Bayesian Learning

What we know so far?

Chapter 1: Machine Learning - Definition and Design constraints.

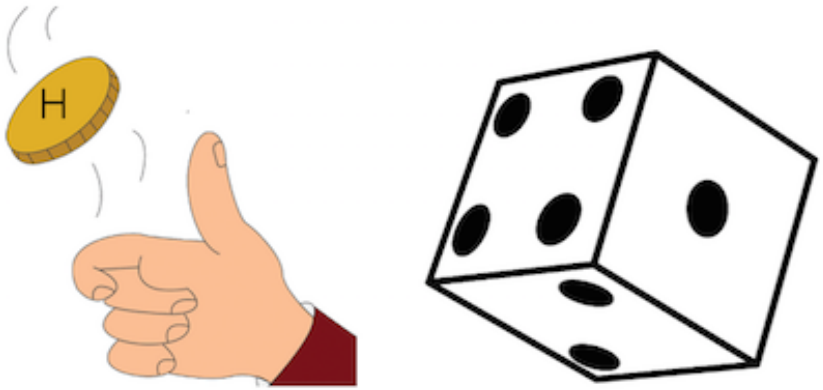
Chapter 2: Concept learning - General and specific ordering, Find - S and Candidate Elimination algorithm to find the consistent hypothesis.

Chapter 3: Decision Trees - Inductive Bias, ID3 algorithm , Overfitting.

Introduction - Definition

Bayesian inference is a method of statistical inference in which **Bayes' theorem** is used to update the **probability for a hypothesis** as more evidence or information becomes available.

The idea of **Bayesian learning** is to compute the posterior probability distribution of the target features of a new example conditioned on its input features and all of the training examples.



Head or Tail ? **Is it 6 ?**

Features

1. Each observed training example can incrementally decrease or increase the estimated probability that a hypothesis is correct
2. Prior knowledge can be combined with observed data to determine the final probability of a hypothesis
3. Bayesian methods can accommodate hypotheses that make probabilistic predictions
4. New instances can be classified by combining the predictions of multiple hypotheses, weighted by their probabilities

Computational cost and having prior knowledge are the only limitations.

Bayes Theorem

It is based on conditional probability.

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

where:

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

Example

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.Does patient have cancer or not?

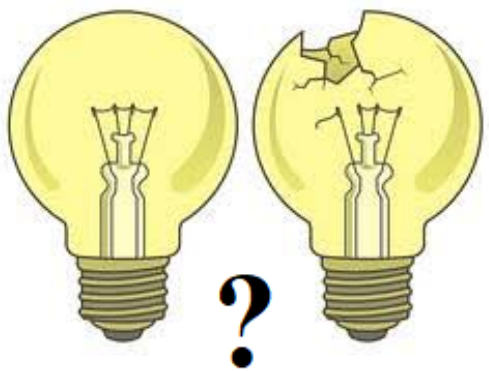
$$P(cancer) = 0.008 \qquad P(\neg cancer) = 0.992$$
$$P(+|cancer) = 0.98 \qquad P(-|cancer) = 0.02$$
$$P(+|\neg cancer) = 0.03 \qquad P(-|\neg cancer) = 0.97$$

P(+ given cancer) P(cancer) = (0.98) 0.008 = 0.0078
P(+ given $\neg cancer$)$P(\neg cancer) = (0.3)0.992 = 0.0298$
Thus $h = \neg$ cancer

Exercise

A desk lamp produced by luminar company was found to be defective (D). There are three factories (A.B.C) where such desk lamps are manufactured. The quality control manager has the information tabulated below. If he randomly selects a lamp and if it is defective, what is the probability that the lamp was manufactured by C?

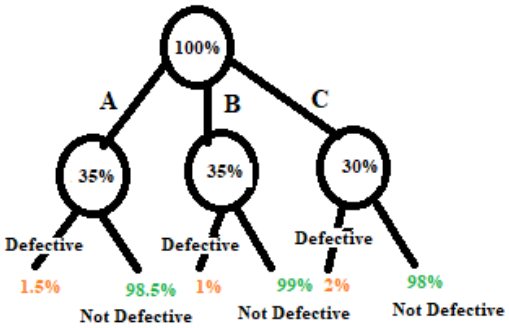
Factory	% of total production	Probability of defective lamp
A	0.35	0.015
B	0.35	0.010
C	0.3	0.020



Solution

$P(A|D) = 0.356 \qquad P(B|D) = 0.237$
$P(C|D) = 0.407$

So the defective lamp is mostly to be manufactured from C.



