

# Bayesian Learning

- Bayes Theorem
- MAP, ML hypotheses
- MAP learners
- Minimum description length principle
- Bayes optimal classifier
- Naive Bayes learner
- Example: Learning over text data
- Bayesian belief networks
- Expectation Maximization algorithm

## Two Roles for Bayesian Methods

Provides practical learning algorithms:

- Naive Bayes learning
- Bayesian belief network learning
- Combine prior knowledge (prior probabilities) with observed data
- Requires prior probabilities

Provides useful conceptual framework

- Provides “gold standard” for evaluating other learning algorithms
- Additional insight into Occam’s razor

## Bayes Theorem

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

- $P(h)$  = prior probability of hypothesis  $h$
- $P(D)$  = prior probability of training data  $D$
- $P(h|D)$  = probability of  $h$  given  $D$
- $P(D|h)$  = probability of  $D$  given  $h$

## Choosing Hypotheses

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

Generally want the most probable hypothesis given the training data

*Maximum a posteriori* hypothesis  $h_{MAP}$ :

$$\begin{aligned} h_{MAP} &= \arg \max_{h \in H} P(h|D) \\ &= \arg \max_{h \in H} \frac{P(D|h)P(h)}{P(D)} \\ &= \arg \max_{h \in H} P(D|h)P(h) \end{aligned}$$

If assume  $P(h_i) = P(h_j)$  then can further simplify, and choose the *Maximum likelihood* (ML) hypothesis

$$h_{ML} = \arg \max_{h_i \in H} P(D|h_i)$$

## Basic Formulas for Probabilities

- *Product Rule*: probability  $P(A \wedge B)$  of a conjunction of two events A and B:

$$P(A \wedge B) = P(A|B)P(B) = P(B|A)P(A)$$

- *Sum Rule*: probability of a disjunction of two events A and B:

$$P(A \vee B) = P(A) + P(B) - P(A \wedge B)$$

- *Theorem of total probability*: if events  $A_1, \dots, A_n$  are mutually exclusive with  $\sum_{i=1}^n P(A_i) = 1$ , then

$$P(B) = \sum_{i=1}^n P(B|A_i)P(A_i)$$

## Brute Force MAP Hypothesis Learner

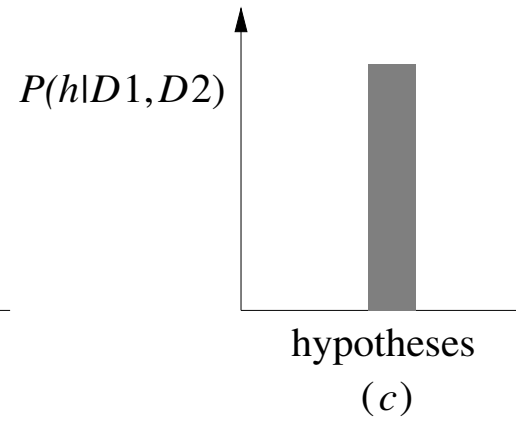
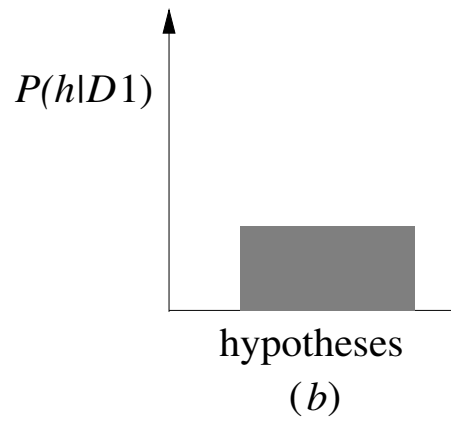
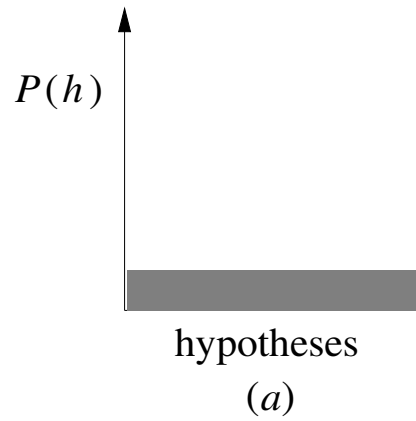
1. For each hypothesis  $h$  in  $H$ , calculate the posterior probability

