

Bayesian Learning

Based on “Machine Learning”, T. Mitchell, McGRAW Hill, 1997, ch. 6

Acknowledgement:

The present slides are an adaptation of slides drawn by T. Mitchell

Two Roles for the Bayesian Methods in Learning

1. Provides practical **learning algorithms**

by combining prior knowledge/probabilities with observed data:

- Naive Bayes learning algorithm
- Expectation Maximization (EM) learning algorithm (scheme): learning in the presence of unobserved variables
- Bayesian Belief Network learning

2. Provides a **useful conceptual framework**

- Serves for evaluating other learning algorithms, e.g. concept learning through general-to-specific hypotheses ordering (FINDS, and CANDIDATEELIMINATION), neural networks, linear regression
- Provides additional insight into Occam's razor

PLAN

2.

1. Basic Notions

Bayes' Theorem

Defining **classes of hypotheses**:

Maximum a Posteriori (MAP) hypotheses

Maximum Likelihood (ML) hypotheses

2. Learning MAP hypotheses

2.1 The brute force MAP hypotheses learning algorithm

2.2 **The Bayes optimal classifier; Gibbs classifier**

2.3 **The Naive Bayes Learner.** Example: Learning over text data

2.4 The Minimum Description Length (MDL) Principle;
MDL hypotheses

3. Learning ML hypotheses

3.1 ML hypotheses in learning real-valued functions

ML hypotheses in learning to predict probabilities

3.2 **The Expectation Maximization (EM) algorithm**

[Application: EM for clustering]

4. **Bayesian Belief Networks**

1 Basic Notions

- **Product Rule:**

probability of a conjunction of two events A and B:

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

- **Bayes' Theorem:**

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

- **Theorem of total probability:**

if events A_1, \dots, A_n are mutually exclusive,
with $\sum_{i=1}^n P(A_i) = 1$, then

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

in particular

$$P(B) = P(B|A)P(A) + P(B|\neg A)P(\neg A)$$

Using Bayes' Theorem for Hypothesis Learning

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

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

Classes of Hypotheses

Maximum Likelihood (ML) hypothesis:

the hypothesis that best explains the training data

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

Maximum a Posteriori (MAP) hypothesis:

the most probable hypothesis given the training data

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

Note: If $P(h_i) = P(h_j), \forall i, j$, then $h_{MAP} = h_{ML}$

Exemplifying MAP Hypotheses

Suppose the following data characterize the lab result for cancer-suspect people.

$$\begin{array}{l|l} P(\text{cancer}) = 0.008 & P(\neg\text{cancer}) = 0.992 \\ \hline P(+|\text{cancer}) = 0.98 & P(-|\text{cancer}) = 0.02 \\ P(+|\neg\text{cancer}) = 0.03 & P(-|\neg\text{cancer}) = 0.97 \end{array} \quad \left| \begin{array}{l} h_1 = \text{cancer}, h_2 = \neg\text{cancer} \\ D = \{+, -\}, P(D | h_1), P(D | h_2) \end{array} \right.$$

Question: Should we diagnose a patient x whose lab result is positive as having cancer?

Answer: No.

Indeed, we have to find $\operatorname{argmax}\{P(\text{cancer}|+), P(\neg\text{cancer}|+)\}$.

Applying Bayes theorem (for $D = \{+\}$):

$$\left. \begin{array}{l} P(+ | \text{cancer})P(\text{cancer}) = 0.98 \times 0.008 = 0.0078 \\ P(+ | \neg\text{cancer})P(\neg\text{cancer}) = 0.03 \times 0.992 = 0.0298 \end{array} \right\} \Rightarrow h_{MAP} = \neg\text{cancer}$$

(We can infer $P(\text{cancer} | +) = \frac{0.0078}{0.0078+0.0298} = 21\%$)

A Special Case

Theorem:

Assume a (fixed) set of instances $\langle x_1, \dots, x_m \rangle$.
and the **noisy-free set** of classifications $D = \langle c(x_1), \dots, c(x_m) \rangle$.
Let H be the hypotheses space.

If $P(h)$ is a **uniform distribution**, i.e. $P(h) = \frac{1}{|H|}$ for all h in H ,
assuming that all hypotheses are mutually exclusive,
then **all hypotheses consistent with D are MAP/ML hypotheses.**

Consequence:

The output of concept learning algorithms through general-to-specific ordering of hypothesis (e.g., FINDS, CANDIDATE-ELIMINATION, see Ch. 2) are MAP/ML hypotheses.

Proof of the theorem

Notation:

$VS_{H,D}$ = the subset of H made of all hypotheses consistent with data in D , i.e. $d_i = h(x_i)$ for all d_i in D .

($VS_{H,D}$ is called the *version space* of concept c relative to H and D .)

