} l(D;\theta) = \frac{\sum_{i=1}^{n} \delta(y_i, 1)}{q} - \frac{\sum_{i=1}^{n} \delta(y_i, -1)}{1-q} = 0$$

Maximum likelihood estimation

$$l(D;\theta) = \sum_{i=1}^{n} \left[-\frac{d}{2}\log(2\pi\sigma_{y_i}^2) - \frac{1}{2\sigma_{y_i}^2}\|\underline{x}_i - \mu_{y_i}\|^2 \right]$$

$$+ \sum_{i=1}^{n} \left[\delta(y_i, 1)\log q + \delta(y_i, -1)\log(1-q) \right]$$

$$\frac{\partial}{\partial q} l(D;\theta) = \frac{\sum_{i=1}^{n} \delta(y_i, 1)}{q} - \frac{\sum_{i=1}^{n} \delta(y_i, -1)}{1-q} = 0$$

$$\Rightarrow \hat{q} = \frac{1}{n}\sum_{i=1}^{n} \delta(y_i, 1)$$

fraction of points labeled +1

Maximum likelihood estimation

$$l(D;\theta) = \sum_{i=1}^{n} \left[-\frac{d}{2}\log(2\pi\sigma_{y_i}^2) - \frac{1}{2\sigma_{y_i}^2}\|\underline{x}_i - \mu_{y_i}\|^2 \right]$$

$$+ \sum_{i=1}^{n} \left[\delta(y_i, 1)\log q + \delta(y_i, -1)\log(1-q) \right]$$

$$\frac{\partial}{\partial \underline{\mu}_y} l(D;\theta) =$$

Maximum likelihood estimation

$$l(D;\theta) = \sum_{i=1}^{n} \left[-\frac{d}{2}\log(2\pi\sigma_{y_i}^2) - \frac{1}{2\sigma_{y_i}^2}\|\underline{x}_i - \mu_{y_i}\|^2 \right]$$

$$+ \sum_{i=1}^{n} \left[\delta(y_i, 1)\log q + \delta(y_i, -1)\log(1-q) \right]$$

$$\frac{\partial}{\partial \underline{\mu}_y} l(D;\theta) = \sum_{i=1}^{n} \delta(y, y_i)\frac{1}{\sigma_y^2}(\underline{x}_i - \mu_y) = 0$$

Maximum likelihood estimation

$$l(D;\theta) = \sum_{i=1}^{n} \left[-\frac{d}{2}\log(2\pi\sigma_{y_i}^2) - \frac{1}{2\sigma_{y_i}^2}\|\underline{x}_i - \mu_{y_i}\|^2 \right]$$

$$+ \sum_{i=1}^{n} \left[\delta(y_i, 1)\log q + \delta(y_i, -1)\log(1-q) \right]$$

$$\frac{\partial}{\partial \underline{\mu}_y} l(D;\theta) = \sum_{i=1}^{n} \delta(y, y_i)\frac{1}{\sigma_y^2}(\underline{x}_i - \mu_y) = 0$$

$$\Rightarrow \hat{\mu}_y = \frac{1}{\sum_{i=1}^{n} \delta(y, y_i)} \sum_{i=1}^{n} \delta(y, y_i)\, \underline{x}_i$$

average of points in class y

Maximum likelihood estimation

$$l(D;\theta) = \sum_{i=1}^{n} \left[-\frac{d}{2}\log(2\pi\sigma_{y_i}^2) - \frac{1}{2\sigma_{y_i}^2}\|\underline{x}_i - \mu_{y_i}\|^2 \right]$$

$$+ \sum_{i=1}^{n} \left[\delta(y_i, 1)\log q + \delta(y_i, -1)\log(1-q) \right]$$

$$\frac{\partial}{\partial \sigma_y^2} l(D;\theta) = \sum_{i=1}^{n} \delta(y, y_i) \left[-\frac{d}{2\sigma_y^2} + \frac{1}{2\sigma_y^4}\|\underline{x}_i - \hat{\mu}_y\|^2 \right] = 0$$

Maximum likelihood estimation

$$l(D;\theta) = \sum_{i=1}^{n} \left[-\frac{d}{2}\log(2\pi\sigma_{y_i}^2) - \frac{1}{2\sigma_{y_i}^2}\|\underline{x}_i - \mu_{y_i}\|^2 \right]$$

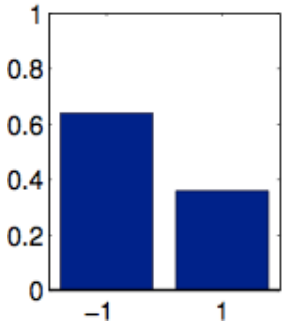
$$+ \sum_{i=1}^{n} \left[\delta(y_i, 1)\log q + \delta(y_i, -1)\log(1-q) \right]$$

$$\frac{\partial}{\partial \sigma_y^2} l(D;\theta) = \sum_{i=1}^{n} \delta(y, y_i) \left[-\frac{d}{2\sigma_y^2} + \frac{1}{2\sigma_y^4}\|\underline{x}_i - \hat{\mu}_y\|^2 \right] = 0$$

$$\Rightarrow \hat{\sigma}_y^2 = \frac{1}{d\sum_{i=1}^{n}\delta(y, y_i)} \sum_{i=1}^{n} \delta(y, y_i)\,\|\underline{x}_i - \hat{\mu}_y\|^2$$

average per dimension squared error in class y

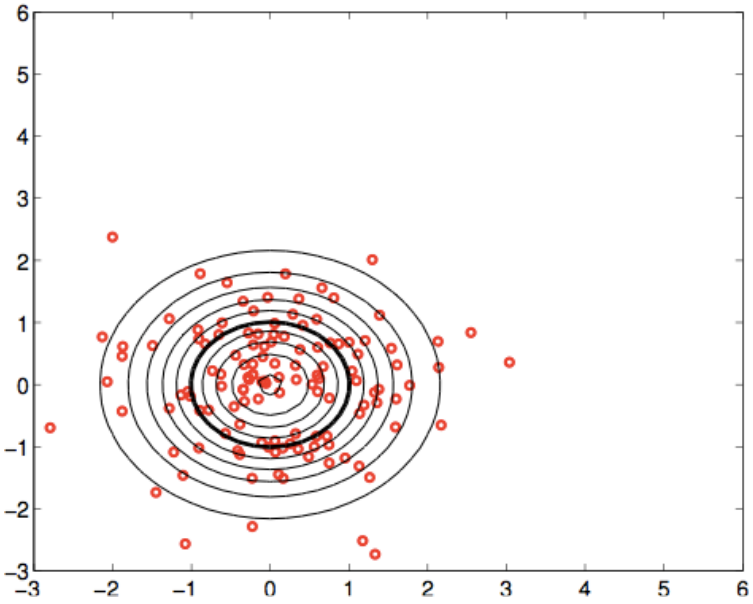
Generative modeling: classification

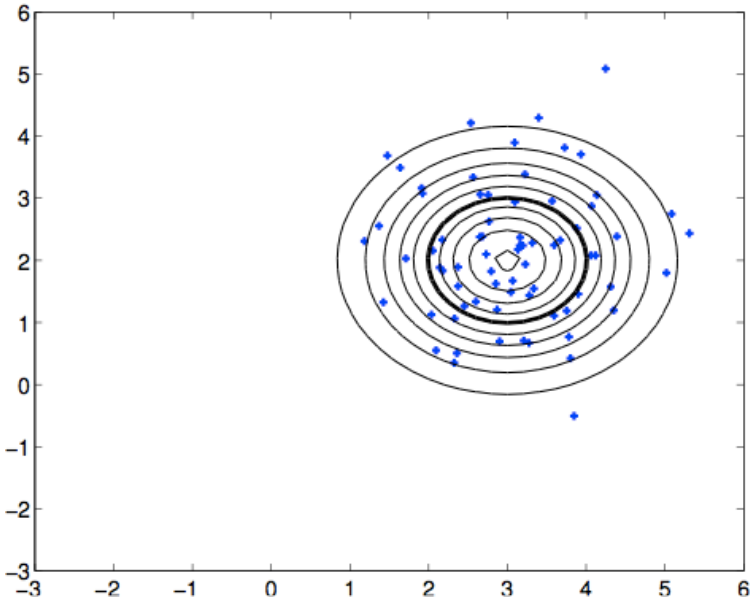


$P(y;\hat{\theta})$

$P(\underline{x}|y=-1;\hat{\theta})$

$P(\underline{x}|y=1;\hat{\theta})$





Decision boundary

- We predict the most likely label for $\underline{x}$ (cf. minimum probability of error classifier)

$$\hat{y} = \arg\max_{y} P(\underline{x}, y; \hat{\theta})$$

- The resulting decision boundary corresponds to all $\underline{x}$ such that

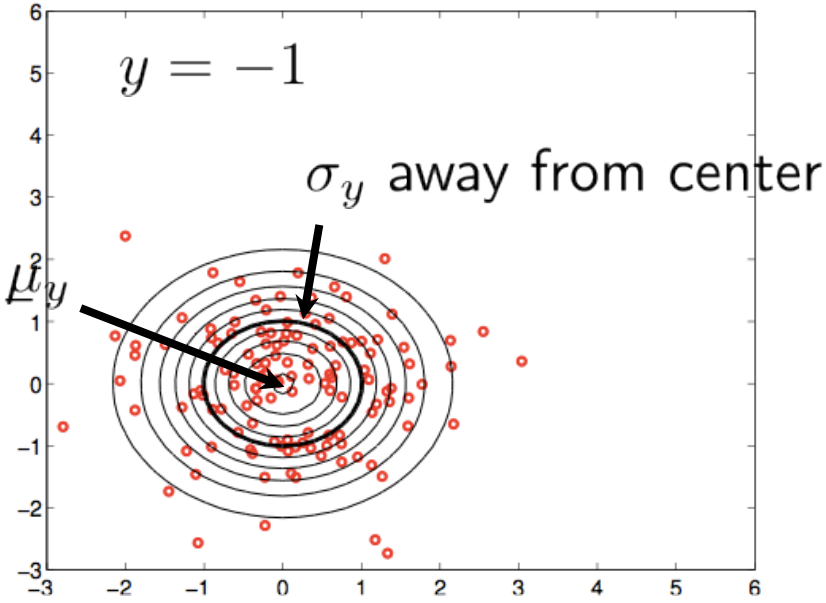
$$\log \frac{P(\underline{x}, y=1; \hat{\theta})}{P(\underline{x}, y=-1; \hat{\theta})} = \log \frac{P(y=1; \hat{\theta})}{P(y=-1; \hat{\theta})} + \log \frac{P(\underline{x}|y=1; \hat{\theta})}{P(\underline{x}|y=-1; \hat{\theta})} = 0$$

Generative modeling

- We can use simple spherical Gaussian models for the class-conditional distributions

$$P(\underline{x}|y;\theta) = N(\underline{x};\, \underline{\mu}_y, \sigma_y^2 I)$$

$$= \frac{1}{(2\pi\sigma_y^2)^{d/2}} \exp\left\{ -\frac{1}{2\sigma_y^2}\|\underline{x} - \underline{\mu}_y\|^2 \right\}$$



Decision boundary

- We predict the most likely label for $\underline{x}$ (cf. minimum probability of error classifier)

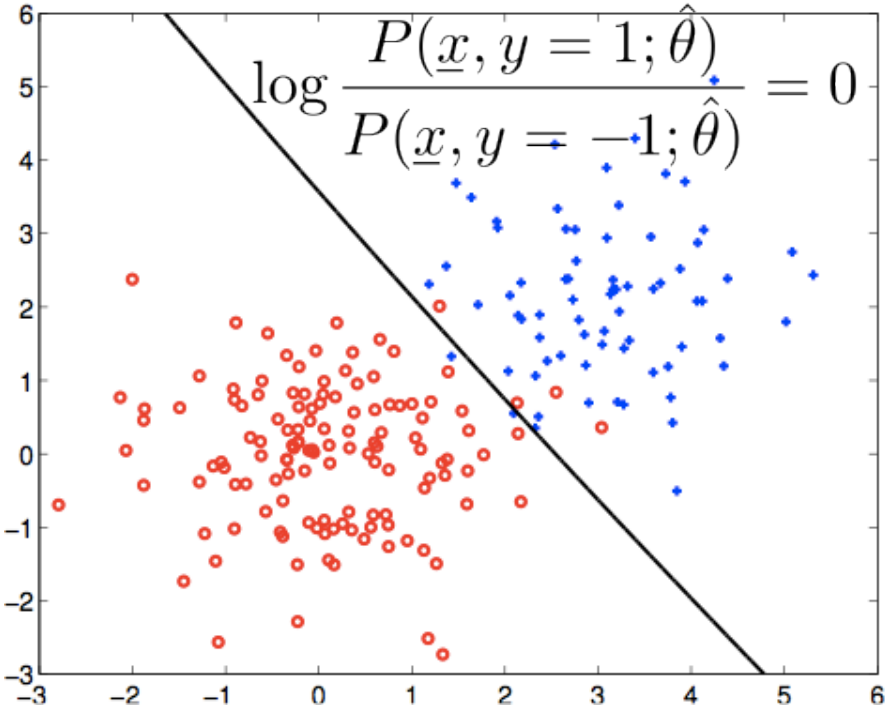
$$\hat{y} = \arg\max_y P(\underline{x}, y; \hat{\theta})$$

- The resulting decision boundary corresponds to all $\underline{x}$ such that

$$\begin{aligned}
\log \frac{P(\underline{x}, y=1; \hat{\theta})}{P(\underline{x}, y=-1; \hat{\theta})} &= \log \frac{P(y=1; \hat{\theta})}{P(y=-1; \hat{\theta})} + \log \frac{P(\underline{x}|y=1; \hat{\theta})}{P(\underline{x}|y=-1; \hat{\theta})} \\
&= \log \frac{\hat{q}}{1-\hat{q}} - \frac{d}{2} \log(\frac{\hat{\sigma}_{+1}^2}{\hat{\sigma}_{-1}^2}) \\
&\quad - \frac{1}{2\sigma_{+1}^2} \|\underline{x} - \underline{\mu}_{+1}\|^2 + \frac{1}{2\sigma_{-1}^2} \|\underline{x} - \underline{\mu}_{-1}\|^2 \\
&= 0
\end{aligned}$$

- This is linear in $\underline{x}$ if $\sigma_{+1}^2 = \sigma_{-1}^2$ (otherwise quadratic)

Decision boundary

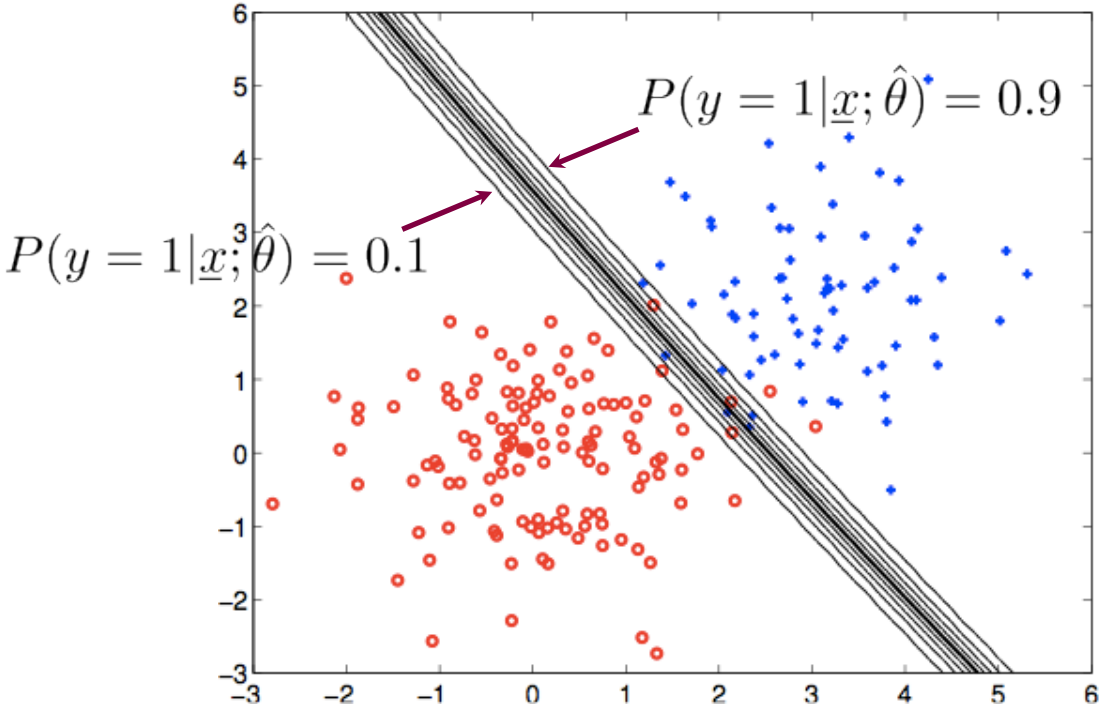


- This is very close to the optimal since the data were generated from two Gaussians with the same variance

