CSCE 496/896 Lecture 8: Bayesian Learning

Stephen D. Scott
(Adapted from Tom Mitchell's slides)

October 25, 1999

Bayesian Methods

Not all hypotheses are created equal (even if they are all consistent with the training data)

Might have reasons (domain information) to favor some hypotheses over others *a priori*

Bayesian methods work with probabilities, and have two main roles:

1. Provide practical learning algorithms:
   - Naive Bayes learning
   - Bayesian belief network learning
   - Combine prior knowledge (prior probabilities) with observed data
   - Requires prior probabilities
2. Provides useful conceptual framework
   - Provides "gold standard" for evaluating other learning algorithms
   - Additional insight into Occam's razor

Outline

- Bayes Theorem
- MAP, ML hypotheses
- MAP learners
- Minimum description length principle
- Bayes optimal classifier/Gibbs algorithm
- Naive Bayes classifier
- Bayesian belief networks
- Expectation Maximization algorithm

Bayes Theorem

In general, an identity for conditional probabilities

For our work, we want to know the probability that a particular $h \in H$ is the correct hypothesis given that we have seen training data D (examples and labels). Bayes theorem lets us do this.

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

- $P(h)$ = prior probability of hypothesis h (might include domain information)
- $P(D)$ = prior probability of training data D
- $P(h \mid D)$ = probability of h given D
- $P(D \mid h)$ = probability of D given h

Note $P(h \mid D)$ increases with $P(D \mid h)$ and $P(h)$ and decreases with $P(D)$

Choosing Hypotheses

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

Generally want the most probable hypothesis given the training data

Maximum a posteriori hypothesis h_{MAP}:

$$\begin{aligned} h_{MAP} &= \underset{h \in H}{\text{argmax}}\, P(h \mid D) \\ &= \underset{h \in H}{\text{argmax}} \frac{P(D \mid h)P(h)}{P(D)} \\ &= \underset{h \in H}{\text{argmax}}\, P(D \mid h)P(h) \end{aligned}$$

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

$$h_{ML} = \underset{h_i \in H}{\text{argmax}}\, P(D \mid h_i)$$

Bayes Theorem Example

Does patient have cancer or not?

> A patient takes a lab test and the result comes back positive. The test returns a correct positive result in only 98% of the cases in which the disease is actually present, and a correct negative result in only 97% of the cases in which the disease is not present. Furthermore, .008 of the entire population have this cancer.

$P(cancer) =$ $\quad$ $P(\neg cancer) =$

$P(+ \mid cancer) =$ $\quad$ $P(- \mid cancer) =$

$P(+ \mid \neg cancer) =$ $\quad$ $P(- \mid \neg cancer) =$

Now consider new patient for whom the test is positive. What is our diagnosis?

$P(+ \mid cancer)P(cancer) =$

$P(+ \mid \neg cancer)P(\neg cancer) =$

So $h_{MAP} =$

Basic Formulas for Probabilities

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

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

- *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 $A_1, \ldots, A_n$ are mutually exclusive with $\sum_{i=1}^{n} P(A_i) = 1$, then

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

Brute Force MAP Hypothesis Learner

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

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

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

$$h_{MAP} = \underset{h \in H}{\text{argmax}}\, P(h \mid D)$$

Problem: what if H exponentially or infinitely large?

Relation to Concept Learning

Consider our usual concept learning task: instance space X, hypothesis space H, training examples D

Consider the Find-S learning algorithm (outputs most specific hypothesis from the version space $VS_{H,D}$)

What would brute-force MAP learner output as MAP hypothesis?

Does Find-S output a MAP hypothesis??

Relation to Concept Learning (cont'd)

Assume fixed set of instances $\langle x_1, \ldots, x_m \rangle$

Assume D is the set of classifications $D = \langle c(x_1), \ldots, c(x_m) \rangle$

Assume no noise and $c \in H$, so choose

$$P(D \mid h) = \begin{cases} 1 & \text{if } d_i = h(x_i) \text{ for all } d_i \in D \\ 0 & \text{otherwise} \end{cases}$$

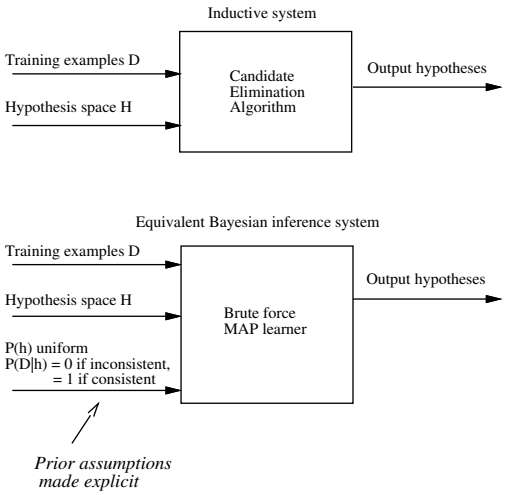
Choose $P(h) = 1/|H| \ \forall h \in H$, i.e. uniform dist.