Basic Formulas for probability

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

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

MAP

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

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

Brute-Force Bayes Concept Learning

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

$$h_{MAP} = \arg\max_{h \in H} P(h|D)$$

Relation to concept Learning

Assumptions

- The training data D is noise free (i.e., $d_i = c(x_i)$).
- The target concept c is contained in the hypothesis space H
- We have no a priori reason to believe that any hypothesis is more probable than any other.

Choose the prior probability such that, for all h in H, it has uniform distribution

$$P(h) = \frac{1}{|H|}$$

Choose the probability of observing target values D for set of instances $(x_1..x_n)$ given the hypothesis h such that :

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

Brute-Force algorithm

First consider the case where h is inconsistent with the training data D.
Since $P(D|h)$ is 0 when h is inconsistent with D, we have

$P(h|D) = \frac{0.P(h)}{P(D)} = 0$ if h is inconsistent with D
The posterior probability of a hypothesis inconsistent with D is zero.

Now consider the case where h is consistent with D.

$$
\begin{aligned}
P(D) &= \sum_{h_i \in H} P(D|h_i) P(h_i) \\
&= \sum_{h_i \in VS_{H,D}} 1 \cdot \frac{1}{|H|} + \sum_{h_i \notin VS_{H,D}} 0 \cdot \frac{1}{|H|} \\
&= \sum_{h_i \in VS_{H,D}} 1 \cdot \frac{1}{|H|} \\
&= \frac{|VS_{H,D}|}{|H|}
\end{aligned}
$$

Contd...

Since $P(D|h) = 1$ when h is consistent with D, we have

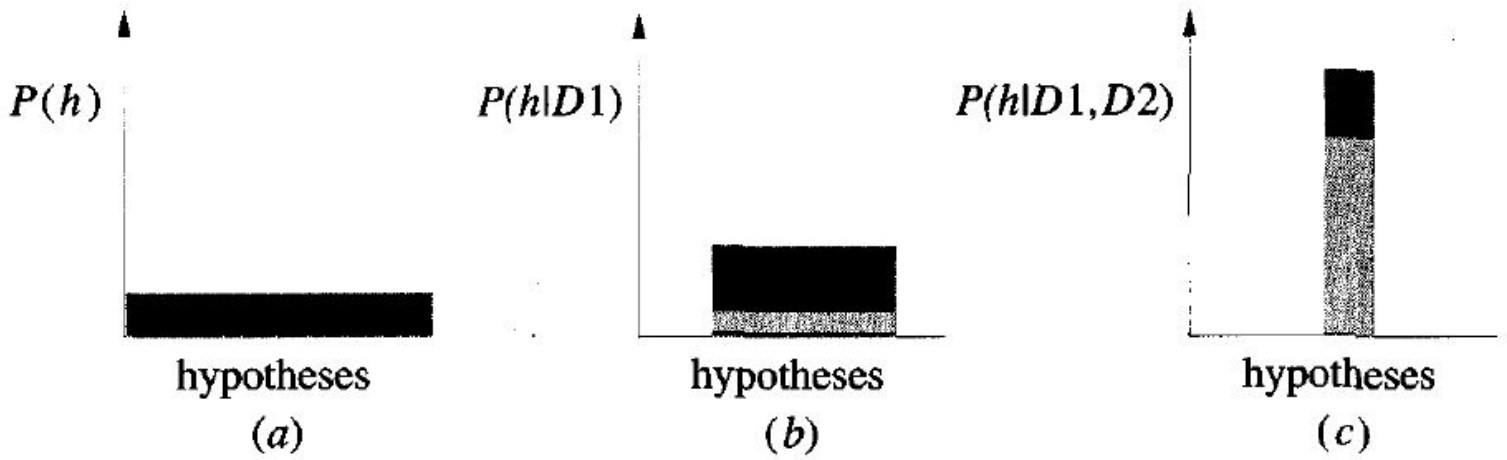
$$P(h|D) = \frac{1.\frac{1}{|H|}}{P(D)} = \frac{1}{|VS_{H,D}|}$$

where $VS_{H,D}$ is the subset of hypotheses from H that are consistent with D (Version Space of H)
Thus

$$P(h|D) = \begin{cases} \frac{1}{|VS_{H,D}|} & \text{if } h \text{ is consistent with } D \\ 0 & \text{otherwise} \end{cases}$$

Contd..

