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, liniar regression
   - Provides additional insight into Occam's razor

PLAN

1. Basic Notions
   Bayes' Theorem
   Defining classes of hypotheses:
      Maximum A posteriori Probability (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;
   2.3 The Gibbs classifier;
   2.4 The Naive Bayes and the Joint Bayes classifiers.
      Example: Learning over text data using Naive Bayes
   2.5 The Minimum Description Length (MDL) Principle;
      MDL hypotheses
3. Learning ML hypotheses
   3.1 ML hypotheses in learning real-valued functions
   3.2 ML hypotheses in learning to predict probabilities
   3.3 The Expectation Maximization (EM) algorithm
4. Bayesian Belief Networks

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

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

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 Probability (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.

$P(cancer) = 0.008$	$P(\neg cancer) = 0.992$	$h_1 = cancer, h_2 = \neg cancer$
$P(+\|cancer) = 0.98$	$P(-\|cancer) = 0.02$	$D = \{+, -\}, P(D \mid h_1), P(D \mid h_2)$
$P(+\|\neg cancer) = 0.03$	$P(-\|\neg cancer) = 0.97$	

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

Answer: No.

Indeed, we have to find $\mathrm{argmax}\{P(cancer|+), P(\neg cancer|+)\}$**.**
Applying Bayes theorem (for D = {+}):

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

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

$\boxed{2}$ 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 — the classification of which can take any value v_j in some set V —, 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:

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

Obviously, $h_{MAP} = h_1$**.**

Let's consider an instance x **such that**

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

Question: What is the most probable classification of x**?**

Answer:

$$\begin{aligned} P(-|h_1) = 0, &\quad P(+|h_1) = 1 \\ P(-|h_2) = 1, &\quad P(+|h_2) = 0 \\ P(-|h_3) = 1, &\quad P(+|h_3) = 0 \end{aligned}$$

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

therefore

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

2.3 The Gibbs Classifier

[Opper and Haussler, 1991]

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 [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!

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

2.4 The Naive Bayes Classifier

When to use it:

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

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

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

$$\begin{aligned} v_{MAP} &= \operatorname*{argmax}_{v_j \in V} P(v_j | a_1, a_2 \ldots a_n) = \operatorname*{argmax}_{v_j \in V} \frac{P(a_1, a_2 \ldots a_n | v_j) P(v_j)}{P(a_1, a_2 \ldots a_n)} \\ &= \operatorname*{argmax}_{v_j \in V} P(a_1, a_2 \ldots a_n | v_j) P(v_j) = \operatorname*{argmax}_{v_j \in V} \prod_i P(a_i | v_j) P(v_j) \overset{not.}{=} v_{NB} \end{aligned}$$

This is the so-called *decision rule* of the Naive Bayes classifier.

The Joint Bayes Classifier

$$
\begin{aligned}
v_{MAP} &= \operatorname*{argmax}_{v_j \in V} P(v_j | a_1, a_2 \ldots a_n) = \ldots \\
&= \operatorname*{argmax}_{v_j \in V} P(a_1, a_2 \ldots a_n | v_j) P(v_j) = \operatorname*{argmax}_{v_j \in V} P(a_1, a_2 \ldots a_n, v_j) \overset{not.}{=} v_{JB}
\end{aligned}
$$

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

The Naive Bayes: An Example

Consider again the *PlayTennis* example, and new instance

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

We compute:

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

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

$\ldots$

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

$P(yes)\; P(sun|yes)\; P(cool|yes)\; P(high|yes)\; P(strong|yes) = 0.0053$

$P(no)\; P(sun|no)\; P(cool|no)\; P(high|no)\; P(strong|no) = 0.0206$

$\rightarrow v_{NB} = no$

A Note on The Conditional Independence Assumption of Attributes

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

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 only need that**

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

[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 $Category : Document \rightarrow \{c_1, \ldots, c_n\}$