Since D is noisy-free, then:

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

It follows that:

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

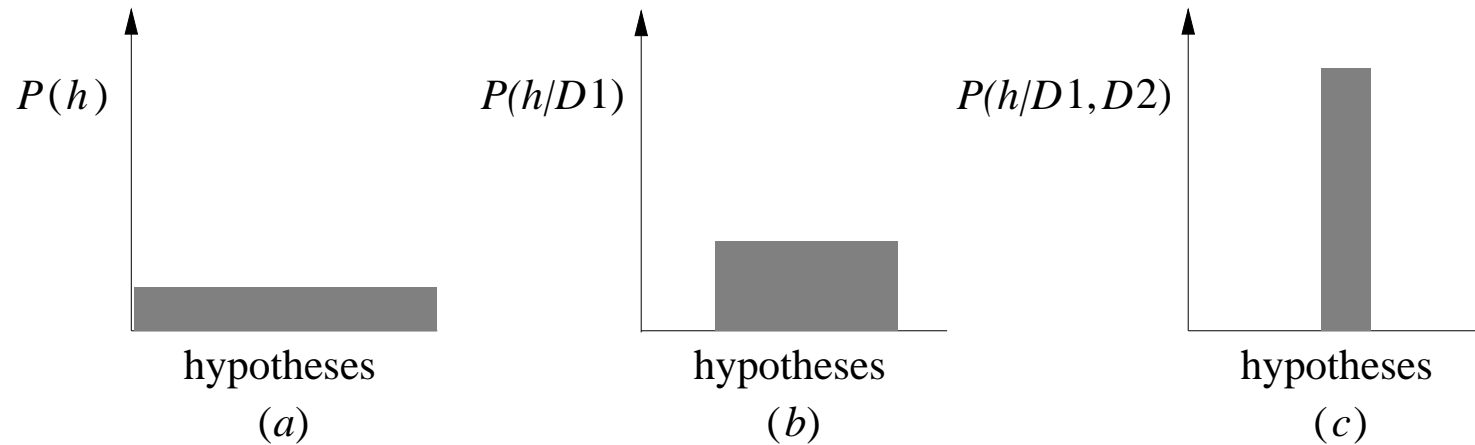
Note: Here above

$P(D)$ was replaced by $\frac{|VS_{H,D}|}{|H|}$ since

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

$$\sum_{h_i \in VS_{H,D}} 1 \cdot \frac{1}{|H|} + \sum_{h_i \notin VS_{H,D}} 0 \cdot \frac{1}{|H|} = \sum_{h_i \in VS_{H,D}} \frac{1}{|H|} = \frac{|VS_{H,D}|}{|H|}$$

Evolution of Posterior Probabilities in FINDS and CANDIDATE-ELIMINATION

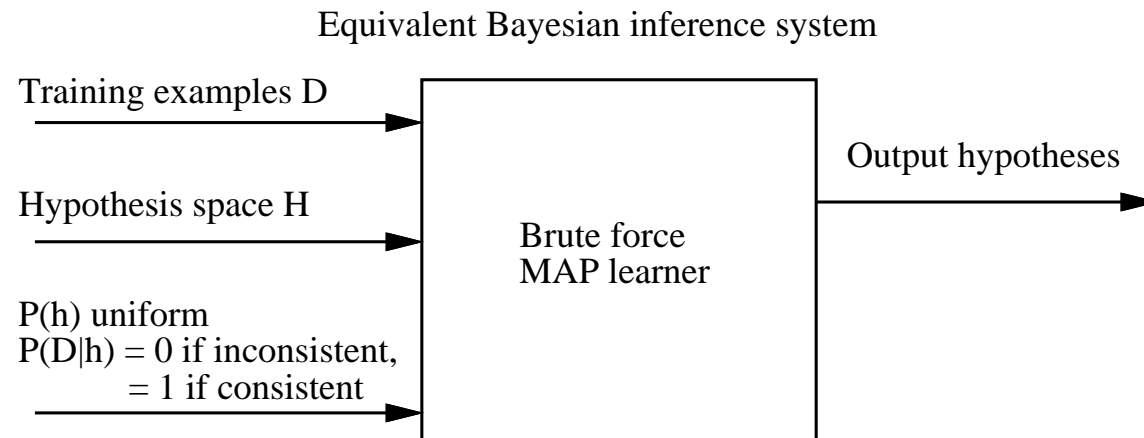
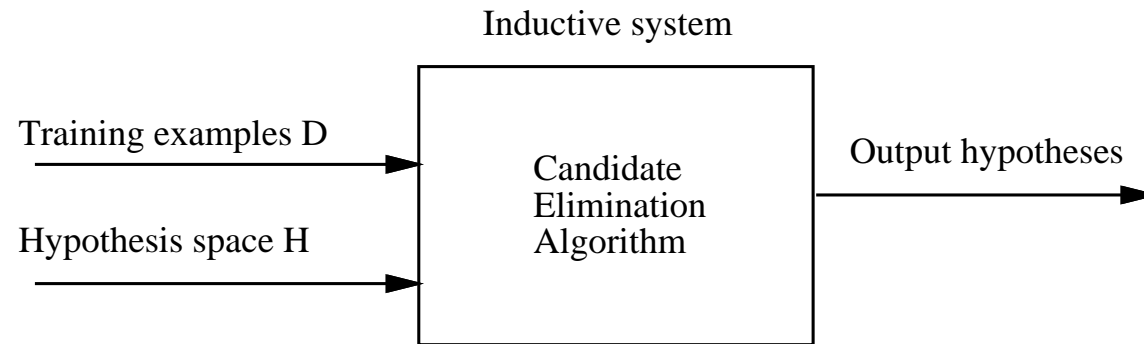


a: Initially, all hypotheses have the same probability, $\frac{1}{|H|}$.

b&c: As training data accumulates, the posterior probability for inconsistent hypotheses becomes 0, while the probability of all (consistent) hypotheses sums to 1.