If h inconsistent with D, then
$P(h \mid D) = (0 \cdot P(h)) / P(D) = 0$

If h consistent with D, then
$P(h \mid D) = (1 \cdot 1/|H|) / P(D) = (1/|H|) / \left(|VS_{H,D}|/|H|\right)$
$= 1/|VS_{H,D}|$ (see Thrm of total prob., slide 7)

Thus if D noise-free and $c \in H$ and $P(h)$ uniform, every consistent hypothesis is a MAP hypothesis

Characterizing Learning Algorithms by Equivalent MAP Learners



So we can characterize algorithms in a Bayesian framework even though they don't directly manipulate probabilities

Other priors will allow Find-S, etc. to output MAP; e.g. $P(h)$ that favors more specific hypotheses

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 $\mu_{e_i} = 0$

Then the maximum likelihood hypothesis h_{ML} is the one that minimizes the sum of squared errors, e.g. a linear unit trained with GD/EG:

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

Learning A Real Valued Function
(cont'd)

$$
\begin{aligned}
h_{ML} &= \underset{h \in H}{\text{argmax}}\ p(D \mid h) \\
&= \underset{h \in H}{\text{argmax}} \prod_{i=1}^{m} p(d_i \mid h) \quad \text{(if } d_i\text{'s cond. indep.)} \\
&= \underset{h \in H}{\text{argmax}} \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\left(\frac{d_i - h(x_i)}{\sigma}\right)^2\right)
\end{aligned}
$$

$(\mu_{e_i} = 0 \Rightarrow \mathsf{E}[d_i \mid h] = h(x_i))$

Maximize natural log instead:

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

Thus have Bayesian justification for minimizing squared error (under certain assumptions)

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 (assume label is [or appears] probabilistically generated)

Want to train neural network to output the probability that x_i has label 1, not the label itself

Using approach similar to previous slide, can show

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

i.e. find h minimizing cross-entropy

For single sigmoid unit, use update rule

$$w_j \leftarrow w_j + \eta \sum_{i=1}^{m} (d_i - h(x_i))\ x_{ij}$$

to find h_{ML} (can also derive EG rule)

Minimum Description Length Principle

Occam's razor: prefer the shortest hypothesis

MDL: prefer the hypothesis h that satisfies

$$h_{MDL} = \underset{h \in H}{\text{argmin}}\ L_{C_1}(h) + L_{C_2}(D \mid h)$$

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

Minimum Description Length Principle
Bayesian Justification

$$
\begin{aligned}
h_{MAP} &= \underset{h \in H}{\text{argmax}}\ P(D \mid h) P(h) \\
&= \underset{h \in H}{\text{argmax}}\ \log_2 P(D \mid h) + \log_2 P(h) \\
(1) \qquad &= \underset{h \in H}{\text{argmin}}\ -\log_2 P(D \mid h) - \log_2 P(h)
\end{aligned}
$$

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

$\rightarrow$ prefer the hypothesis that minimizes

$$length(h) + length(misclassifications)$$

Caveat: $h_{MDL} = h_{MAP}$ doesn't apply for arbitrary encodings (need $P(h)$ and $P(D \mid h)$ to be optimal); merely a guide

Bayes Optimal Classifier

- So far we've sought the most probable hypothesis given the data D, i.e. h_{MAP}
- But given new instance x, $h_{MAP}(x)$ is not the most probable classification!
- Consider three possible hypotheses:

  $P(h_1 \mid D) = 0.4,\ P(h_2 \mid D) = 0.3,\ P(h_3 \mid D) = 0.3$

  Given new instance x,

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

- $h_{MAP}(x) =$
- What's the most probable classification of x?