The evolution of probabilities associated with hypotheses is depicted in the figure.
Initially (a) all hypotheses have the same probability.
As training data accumulates (b and c), the posterior probability for inconsistent hypotheses becomes zero while the total probability summing to one is shared equally among the remaining consistent hypotheses.
Every consistent hypothesis is therefore a MAP hypothesis.



MAP Hypotheses and Consistent Learners

- A learning algorithm is a consistent learner if it outputs a hypothesis that commits zero errors over the training examples.
- Every consistent learner outputs a MAP hypothesis, if we assume
  - a uniform prior probability distribution over H (i.e., $P(h_i) = P(h_j)$ for all i, j), and
  - deterministic, noise free training data (i.e., $P(D|h) = 1$ if D and h are consistent, and 0 otherwise).
- Because FIND-S outputs a consistent hypothesis, it will output a MAP hypothesis under the probability distributions P(h) and $P(D|h)$ defined above.

Risk factor in Bayes theorem

We know that

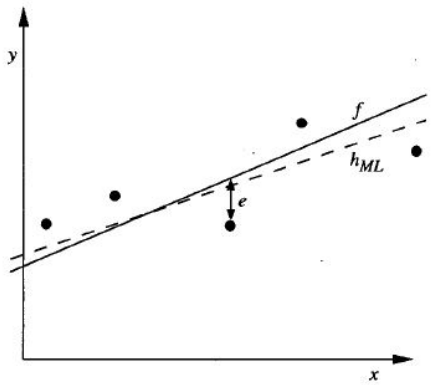
$$Posterior \propto Likelihood * Prior$$

Consider a two class category problem. Suppose both classes are equally likely (Prior probability is 0.5 for each), then the posterior solely depends on likelihood estimate. Similarly, if the likelihood is same for both class, then the posterior classification is purely based on prior.
This bias in classification is taken care by introducing a risk factor or loss function in the decision boundary.

Maximum Likelihood and Least Squared Error Hypothesis

1. Many learning approaches such as linear regression, and polynomial curve fitting try to learn a continuous-valued target function.
2. Under certain assumptions any learning algorithm that minimizes the squared error between the output hypothesis predictions and the training data will output a MAXIMUM LIKELIHOOD HYPOTHESIS.
3. The significance of this result is that it provides a Bayesian justification (under certain assumptions) for many neural network and other curve fitting methods that attempt to minimize the sum of squared errors over the training data.

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:

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

Learning A Real Valued Function

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

Maximize natural log of this instead...

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

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

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

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

MDL: prefer the hypothesis h that minimizes

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

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

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

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

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

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 - Example

Question: What is the most probable classification of x ?

Answer:

$$P(-|h_1) = 0, \quad P(+|h_1) = 1$$
$$P(-|h_2) = 1, \quad P(+|h_2) = 0$$
$$P(-|h_3) = 1, \quad P(+|h_3) = 0$$

$$\sum_{h_i \in H} P(+|h_i)P(h_i|D) = 0.4$$

$$\sum_{h_i \in H} P(-|h_i)P(h_i|D) = 0.6$$

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

Gibbs Classifier

Note: The Bayes optimal classifier provides the best result, but it can be expensive if there are many hypotheses.

Gibbs algorithm:

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

Surprising fact [Haussler et al. 1994]: If the target concept is selected randomly according to the $P(h|D)$ distribution, then the expected error of Gibbs Classifier is no worse than twice the expected error of the Bayes optimal classifier!

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

Naive Bayes Classifier - Learning

LEARN_NAIVE_BAYES_TEXT(*Examples, Vocabulary*)

1. **Collect all words and other tokens that occur in** $Examples$

   $Vocabulary \leftarrow$ **all distinct words and other tokens in** $Examples$

2. **Calculate the required** $P(v_j)$ **and** $P(w_k|v_j)$ **probability terms**

   For each target value v_j **in** V

   $docs_j \leftarrow$ **the subset of** $Examples$ **for which the target value is** v_j