Characterizing some ML Algos by Equivalent MAP Learners

Note the similarity
with characterizing
the inductive bias
of a learner.



*Prior assumptions
made explicit*

Important Remark

Since now, we have seen a special case of Bayesian reasoning, because we considered only $P(D | h) = 0$ or 1 , and no noisy data.

In the sequel we will work without these restrictions.

2 Learning MAP Hypothesis

2.1 The Brute Force MAP Hypothesis Learning Algorithm

Training:

Choose the hypothesis with the highest posterior probability

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

Testing:

Given x , compute $h_{MAP}(x)$

Drawback:

Requires to compute all probabilities $P(D|h)$ and $P(h)$.

2.2 The Bayes Optimal Classifier:

The Most Probable Classification of New Instances

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

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

Answer: $P(v_j|D) = \sum_{h_i \in H} P(v_j|h_i)P(h_i|D)$

Therefore, the **Bayes optimal classification** of x is:

$$\operatorname{argmax}_{v_j \in V} \sum_{h_i \in H} P(v_j|h_i)P(h_i|D)$$

Remark: $h_{MAP}(x)$ is not the most probable classification of x !
(See the next example.)

The Bayes Optimal Classifier: An Example

Let us consider three possible hypotheses:

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

Given new instance x such that

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

Question: What is the most probable classification of x ?

Answer:

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

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

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

$$\sum_{h_i \in H} P(+|h_i)P(h_i|D) = 0.4 \quad \text{and} \quad \sum_{h_i \in H} P(-|h_i)P(h_i|D) = 0.6$$

therefore

$$\operatorname{argmax}_{v_j \in V} \sum_{h_i \in H} P(v_j|h_i)P(h_i|D) = -$$

Gibbs Classifier

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

Gibbs algorithm:

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

Surprising fact:

Suppose a correct, uniform prior distribution over H .

If the target concept is selected randomly according to that distribution, then

- pick any hypothesis from $V S_{H,D}$ with uniform probability;
- its expected error is no worse than twice the expected error of the Bayes optimal classifier!

$$E[\text{error}_{Gibbs}] \leq 2E[\text{error}_{BayesOptimal}]$$

2.3 The Naive Bayes Classifier

When to use it:

- The target function f takes value from a finite set $V = \{v_1, \dots, v_k\}$
- Moderate or large training data set is available
- The attributes $\langle a_1, \dots, a_n \rangle$ that describe instances are conditionally independent w.r.t. to the given classification:

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

The most probable value of $f(x)$ is:

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

The Naive Bayes Classifier: Remarks

1. Along with decision trees, neural networks, k-nearest neighbours, the Naive Bayes Classifier is **one of the most practical** learning methods.
2. Compared to the previously presented learning algorithms, the Naive Bayes Classifier **does no search** through the hypothesis space;
the output hypothesis is simply formed by estimating the **parameters** $P(v_j)$, $P(a_i|v_j)$.

The Naive Bayes Classification Algorithm

NAIVE_BAYES_LEARN(*examples*)

for each target value v_j

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

for each attribute value a_i of each attribute a

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

CLASSIFY_NEW_INSTANCE(x)

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

The Naive Bayes: An Example

Consider again the *PlayTennis* example, and new instance

$\langle \text{Outlook} = \text{sun}, \text{Temp} = \text{cool}, \text{Humidity} = \text{high}, \text{Wind} = \text{strong} \rangle$

We compute:

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

$$P(\text{yes}) = \frac{9}{14} = 0.64 \quad P(\text{no}) = \frac{5}{14} = 0.36$$

...

$$P(\text{strong} | \text{yes}) = \frac{3}{9} = 0.33 \quad P(\text{strong} | \text{no}) = \frac{3}{5} = 0.60$$

$$P(\text{yes}) P(\text{sun} | \text{yes}) P(\text{cool} | \text{yes}) P(\text{high} | \text{yes}) P(\text{strong} | \text{yes}) = 0.0053$$

$$P(\text{no}) P(\text{sun} | \text{no}) P(\text{cool} | \text{no}) P(\text{high} | \text{no}) P(\text{strong} | \text{no}) = 0.0206$$

$$\rightarrow v_{NB} = \text{no}$$

A Note on The Conditional Independence Assumption of Attributes

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

It is often violated in practice ...but it works surprisingly well anyway.

Note that we don't need estimated posteriors $\hat{P}(v_j|x)$ to be correct; we need only that

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

[Domingos & Pazzani, 1996] analyses this phenomenon.