Bayes Optimal Classifier
(cont'd)

Bayes optimal classification:

$$\underset{v_j \in V}{\text{argmax}} \sum_{h_i \in H} P(v_j \mid h_i) P(h_i \mid D)$$

where V is set of possible labels (e.g. $\{+, -\}$)

Example:

$P(h_1 \mid D) = 0.4,\ \ P(- \mid h_1) = 0,\ \ P(+ \mid h_1) = 1$
$P(h_2 \mid D) = 0.3,\ \ P(- \mid h_2) = 1,\ \ P(+ \mid h_2) = 0$
$P(h_3 \mid D) = 0.3,\ \ P(- \mid h_3) = 1,\ \ P(+ \mid h_3) = 0$

therefore

$$\sum_{h_i \in H} P(+ \mid h_i) P(h_i \mid D) = 0.4$$
$$\sum_{h_i \in H} P(- \mid h_i) P(h_i \mid D) = 0.6$$

and

$$\underset{v_j \in V}{\text{argmax}} \sum_{h_i \in H} P(v_j \mid h_i) P(h_i \mid D) = -$$

On avg, no other classifier using same prior and same hyp. space can outperform Bayes optimal!

Also works for nondeterministic labels (hyps)

Gibbs Algorithm

Bayes optimal classifier provides best result, but can be expensive or impossible if many hypotheses

Gibbs algorithm:

1. Randomly choose one hyp, according to $P(h \mid D)$
2. Use this to classify new instance

Surprising fact: Assume target concepts are drawn at random from H according to priors on H. Then:

$$\mathrm{E}\left[error_{Gibbs}\right] \leq 2\,\mathrm{E}\left[error_{Bayes\ Optimal}\right]$$

i.e. if prior correct and $c \in H$, then avg error at most twice best possible!

E.g. Suppose correct, uniform prior distribution over H. Then

- Pick any hyp from VS with uniform prob.
- Expected error no worse than twice Bayes opt

Still have to be able to choose random hyp!

Naive Bayes Classifier

Along with decision trees, neural networks, nearest nbr, one of the most practical learning methods

When to use

- Moderate or large training set available
- Attributes that describe instances are conditionally independent given classification

Successful applications:

- Diagnosis
- Classifying text documents

Naive Bayes Classifier
(cont'd)

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

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

$$\begin{aligned} v_{MAP} &= \underset{v_j \in V}{\text{argmax}}\, P(v_j \mid a_1, a_2, \ldots, a_n) \\ &= \underset{v_j \in V}{\text{argmax}}\, \frac{P(a_1, a_2, \ldots, a_n \mid v_j)\, P(v_j)}{P(a_1, a_2, \ldots, a_n)} \\ &= \underset{v_j \in V}{\text{argmax}}\, P(a_1, a_2, \ldots, a_n \mid v_j)\, P(v_j) \end{aligned}$$

Problem with estimating probs from training data: estimating $P(v_j)$ easily done by counting, but there are exponentially (in n) many combs. of values of $a_1, \ldots, a_n$, so can't get estimates for most combs

Naive Bayes assumption:

$$P(a_1, a_2, \ldots, a_n \mid v_j) = \prod_i P(a_i \mid v_j)$$

so Naive Bayes classifier:

$$v_{NB} = \underset{v_j \in V}{\text{argmax}}\, P(v_j) \prod_i P(a_i \mid v_j)$$

Now have only polynomial number of probs to est

Naive Bayes Algorithm

Naive_Bayes_Learn

1. For each target value v_j
   (a) $\hat{P}(v_j) \leftarrow$ estimate $P(v_j)$
   (b) For each attribute value a_i of each attrib a
      i. $\hat{P}(a_i \mid v_j) \leftarrow$ estimate $P(a_i \mid v_j)$

Classify_New_Instance(x)

$$v_{NB} = \underset{v_j \in V}{\text{argmax}}\, \hat{P}(v_j) \prod_{a_i \in x} \hat{P}(a_i \mid v_j)$$

Naive Bayes Example

Training Examples (Table 3.2):

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Example to classify:

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

Assign label $v_{NB} = \text{argmax}_{v_j \in V}\ P(v_j) \prod_i P(a_i \mid v_j)$

$P(y) \cdot P(sun \mid y) \cdot P(cool \mid y) \cdot P(high \mid y) \cdot P(strong \mid y)$

$= (9/14) \cdot (2/9) \cdot (3/9) \cdot (3/9) \cdot (3/9) = 0.0053$

$P(n)\ P(sun \mid n)\ P(cool \mid n)\ P(high \mid n)\ P(strong \mid n)$

$= (5/14) \cdot (3/5) \cdot (1/5) \cdot (4/5) \cdot (3/5) = 0.0206$

So $v_{NB} = n$

Naive Bayes
Subtleties

- Conditional independence assumption is often violated, i.e.