Probabilistic predictions

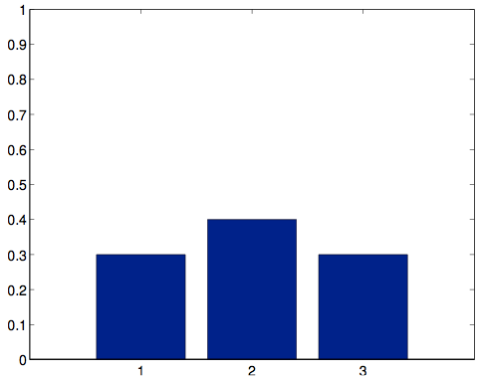
- The model also permits us to evaluate probabilities over the possible class labels such as

$$P(y = 1|\underline{x}; \hat{\theta}) = \frac{P(\underline{x}, y = 1; \hat{\theta})}{\sum_{y' \in \{-1,1\}} P(\underline{x}, y'; \hat{\theta})}$$



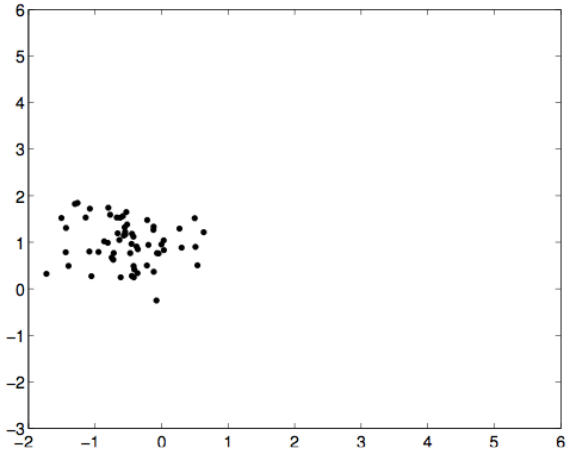
A mixture model

three types $z = 1, 2, 3$

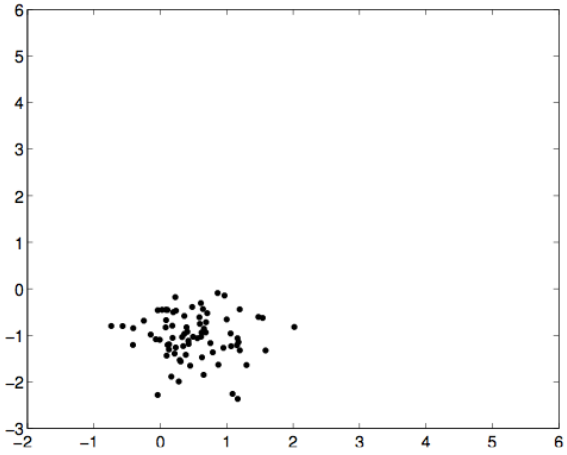


$P(z; \hat{\theta})$

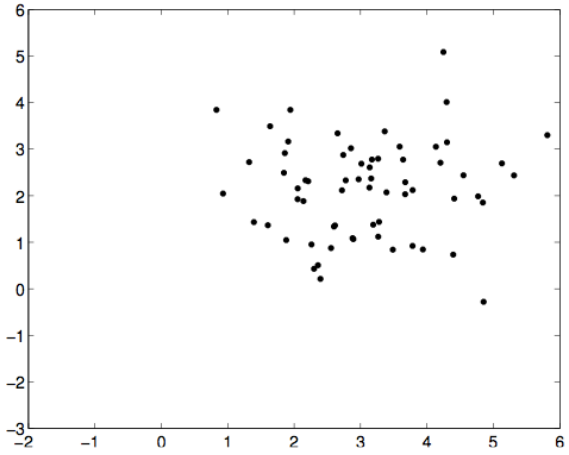
$P(\underline{x}|z = 1; \hat{\theta})$



$P(\underline{x}|z = 2; \hat{\theta})$

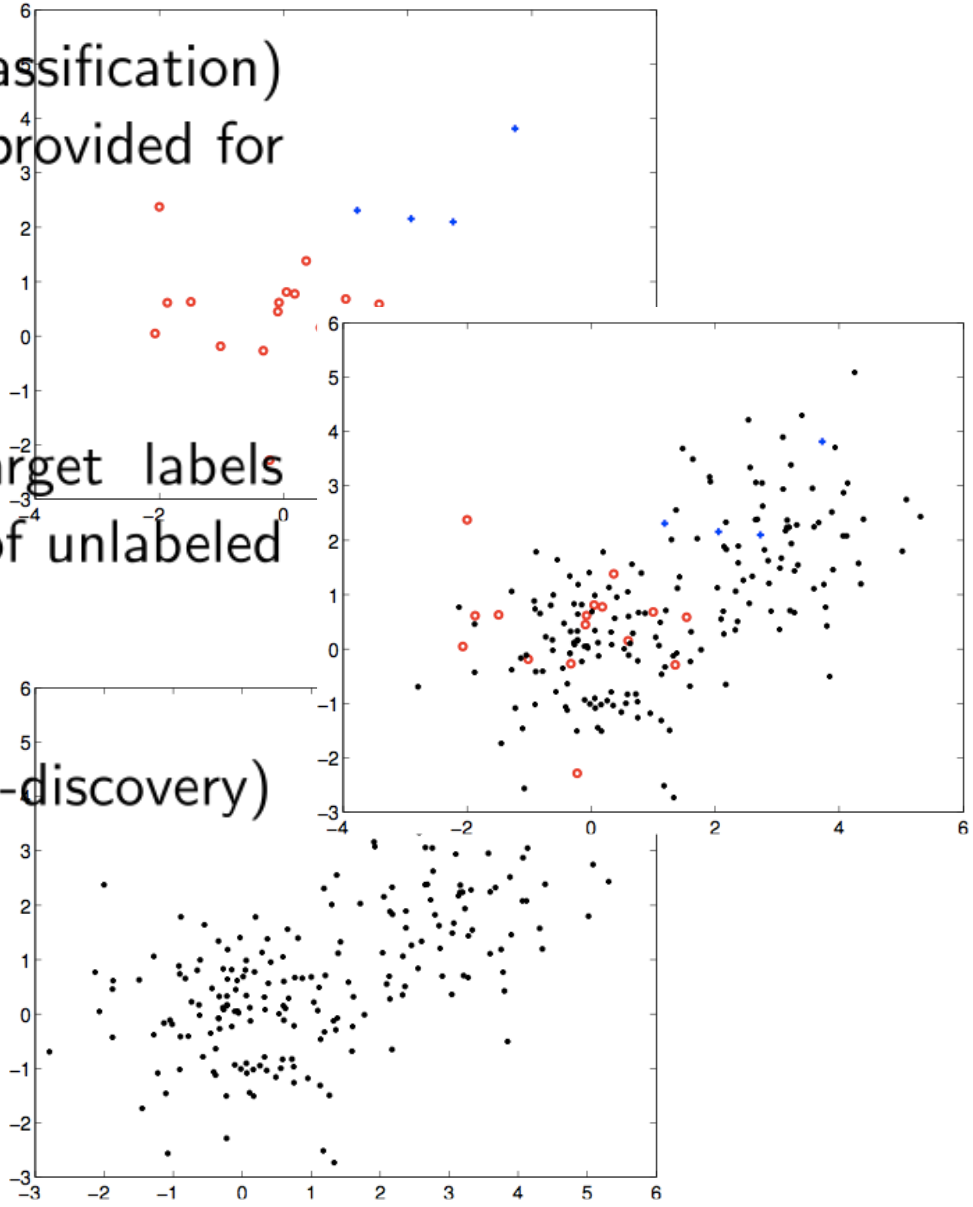


$P(\underline{x}|z = 3; \hat{\theta})$



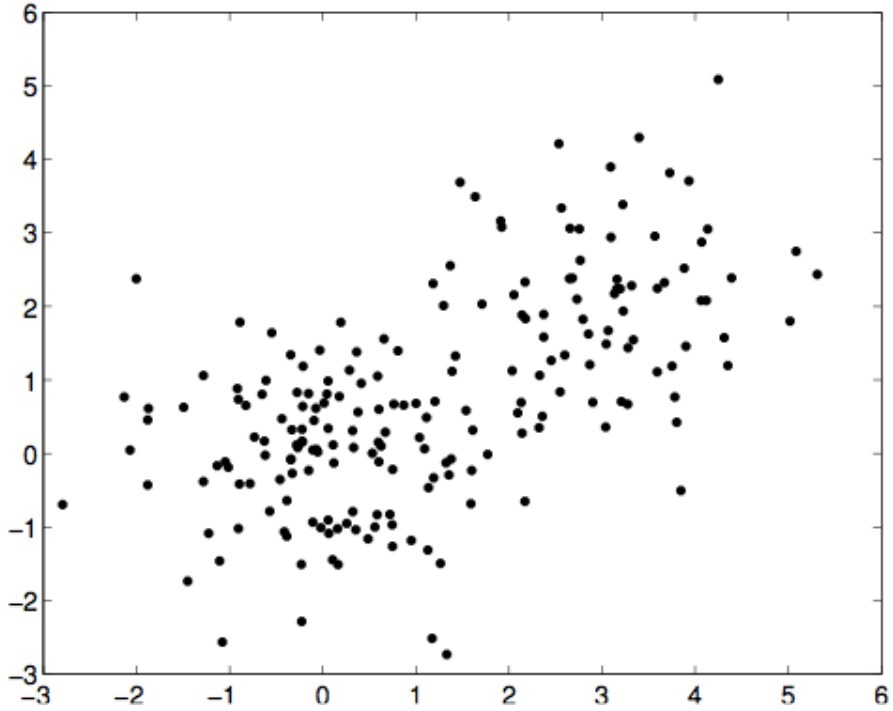
Classification - class discovery

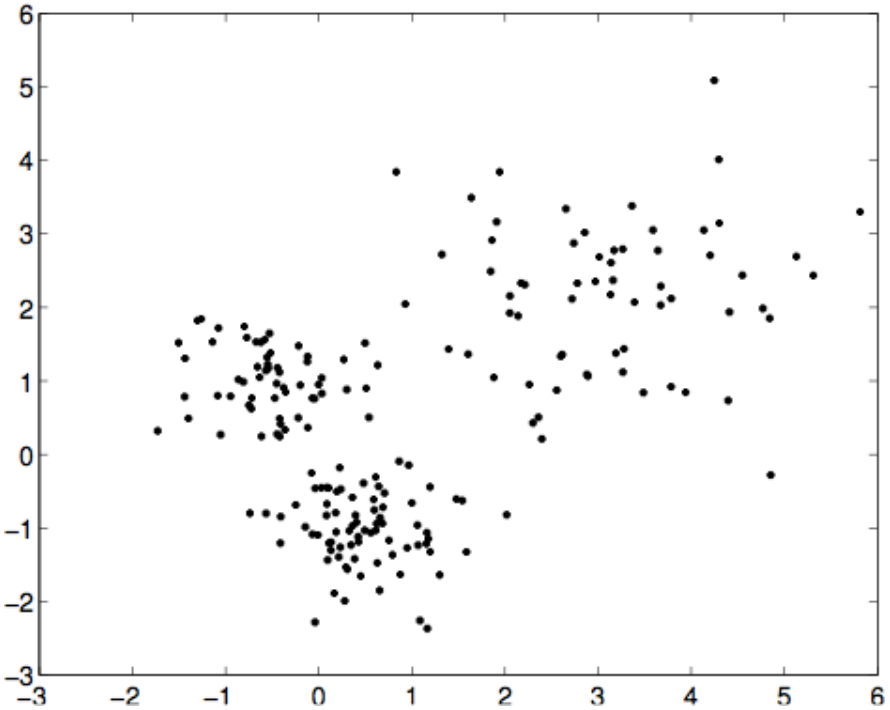
- Supervised learning (e.g., classification)
  - target labels (responses) provided for each example
- Semi-supervised learning
  - in addition to a few target labels we have a large number of unlabeled examples
- Unsupervised learning (class-discovery)
  - all examples are unlabeled



Class or type discovery

- There may be many underlying classes or "types" of examples
- Each class may itself consist of different sub-types
- Our goal is to discover how the data may have been generated from a "mixture" of underlying types





Mixtures and latent variables

- A k-component mixture model with parameters θ

$$P(\underline{x}, z; \theta) = P(\underline{x}|z; \theta)P(z; \theta)$$

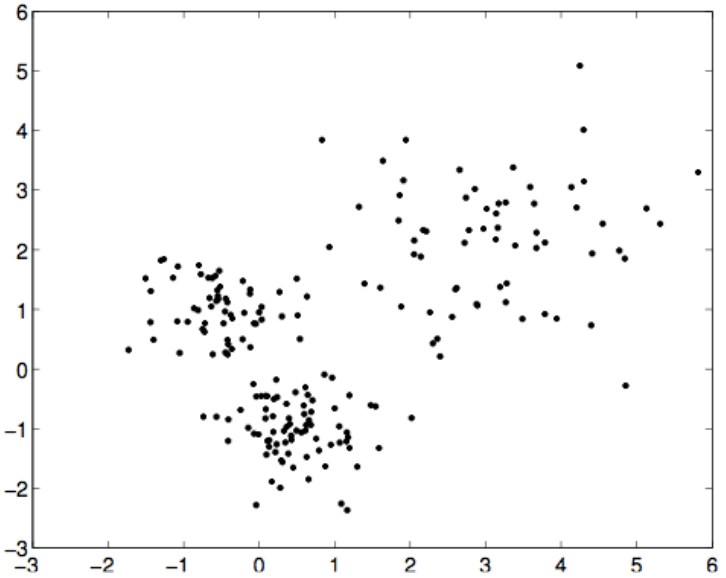
$$z \sim P(z; \theta) \text{ sample type } z = 1, \ldots, k$$

$$\underline{x} \sim P(\underline{x}|z; \theta) \text{ sample } \underline{x} \text{ given type } z$$

- The type z is latent (unobserved). The marginal distribution over $\underline{x}$ is given by the mixture

$$P(\underline{x}; \theta) = \sum_{z=1}^{k} P(\underline{x}|z; \theta)P(z; \theta)$$

$$\underline{x} \sim P(\underline{x}; \theta) \text{ sample } \underline{x}$$



Mixture estimation

- Consider a $k-$component mixture of spherical Gaussians

$$P(z;\theta) = \theta_z, \quad \sum_{z=1}^{k} \theta_z = 1$$

$$P(\underline{x}|z;\theta) = N(\underline{x};\, \underline{\mu}_z, \sigma_z^2 I)$$

where $\theta = \{\underline{\mu}_1, \ldots, \underline{\mu}_k, \sigma_1^2, \ldots, \sigma_k^2, \theta_1, \ldots, \theta_k\}$.

- We estimate the mixture parameters by maximizing the log-likelihood of the observed data $D = \{\underline{x}_1, \ldots, \underline{x}_n\}$

$$l(D;\theta) = \sum_{i=1}^{n} \log P(\underline{x}_i;\theta) = \sum_{i=1}^{n} \log \sum_{z=1}^{k} P(\underline{x}_i|z;\theta) P(z;\theta)$$

- This would be much easier if we knew the value of z corresponding to each $\underline{x}_i$ (complete data).