Naive Bayes Classification: The problem of unseen data

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

It follows that $\hat{P}(a_i|v_j) = 0$, and $\hat{P}(v_j) \prod_i \hat{P}(a_i|v_j) = 0$

The typical **solution** is to (re)define $P(a_i|v_j)$, for each value v_j of a_i :

$\hat{P}(a_i|v_j) \leftarrow \frac{n_c + mp}{n + m}$, 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 a prior estimate for $\hat{P}(a_i|v_j)$
(for instance, if the attribute a has k values, then $p = \frac{1}{k}$)
- m is a weight given to that prior estimate
(i.e. number of “virtual” examples)

Using the Naive Bayes Learner: Learning to Classify Text

- Learn which news articles are of interest

Target concept *Interesting?* : *Document* $\rightarrow \{+, -\}$

- Learn to classify web pages by topic

Target concept *Category* : *Document* $\rightarrow \{c_1, \dots, c_n\}$

Naive Bayes is among most effective algorithms

Learning to Classify Text: Main Design Issues

1. Represent each document by a vector of words

- one attribute per word position in document

2. Learning:

- use training examples to estimate $P(+)$, $P(-)$, $P(doc|+)$, $P(doc|-)$
- Naive Bayes conditional independence assumption:

$$P(doc|v_j) = \prod_{i=1}^{length(doc)} P(a_i = w_k|v_j)$$

where $P(a_i = w_k|v_j)$ is probability that word in position i is w_k , given v_j

- Make one more assumption:

$$\forall i, m \ P(a_i = w_k|v_j) = P(a_m = w_k|v_j) = P(w_k|v_j)$$

i.e. attributes are (not only indep. but) also identically distributed

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 $Text_j$

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

CLASSIFY_NAIVE_BAYES_TEXT(*Doc*)

positions ← all word positions in *Doc* that contain tokens from *Vocabulary*

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

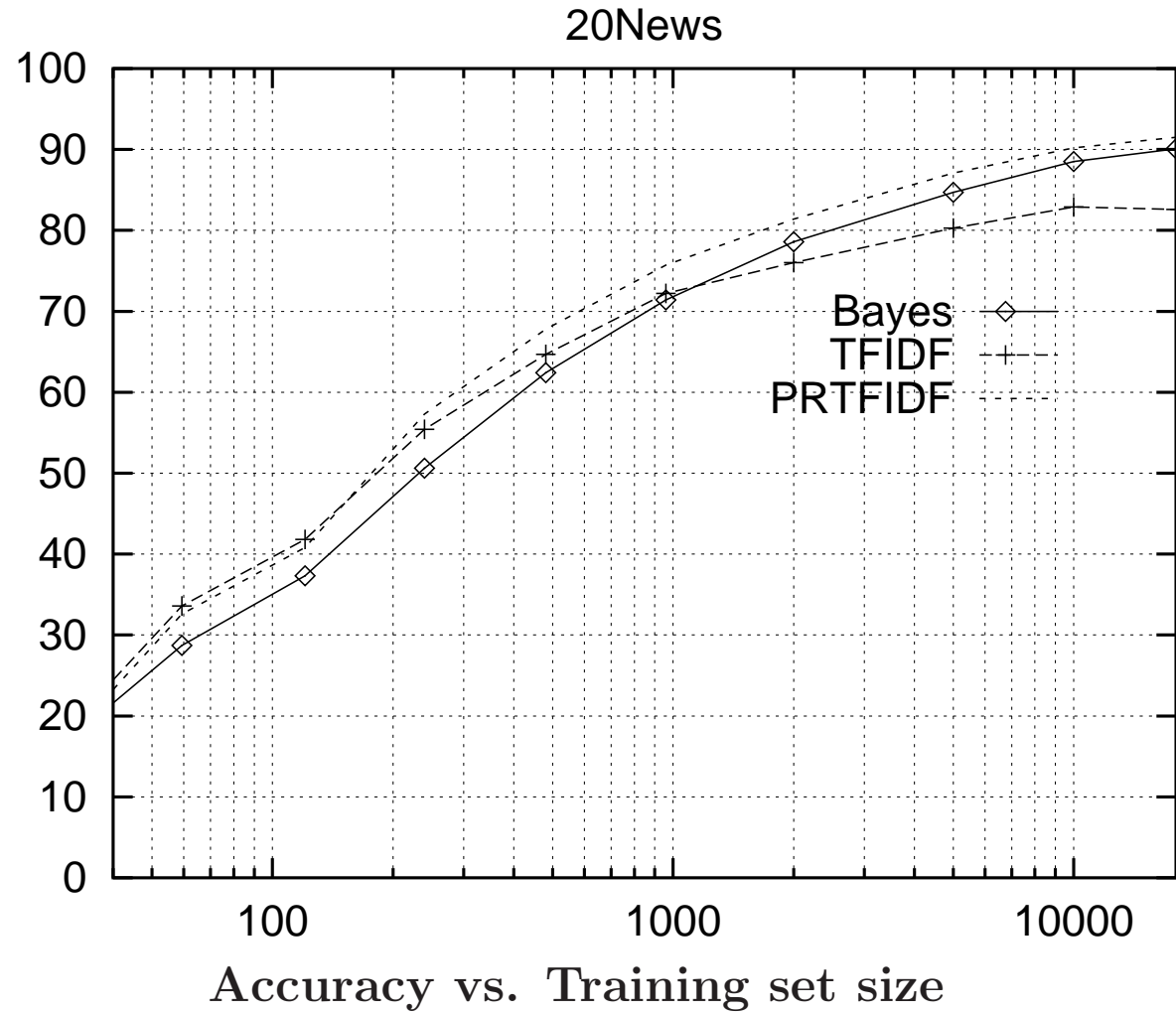
Application: Learning to Classify Usenet News Articles