   $P(v_j) \leftarrow \frac{|docs_j|}{|Examples|}$

   $Text_j \leftarrow$ **a single doc. created by concat. all members of** $docs_j$

   $n \leftarrow$ **the total number of words in** $Text_j$

   For each word w_k **in** $Vocabulary$

   $n_k \leftarrow$ **the number of times word** w_k **occurs in** $Text_j$

   $P(w_k|v_j) \leftarrow \frac{n_k+1}{n+|Vocabulary|}$ **(here we use the** m**-estimate)**

Naive Bayes Classifier - classification

CLASSIFY_NAIVE_BAYES_TEXT(Doc)

$positions \leftarrow$ **all word positions in** Doc **that contain tokens from** $Vocabulary$

Return $v_{NB} = \text{argmax}_{v_j \in V} P(v_j) \prod_{i \in positions} P(a_i = w_k | v_j)$

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

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

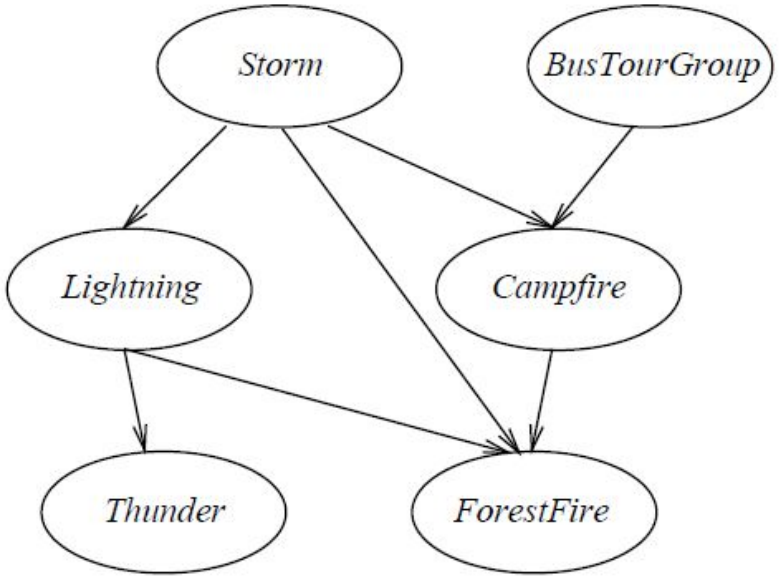
Example: *Thunder* is conditionally independent of *Rain*, given *Lightning*

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

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

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 Nets

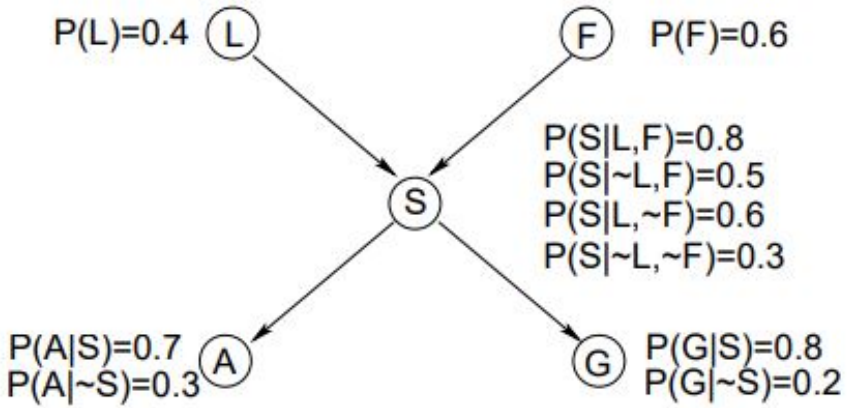
Question: Given a Bayes net, can one infer the probabilities of values of one or more network variables, given the observed values of (some) others?

Example:

Given the Bayes net

compute:

(a) $P(S)$
(b) $P(A, S)$
(b) $P(A)$



Inference in Bayesian Networks

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

- 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

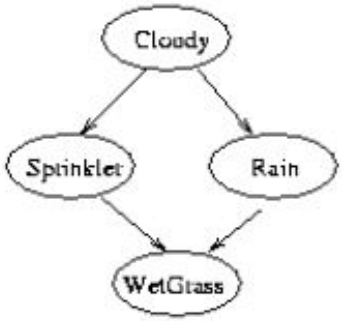
- Exact inference methods work well for some network structures
- Monte Carlo methods "simulate" the network randomly to calculate approximate solutions