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

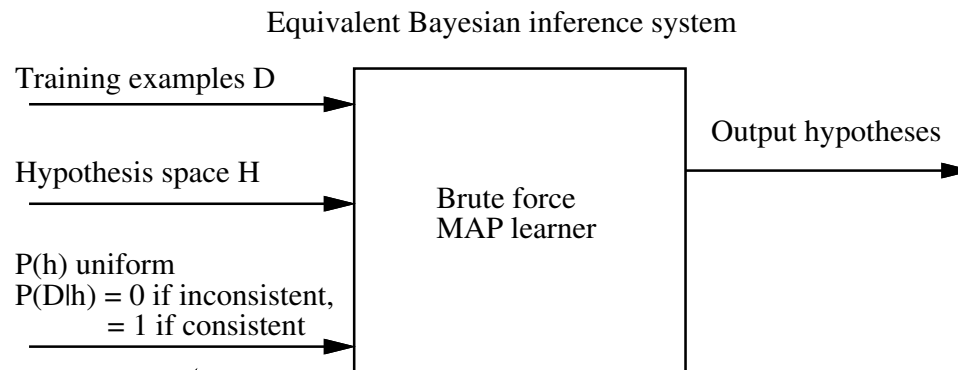
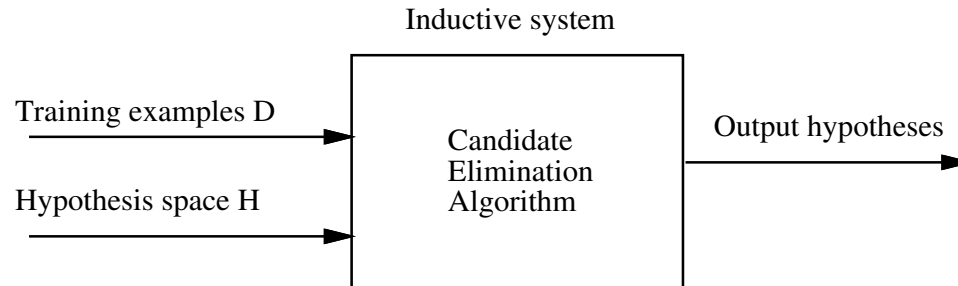
2. Output the hypothesis  $h_{MAP}$  with the highest posterior probability

$$h_{MAP} = \operatorname{argmax}_{h \in H} P(h|D)$$

# Evolution of Posterior Probabilities

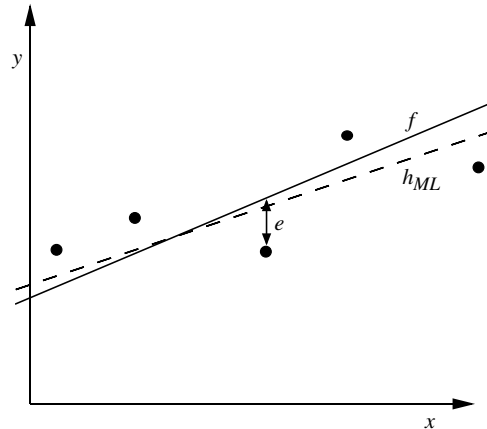


# Characterizing Learning Algorithms by Equivalent MAP Learners



*Prior assumptions  
made explicit*

## Learning A Real Valued Function



Consider any real-valued target function  $f$

Training examples  $\langle x_i, d_i \rangle$ , where  $d_i$  is noisy training value

- $d_i = f(x_i) + e_i$
- $e_i$  is random variable (noise) drawn independently for each  $x_i$  according to some Gaussian distribution with mean=0

Then the maximum likelihood hypothesis  $h_{ML}$  is the one that minimizes the sum of squared errors:

$$h_{ML} = \arg \min_{h \in H} \sum_{i=1}^m (d_i - h(x_i))^2$$

Let's Prove this!

# Learning A Real Valued Function

$$\begin{aligned}h_{ML} &= \operatorname{argmax}_{h \in H} p(D|h) \\ &= \operatorname{argmax}_{h \in H} \prod_{i=1}^m p(d_i|h) \\ &= \operatorname{argmax}_{h \in H} \prod_{i=1}^m \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{d_i - h(x_i)}{\sigma}\right)^2}\end{aligned}$$

Maximize natural log of this instead...

$$\begin{aligned}h_{ML} &= \operatorname{argmax}_{h \in H} \sum_{i=1}^m \ln \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2} \left( \frac{d_i - h(x_i)}{\sigma} \right)^2 \\ &= \operatorname{argmax}_{h \in H} \sum_{i=1}^m -\frac{1}{2} \left( \frac{d_i - h(x_i)}{\sigma} \right)^2 \\ &= \operatorname{argmax}_{h \in H} \sum_{i=1}^m - (d_i - h(x_i))^2 \\ &= \operatorname{argmin}_{h \in H} \sum_{i=1}^m (d_i - h(x_i))^2\end{aligned}$$

Proof Done!