Given 1000 training documents from each of the 20 newsgroups, learn to classify new documents according to which newsgroup it came from

comp.graphics	misc.forsale
comp.os.ms-windows.misc	rec.autos
comp.sys.ibm.pc.hardware	rec.motorcycles
comp.sys.mac.hardware	rec.sport.baseball
comp.windows.x	rec.sport.hockey
alt.atheism	sci.space
soc.religion.christian	sci.crypt
talk.religion.misc	sci.electronics
talk.politics.mideast	sci.med
talk.politics.misc	
talk.politics.guns	

Naive Bayes: 89% classification **accuracy** (having used 2/3 of each group for training; eliminated rare words, and the 100 most freq. words)

Learning Curve for 20 Newsgroups



2.4 The Minimum Description Length (MDL) Principle

Occam's razor: prefer the shortest hypothesis

Bayes analysis: prefer the hypothesis h_{MAP}

$$\begin{aligned} h_{MAP} &= \operatorname{argmax}_{h \in H} P(D|h)P(h) = \operatorname{argmax}_{h \in H} (\log_2 P(D|h) + \log_2 P(h)) \\ &= \operatorname{argmin}_{h \in H} (-\log_2 P(D|h) - \log_2 P(h)) \end{aligned}$$

Interesting fact from the Information Theory:

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

So we can interpret:

$-\log_2 P(h)$: the length of h under the optimal code

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

Therefore we prefer the hypothesis h that minimizes...

Bayes Analysis and the MDL Principle

We saw that a MAP learner prefers the hypothesis h that minimizes $L_{C_1}(h) + L_{C_2}(D|h)$, where $L_C(x)$ is the description length of x under encoding C

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

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

- $L_{C_1}(h)$ is the number of bits to describe tree h
- $L_{C_2}(D|h)$ is the number of bits to describe D given h

In literature, the application of MDL to practical problems often include arguments justifying the choice of the encodings C_1 and C_2 .

For instance:

$L_{C_2}(D|h) = 0$ if examples are classified perfectly by h ,
and both the transmitter and the receiver know h .

Therefore, in this situation we need only to describe exceptions. So:

$$h_{MDL} = \operatorname{argmin}_{h \in H} (\operatorname{length}(h) + \operatorname{length}(\operatorname{misclassifications}))$$

In general, **MDL trades off hypothesis size for training errors:**

it might select a shorter hypothesis that makes few errors over a longer hypothesis that perfectly classifies the data!

Consequence: In learning (for instance) decision trees, (using) the MDL principle can work as an alternative to pruning.

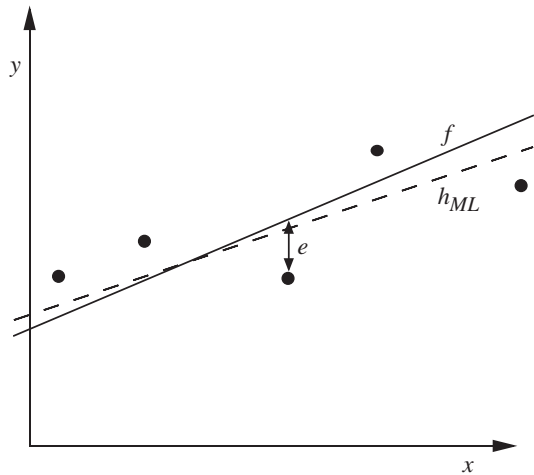
The MDL Principle: Back to Occam's Razor

MDL hypotheses are not necessarily also the best/MAP ones.

(For that, we should know all the probabilities $P(D|h)$ and $P(h)$.)

3 Learning Maximum Likelihood (ML) Hypothesis

3.1 ML Hypotheses in Learning Real Valued Functions



Problem: Consider learning a real-valued target function f from the training examples $\langle x_i, d_i \rangle$, with

x_i assumed fixed (to simplify)

d_i 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.

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

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

Proof

Note: We use the **probability density function** for a continuous random variable x :

$$p(x_0) = \lim_{\epsilon \rightarrow 0} \frac{1}{\epsilon} P(x_0 \leq x < x_0 + \epsilon)$$

$$h_{ML} = \operatorname{argmax}_{h \in H} p(D|h) = \operatorname{argmax}_{h \in H} \prod_{i=1}^m p(d_i|h) = \operatorname{argmax}_{h \in H} \prod_{i=1}^m p(e_i|h)$$

$$= \operatorname{argmax}_{h \in H} \prod_{i=1}^m p(d_i - f(x_i)|h) = \operatorname{argmax}_{h \in H} \prod_{i=1}^m p(d_i - h(x_i)|h)$$

$$= \operatorname{argmax}_{h \in H} \prod_{i=1}^m \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{d_i - h(x_i)}{\sigma}\right)^2} = \operatorname{argmax}_{h \in H} \left(\sum_{i=1}^m \ln \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2} \left(\frac{d_i - h(x_i)}{\sigma} \right)^2 \right)$$

$$= \operatorname{argmax}_{h \in H} \sum_{i=1}^m -\frac{1}{2} \left(\frac{d_i - h(x_i)}{\sigma} \right)^2 = \operatorname{argmax}_{h \in H} \sum_{i=1}^m -(d_i - h(x_i))^2$$

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

Generalisations

1. Similar derivations can be performed starting with **other assumed noise distributions** (than Gaussians), producing **different results**.