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

Classify_naive_Bayes_text(Doc)

```
$positions \leftarrow$ all word positions in $Doc$ that contain tokens from $Vocabulary$
Return $v_{NB} = \text{argmax}_{v_j \in V} P(v_j) \prod_{i \in positions} P(a_i = w_k | v_j)$
```

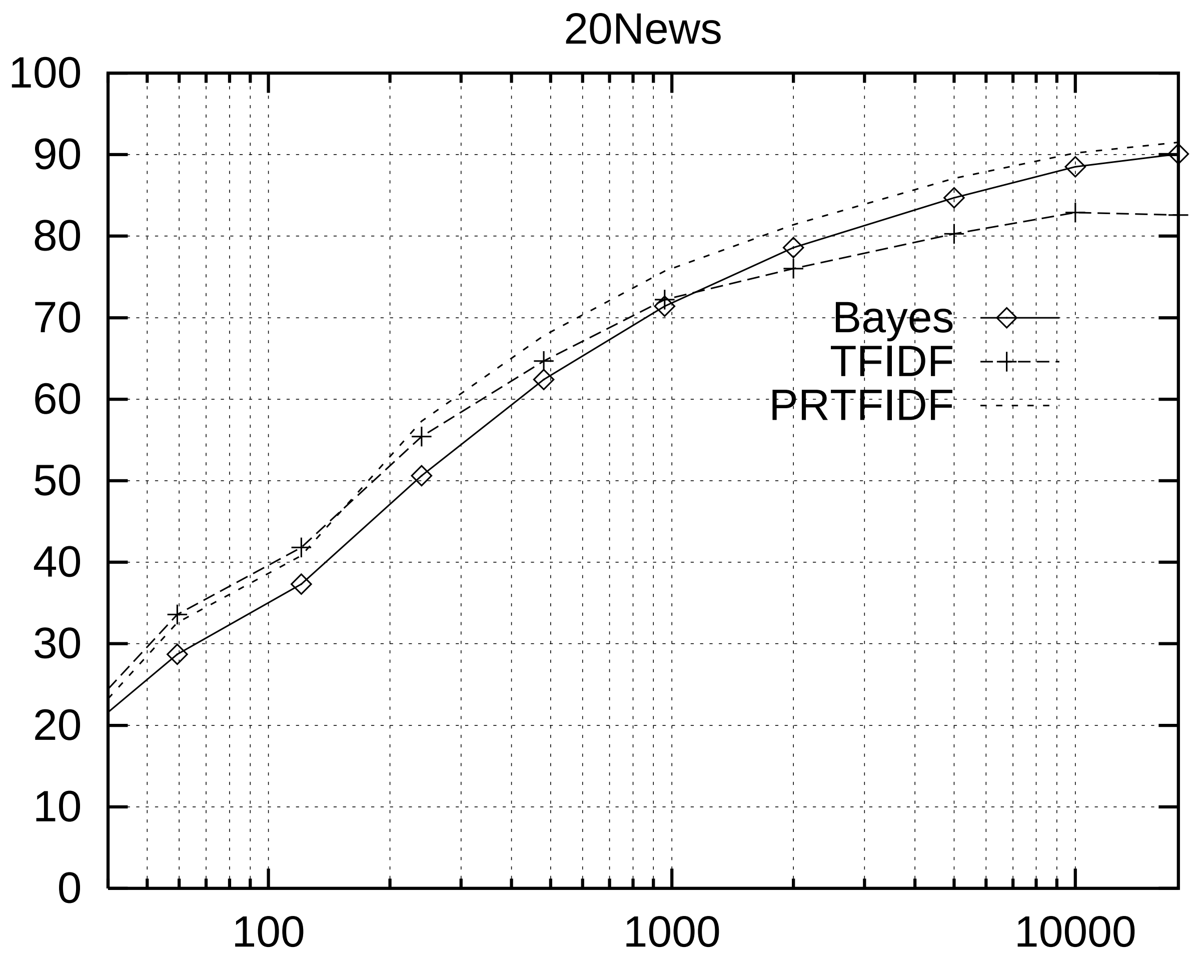
Application: Learning to Classify Usenet News Articles

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



Accuracy vs. Training set size

2.5 The Minimum Description Length Principle

Occam's razor: prefer the shortest hypothesis

Bayes analysis: prefer the hypothesis h_{MAP}

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

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

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

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:

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

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.