Mixture estimation: complete data

- If we knew the type z_i corresponding to each $\underline{x}_i$ then we would maximize the *complete log-likelihood*

$$\sum_{i=1}^{n} \log P(\underline{x}_i, z_i; \theta) = \sum_{i=1}^{n} \sum_{z=1}^{k} \delta(z, z_i) \log P(\underline{x}_i, z; \theta)$$

Mixture estimation: complete data

- If we knew the type z_i corresponding to each $\underline{x}_i$ then we would maximize the *complete log-likelihood*

$$\sum_{i=1}^{n} \log P(\underline{x}_i, z_i; \theta) = \sum_{i=1}^{n} \sum_{z=1}^{k} \delta(z, z_i) \log P(\underline{x}_i, z; \theta)$$

- Just as in the classification context, finding the maximizing parameters is simple (based on separately estimating the Gaussians from the points assigned to them)

$$\hat{\theta}_z = \frac{\sum_{i=1}^{n} \delta(z, z_i)}{n}$$

$$\hat{\mu}_z = \frac{1}{\sum_{i=1}^{n} \delta(z, z_i)} \sum_{i=1}^{n} \delta(z, z_i)\, \underline{x}_i$$

$$\hat{\sigma}_z^2 = \frac{1}{d \sum_{i=1}^{n} \delta(z, z_i)} \sum_{i=1}^{n} \delta(z, z_i) \left\| \underline{x}_i - \hat{\mu}_z \right\|^2$$

The EM-algorithm

Initialize: select the initial parameters $\theta^{(0)}$

E-step: fix the current parameters $\theta^{(m)}$, evaluate posterior assignments (divide each point softly across the mixture components)

$$p(z|i) = P(z|\underline{x}_i; \theta^{(m)}), \quad z = 1, \ldots, k, \quad i = 1, \ldots, n$$

M-step: fix the posterior assignments $p(z|i)$, find new parameters $\theta^{(m+1)}$ by maximizing the expected complete log-likelihood

$$\sum_{i=1}^{n} \sum_{z=1}^{k} p(z|i) \log P(\underline{x}_i, z; \theta)$$

with respect to θ

Mixture estimation: E-step

- The difficulty now is that we have to *complete* the data by inferring values for the missing types z_i

Mixture estimation: E-step

- The difficulty now is that we have to *complete* the data by inferring values for the missing types z_i
- Given the mixture parameters θ our best guess about the value of z_i corresponding to $\underline{x}_i$ is given by the posterior

$$p(z|i) = P(z|\underline{x}_i;\theta) = \frac{P(\underline{x}_i|z;\theta)P(z;\theta)}{\sum_{z'=1}^{k} P(\underline{x}_i|z';\theta)P(z';\theta)}$$

  Note that these "soft" posterior assignments $p(z|i)$ depend on the current setting of the parameters θ

Mixture estimation: E-step

- The difficulty now is that we have to *complete* the data by inferring values for the missing types z_i
- Given the mixture parameters θ, our best guess about the value of z_i corresponding to $\underline{x}_i$ is given by the posterior

$$p(z|i) = P(z|\underline{x}_i;\theta) = \frac{P(\underline{x}_i|z;\theta)P(z;\theta)}{\sum_{z'=1}^{k} P(\underline{x}_i|z';\theta)P(z';\theta)}$$

  Note that these "soft" posterior assignments $p(z|i)$ depend on the current setting of the parameters θ

- The soft assignments help us stochastically fill-in the missing type values. We therefore maximize the *expected log-likelihood* with respect to θ for fixed soft assignments

$$\sum_{i=1}^{n}\sum_{z=1}^{k} p(z|i)\log P(\underline{x}_i, z;\theta)$$

Mixture estimation: M-step

- As we have filled in the missing values, the estimation step is again easy. We simply use the "soft" posterior assignments $p(z|i)$ in place of the indicators $\delta(z, z_i)$

$$\hat{\theta}_z = \frac{\sum_{i=1}^{n} p(z|i)}{n}$$

$$\hat{\mu}_z = \frac{1}{\sum_{i=1}^{n} p(z|i)} \sum_{i=1}^{n} p(z|i)\, \underline{x}_i$$

$$\hat{\sigma}_z^2 = \frac{1}{d \sum_{i=1}^{n} p(z|i)} \sum_{i=1}^{n} p(z|i)\, \|\underline{x}_i - \hat{\underline{\mu}}_z\|^2$$

- These new parameters $\hat{\theta}$ can in turn be used to derive better soft assignments and so on.

[E-step]

The EM-algorithm

Initialize: select the initial parameters $\theta^{(0)}$

E-step: fix the current parameters $\theta^{(m)}$, evaluate posterior assignments (divide each point softly across the mixture components)

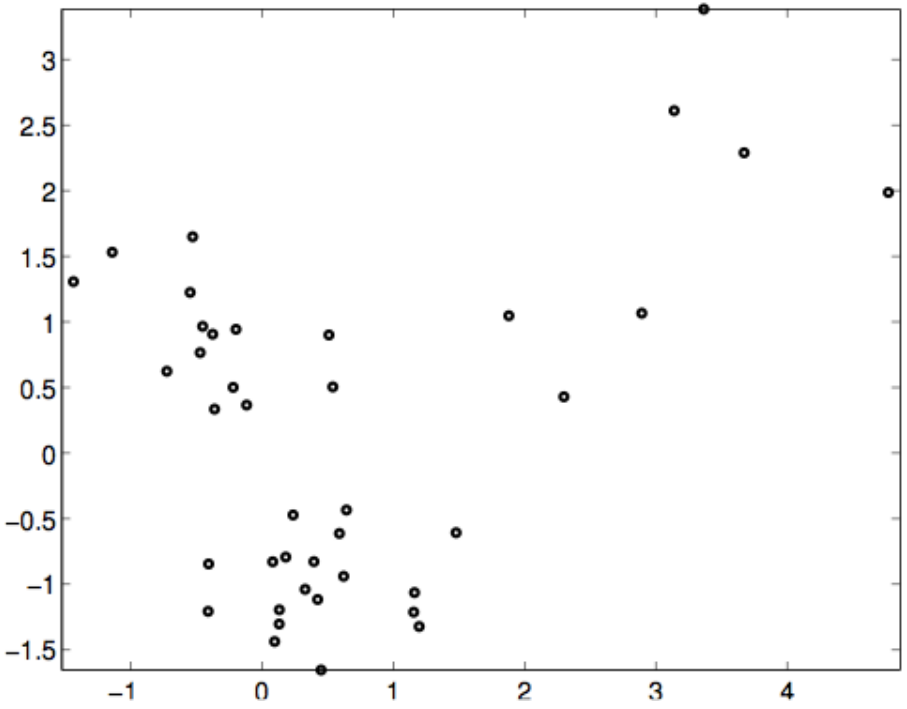
$$p(z|i) = P(z|\underline{x}_i; \theta^{(m)}), \quad z = 1, \ldots, k, \quad i = 1, \ldots, n$$

M-step: fix the posterior assignments $p(z|i)$, find new parameters $\theta^{(m+1)}$ by maximizing the expected complete log-likelihood

$$\sum_{i=1}^{n} \sum_{z=1}^{k} p(z|i) \log P(\underline{x}_i, z; \theta)$$

with respect to θ

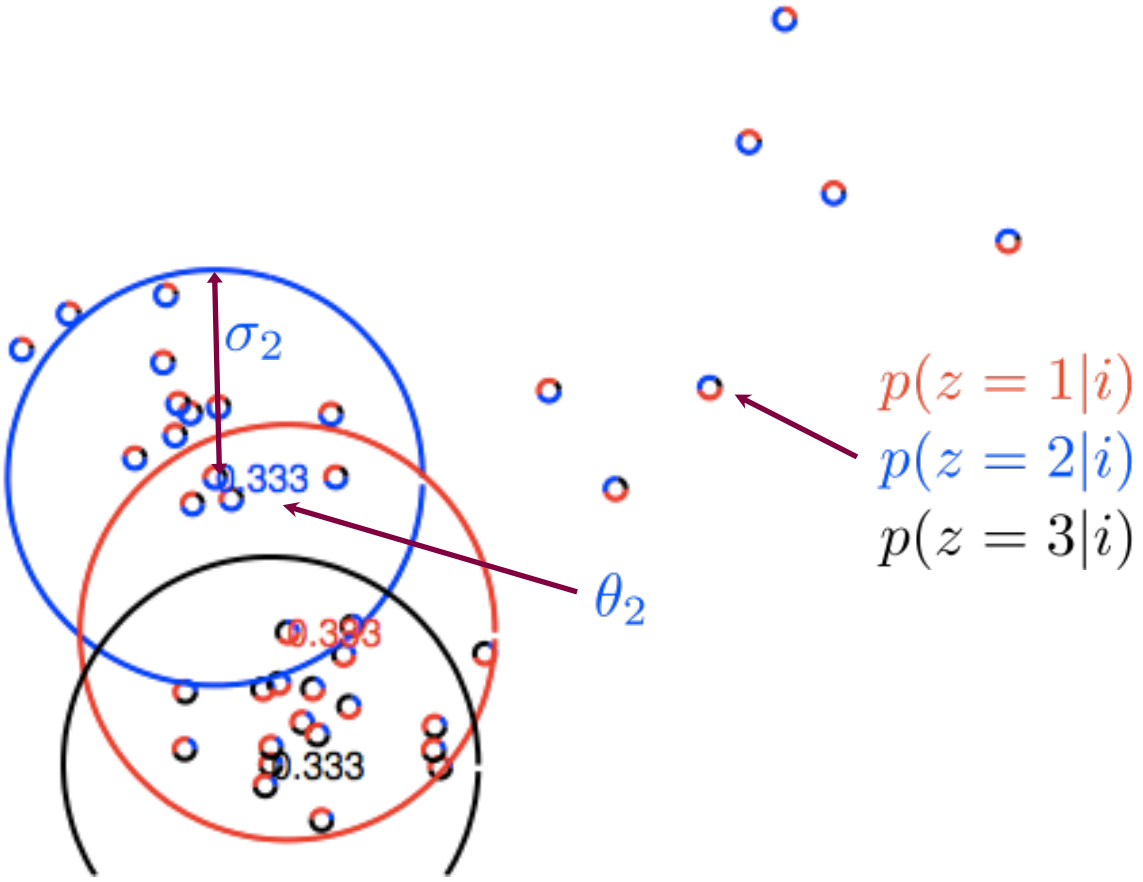
Mixture of Gaussians example



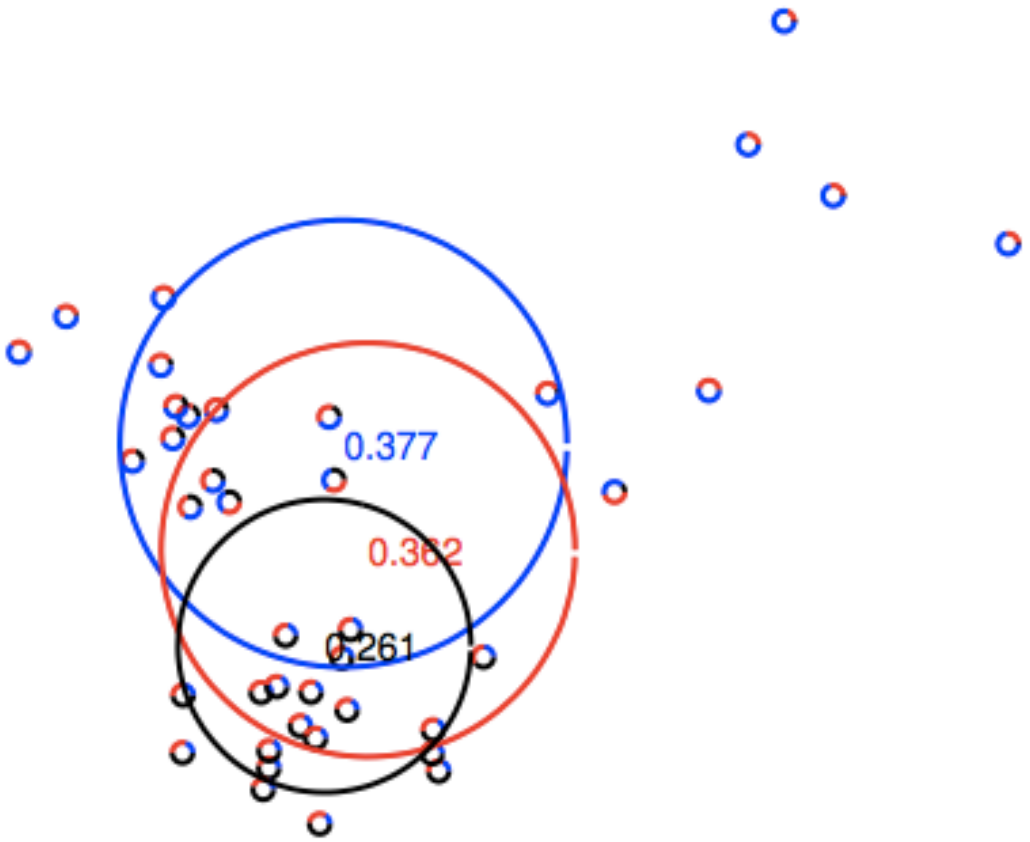
- The EM-algorithm is iterative and requires an initialization
  - $\theta_z = 1/k,\ z = 1, \ldots, k$
  - each μ_z is set to a randomly selected point
  - each σ_z^2 is set to the mean squared distance of points to the overall mean