## Learning to Predict Probabilities

Consider predicting survival probability from patient data

Training examples  $\langle x_i, d_i \rangle$ , where  $d_i$  is 1 or 0

Want to train neural network to output a *probability* given  $x_i$  (not a 0 or 1)

In this case can show

$$h_{ML} = \operatorname{argmax}_{h \in H} \sum_{i=1}^m d_i \ln h(x_i) + (1 - d_i) \ln(1 - h(x_i))$$

Weight update rule for a sigmoid unit:

$$w_{jk} \leftarrow w_{jk} + \Delta w_{jk}$$

where  $\Delta w_{jk} = \eta \sum_{i=1}^m (d_i - h(x_i)) x_{ijk}$

# Minimum Description Length Principle

Occam's razor: prefer the shortest hypothesis

MDL: prefer the hypothesis  $h$  that minimizes

$$h_{MDL} = \operatorname{argmin}_{h \in H} L_{C_1}(h) + L_{C_2}(D|h)$$

where  $L_C(x)$  is the description length of  $x$  under encoding  $C$

Example:  $H$  = decision trees,  $D$  = training data labels

- $L_{C_1}(h)$  is # bits to describe tree  $h$
- $L_{C_2}(D|h)$  is # bits to describe  $D$  given  $h$ 
  - Note  $L_{C_2}(D|h) = 0$  if examples classified perfectly by  $h$ . Need only describe exceptions
- Hence  $h_{MDL}$  trades off tree size for training errors

## Minimum Description Length Principle

$$\begin{aligned} h_{MAP} &= \arg \max_{h \in H} P(D|h)P(h) \\ &= \arg \max_{h \in H} \log_2 P(D|h) + \log_2 P(h) \\ &= \arg \min_{h \in H} -\log_2 P(D|h) - \log_2 P(h) \end{aligned} \quad (1)$$

Interesting fact from information theory:

The optimal (shortest expected coding length) code for an event with probability  $p$  is  $-\log_2 p$  bits.

So interpret (1):

- $-\log_2 P(h)$  is length of  $h$  under optimal code
- $-\log_2 P(D|h)$  is length of  $D$  given  $h$  for optimal code

**prefer to minimize:**  $length(h) + length(misclassifications)$

## Most Probable Classification of New Instances

So far we've sought the most probable *hypothesis* given the data  $D$  (i.e.,  $h_{MAP}$ )

Given new instance  $x$ , what is its most probable *classification*?

- $h_{MAP}(x)$  is not the most probable classification!

Consider:

- Three possible hypotheses:

$$P(h_1|D) = .4, P(h_2|D) = .3, P(h_3|D) = .3$$

- Given new instance  $x$ ,

$$h_1(x) = +, h_2(x) = -, h_3(x) = -$$

- What's most probable classification of  $x$ ?

# Bayes Optimal Classifier

**Bayes optimal classification:**

$$\arg \max_{v_j \in V} \sum_{h_i \in H} P(v_j | h_i) P(h_i | D)$$

**Example:**

$$P(h_1 | D) = .4, P(- | h_1) = 0, P(+ | h_1) = 1$$

$$P(h_2 | D) = .3, P(- | h_2) = 1, P(+ | h_2) = 0$$

$$P(h_3 | D) = .3, P(- | h_3) = 1, P(+ | h_3) = 0$$

therefore

$$\sum_{h_i \in H} P(+|h_i)P(h_i|D) = .4$$

$$\sum_{h_i \in H} P(-|h_i)P(h_i|D) = .6$$

and

$$\arg \max_{v_j \in V} \sum_{h_i \in H} P(v_j|h_i)P(h_i|D) = -$$

## Naive Bayes Classifier

Assume target function  $f : X \rightarrow V$ , where each instance  $x$  described by attributes  $\langle a_1, a_2 \dots a_n \rangle$ .