$$P(a_1, a_2, \ldots, a_n \mid v_j) \neq \prod_i P(a_i \mid v_j)$$

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

$$\underset{v_j \in V}{\text{argmax}}\, \hat{P}(v_j) \prod_i \hat{P}(a_i \mid v_j) = \underset{v_j \in V}{\text{argmax}}\, P(v_j) P(a_1, \ldots, a_n \mid v_j)$$

Sufficient conditions given in [Domingos & Pazzani, 1996]

Naive Bayes
Subtleties (cont'd)

- What if none of the training instances with target value v_j have attribute value a_i? Then

$$\hat{P}(a_i \mid v_j) = 0, \text{ and } \hat{P}(v_j) \prod_i \hat{P}(a_i \mid v_j) = 0$$

Typical solution is to use as estimate:

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

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 $\hat{P}(a_i \mid v_j)$
- m is weight given to prior (i.e. number of "virtual" examples)

Naive Bayes
Application: Learning to Classify Text

- Target concept $Interesting? : Document \rightarrow \{+, -\}$
- Each document is a vector of words (i.e. one attribute per word position), e.g. $a_1 =$ "our", $a_2 =$ "approach", etc.
- Naive Bayes very effective despite obvious violation of conditional independence assumption
- See Section 6.10 for more detail

Bayesian Belief Networks

- Sometimes Naive Bayes assumption of conditional independence too restrictive
- But inferring probabilities is intractable without some such assumptions
- Bayesian Belief networks (also called Bayes Nets) describe conditional independence among subsets of variables
- Allows combining prior knowledge about dependencies among variables with observed training data

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

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

more compactly, we write

$$P(X \mid Y, Z) = P(X \mid Z)$$

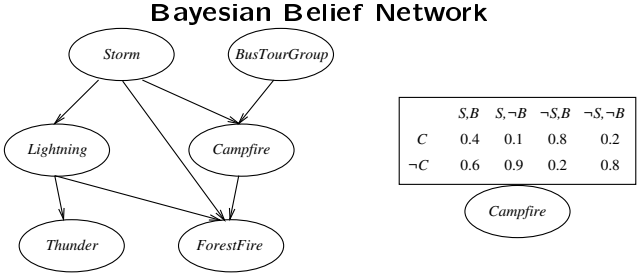
Example: $Thunder$ is conditionally independent of $Rain$, given $Lightning$

$$P(Thunder \mid Rain, Lightning) = P(Thunder \mid Lightning)$$

Naive Bayes uses cond indep and product rule (slide 7) to justify

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

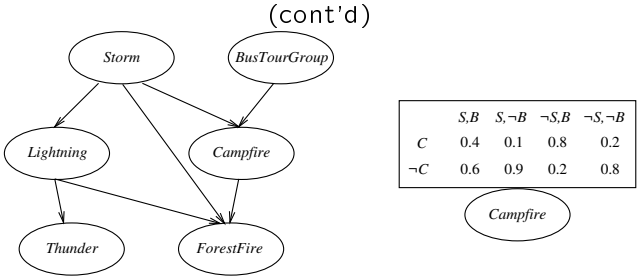
Bayesian Belief Network



Network represents a set of conditional independence assertions:

- Each node is asserted to be conditionally independent of its nondescendants, given its immediate predecessors
- Directed acyclic graph

Bayesian Belief Network (cont'd)



Represents joint probability distribution over all network variables $\langle Y_1, \ldots, Y_n \rangle$, e.g. $P(Storm, BusTourGroup, \ldots, ForestFire)$

- In general, for y_i = value of Y_i

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

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

- E.g. $P(S, B, C, \neg L, \neg T, \neg F) =$

$$P(S) \cdot P(B) \cdot \underbrace{P(C \mid B, S)}_{0.4} \cdot P(\neg L \mid S) \cdot P(\neg T \mid \neg L) \cdot P(\neg F \mid S, \neg L, \neg C)$$

(compute cond probs using product rule)

Inference in Bayesian Networks

Want to infer the probabilities of values of one or more network variables (attributes), given observed values of others, i.e. want the probability distribution of a subset of variables given the values of a subset of the other variables

- Bayes net contains all information needed for this inference: can simply brute force try all combinations of values of the unknown variables
- Of course, this takes time exponential in number of unknowns
- 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

We know how to use Bayesian Networks, but how do we learn one?