Mixture of Gaussians example

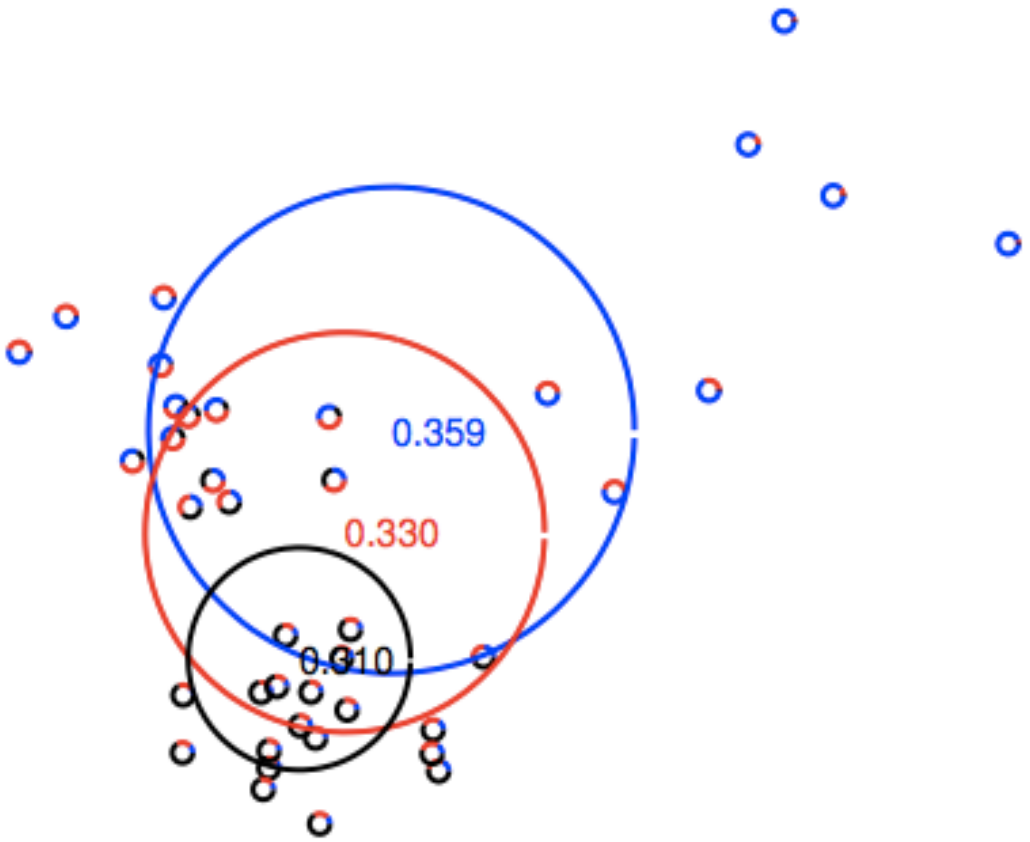
- initial 3-component mixture



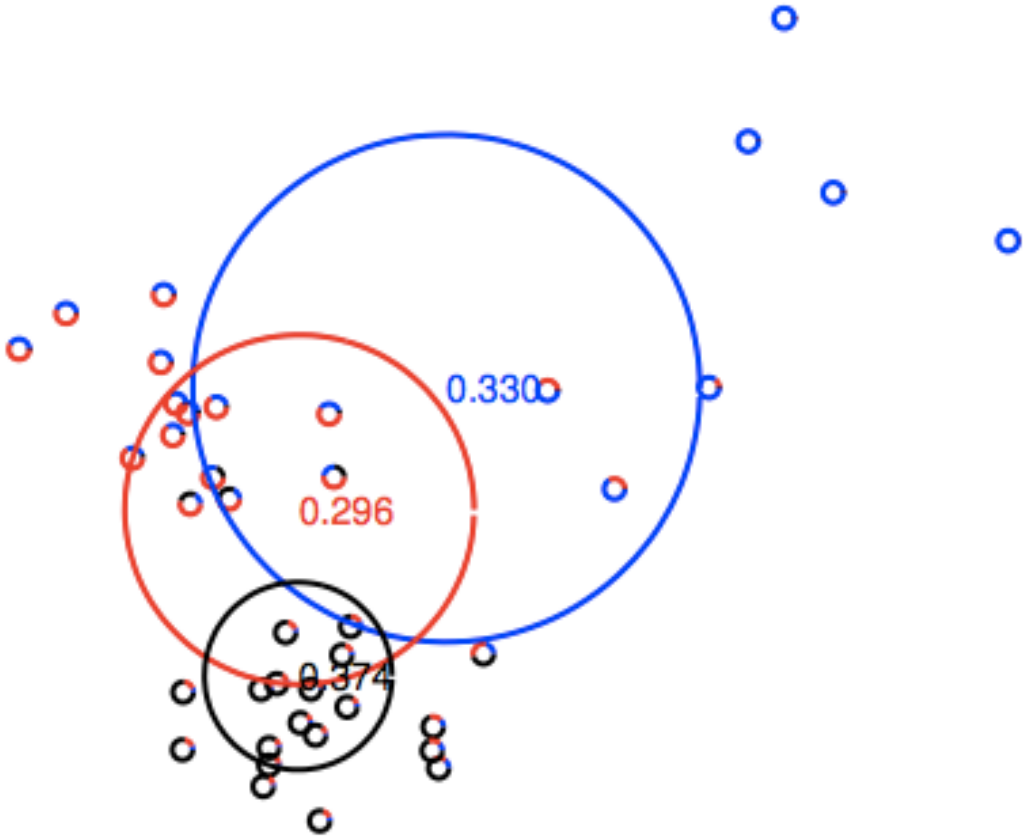
Mixture of Gaussians example



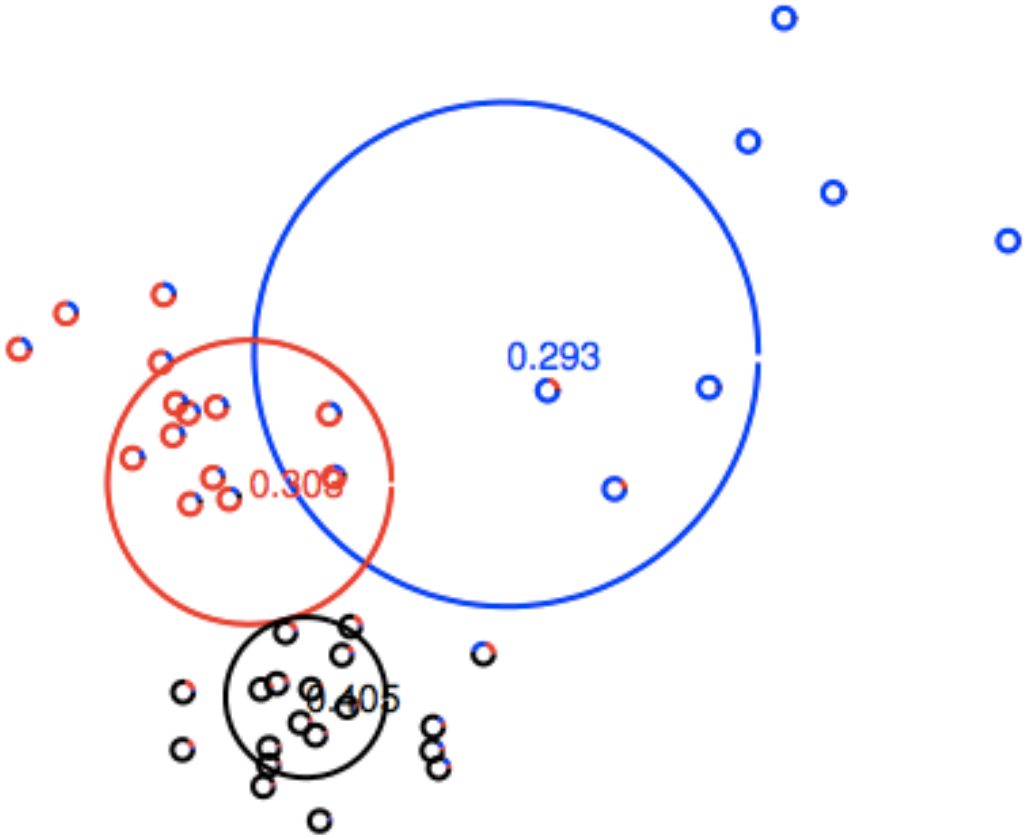
Mixture of Gaussians example



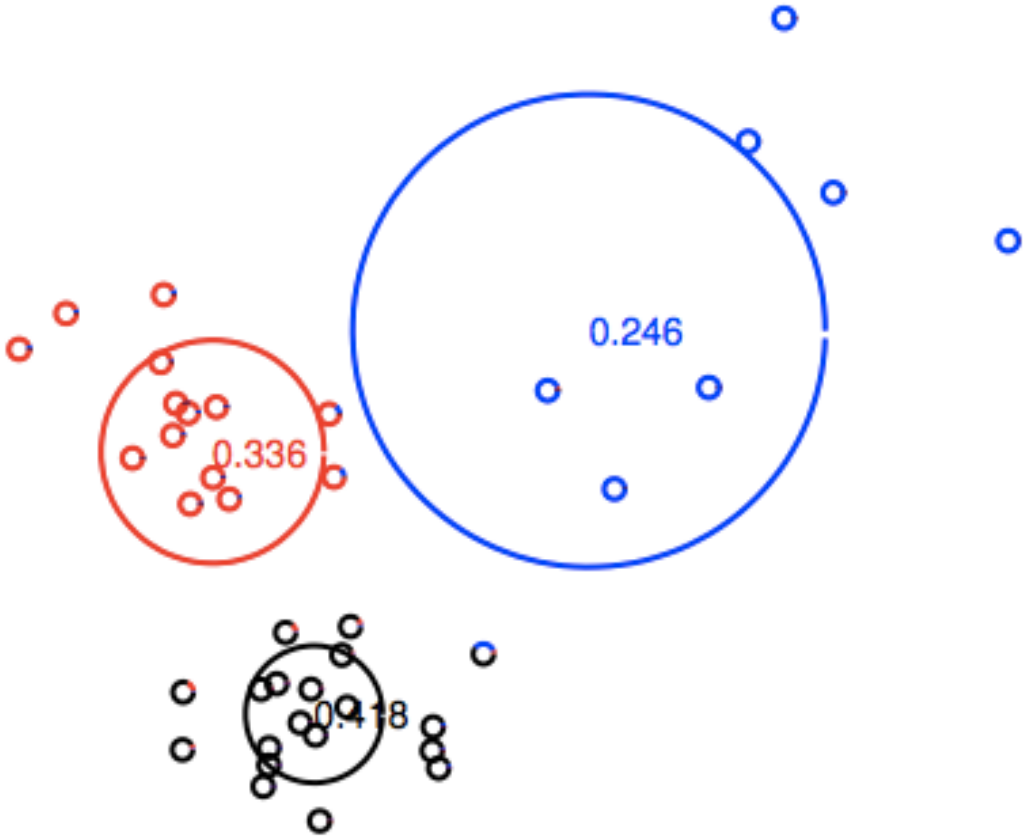
Mixture of Gaussians example



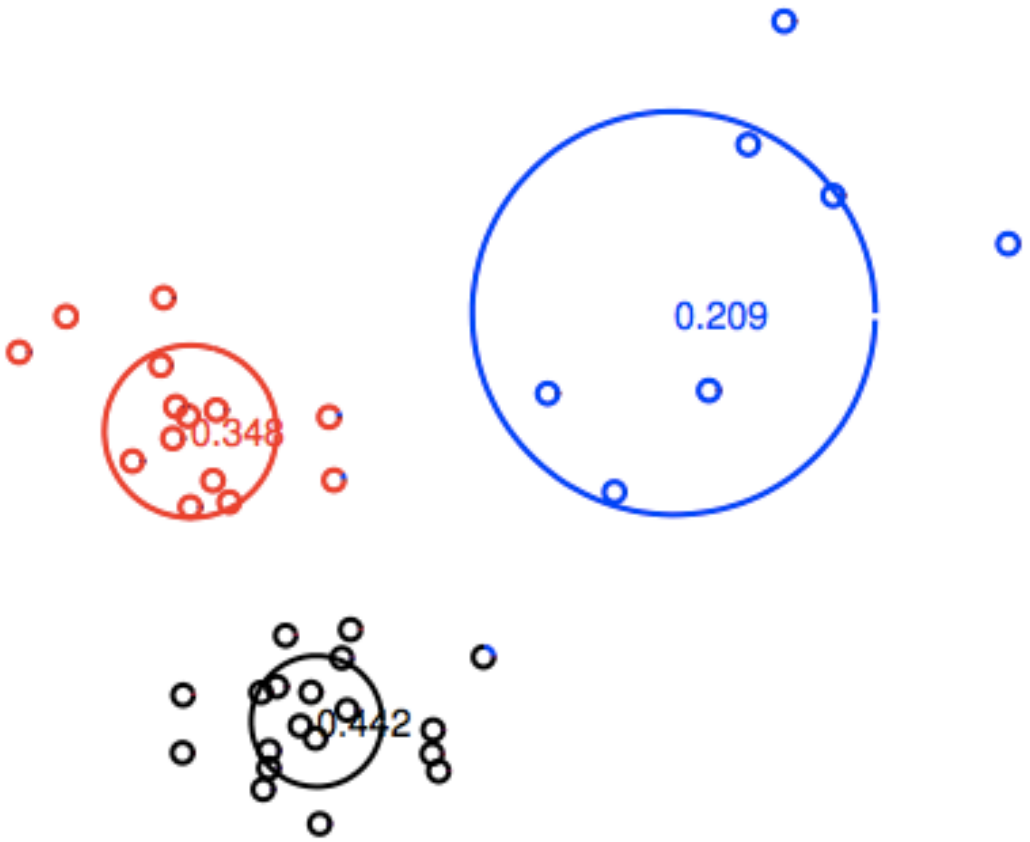
Mixture of Gaussians example



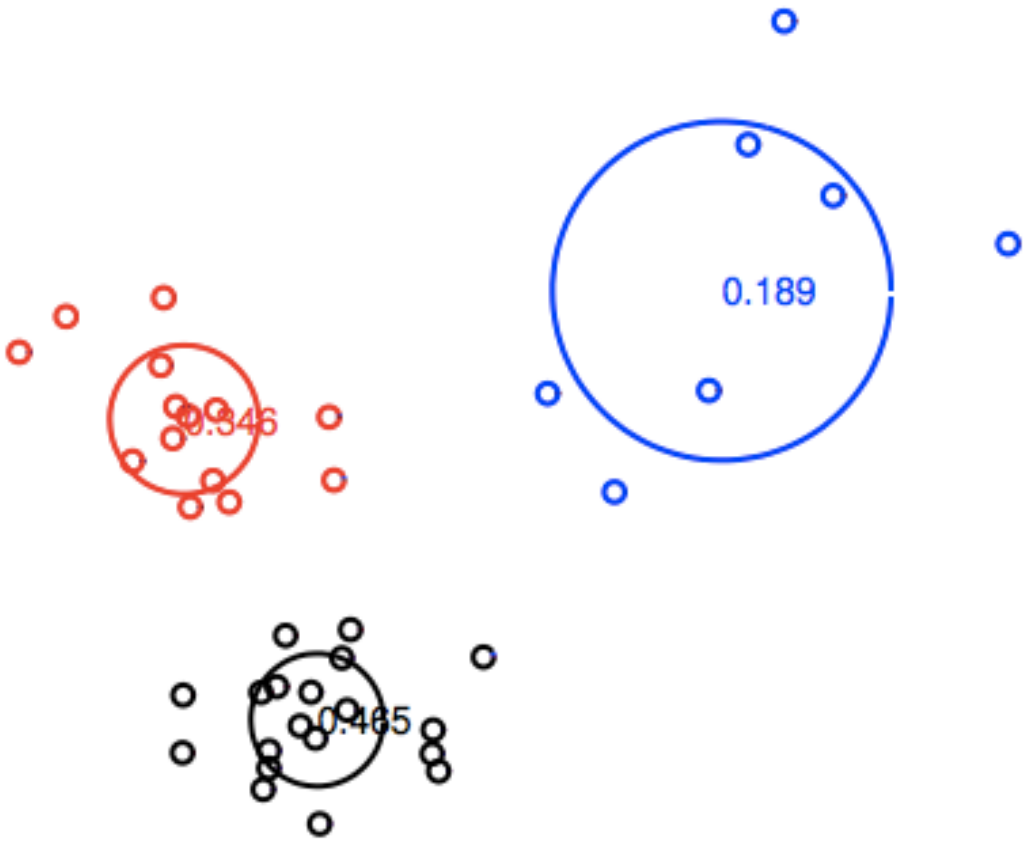
Mixture of Gaussians example



Mixture of Gaussians example

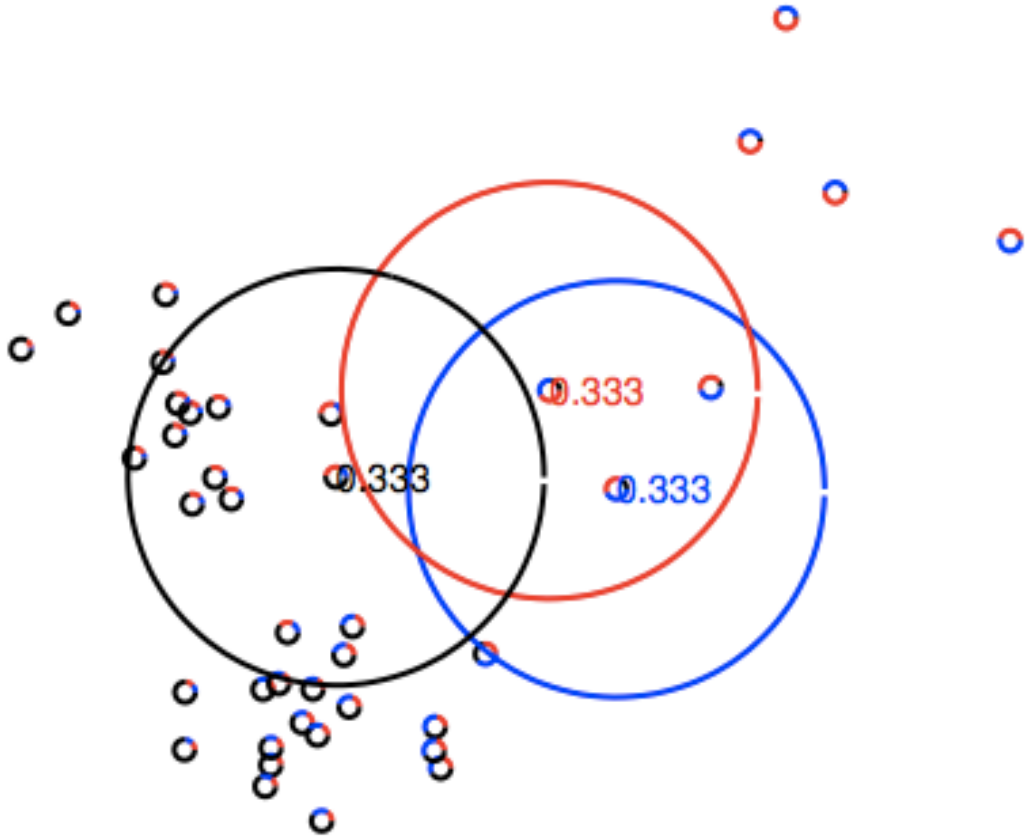


Mixture of Gaussians example

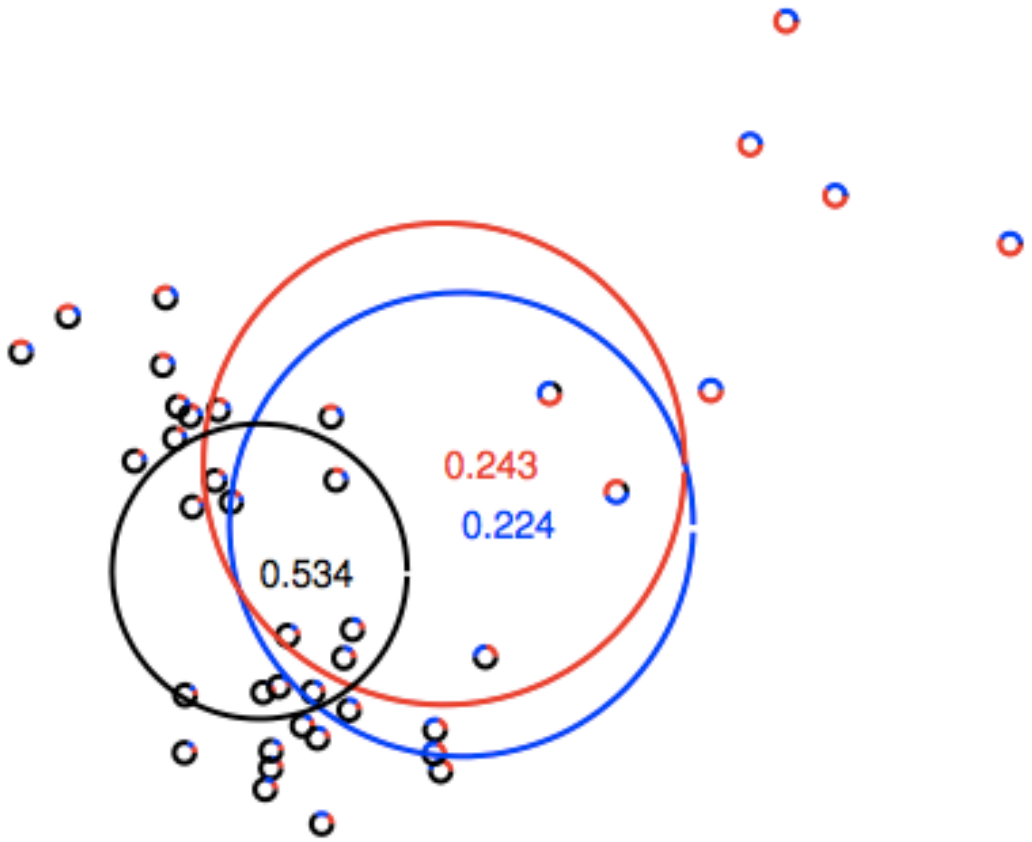


Mixture of Gaussians example

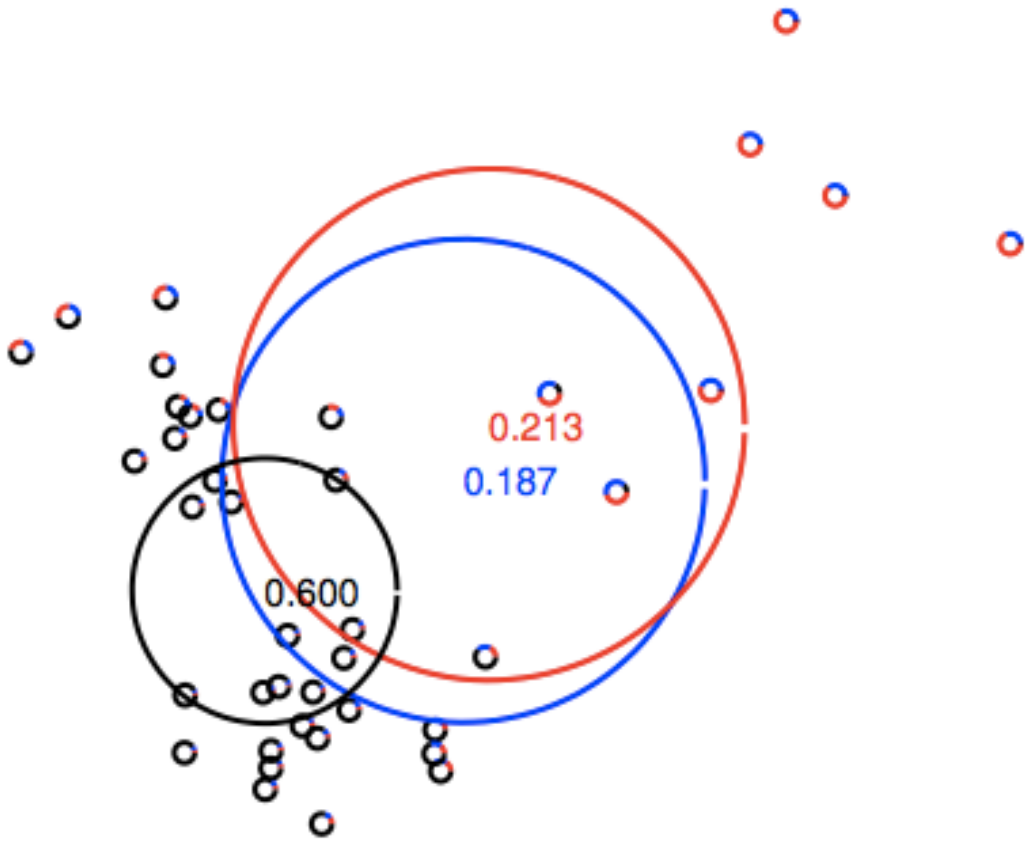