Bayes Nets - Examples

Example

KP Murphy's web site:
http://www.cs.berkeley.edu/~murphyk/Bayes/bayes.html

P(C=F)	P(C=T)
0.5	0.5



P(S|C)

C	P(S=F)	P(S=T)
F	0.5	0.5
T	0.9	0.1

P(R|C)

C	P(R=F)	P(R=T)
F	0.8	0.2
T	0.2	0.8

S	R	P(W=F)	P(W=T)
F	F	1.0	0.0
T	F	0.1	0.9
F	T	0.1	0.9
T	T	0.01	0.99

P(W|S,R)

Joint: $P(C,R,S,W)=P(C)*P(R|C)*P(S|C,R)*P(W|C,R,S)$

or $P(C,R,S,W)=P(C)*P(S|C)*P(R|C)*P(W|S,R)$

Independencies: $\underline{\boldsymbol{I(S;R|C),\ I(R;S|C)}}$

Dependencies: $P(S|C),\ P(R|C),\ P(W|S,R)$

Bayesian Inference

Which event is more likely, wet grass observed and it is because of

- sprinkler:

  $P(S=1|W=1)=P(S=1,W=1)/P(W=1)=0.430$

- rain:

  $P(R=1|W=1)=P(R=1,W=1)/P(W=1)=0.708$

Algorithms exist that can answer such questions given the Bayesian Network

Bayes Nets - Examples

Example for BN construction: Fire Diagnosis

You want to diagnose whether there is a fire in a building

- You receive a noisy report about whether everyone is leaving the building
- If everyone is leaving, this may have been caused by a fire alarm
- If there is a fire alarm, it may have been caused by a fire or by tampering
- If there is a fire, there may be smoke

Example for BN construction: Fire Diagnosis

First you choose the variables. In this case, all are Boolean:

- Tampering is true when the alarm has been tampered with
- Fire is true when there is a fire
- Alarm is true when there is an alarm
- Smoke is true when there is smoke
- Leaving is true if there are lots of people leaving the building
- Report is true if the sensor reports that lots of people are leaving the building

Example for BN construction: Fire Diagnosis

- Using the total ordering of variables:
  - Let's say Fire; Tampering; Alarm; Smoke; Leaving; Report.
- Now choose the parents for each variable by evaluating conditional independencies
  - Fire is the first variable in the ordering. It does not have parents.
  - Tampering independent of fire (learning that one is true would not change your beliefs about the probability of the other)
  - Alarm depends on both Fire and Tampering: it could be caused by either or both
  - Smoke is caused by Fire, and so is independent of Tampering and Alarm given whether there is a Fire
  - Leaving is caused by Alarm, and thus is independent of the other variables given Alarm
  - Report is caused by Leaving, and thus is independent of the other variables given Leaving

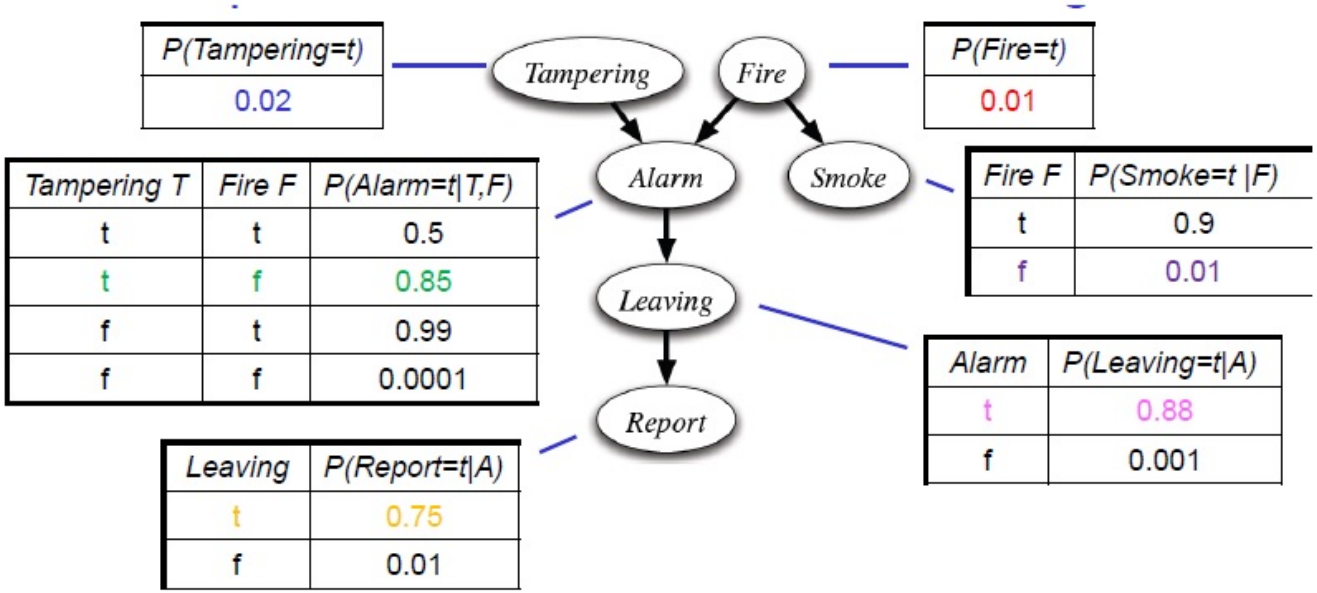
Example for BN construction: Fire Diagnosis