Most probable value of  $f(x)$  is:

$$v_{MAP} = \operatorname{argmax}_{v_j \in V} P(v_j | a_1, a_2 \dots a_n)$$

$$\begin{aligned} v_{MAP} &= \operatorname{argmax}_{v_j \in V} \frac{P(a_1, a_2 \dots a_n | v_j) P(v_j)}{P(a_1, a_2 \dots a_n)} \\ &= \operatorname{argmax}_{v_j \in V} P(a_1, a_2 \dots a_n | v_j) P(v_j) \end{aligned}$$

Naive Bayes assumption:

$$P(a_1, a_2 \dots a_n | v_j) = \prod_i P(a_i | v_j)$$

which gives

**Naive Bayes classifier:**  $v_{NB} = \operatorname{argmax}_{v_j \in V} P(v_j) \prod_i P(a_i | v_j)$

# Naive Bayes Algorithm

Naive\_Bayes\_Learn(*examples*)

For each target value  $v_j$

$$\hat{P}(v_j) \leftarrow \text{estimate } P(v_j)$$

For each attribute value  $a_i$  of each attribute  $a$

$$\hat{P}(a_i|v_j) \leftarrow \text{estimate } P(a_i|v_j)$$

Classify\_New\_Instance( $x$ )

$$v_{NB} = \operatorname{argmax}_{v_j \in V} \hat{P}(v_j) \prod_{a_i \in x} \hat{P}(a_i|v_j)$$

## Naive Bayes: Example

Consider *PlayTennis* again, and new instance

$\langle \text{Outlk} = \text{sun}, \text{Temp} = \text{cool}, \text{Humid} = \text{high}, \text{Wind} = \text{strong} \rangle$

Want to compute:

$$v_{NB} = \operatorname{argmax}_{v_j \in V} P(v_j) \prod_i P(a_i | v_j)$$

$$P(y) P(\text{sun}|y) P(\text{cool}|y) P(\text{high}|y) P(\text{strong}|y) = .005$$

$$P(n) P(\text{sun}|n) P(\text{cool}|n) P(\text{high}|n) P(\text{strong}|n) = .021$$

$$\rightarrow v_{NB} = n$$

## Naive Bayes: Subtleties

1. Conditional independence assumption is often violated

$$P(a_1, a_2 \dots a_n | v_j) = \prod_i P(a_i | v_j)$$

- ...but it works surprisingly well anyway. Note don't need estimated posteriors  $\hat{P}(v_j | x)$  to be correct; need only that

$$\operatorname{argmax}_{v_j \in V} \hat{P}(v_j) \prod_i \hat{P}(a_i | v_j) = \operatorname{argmax}_{v_j \in V} P(v_j) P(a_1 \dots, a_n | v_j)$$

- see [Domingos & Pazzani, 1996] for analysis
- Naive Bayes posteriors often unrealistically close to 1 or 0

## Naive Bayes: Subtleties

2. what if none of the training instances with target value  $v_j$  have attribute value  $a_i$ ? Then

$$\hat{P}(a_i|v_j) = 0, \text{ and...}$$

$$\hat{P}(v_j) \prod_i \hat{P}(a_i|v_j) = 0$$

Typical solution is Bayesian estimate for  $\hat{P}(a_i|v_j)$

$$\hat{P}(a_i|v_j) \leftarrow \frac{n_c + mp}{n + m}$$

where

- $n$  is number of training examples for which  $v = v_j$ ,
- $n_c$  number of examples for which  $v = v_j$  and  $a = a_i$
- $p$  is prior estimate for  $\hat{P}(a_i|v_j)$
- $m$  is weight given to prior (i.e. number of “virtual” examples)

# Bayesian Belief Networks

Interesting because:

- Naive Bayes assumption of conditional independence too restrictive
  - But it's intractable without some such assumptions...
  - Bayesian Belief networks describe conditional independence among *subsets* of variables
- allows combining prior knowledge about (in)dependencies among variables with observed training data

(also called Bayes Nets)