- initial 3-component mixture



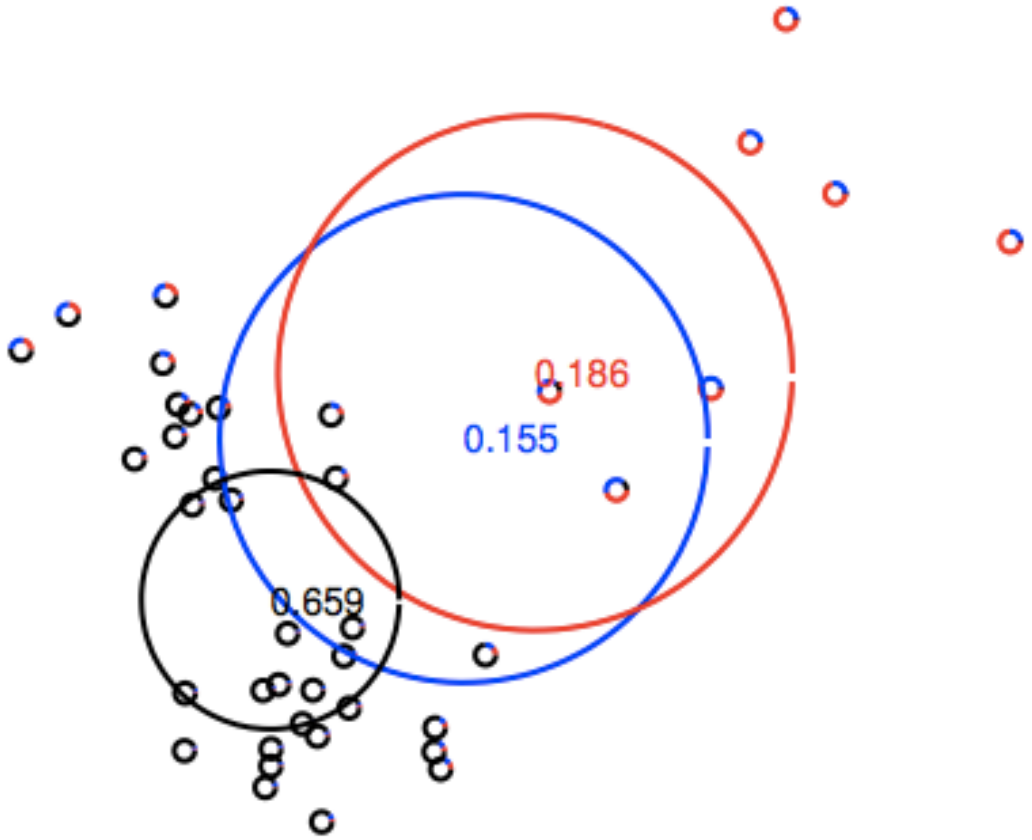
Mixture of Gaussians example



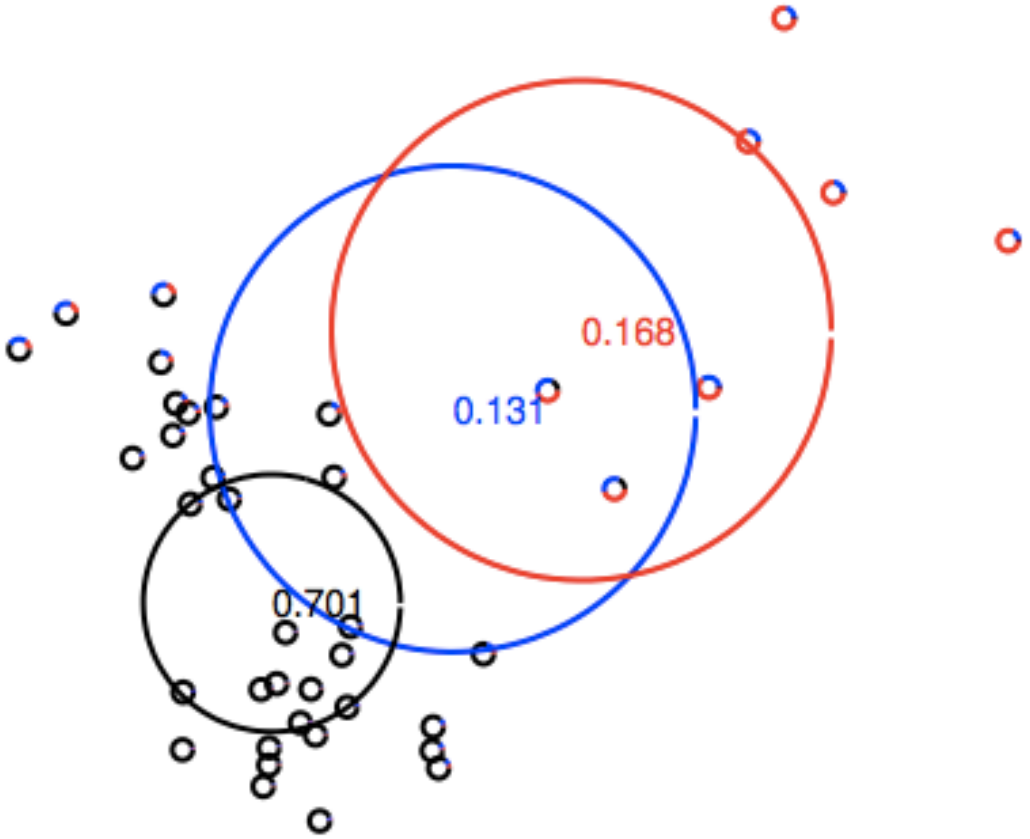
Mixture of Gaussians example



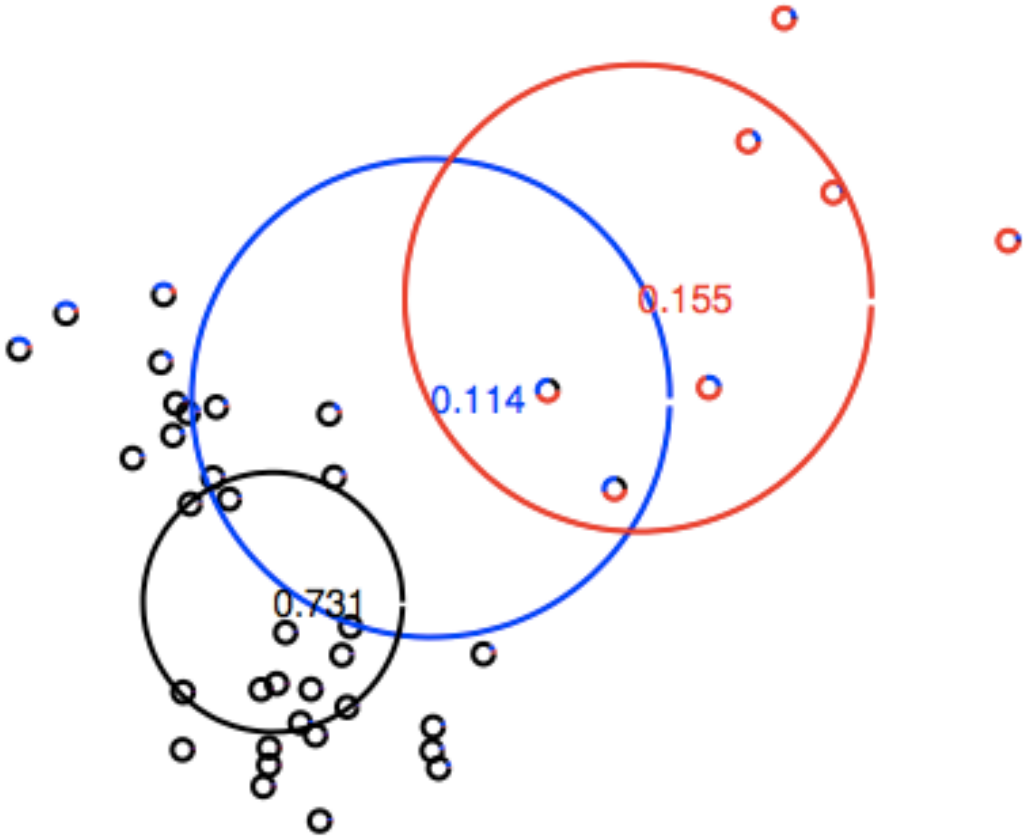
Mixture of Gaussians example



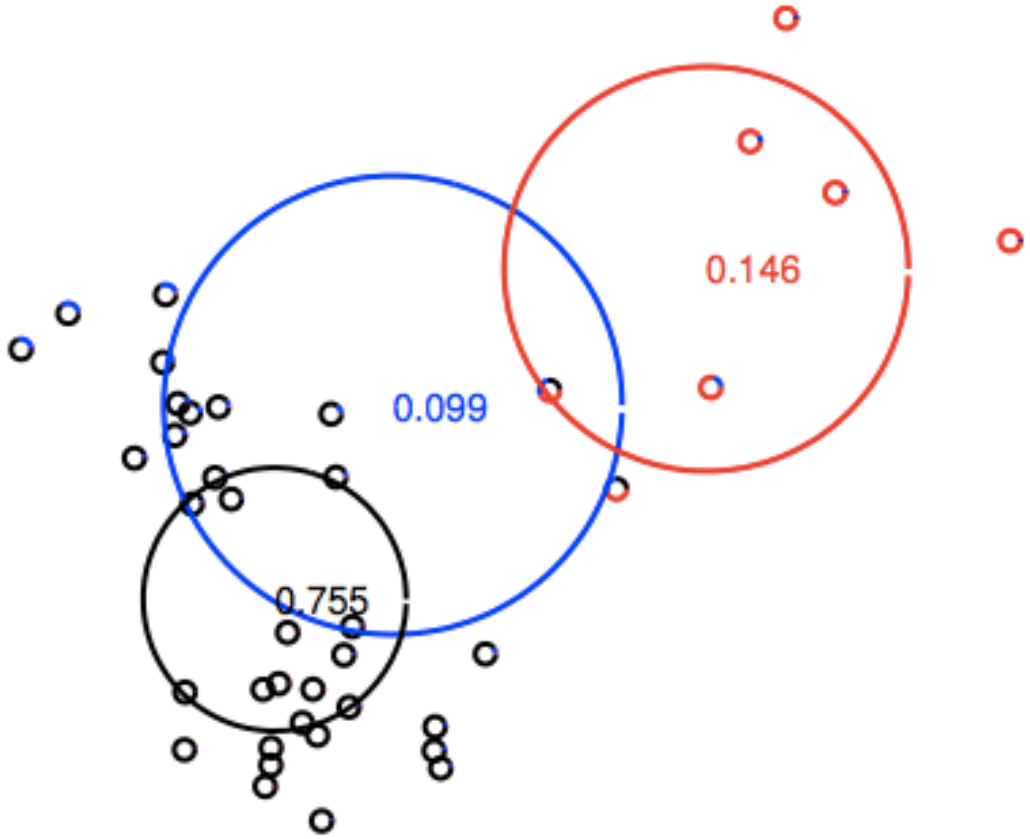
Mixture of Gaussians example



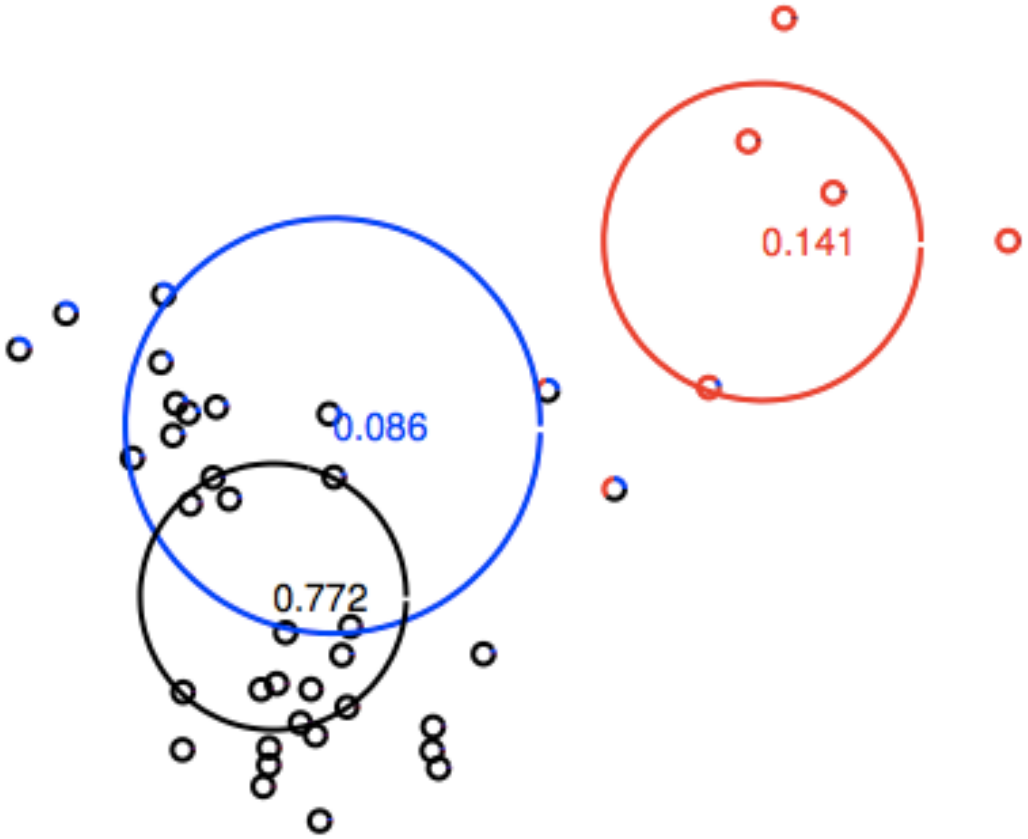
Mixture of Gaussians example



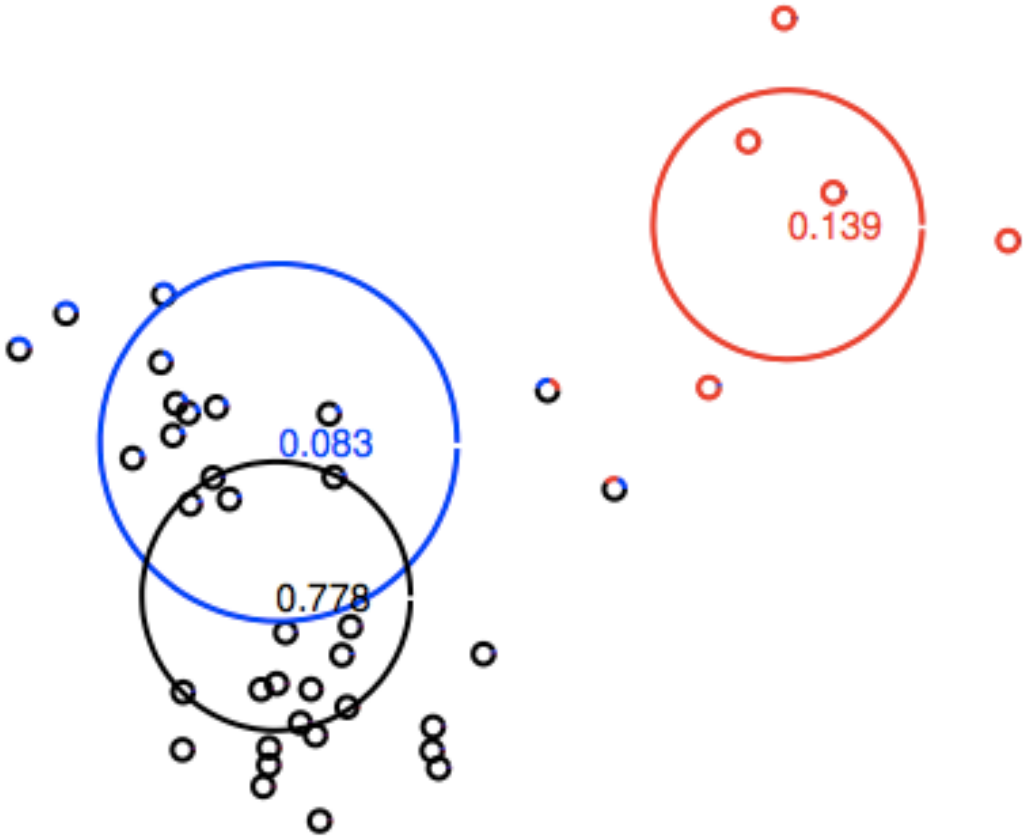
Mixture of Gaussians example



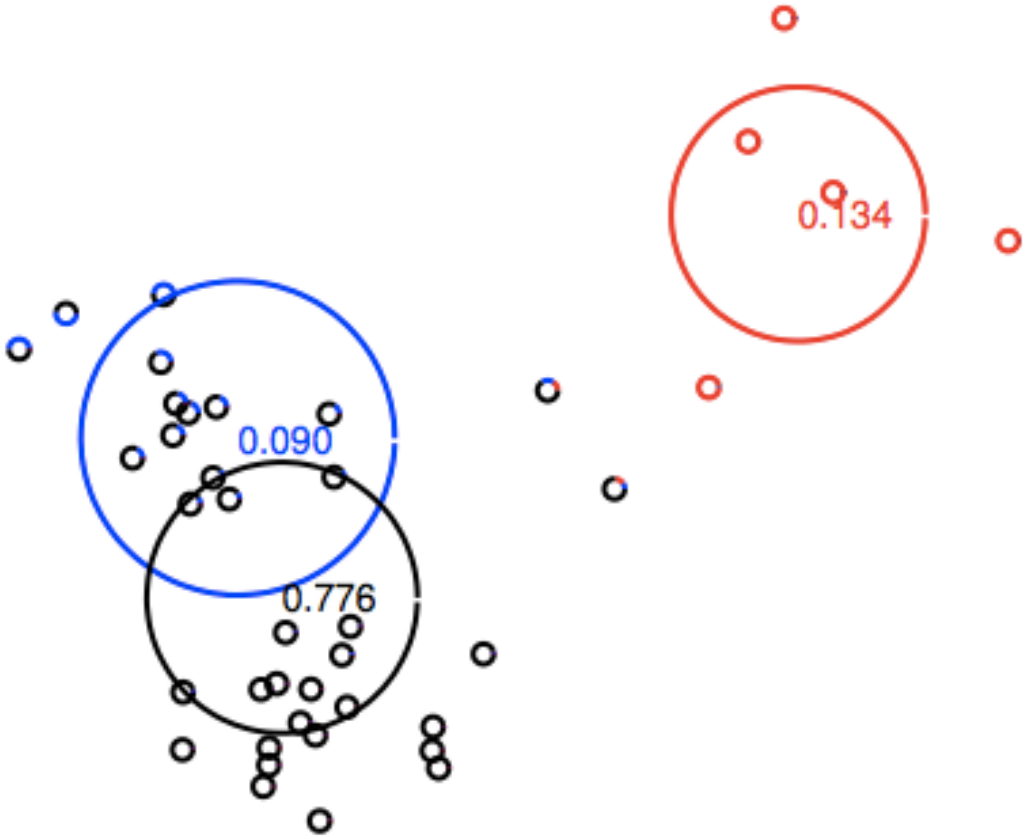
Mixture of Gaussians example



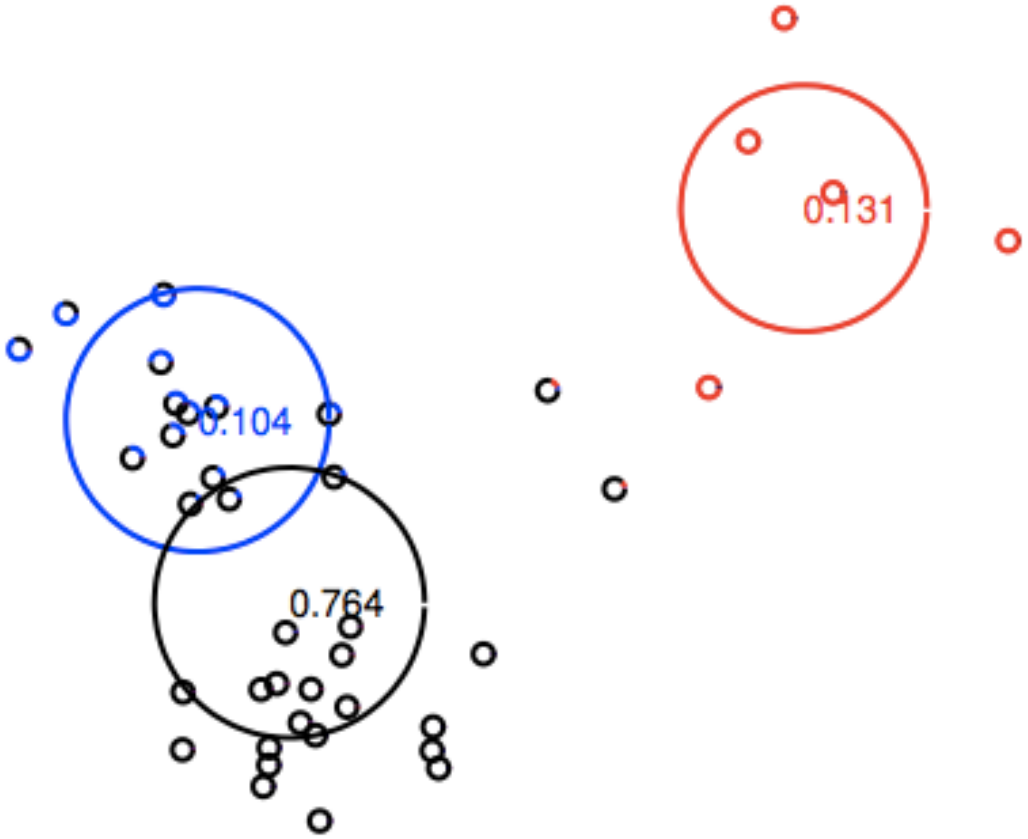
Mixture of Gaussians example



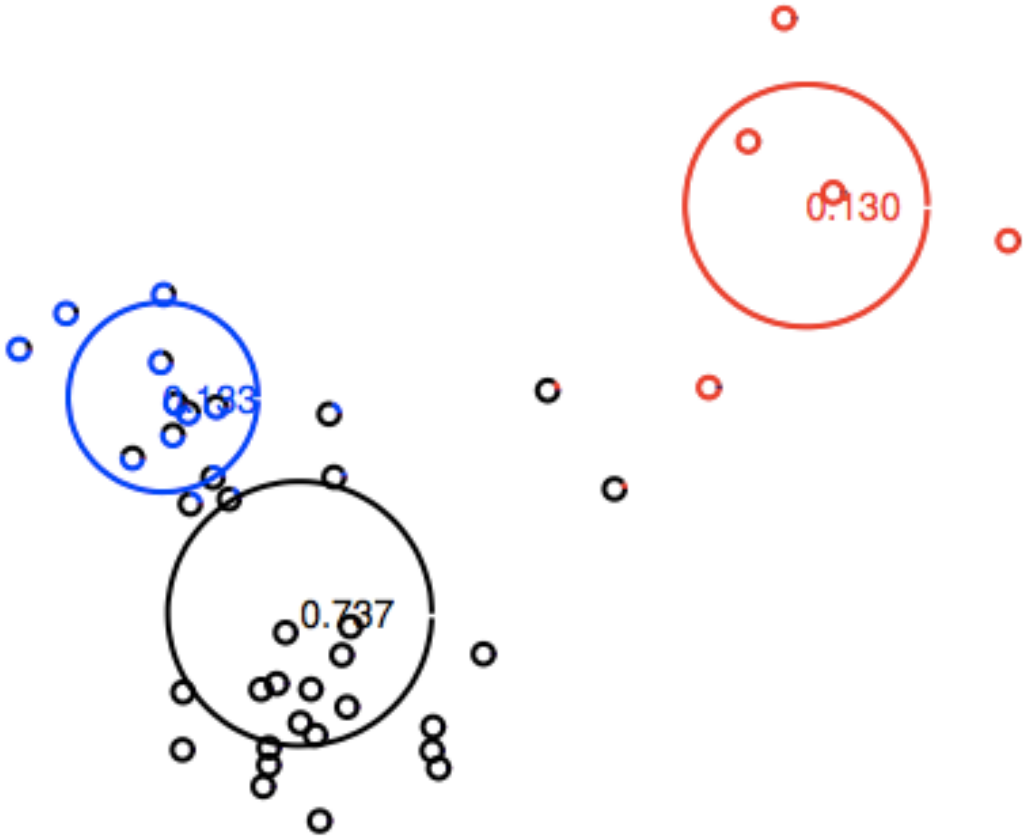
Mixture of Gaussians example