- This results in the following Bayesian network

- P(Tampering, Fire, Alarm, Smoke, Leaving, Report)
  = P(Tampering) × P(Fire) × P(Alarm|Tampering,Fire)
  × P(Smoke|Fire) × P(Leaving|Alarm) × P(Report|Leaving)

Bayes Nets - Examples

Suppose the conditional probabilities are given,
find P (Tampering = true, Fire = false, Smoke = false, Leaving = true and Report = true)



P(Tampering=t)
0.02

P(Fire=t)
0.01

Tampering T	Fire F	P(Alarm=t|T,F)
t	t	0.5
t	f	0.85
f	t	0.99
f	f	0.0001

Fire F	P(Smoke=t |F)
t	0.9
f	0.01

Alarm	P(Leaving=t|A)
t	0.88
f	0.001

Leaving	P(Report=t|A)
t	0.75
f	0.01

P(Tampering=t, Fire=f, Alarm=t, Smoke=f, Leaving=t, Report=t)
= P(Tampering=t) × P(Fire=f) × P(Alarm=t|Tampering=t,Fire=f)
× P(Smoke=f|Fire=f) × P(Leaving=t|Alarm=t)
× P(Report=t|Leaving=t)
= 0.02 × (1-0.01) × 0.85 × (1-0.01) × 0.88 × 0.75

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

- 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 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 $Y_i = Campfire$, then u_{ik} might be
$\langle Storm = T, BusTourGroup = F \rangle$
Perform gradient ascent by repeatedly

- update all w_{ijk} using training data D

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

- then, renormalize the w_{ijk} to assure
  - $\sum_j w_{ijk} = 1$
  - $0 \leq w_{ijk} \leq 1$

Expectation Maximization -

1. An iterative algorithm for maximizing likelihood when the model contains unobserved latent variables.
2. The algorithm iterate between E-step (expectation) and M-step (maximization).
3. E-step: create a function for the expectation of the log-likelihood, evaluated using the current estimate for the parameters.
4. M-step: compute parameters maximizing the expected log-likelihood found on the E step.
5. It is useful when some of the random variables involved are not observed, i.e., considered missing or incomplete.
6. It is also useful when direct maximizing the target likelihood function is difficult.

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

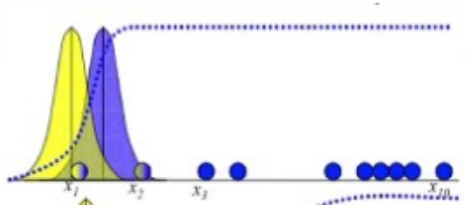
EM Algorithm: 1D example

We have data points belonging to 2 classes (Blue / Yellow).
We can find the Gaussian curve that can fit these data

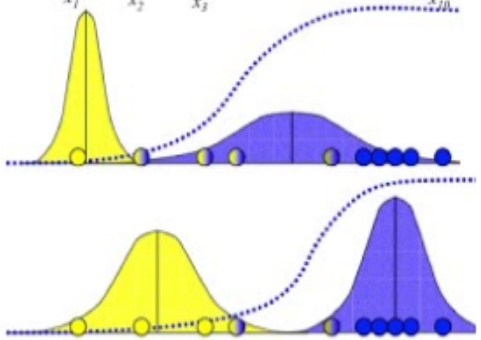
What if we had just the data points but no information about its class?
How to fit the Gaussian curve?