## Conditional Independence

**Definition:**  $X$  is *conditionally independent* of  $Y$  given  $Z$  if the probability distribution governing  $X$  is independent of the value of  $Y$  given the value of  $Z$ ; that is, if

$$(\forall x_i, y_j, z_k) P(X = x_i | Y = y_j, Z = z_k) = P(X = x_i | Z = z_k)$$

more compactly, we write

$$P(X|Y, Z) = P(X|Z)$$

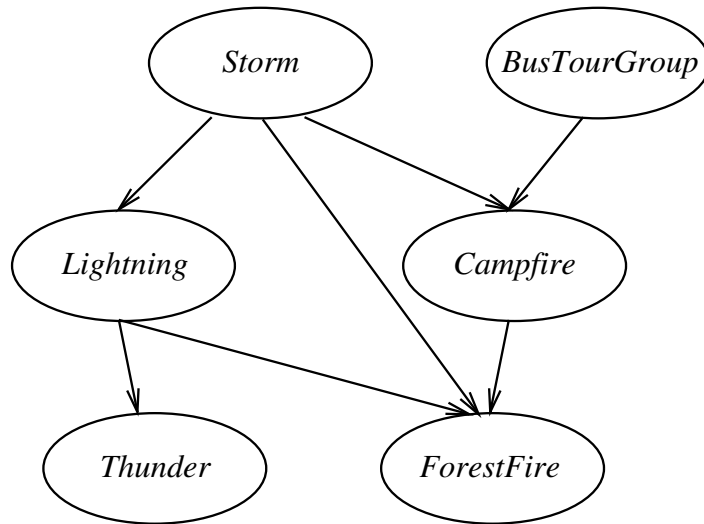
Example: *Thunder* is conditionally independent of *Rain*, given *Lightning*

$$P(\textit{Thunder}|\textit{Rain}, \textit{Lightning}) = P(\textit{Thunder}|\textit{Lightning})$$

Naive Bayes uses cond. indep. to justify

$$\begin{aligned} P(X, Y|Z) &= P(X|Y, Z)P(Y|Z) \\ &= P(X|Z)P(Y|Z) \end{aligned}$$

# Bayesian Belief Network



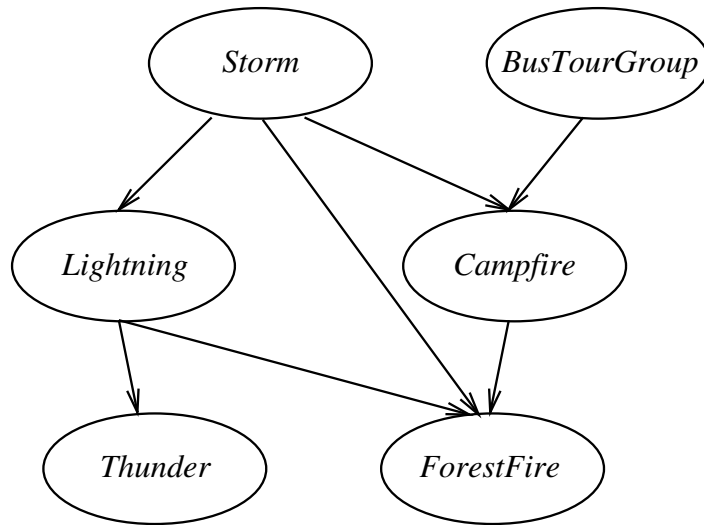
	$S, B$	$S, \neg B$	$\neg S, B$	$\neg S, \neg B$
$C$	0.4	0.1	0.8	0.2
$\neg C$	0.6	0.9	0.2	0.8



Network represents a set of conditional independence assertions:

- Nodes are asserted to be conditionally independent of nondescendants, given their immediate predecessors.
- Directed acyclic graph

# Bayesian Belief Network



	$S, B$	$S, \neg B$	$\neg S, B$	$\neg S, \neg B$
$C$	0.4	0.1	0.8	0.2
$\neg C$	0.6	0.9	0.2	0.8



Represents joint probability distribution over all variables

- e.g.,  $P(\text{Storm}, \text{BusTourGroup}, \dots, \text{ForestFire})$

- in general,

$$P(y_1, \dots, y_n) = \prod_{i=1}^n P(y_i | Parents(Y_i))$$

where  $Parents(Y_i)$  denotes immediate predecessors of  $Y_i$  in graph

- so, joint distribution is fully defined by graph, plus the  $P(y_i | Parents(Y_i))$

## Inference in Bayesian Networks

How can one infer the (probabilities of) values of one or more network variables, given observed values of others?