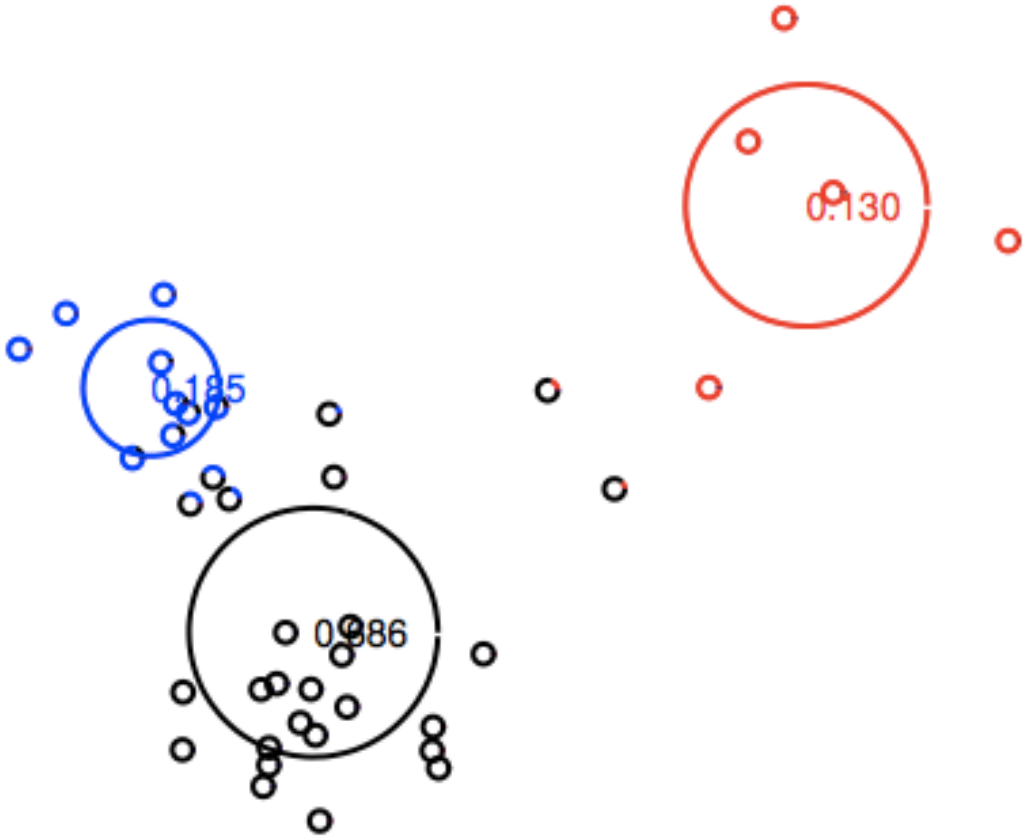
Mixture of Gaussians example



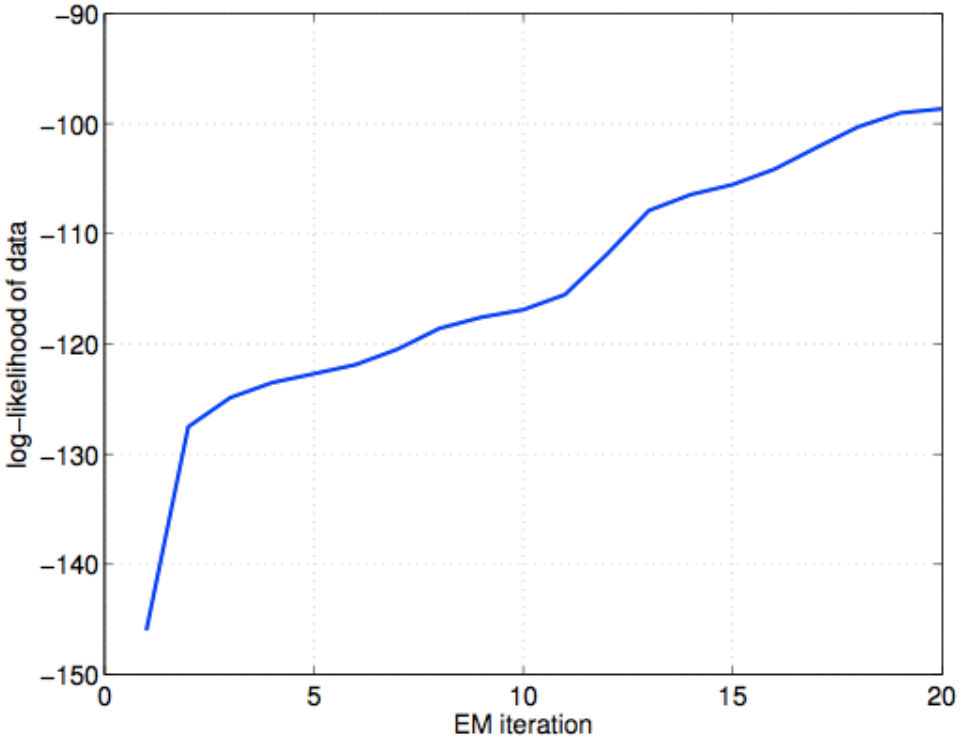
Mixture of Gaussians example



Mixture of Gaussians example



The EM algorithm



- The EM-algorithm monotonically increases the log-likelihood of the training data

$$l(D;\theta^{(0)}) < l(D;\theta^{(1)}) < \ldots < l(D;\theta^{(m)})$$

The EM-algorithm

Initialize: select the initial parameters $\theta^{(0)}$

E-step: fix the current parameters $\theta^{(m)}$, evaluate posterior assignments (divide each point softly across the mixture components)

$$p(z|i) = P(z|\underline{x}_i; \theta^{(m)}), \quad z = 1, \ldots, k, \quad i = 1, \ldots, n$$