EM algorithm:
1) Take any two random gaussain curves and fit the data points

2) Update the parameters of gaussian curves and fit the data points



3) Repeat this until desired level of accuracy is reached.



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

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

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 $X = \{x_1, \ldots, x_m\}$
- 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 $y_i = x_i z_i$
  - 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

EM Algorithm example

Let $(x_1, x_2, x_3) = (2, 4, 7)$ be our three datapoints, presumed to have each been generated from one of two Gaussians.

The stdev of both Gaussians are given: $\sigma_1 = \sigma_2 = 1/\sqrt{2}$.
The prior over the two Gaussians in also given: $\lambda_1 = \lambda_2 = 0.5$
Let j be an index over the Gaussians, i an index over the data points, and k an index over the E-M iterations.

We are trying to derive values for the two means $\boldsymbol{\mu} = (\mu_1, \mu_2)$ that maximize the likelihood of the given data, i.e.:

$$\boldsymbol{\mu}_{ML} = \arg\max_{\mu_1, \mu_2} \prod_i \sum_j \lambda_j e^{-\frac{(x_i - \mu_j)^2}{2\sigma^2}}$$

Let us initialize the Gaussian means to some reasonable values (inside the data range, and integer valued, to make calculation easy): $\mu_1^{[0]} = 3, \mu_2^{[0]} = 6$.

Let $z_{i,j} = 1$ if x_i was generated by Gaussian j, and 0 otherwise. The $z_{i,j}$'s are our latent variables.

The likelihood function can be written as:

$$L(x_i|\mu^{[k]}) = \sum_j \lambda_j L(x_i|\mu = \mu_j^{[k]})$$

EM Algorithm example

The E-Step is:

$$E[z_{i,j}|\mu^{[k]}] = \frac{\lambda_j L(x_i|\mu = \mu_j^{[k]})}{\sum_{j'} \lambda_{j'} L(x_i|\mu = \mu_{j'}^{[k]})}$$

and the M-step is:

$$\mu_j^{[k+1]} = \frac{\sum_i E[z_{i,j}|\mu^{[k]}] \cdot x_i}{\sum_i E[z_{i,j}|\mu^{[k]}]}$$

EM Algorithm example

i	1	2	3
x_i	2.0	4.0	7.0
$L(x_i \mid \mu = \mu_1^{[0]})$	$\frac{1}{\sqrt{\pi}} e^{-1}$	$\frac{1}{\sqrt{\pi}} e^{-1}$	$\frac{1}{\sqrt{\pi}} e^{-16}$
$L(x_i \mid \mu = \mu_2^{[0]})$	$\frac{1}{\sqrt{\pi}} e^{-16}$	$\frac{1}{\sqrt{\pi}} e^{-4}$	$\frac{1}{\sqrt{\pi}} e^{-1}$
$L(x_i \mid \mu^{[\mathbf{0}]})$	$\frac{1}{2\sqrt{\pi}}(e^{-1} + e^{-16})$	$\frac{1}{2\sqrt{\pi}}(e^{-1} + e^{-4})$	$\frac{1}{2\sqrt{\pi}}(e^{-16} + e^{-1})$
$E[z_{i,1} \mid \mu^{[0]}]$	$\frac{e^{-1}}{e^{-1}+e^{-16}} \approx 1$	$\frac{e^{-1}}{e^{-1}+e^{-4}} \approx 0.953$	$\frac{e^{-16}}{e^{-1}+e^{-16}} \approx 0$
$E[z_{i,2} \mid \mu^{[0]}]$	$\frac{e^{-16}}{e^{-1}+e^{-16}} \approx 0$	$\frac{e^{-4}}{e^{-1}+e^{-4}} \approx 0.047$	$\frac{e^{-1}}{e^{-1}+e^{-16}} \approx 1$

And therefore:

$$\mu_1^{[1]} \approx \frac{1 * 2.0 + 0.953 * 4.0 + 0 * 7.0}{1 + 0.953} \approx 2.976$$

and

$$\mu_2^{[1]} \approx \frac{0 * 2.0 + 0.047 * 4.0 + 1 * 7.0}{1 + 0.047} \approx 6.865$$