- Bayes net contains all information needed for this inference
- Single variable with unknown value is easy to infer.
- In general case, problem is NP hard

In practice, can succeed in many cases

- Exact inference methods work well for some network structures
- Monte Carlo methods “simulate” the network randomly to calculate approximate solutions

# Learning of Bayesian Networks

Several variants of this learning task

- Network structure might be *known* or *unknown*
- Training examples might provide values of *all* network variables, or just *some*

If structure known and observe all variables

- Then it's easy as training a Naive Bayes classifier

## Learning Bayes Nets

Suppose structure known, variables partially observable

e.g., observe *ForestFire*, *Storm*, *BusTourGroup*, *Thunder*, but not *Lightning*, *Campfire*...

- Similar to training neural network with hidden units
- In fact, can learn network conditional probability tables using gradient ascent!
- Converge to network  $h$  that (locally) maximizes  $P(D|h)$

## Gradient Ascent for Bayes Nets

Let  $w_{ijk}$  denote one entry in the conditional probability table for variable  $Y_i$  in the network

$$w_{ijk} = P(Y_i = y_{ij} | Parents(Y_i) = \text{the list } u_{ik} \text{ of values})$$

e.g., if  $Y_i = \text{Campfire}$ , then  $u_{ik}$  might be  $\langle \text{Storm} = T, \text{BusTourGroup} = F \rangle$

Perform gradient ascent by repeatedly

1. update all  $w_{ijk}$  using training data  $D$

$$w_{ijk} \leftarrow w_{ijk} + \eta \sum_{d \in D} \frac{P_h(y_{ij}, u_{ik} | d)}{w_{ijk}}$$

2. then, renormalize the  $w_{ijk}$  to assure

- $\sum_j w_{ijk} = 1$
- $0 \leq w_{ijk} \leq 1$

## More on Learning Bayes Nets

EM algorithm can also be used. Repeatedly:

1. Calculate probabilities of unobserved variables, assuming  $h$
2. Calculate new  $w_{ijk}$  to maximize  $E[\ln P(D|h)]$  where  $D$  now includes both observed and (calculated probabilities of) unobserved variables

When structure unknown...

- Algorithms use greedy search to add/subtract edges and nodes
- Active research topic

## Summary: Bayesian Belief Networks

- Combine prior knowledge with observed data
- Impact of prior knowledge (when correct!) is to lower the sample complexity
- Active research area
  - Extend from boolean to real-valued variables
  - Parameterized distributions instead of tables
  - Extend to first-order instead of propositional systems
  - More effective inference methods
  - ...

## Expectation Maximization (EM)

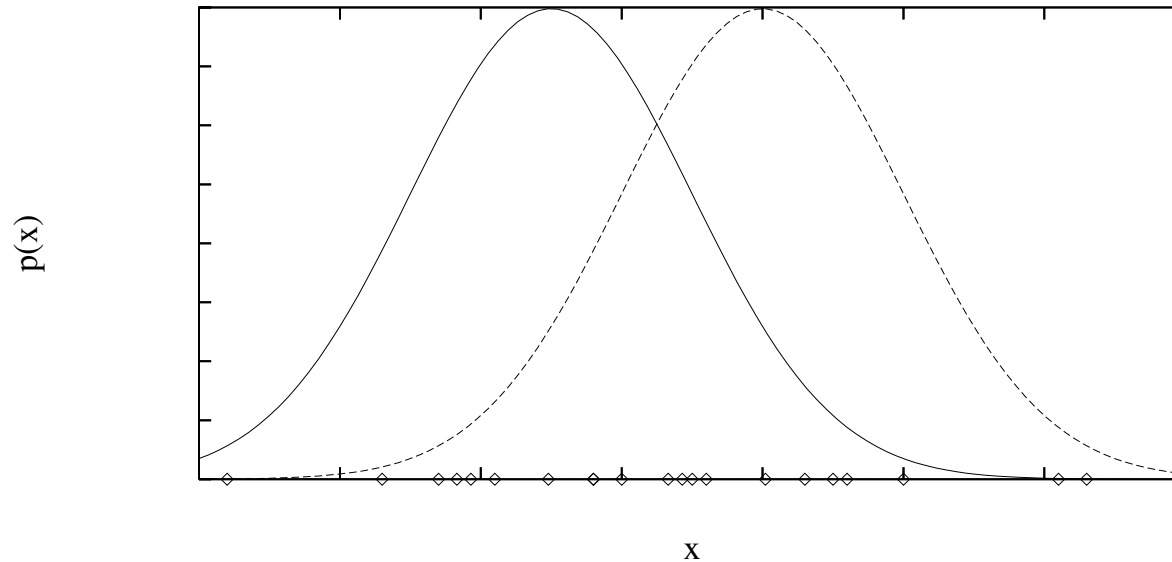
When to use:

- Data is only partially observable
- Unsupervised clustering (target value unobservable)
- Supervised learning (some instance attributes unobservable)

Some uses:

- Train Bayesian Belief Networks
- Unsupervised clustering (AUTOCLASS)
- Learning Hidden Markov Models

## Generating Data from Mixture of $k$ Gaussians



Each instance  $x$  generated by

1. Choosing one of the  $k$  Gaussians with uniform probability
2. Generating an instance at random according to that Gaussian

## EM for Estimating $k$ Means

Given:

- Instances from  $X$  generated by mixture of  $k$  Gaussian distributions
- Unknown means  $\langle \mu_1, \dots, \mu_k \rangle$  of the  $k$  Gaussians
- Don't know which instance  $x_i$  was generated by which Gaussian

Determine:

- Maximum likelihood estimates of  $\langle \mu_1, \dots, \mu_k \rangle$

Think of full description of each instance as

$y_i = \langle x_i, z_{i1}, z_{i2} \rangle$ , where

- $z_{ij}$  is 1 if  $x_i$  generated by  $j$ th Gaussian
- $x_i$  observable
- $z_{ij}$  unobservable

## EM for Estimating $k$ Means

EM Algorithm: Pick random initial  $h = \langle \mu_1, \mu_2 \rangle$ , then iterate

**E step:** Calculate the expected value  $E[z_{ij}]$  of each hidden variable  $z_{ij}$ , assuming the current hypothesis  $h = \langle \mu_1, \mu_2 \rangle$  holds.

$$\begin{aligned} E[z_{ij}] &= \frac{p(x = x_i | \mu = \mu_j)}{\sum_{n=1}^2 p(x = x_i | \mu = \mu_n)} \\ &= \frac{e^{-\frac{1}{2\sigma^2}(x_i - \mu_j)^2}}{\sum_{n=1}^2 e^{-\frac{1}{2\sigma^2}(x_i - \mu_n)^2}} \end{aligned}$$

**M step:** Calculate a new maximum likelihood hypothesis  $h' = \langle \mu'_1, \mu'_2 \rangle$ , assuming the value taken on by each hidden variable  $z_{ij}$  is its expected value  $E[z_{ij}]$  calculated above. Replace  $h = \langle \mu_1, \mu_2 \rangle$  by  $h' = \langle \mu'_1, \mu'_2 \rangle$ .

$$\mu_j \leftarrow \frac{\sum_{i=1}^m E[z_{ij}] x_i}{\sum_{i=1}^m E[z_{ij}]}$$

## EM Algorithm

Converges to local maximum likelihood  $h$   
and provides estimates of hidden variables  $z_{ij}$

In fact, local maximum in  $E[\ln P(Y|h)]$

- $Y$  is complete (observable plus unobservable variables) data
- Expected value is taken over possible values of unobserved variables in  $Y$

## General EM Problem

Given:

- Observed data  $X = \{x_1, \dots, x_m\}$
- Unobserved data  $Z = \{z_1, \dots, z_m\}$
- Parameterized probability distribution  $P(Y|h)$ , where
  - $Y = \{y_1, \dots, y_m\}$  is the full data  $y_i = x_i \cup z_i$
  - $h$  are the parameters

Determine:

- $h$  that (locally) maximizes  $E[\ln P(Y|h)]$

Many uses:

- Train Bayesian belief networks
- Unsupervised clustering (e.g.,  $k$  means)
- Hidden Markov Models

## General EM Method

Define likelihood function  $Q(h'|h)$  which calculates  $Y = X \cup Z$  using observed  $X$  and current parameters  $h$  to estimate  $Z$

$$Q(h'|h) \leftarrow E[\ln P(Y|h')|h, X]$$

## EM Algorithm:

*Estimation (E) step:* Calculate  $Q(h'|h)$  using the current hypothesis  $h$  and the observed data  $X$  to estimate the probability distribution over  $Y$ .

$$Q(h'|h) \leftarrow E[\ln P(Y|h')|h, X]$$

*Maximization (M) step:* Replace hypothesis  $h$  by the hypothesis  $h'$  that maximizes this  $Q$  function.

$$h \leftarrow \operatorname{argmax}_{h'} Q(h'|h)$$