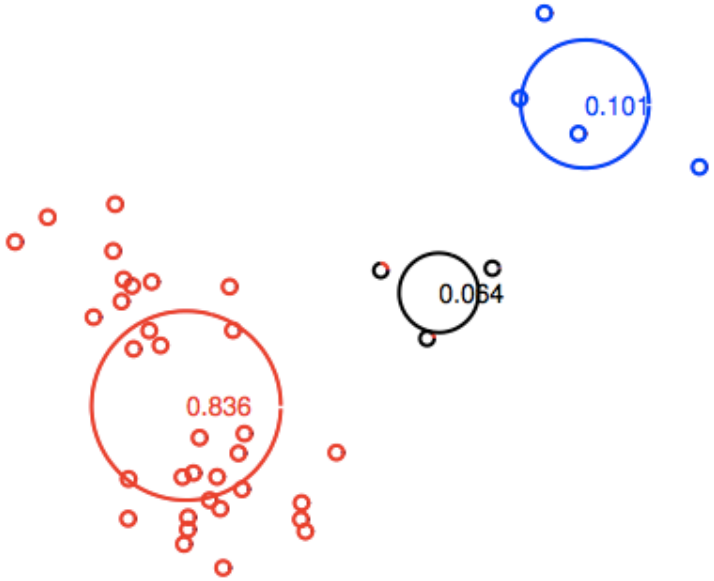
M-step: fix the posterior assignments $p(z|i)$, find new parameters $\theta^{(m+1)}$ by maximizing the expected complete log-likelihood

$$\sum_{i=1}^{n} \sum_{z=1}^{k} p(z|i) \log P(\underline{x}_i, z; \theta)$$

with respect to θ

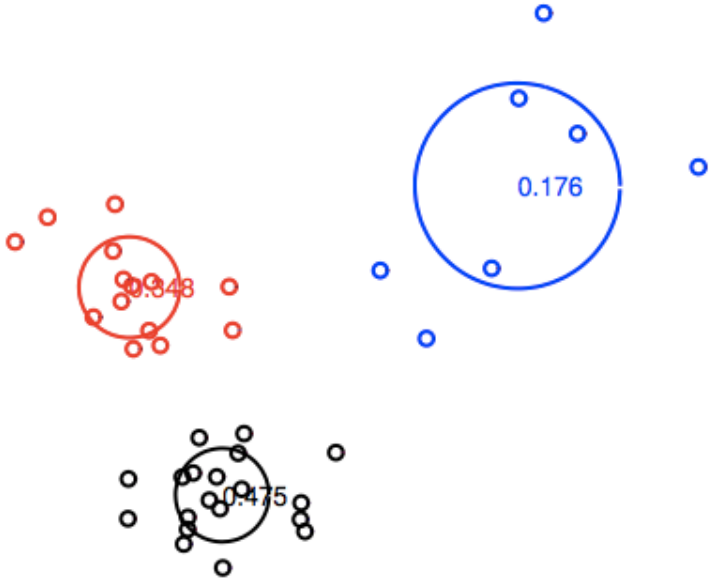
Locally optimal solutions

- The EM-algorithm is guaranteed to find a locally optimal solution by monotonically increasing the log-likelihood (the estimation problem with respect to θ is typically not convex)
- Whether the algorithm converges to the globally optimal solution depends on the initialization

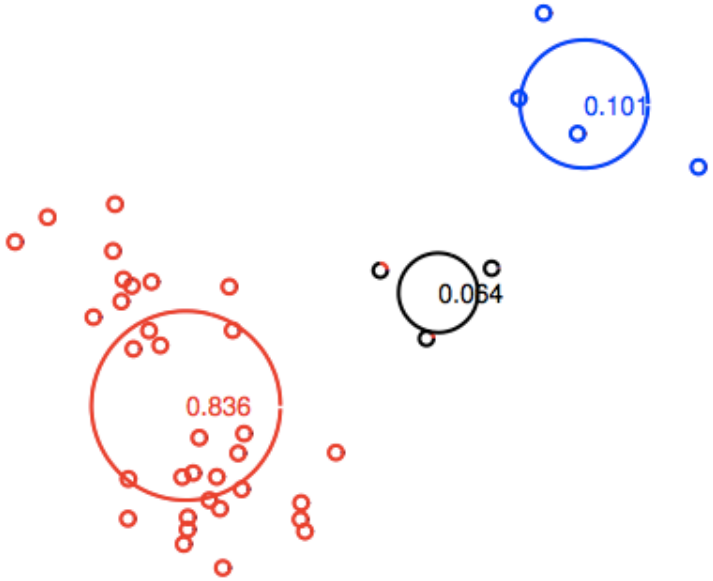


Locally optimal solutions

- The EM-algorithm is guaranteed to find a locally optimal solution by monotonically increasing the log-likelihood (the estimation problem with respect to θ is typically not convex)
- Whether the algorithm converges to the globally optimal solution depends on the initialization



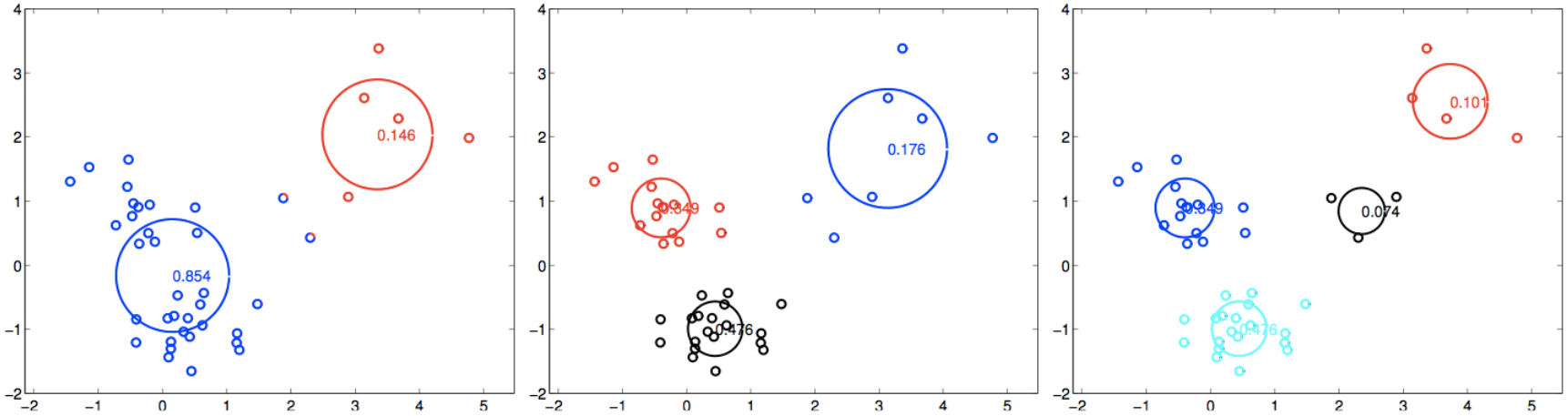
$$l(D;\hat{\theta}) = -98.64$$



$$l(D;\hat{\theta}) = -114.82$$

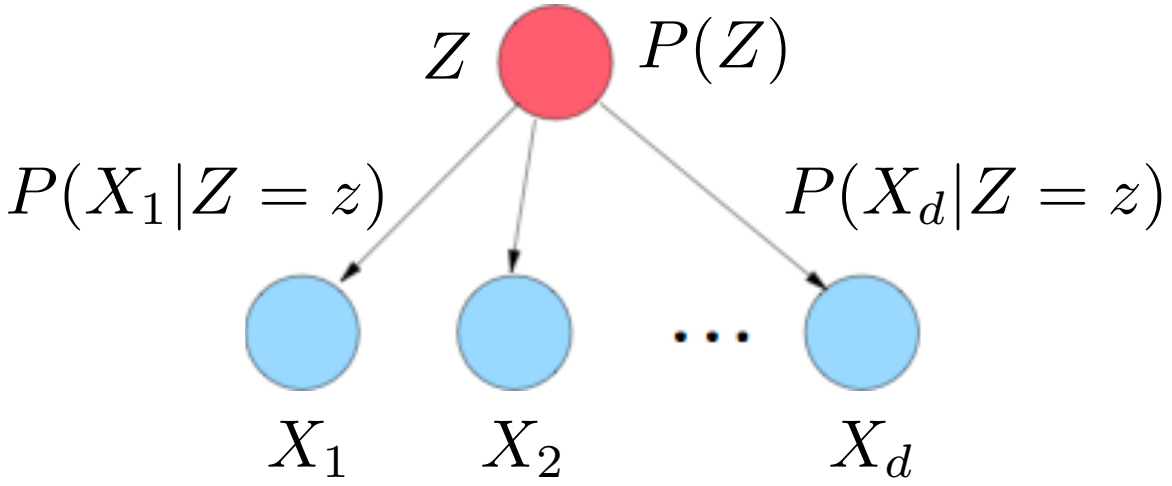
Model selection

- We can run the EM-algorithm with different numbers of components. Need to specify a criterion for selecting among the different models.



$l(D;\hat{\theta}) = -118.25$ $l(D;\hat{\theta}) = -98.64$ $l(D;\hat{\theta}) = -94.11$

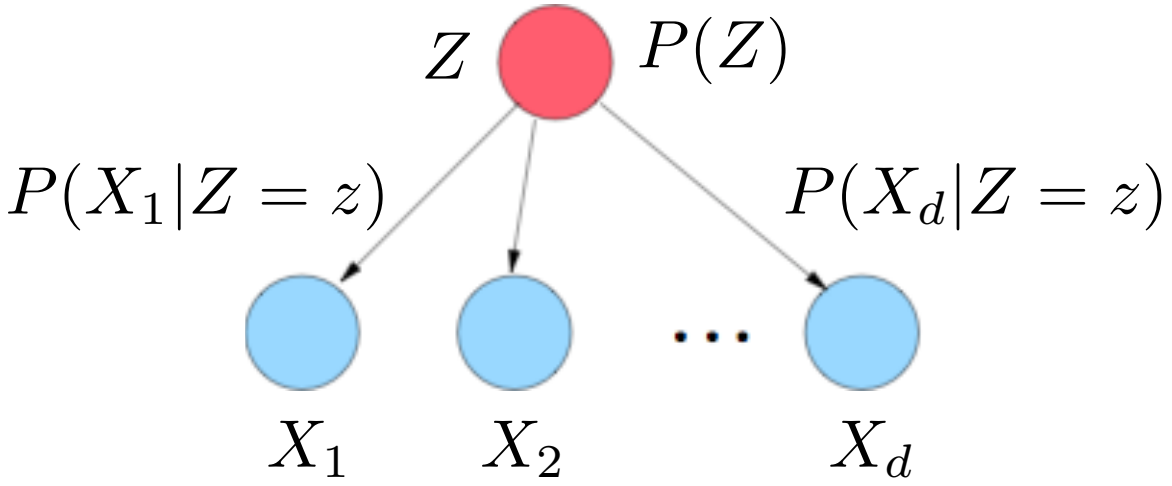
Graphical models / Bayes Nets



This is an example of a Graphical Model (a.k.a. Bayesian Network, Bayes Net, Probabilistic Graphical Model)

- Nodes are random variables
- Edges are conditional probability distributions

Graphical models / Bayes Nets

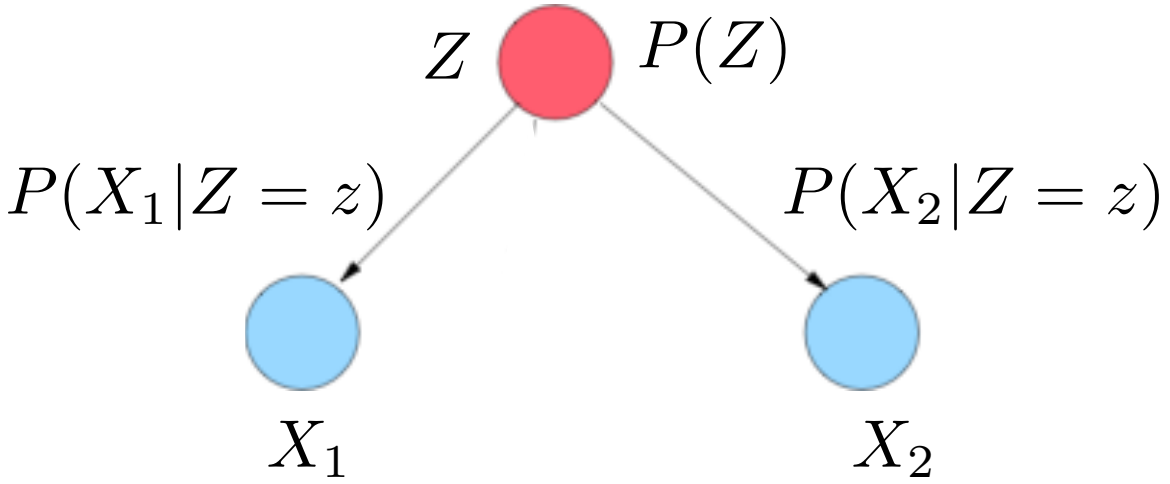


A graphical model factors the joint probability distribution (over all random variables)

→ Much fewer table entries are needed than in the full joint!

This is subject to the (conditional) independence assumptions specified by the graph.

Graphical models / Bayes Nets



Example: Binary random variables: $z, x_i, x_2 \in \{0, 1\}$

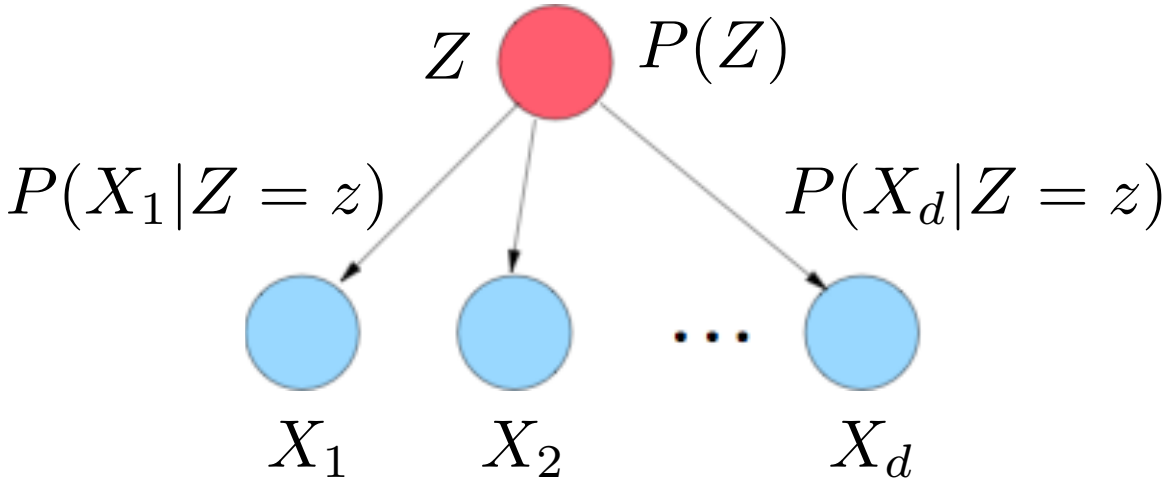
- The joint probability table $P(X_1, X_2, Z)$ has $2^3 = 8$ entries.
- Assuming the graph's independence structure, the joint factors into:

$$P(X_1, X_2, Z) = P(Z)P(X_1|Z)P(X_2|Z)$$

- To specify this Bayes Net requires only 5 parameters:

$$p(Z = 1)\, p(X_1 = 1|Z = 1), p(X_1 = 1|Z = 0)$$
$$p(X_2 = 1|Z = 1), p(X_2 = 1|Z = 0)$$

Naïve Bayes model



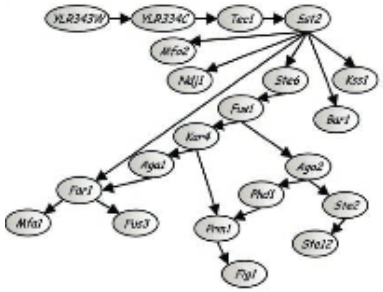
A k-component mixture over d features X_i that assumes conditional independence among the X_i's, given Z:

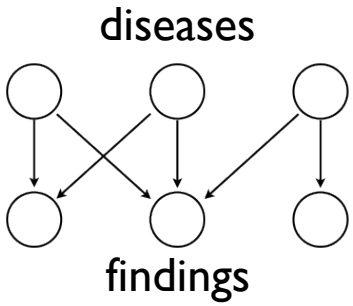
Fix the value of Z = z,

then X_i is independent of X_j for all i,j $\in \{1, \ldots, d\}$

Bayesian networks

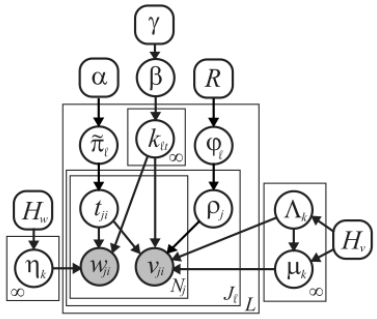
- Bayesian networks provide a simple language for succinctly expressing, using, and learning probabilistic information
- In a Bayesian network, nodes correspond to random variables and directed edges indicate dependencies
  - the graph provides a qualitative description of how the variables relate to each other
  - the probability distribution underlying the graph quantifies numerically how the variables depend on each other
- Both the graph structure and the associated distribution are important in applications
  - diagnostic tasks, medicine
  - data analysis in biology
  - etc.