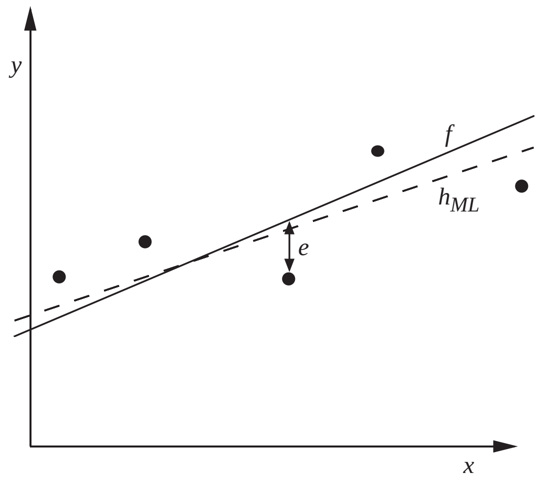
The MDL Principle: Back to Occam's Rasor

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

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

$\boxed{3}$ Learning Maximum Likelihood (ML) Hypothesis

3.1 Learning Real Valued Functions: ML Hypotheses as Least Suquered Error Hypotheses



Problem: Consider learning a real-valued target function $f : X \to \mathbb{R}$ from D, a training set consisting of examples $\langle x_i, d_i \rangle$, $i = 1, \ldots, m$ 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

Considering H, a certain class of functions $h : X \to \mathbb{R}$ such that $h(x_i) = f(x_i)$ and assuming that x_i are mutually independent given h,
the maximum likelihood hypothesis h_{ML} is the one that minimizes the sum of squared errors:

$$h_{ML} \overset{def.}{=} \operatorname*{argmax}_{h \in H} P(D|h) = \arg\min_{h \in H} \sum_{i=1}^{m} (d_i - h(x_i))^2$$

Proof

Note: We will use the probability density function:

$$p(x_0) \overset{def.}{=} \mathbf{lim}_{\epsilon \to 0} \frac{1}{\epsilon} P(x_0 \le x < x_0 + \epsilon)$$

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

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.

3.2 ML hypotheses for Learning Probability Functions

Let us consider a non-deterministic function (i.e. one-to-many relation) $f : X \to \{0, 1\}$.

Given a set of independently drawn examples $D = \{< x_1, d_1 >, \ldots, < x_m, d_m >\}$ where $d_i = f(x_i) \in \{0, 1\}$,

we would like to learn a ML hypothesis for the probability function $g(x) \overset{def.}{=} P(f(x) = 1)$.

For example, $h(x_i) = 0.92$ if $P(\{< x_i, d_i > | d_i = 1\}) = 0.92$.

Proposition: In this setting, $h_{ML} = \boldsymbol{argmax}_{h \in H}\, P(D \mid h)$ maximizes the sum $\sum_{i=1}^{m} [d_i \,\boldsymbol{ln}\, h(x_i) + (1 - d_i)\, \boldsymbol{ln}\, (1 - h(x_i))]$.