2. It was assumed that
 - a. the **noise** affects only $f(x_i)$, and
 - b. no noise was recorded in the **attribute values** for the given examples x_i .

Otherwise, the analysis becomes significantly more complex.

ML Hypotheses for Learning to Predict Probabilities

Consider some training examples $\langle x_1, d_1 \rangle, \langle x_2, d_2 \rangle, \dots, \langle x_m, d_m \rangle$, where d_i is either 1 or 0 for $i = 1, \dots, m$.

We want to train a neural network to output a *probability* h given x_i . (E.g. $h(x_i) = 0.92$ if $P(\{ \langle x_i, d_i \rangle \mid d_i = 1 \}) = 0.92$)

In this case we can show (see [Mitchell, 1997] page 169) that

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

As a consequence (see [Mitchell, 1997] pages 170-171), the weight update rule for a sigmoid unit in a single layer NN with the output h_{ML} is:

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

where x_{ijk} is the k -th input to the unit j for the i -th training example.

3.2 The Expectation Maximization (EM) Algorithm

[Dempster et al, 1977]

Find (local) Maximum Likelihood hypotheses when data is only partially observable:

- Unsupervised learning (i.e., clustering):
the target value is unobservable
- Supervised learning:
some instance attributes are unobservable

Some applications:

- Non-hierarchical clustering:
Estimate the means of k Gaussians
- Train Radial Basis Function Networks
- Train Bayesian Belief Networks
- Learn Hidden Markov Models
- Learn Probabilistic Context Free Grammars

The General EM Problem

Given

- observed data $X = \{x_1, \dots, x_m\}$
(independently drawn instances)
- unobserved data $Z = \{z_1, \dots, z_m\}$
- the parameterized probability distribution $P(Y|h)$,
where $Y = \{y_1, \dots, y_m\}$ is the full data $y_i = x_i \cup z_i$

determine

\bar{h} that (locally) maximizes $P(Y|h)$

The Essence of the EM Approach

Start with h_0 , an arbitrarily/conveniently chosen value of h .

Repeatedly

1. Use the observed data X and the current hypothesis h_i to **estimate the unobserved** variables Z .
2. The expected values of the unobserved variables Z are used to **calculate an improved hypothesis** h_{i+1} .

The General EM Algorithm

Repeat the following two steps until convergence is reached:

Estimation (E) step:

Calculate the log **likelihood function**

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

where $Y = X \cup Z$.

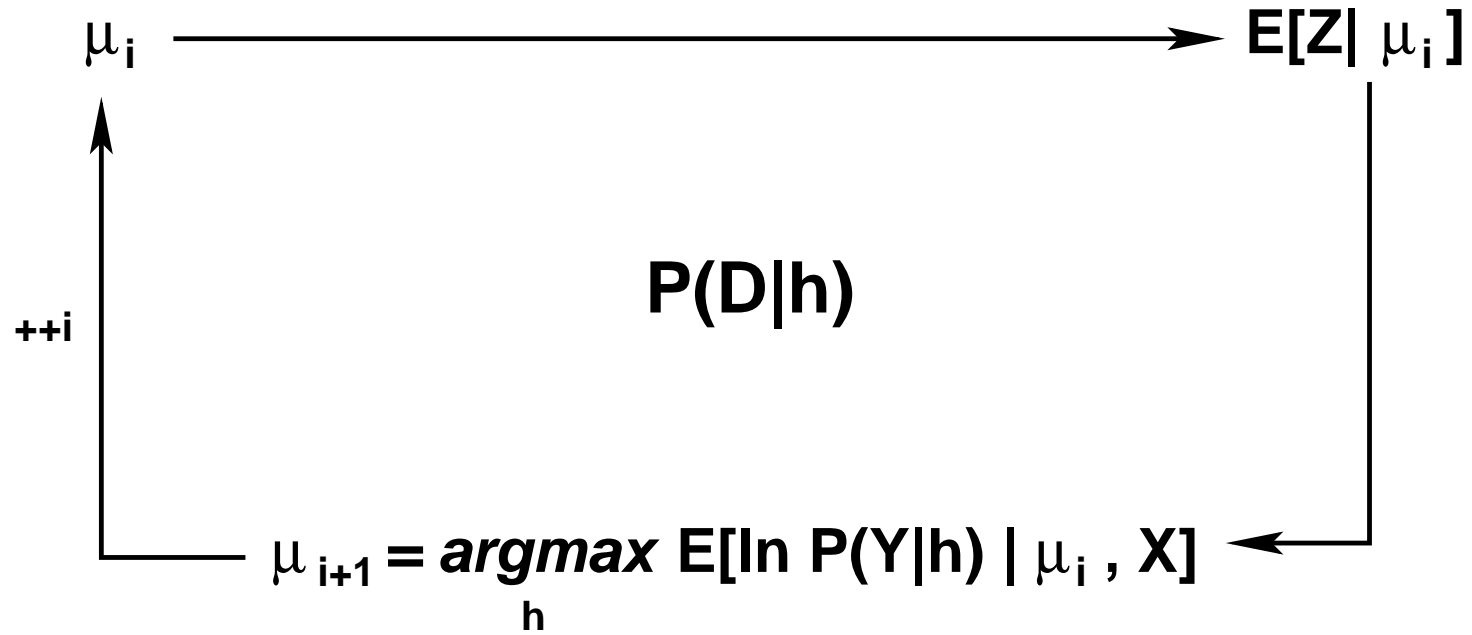
Maximization (M) step:

Replace hypothesis h_i by the hypothesis h_{i+1} that maximizes this Q function.

$$h_{i+1} \leftarrow \operatorname{argmax}_h Q(h|h_i)$$

Baum-Welch Theorem

When Q is continuous, it can be shown that EM converges to a stationary point (local maximum) of the likelihood function $P(Y|h)$.