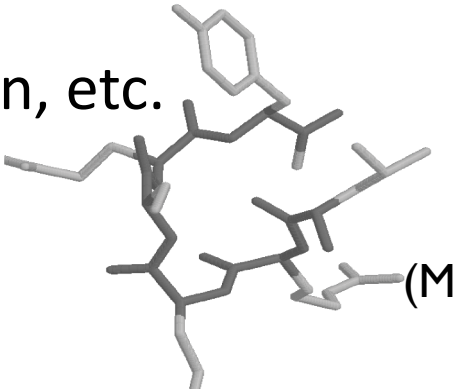




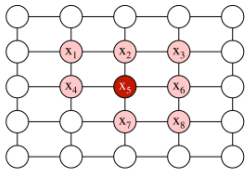
Applications of Bayes Nets

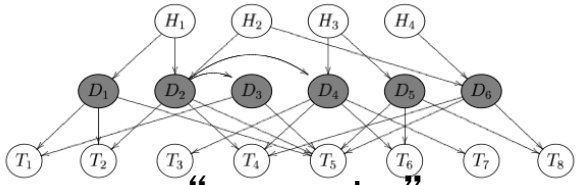
- Computer vision
  - scene analysis, stereo matching, etc.
- Molecular biology
  - networks, molecular design, etc.
- Information retrieval / NLP
  - topic models, parsing
- Diagnosis
  - fault, medical
- Signal processing
  - deconvolution, sensor networks
- Communication
  - decoding algorithms



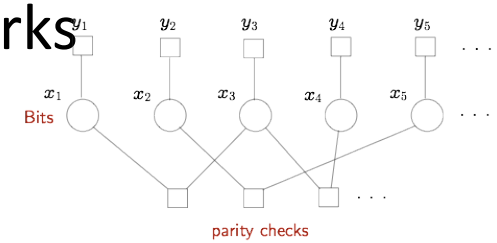


(Meltzer et al.)

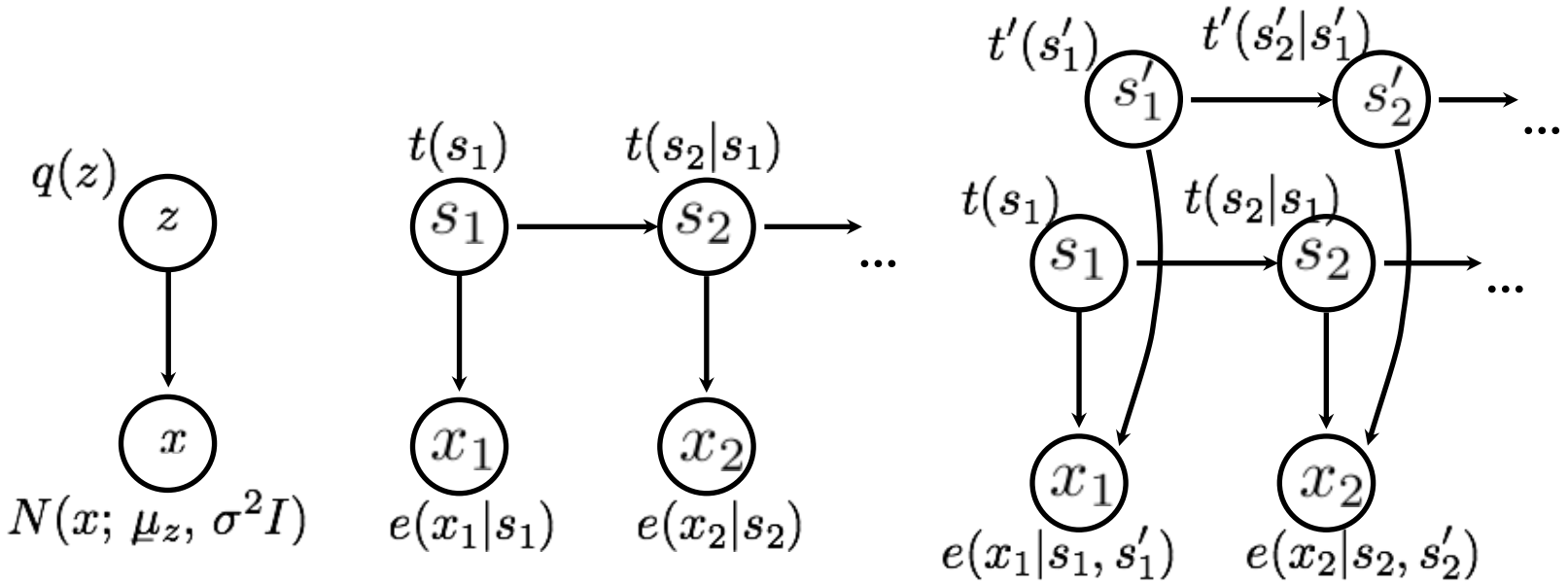




"promedas"



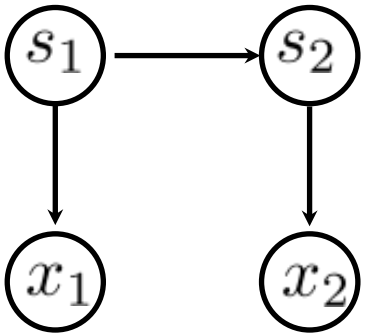
Graphs and probabilities



- Bayesian networks as probability models are represented by directed acyclic graphs (DAGs) where the nodes specify variables and the directed edges identify dependencies
- Formally, the graph encodes conditional independence properties about the variables (cf. Markov properties)
- Any probability distribution we associate with the graph has to be consistent with such independence properties

Bayesian networks

- Some basic terminology:
  - s_1 is a "parent" of x_1
  - s_1 and s_2 are "ancestors" of x_2; x_1 is not.
  - x_1 is a "child" of s_1
  - x_1, s_2 and x_2 are all "descendants" of s_1



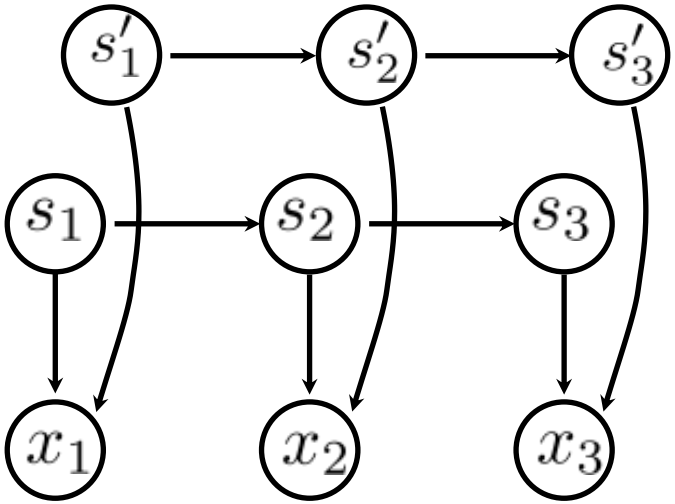
- Joint probability distribution factors as:

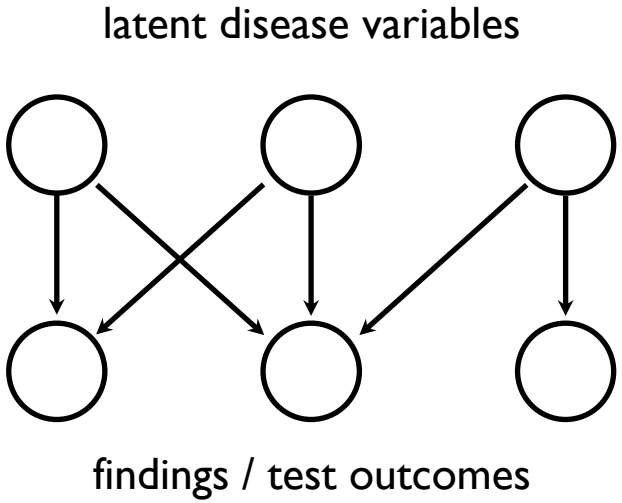
$$P(V_1, V_2, \ldots, V_k) = \prod_{i=1}^{k} P(v_i | \texttt{Parents}(V_i))$$

Bayesian networks

Examples:

- a factorial HMM, where observations depend on two Markov chains running in parallel (two speakers, one microphone)
- a two layer model for medical diagnosis





Learning Bayesian networks

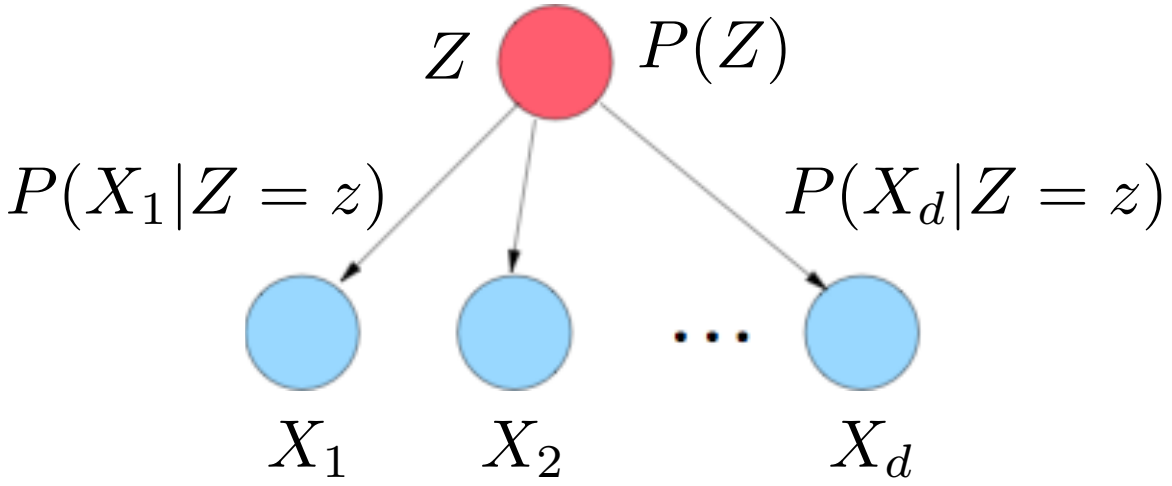
$D =$

complete data

x_1	x_2				$\cdots$					x_n
2	2	1	1	1	2	1	1	2	1	2
1	2	2	2	2	2	3	1	1	2	2
1	1	1	1	2	1	1	1	1	1	1
2	2	3	1	1	2	3	3	2	3	2
1	2	2	2	2	2	3	1	1	2	2
1	1	3	1	1	1	3	1	1	1	3
					$\cdots$					

- **Parameter estimation:** find the maximum likelihood (or Bayesian) estimates of parameters for a graph G
- **Model selection:** appropriately score each G based on its degree of fit to the data
- **Structure search:** find the highest scoring structure $\hat{G}$

Naïve Bayes model



Estimation from labeled training data: Filter the data by labels z:

For each label z, fit a probability model, $P(X_i|Z)$, for each X_i.

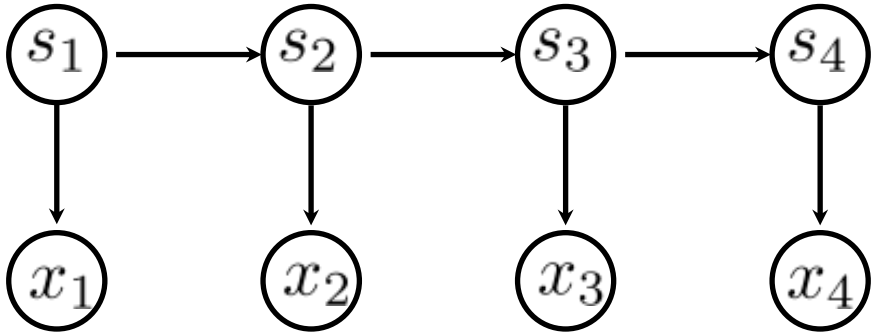
E.g. if $P(X_i|Z)$ is modeled with a Gaussian, then for each i, just need to compute $\hat{\mu}_i, \hat{\sigma}_i^2$ from the training data.

Prediction on a new example $x = (x_1, \ldots, x_d)$:

$$\hat{z} = \arg\max_z P(Z|x_1, \ldots, x_d) = \arg\max_z P(Z) \prod_{i=1}^{d} P(x_i|Z)$$

The Hidden Markov Model (HMM)

- Each node corresponds to a variable, either state s_t or observation x_t, and the arcs represent dependencies such as "s_2 depends on s_1"



HMMs

- Hidden Markov models are widely used models
- speech recognition
  - e.g., each word is a Markov sequence of phonemes which are then coupled to acoustic measurements
- computational biology
  - e.g., gene structure labels (coding region, etc.) form a Markov chain of states and each state generates a base-pair (sequence)
- natural language processing
  - e.g., part of speech tags (noun, verb, etc) form a Markov chain and the tags give rise to observed words

HMM problems

- Three problems we need to solve in the context of HMMs

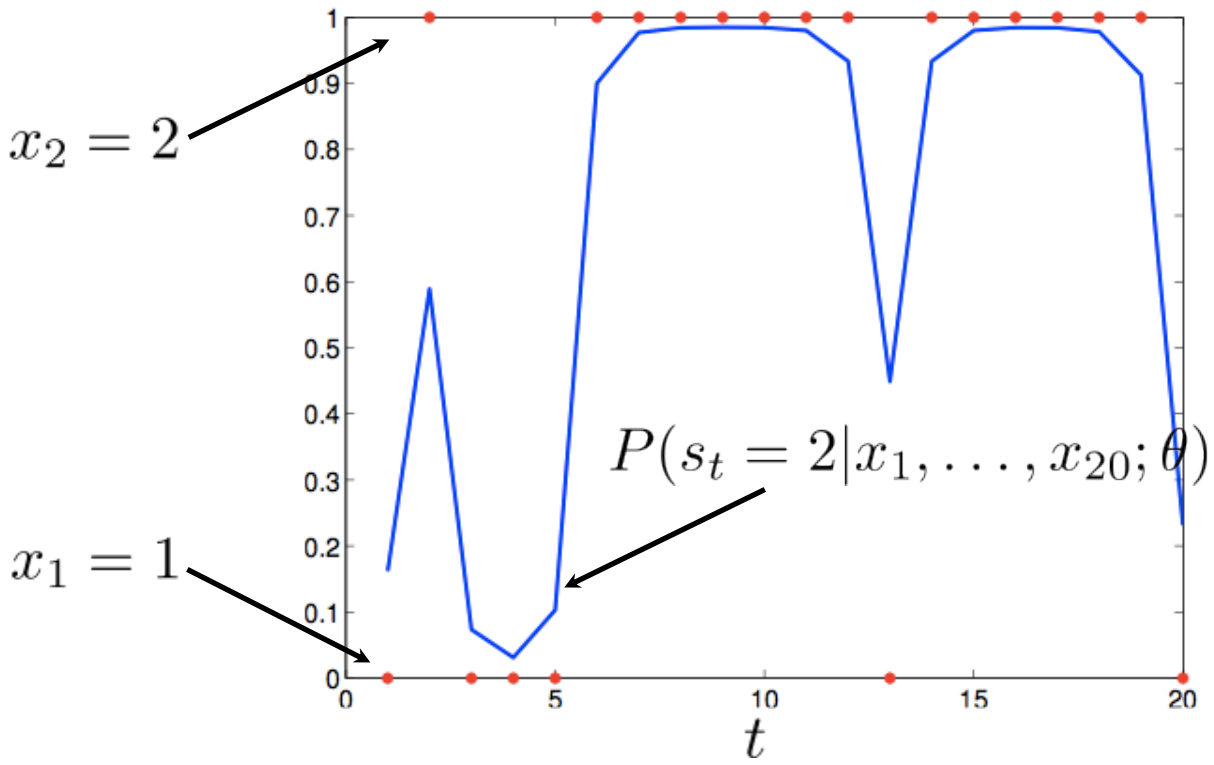
1) How to evaluate the probability of any observation sequence $P(x_1, \ldots, x_N; \theta)$
2) How to estimate the HMM parameters $\{q(s_1)\}$, $\{q(s_i|s_{i-1})\}$ and $\{q_e(x_i|s_i)\}$ on the basis of n observed sequences of varying length
3) How to find the most likely hidden state sequence $\hat{s}_1, \ldots, \hat{s}_N$ corresponding to observations $x_1, \ldots, x_N$

HMM example

- Underlying HMM

$$q(s_1) : \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}, \quad q(s_t|s_{t-1}) : \begin{bmatrix} 0.75 & 0.25 \\ 0.25 & 0.75 \end{bmatrix}$$

$$q_e(x_t|s_t) : \begin{bmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{bmatrix}$$



HMM problems: common techniques

1) Compute probability of an observation sequence: Baum-Welch, forward-backward, alpha-beta algorithms.
   - Iterative, two-stage message passing algorithms
2) Estimate HMM parameters: apply EM to iteratively maximize the likelihood of the training data given the parameters.
   - Relies on some quantities computed in 1)

3) Compute most likely hidden state sequence for an observation sequence: Viterbi Algorithm: uses dynamic programming.

Optimal substructure:

- Due to the Markov property: the most probable path (through the hidden states) for the rest of any sequence, only depends on the state in which it starts.

→ For each state, only need to keep track of most probable path that ends in that state (not all possible paths to that state).