Proof:

$$P(D \mid h) = \Pi_{i=1}^{m} P(x_i, d_i \mid h) = \Pi_{i=1}^{m} P(d_i \mid x_i, h) \cdot P(x_i \mid h)$$

It can be assumed that x_i is independent of h, therefore:

$$P(D \mid h) = \Pi_{i=1}^{m} P(d_i \mid x_i, h) \cdot P(x_i)$$

Proof (continued):

What we wanted to compute is $h(x_i) = P(d_i = 1 \mid x_i, h)$.
In a more general form:

$$P(d_i \mid x_i, h) = \begin{cases} h(x_i) & \textbf{\textit{if}} \;\; d_i = 1 \\ 1 - h(x_i) & \textbf{\textit{if}} \;\; d_i = 0 \end{cases}$$

In a more convenient mathematical form: $P(d_i \mid x_i, h) = h(x_i)^{d_i}(1 - h(x_i))^{1-d_i}$.

$$\begin{aligned} \Rightarrow h_{ML} &= \textbf{\textit{argmax}}_{h \in H} \, \Pi_{i=1}^{m}[h(x_i)^{d_i}(1 - h(x_i))^{1-d_i} P(x_i)] \\ &= \textbf{\textit{argmax}}_{h \in H} \, \Pi_{i=1}^{m} h(x_i)^{d_i}(1 - h(x_i))^{1-d_i} \cdot \Pi_{i=1}^{m} P(x_i) \\ &= \textbf{\textit{argmax}}_{h \in H} \, \Pi_{i=1}^{m} h(x_i)^{d_i}(1 - h(x_i))^{1-d_i} \\ &= \textbf{\textit{argmax}}_{h \in H} \sum_{i=1}^{m}[d_i \, \textbf{\textit{ln}} \, h(x_i) + (1 - d_i) \, \textbf{\textit{ln}} \, (1 - h(x_i))] \end{aligned}$$

Note: **The quantity $-\sum_{i=1}^{m}[d_i \, \textbf{\textit{ln}} \, h(x_i) + (1 - d_i) \, \textbf{\textit{ln}} \, (1 - h(x_i))]$ is called cross-entropy; the above h_{ML} minimizes this quantity.**

3.3 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 Gausseans
- Learn Hidden Markov Models
- Learn Probabilistic Context Free Grammars
- Train Radial Basis Function Networks
- Train Bayesian Belief Networks

The General EM Problem

Given

- **observed data** $X = \{x_1, \ldots, x_m\}$
  independently generated using the parameterized distributions/hypotheses $h_1, \ldots, h_m$
- **unobserved data** $Z = \{z_1, \ldots, z_m\}$

determine

$\hat{h}$ **that (locally) maximizes** $P(X|h)$.

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 $h^{(t)}$ to estimate [the probabilities associated to the values of] the unobserved variables Z, and further on compute their expectations, $E[Z]$.**
2. **The expected values of the unobserved variables Z are used to calculate an improved hypothesis $h^{(t+1)}$, based on maximizing the mean of a log-verosimility function: $E[\ln P(Y|h)|X, h^{(t)}]$, where $Y = \{y_1, \ldots, y_m\}$ is the complete (observed and unobserved) data, i.e. $y_i = (x_i, z_i)$, for $i = 1, \ldots, m$.**

The General EM Algorithm

Repeat the following two steps until convergence is reached:

Estimation (E) step:

Calculate the log likelihood function

$$Q(h|h^{(t)}) \overset{not.}{=} E[\ln P(Y|h)|X, h^{(t)}]$$

where $Y = X \cup Z$**.**

Maximization (M) step:

Replace hypothesis $h^{(t)}$ **by the hypothesis** $h^{(t+1)}$ **that maximizes this** Q **function.**

$$h^{(t+1)} \leftarrow \underset{h}{\operatorname{argmax}}\, Q(h|h^{(t)})$$

The EM algorithmic Schema

- ○ **idea:** replace missing values by estimated values
- initialize parameters with arbitrary values
- estimate *missing values* based on *current parameter values*
- re-estimate *parameters* using the *complete* data
- repeat the previous two steps until *convergence*

$$\begin{tikzcd}
t=0 \arrow[d] & & \\
h^{(t)} \arrow[rr] & & E[Z \mid X, h^{(t)}] \arrow[ddl] \\
& \ln P(X|h) & \\
& h^{(t+1)} = \mathop{\mathit{argmax}}_{h} \; E_{P(Z|X;\, h^{(t)})}\,[\ln P(X,Z|h)] \arrow[uul, "++t"] &
\end{tikzcd}$$