4 Bayesian Belief Networks

(also called Bayes Nets)

Interesting because:

- The Naive Bayes assumption of conditional independence of attributes is too restrictive.
(But it's intractable without some such assumptions...)
- Bayesian Belief networks describe **conditional independence among *subsets* of variables**.
- It allows the combination of prior knowledge about (in)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 a value of Z :

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

More compactly, we write $P(X|Y, Z) = P(X|Z)$

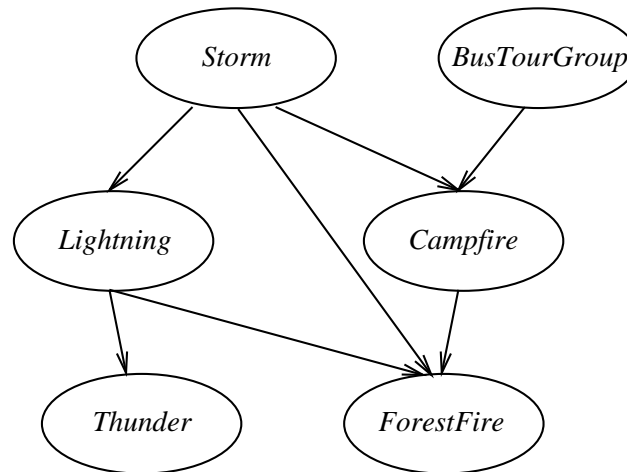
Note: Naive Bayes uses conditional independence to justify

$$P(A_1, A_2 | V) = P(A_1 | A_2, V) P(A_2 | V) = P(A_1 | V) P(A_2 | V)$$

Generalizing the above definition:

$$P(X_1 \dots X_l | Y_1 \dots Y_m, Z_1 \dots Z_n) = P(X_1 \dots X_l | Z_1 \dots Z_n)$$

A Bayes Net



	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



The network is defined by

- A directed acyclic graph, representing a set of conditional independence assertions:

Each node — representing a random variable — is asserted to be conditionally independent of its nondescendants, given its immediate predecessors.

Example: $P(\text{Thunder} | \text{ForestFire}, \text{Lightning}) = P(\text{Thunder} | \text{Lightning})$

- A table of local conditional probabilities for each node/variable.

A Bayes Net (Cont'd)

represents **the joint probability distribution** over all variables Y_1, Y_2, \dots, Y_n :

This joint distribution is fully defined by the graph, plus the conditional probabilities:

$$P(y_1, \dots, y_n) = P(Y_1 = y_1, \dots, Y_n = y_n) = \prod_{i=1}^n P(y_i | Parents(Y_i))$$

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

In our **example**: $P(Storm, BusTourGroup, \dots, ForestFire)$

Inference in Bayesian Nets

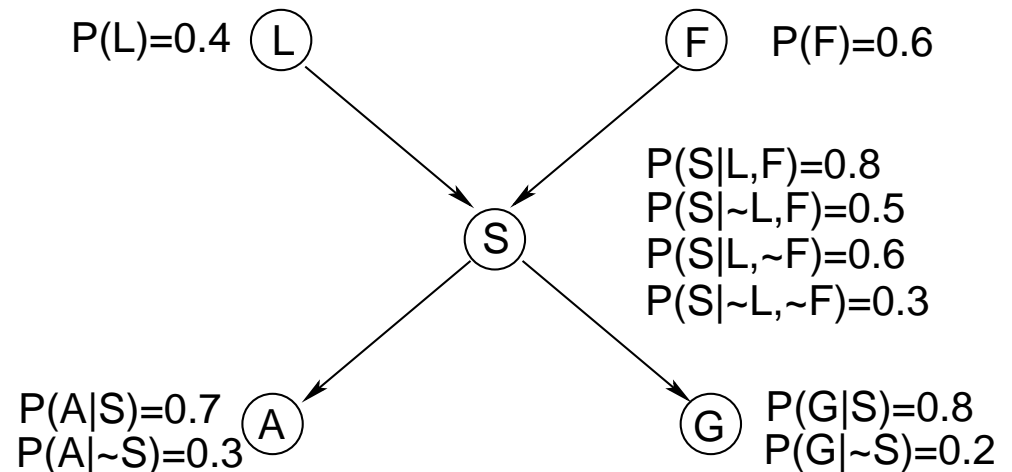
Question: Given a Bayes net, can one infer the probabilities of values of one or more network variables, given the observed values of (some) others?

Example:

Given the Bayes net

compute:

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



Inference in Bayesian Nets (Cont'd)

Answer(s):

- If only one variable is of unknown (probability) value, then it is easy to infer it
- In the general case, we can compute the probability distribution for any subset of network variables, given the distribution for any subset of the remaining variables. But...
- The exact inference of probabilities for an arbitrary Bayes net is an NP-hard problem!!

