Minimum Description Length Principle

Occam's razor: prefer the shortest hypothesis

MDL: prefer the hypothesis h that minimizes

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

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

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

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

$\rightarrow$ prefer the hypothesis that minimizes

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

Gibbs Classifier

Bayes optimal classifier provides best result, but can be expensive if many hypotheses.
Gibbs algorithm:

1. Choose one hypothesis at random, according to $P(h|D)$
2. Use this to classify new instance

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

$$E[error_{Gibbs}] \leq 2E[error_{BayesOptimal}]$$

Suppose correct, uniform prior distribution over H, then

- Pick any hypothesis from VS, with uniform probability
- Its expected error no worse than twice Bayes optimal

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

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

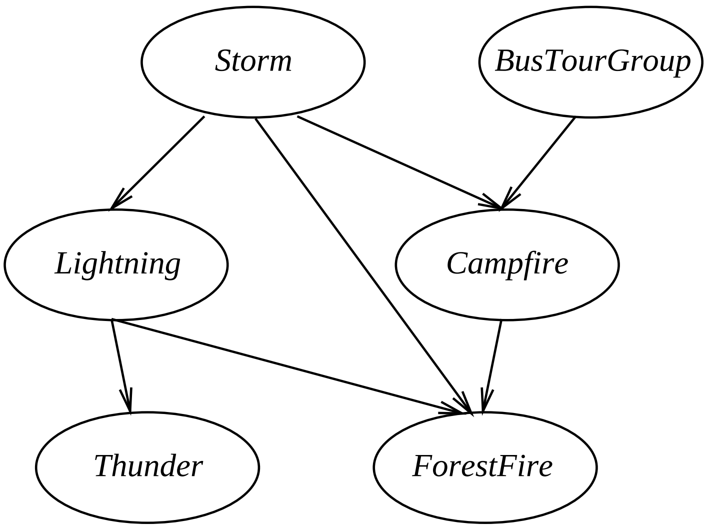
Example: $Thunder$ is conditionally independent of $Rain$, given $Lightning$

$$P(Thunder|Rain, Lightning) = P(Thunder|Lightning)$$

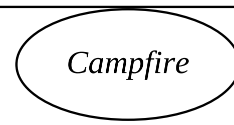
Naive Bayes uses cond. indep. to justify

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

Bayesian Belief Network



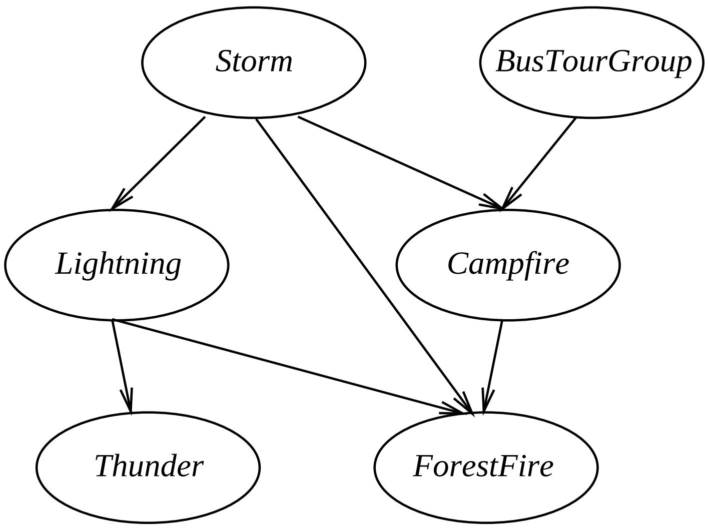
	S,B	$S,\neg B$	$\neg S,B$	$\neg S,\neg B$
C	0.4	0.1	0.8	0.2
$\neg C$	0.6	0.9	0.2	0.8



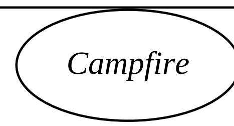
Network represents a set of conditional independence assertions:

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

Bayesian Belief Network



	S,B	*S,¬B*	*¬S,B*	*¬S,¬B*
C	0.4	0.1	0.8	0.2
¬C	0.6	0.9	0.2	0.8



Represents joint probability distribution over all variables

- e.g., $P(Storm, BusTourGroup, \ldots, ForestFire)$
- in general,
$$P(y_1, \ldots, y_n) = \prod_{i=1}^{n} P(y_i | Parents(Y_i))$$
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
- If only one variable with unknown value, easy to infer it
- 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

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

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

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/substract 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)

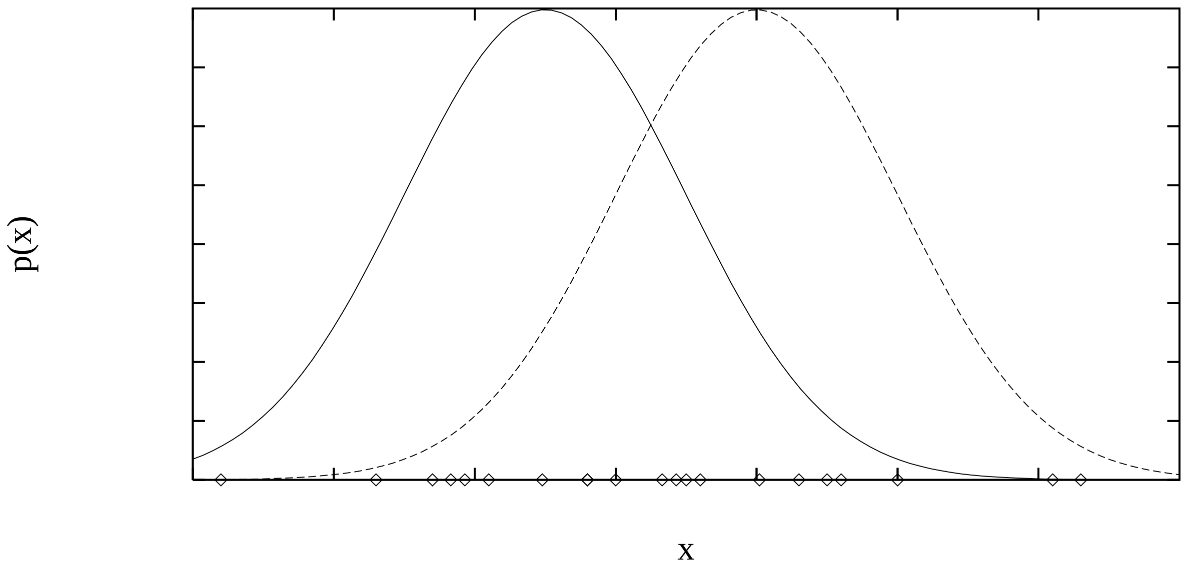
When to use:

- 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 $\langle \mu_1, \ldots, \mu_k \rangle$ 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

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.

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

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 $h' = \langle \mu'_1, \mu'_2 \rangle$.

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

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 $X = \{x_1, \ldots, x_m\}$
- Unobserved data $Z = \{z_1, \ldots, z_m\}$
- Parameterized probability distribution $P(Y|h)$, where
  - $Y = \{y_1, \ldots, y_m\}$ 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.

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