Baum-Welch Theorem

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

$\boxed{4}$ 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 $P(X|Y, Z) = P(X|Z)$

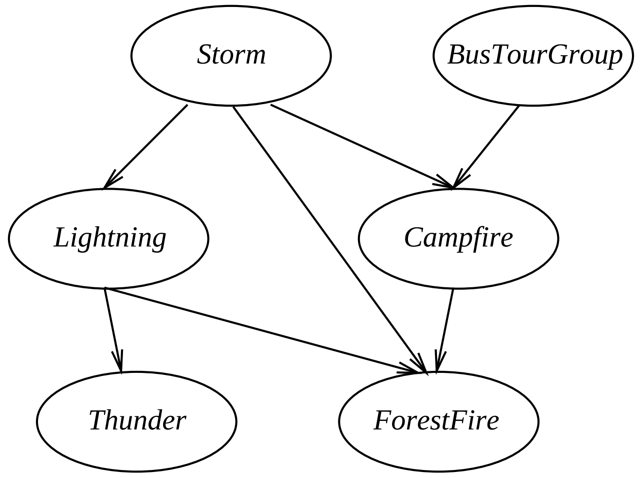
Note: **Naive Bayes uses conditional independence to justify**

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

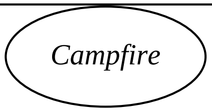
Generalizing the above definition:

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

A Bayes Net



	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



The network is defined by

- A directed acyclic graph, represening 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: $P(Thunder|ForestFire, Lightning) = P(Thunder|Lightning)$

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

A Bayes Net (Cont'd)

represents the joint probability distribution over all variables $Y_1, Y_2, \ldots, Y_n$:

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

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

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

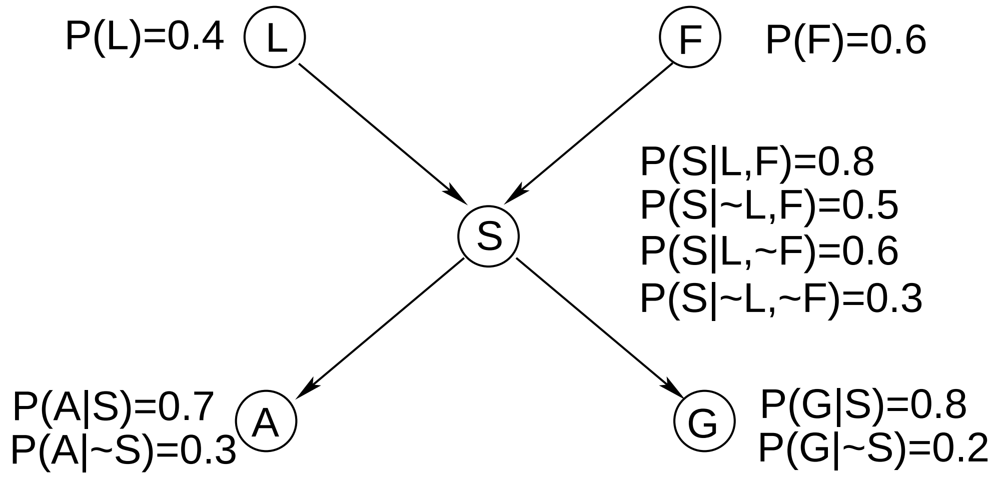
In our example: $P(Storm, BusTourGroup, \ldots, ForestFire)$

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

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

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**

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

2. **renormalize the w_{ijk} to assure**

$$\sum_j w_{ijk} = 1 \textbf{ and } 0 \leq w_{ijk} \leq 1$$

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 $U_i = Parents(Y_i)$:

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

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

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

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

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/substract 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.