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 all variables observed, then it's as easy as training a Naive Bayes classifier (just count occurrences as before)

Learning of Bayesian Networks
(cont'd)

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 $P(D \mid h)$, i.e. search for ML hypothesis

Gradient Ascent for Bayes Nets

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

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

e.g. if $Y_i = Campfire$, then u_{ik} might be $\langle S, \neg B \rangle$

Perform gradient ascent by repeating:

1. Update all w_{ijk} using training data D

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

2. Then renormalize the w_{ijk} to assure $\sum_j w_{ijk} = 1$ and $0 \leq w_{ijk} \leq 1$

I.e. each $d \in D$ is a tuple of specified attribute values* and P_h is probability conditioned on h (current network)

E.g. $y_{ij} = C$, $u_{ik} = \langle S, \neg B \rangle$, $d_1 = (S, B, L)$, $d_2 = (L, T)$, then $P_h(y_{ij}, u_{ik} \mid d_1) = 0$, $P_h(y_{ij}, u_{ik} \mid d_2)$ is computable via procedure on Slides 30–31

*Note that in this context, any attribute could be considered a label, so each d is an example.

More on Learning Bayes Nets

- EM (expectation maximization) algorithm can also be used (more on this later)
- When structure unknown
  - Algorithms use greedy search to add/substract edges and nodes
  - Active research topic

Bayesian Belief Networks
Summary

- 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
  - More effective inference methods

Expectation Maximization (EM)

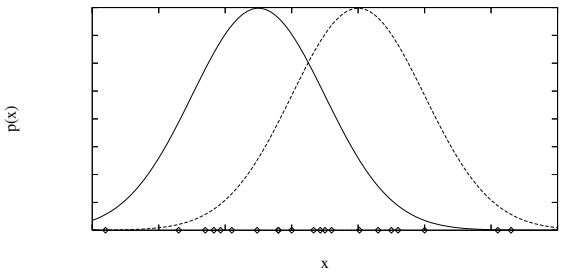
Use when data only partially observable

- Unsupervised clustering (target value unobservable)
- Supervised learning (some instance attributes unobservable)

Applications:

- 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 $\langle \mu_1, \ldots, \mu_k \rangle$ of the k Gaussians
- Don't know which instance x_i was generated by which Gaussian

Determine maximum likelihood estimates of $\langle \mu_1, \ldots, \mu_k \rangle$

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 jth Gaussian
- x_i observable
- z_{ij} unobservable

EM for Estimating k Means
(Here $k = 2$)

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

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

$$\mathrm{E}\left[z_{ij}\right] = \frac{p(x = x_i \mid \mu = \mu_j)}{\sum_{n=1}^{2} p(x = x_i \mid \mu = \mu_n)}$$

$$= \frac{e^{-\frac{1}{2\sigma^2}(x_i - \mu_j)^2}}{\sum_{n=1}^{2} e^{-\frac{1}{2\sigma^2}(x_i - \mu_n)^2}}$$

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 $\mathrm{E}\left[z_{ij}\right]$ calculated above. Replace $h = \langle \mu_1, \mu_2 \rangle$ by $h' = \langle \mu'_1, \mu'_2 \rangle$.

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

EM Algorithm

Miscellany

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

In fact, local maximum in $\mathrm{E}[\ln P(Y \mid h)]$

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

General EM Problem

Given:

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

Determine h that locally maximizes $\mathrm{E}[\ln P(Y \mid h)]$

Many uses:

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

General EM Method

Since $Y = X \cup Z$ is distributed according to parameters we want to estimate (that govern Z) and observed data X, we will use current estimates (h) in computation, adjust h, and iterate

Define likelihood function $Q(h' \mid h)$ which estimates prob distribution over Y using observed X and current parameters h to estimate Z

$$Q(h' \mid h) \leftarrow \mathrm{E}\left[\ln P(Y \mid h') \mid h, X\right]$$

EM Algorithm:

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

$$Q(h' \mid h) \leftarrow \mathrm{E}\left[\ln P(Y \mid h') \mid h, X\right]$$

Maximization (M) step: Replace hypothesis h by hypothesis h' that maximizes Q

$$h \leftarrow \underset{h'}{\operatorname{argmax}}\, Q(h' \mid h)$$

Using EM to Learn Probabilities in a Bayes Net

Repeatedly:

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