Inference in Bayesian Nets (Cont'd)

In practice, we can succeed in many cases:

- Exact inference methods work well for some net structures.
- Monte Carlo methods “simulate” the network randomly to calculate approximate solutions [Pradham & Dagum, 1996].

(In theory even approximate inference of probabilities in Bayes Nets can be NP-hard!! [Dagum & Luby, 1993])

Learning Bayes Nets (I)

There are **several variants** of this learning task

- The network structure might be either *known* or *unknown* (i.e., it has to be inferred from the training data).
- The training examples might provide values of *all* network variables, or just for *some* of them.

The simplest case:

If the structure is known and we can observe the values of all variables, then it is easy to estimate the conditional probability table entries (analogous to training a Naive Bayes classifier).

Learning Bayes Nets (II)

When

- the structure of the Bayes Net is known, and
- the variables are only partially observable in the training data

learning the entries in the conditional probabilities tables is similar to (learning the weights of hidden units in) training a neural network with hidden units:

- We can learn the net's conditional probability tables using the gradient ascent!
- Converge to the 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 the variable Y_i in the network

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

It can be shown (see the next two slides) that

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

therefore perform gradient ascent by repeatedly

1. **update all** w_{ijk} using the training data D
2. **renormalize the** w_{ijk} to assure

$$w_{ijk} \leftarrow w_{ijk} + \eta \sum_{d \in D} \frac{P_h(y_{ij}, u_{ik} | d)}{w_{ijk}} \quad \sum_j w_{ijk} = 1 \text{ and } 0 \leq w_{ijk} \leq 1$$

Gradient Ascent for Bayes Nets: Calculus

$$\frac{\partial \ln P_h(D)}{\partial w_{ijk}} = \frac{\partial}{\partial w_{ijk}} \ln \prod_{d \in D} P_h(d) = \sum_{d \in D} \frac{\partial \ln P_h(d)}{\partial w_{ijk}} = \sum_{d \in D} \frac{1}{P_h(d)} \frac{\partial P_h(d)}{\partial w_{ijk}}$$

Summing over all values $y_{ij'}$ of Y_i , and $u_{ik'}$ of $U_i = \text{Parents}(Y_i)$:

$$\begin{aligned} \frac{\partial \ln P_h(D)}{\partial w_{ijk}} &= \sum_{d \in D} \frac{1}{P_h(d)} \frac{\partial}{\partial w_{ijk}} \sum_{j'k'} P_h(d|y_{ij'}, u_{ik'}) P_h(y_{ij'}, u_{ik'}) \\ &= \sum_{d \in D} \frac{1}{P_h(d)} \frac{\partial}{\partial w_{ijk}} \sum_{j'k'} P_h(d|y_{ij'}, u_{ik'}) P_h(y_{ij'}|u_{ik'}) P_h(u_{ik'}) \end{aligned}$$

Note that $w_{ijk} \equiv P_h(y_{ij}|u_{ik})$, therefore...

Gradient Ascent for Bayes Nets: Calculus (Cont'd)

$$\begin{aligned}
 \frac{\partial \ln P_h(D)}{\partial w_{ijk}} &= \sum_{d \in D} \frac{1}{P_h(d)} \frac{\partial}{\partial w_{ijk}} P_h(d|y_{ij}, u_{ik}) w_{ijk} P_h(u_{ik}) \\
 &= \sum_{d \in D} \frac{1}{P_h(d)} P_h(d|y_{ij}, u_{ik}) P_h(u_{ik}) \quad (\text{applying Bayes th.}) \\
 &= \sum_{d \in D} \frac{1}{P_h(d)} \frac{P_h(y_{ij}, u_{ik}|d) P_h(d) P_h(u_{ik})}{P_h(y_{ij}, u_{ik})} \\
 &= \sum_{d \in D} \frac{P_h(y_{ij}, u_{ik}|d) P_h(u_{ik})}{P_h(y_{ij}, u_{ik})} = \sum_{d \in D} \frac{P_h(y_{ij}, u_{ik}|d)}{P_h(y_{ij}|u_{ik})} \\
 &= \sum_{d \in D} \frac{P_h(y_{ij}, u_{ik}|d)}{w_{ijk}}
 \end{aligned}$$

Learning Bayes Nets (II, Cont'd)

The **EM** algorithm can also be used.

Repeatedly:

1. Calculate/estimate from data the probabilities of unobserved variables w_{ijk} ,
assuming that the hypothesis h holds
2. Calculate a new h (i.e. new values of w_{ijk}) so to maximize $E[\ln P(D|h)]$,
where D now includes both the observed and the unobserved variables.

Learning Bayes Nets (III)

When the **structure is unknown**, algorithms usually use greedy search to trade off network complexity (add/subtract edges/nodes) against degree of fit to the data.

Example: [Cooper & Herscovitz, 1992] the *K2 algorithm*:

When data is fully observable, use a score metric to choose among alternative networks.

They report an experiment on (re-learning) a network with 37 nodes and 46 arcs describing anesthesia problems in a hospital operating room. Using 3000 examples, the program succeeds almost perfectly: it misses one arc and adds an arc which is not in the original net.