
Aims

This lecture will enable you to describe machine learning in the framework of Bayesian statistics and reproduce key algorithms derived from this approach. Following it you should be able to:

- reproduce basic definitions of useful probabilities
- derive Bayes theorem and the formulae for MAP and ML hypotheses
- describe concept learning in Bayesian terms
- derive formula for maximum likelihood hypothesis for numerical prediction by minimising sum of squared errors
- define the Minimum Description Length principle
- reproduce Bayes Optimal Classifier formula
- describe the Gibbs and Naive Bayes algorithms

[Recommended reading: Mitchell, Chapter 6]
[Recommended exercises: 6.1, 6.2, (6.4)]

Relevant WEKA programs:
weka.classifiers.bayes package (BayesNet, NaiveBayes, etc.)

See also the R Project for Statistical Computing:
http://www.r-project.org/
Uncertainty

As far as the laws of mathematics refer to reality, they are not certain; as far as they are certain, they do not refer to reality.

–Albert Einstein

Two Roles for Bayesian Methods

Provides practical learning algorithms:

- Naive Bayes learning
- Bayesian belief network learning
- Combine prior knowledge (prior probabilities) with observed data
- Requires prior probabilities

Provides useful conceptual framework:

- Provides “gold standard” for evaluating other learning algorithms
- Additional insight into Occam’s razor

Bayes Theorem

\[ P(h|D) = \frac{P(D|h)P(h)}{P(D)} \]

- \( P(h) \) = prior probability of hypothesis \( h \)
- \( P(D) \) = prior probability of training data \( D \)
- \( P(h|D) \) = probability of \( h \) given \( D \)
- \( P(D|h) \) = probability of \( D \) given \( h \)

Choosing Hypotheses

\[ P(h|D) = \frac{P(D|h)P(h)}{P(D)} \]

Generally want the most probable hypothesis given the training data

*Maximum a posteriori* hypothesis \( h_{MAP} \):

\[
\begin{align*}
  h_{MAP} &= \arg \max_{h \in H} P(h|D) \\
  &= \arg \max_{h \in H} \frac{P(D|h)P(h)}{P(D)} \\
  &= \arg \max_{h \in H} P(D|h)P(h)
\end{align*}
\]
Choosing Hypotheses

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

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

Applying Bayes Theorem

Does patient have cancer or not?

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

$$P(\oplus | cancer) = .98 \quad P(\ominus | cancer) = .92$$
$$P(\oplus | \neg cancer) = .03 \quad P(\ominus | \neg cancer) = .97$$

Thus $h_{MAP} = \ldots$
Applying Bayes Theorem

Does patient have cancer or not?
We can find the maximum a priori (MAP) hypothesis

\[
P(\oplus \mid \text{cancer})P(\text{cancer}) = 0.98 \times 0.008 = 0.00784
\]
\[
P(\oplus \mid \neg \text{cancer})P(\neg \text{cancer}) = 0.03 \times 0.992 = 0.02976
\]

Thus \( h_{MAP} = \neg \text{cancer}. \)

Also note: posterior probability of hypothesis \text{cancer} higher than prior.

Basic Formulas for Probabilities

Also worth remembering:

- **Conditional Probability**: probability of \( A \) given \( B \):
  \[
P(A \mid B) = \frac{P(A \land B)}{P(B)}
\]
- Rearrange sum rule to get:
  \[
P(A \land B) = P(A) + P(B) - P(A \lor B)
\]

**Exercise**: Derive Bayes Theorem.

Brute Force MAP Hypothesis Learner

1. For each hypothesis \( h \) in \( H \), calculate the posterior probability
   \[
P(h \mid D) = \frac{P(D \mid h)P(h)}{P(D)}
\]
2. Output the hypothesis \( h_{MAP} \) with the highest posterior probability
   \[
h_{MAP} = \arg\max_{h \in H} P(h \mid D)
\]
Relation to Concept Learning

Consider our usual concept learning task

- instance space $X$, hypothesis space $H$, training examples $D$
- consider the FindS learning algorithm (outputs most specific hypothesis from the version space $V_{S_H,D}$)

What would Bayes rule produce as the MAP hypothesis?

Does FindS output a MAP hypothesis??

Relation to Concept Learning

Assume fixed set of instances $(x_1, \ldots, x_m)$
Assume $D$ is the set of classifications $D = (c(x_1), \ldots, c(x_m))$
Choose $P(D|h)$:

- $P(D|h) = 1$ if $h$ consistent with $D$
- $P(D|h) = 0$ otherwise

Choose $P(h)$ to be uniform distribution:

- $P(h) = \frac{1}{|H|}$ for all $h$ in $H$

Then:

$$P(h|D) = \begin{cases} \frac{1}{|V_{S_H,D}|} & \text{if } h \text{ is consistent with } D \\ 0 & \text{otherwise} \end{cases}$$
Evolution of Posterior Probabilities

\[ P(h) \quad P(h|D_1) \quad P(h|D_1, D_2) \]

(a) hypotheses

(b) hypotheses

(c) hypotheses

Relation to Concept Learning

Every hypothesis consistent with \( D \) is a MAP hypothesis, if

- uniform probability over \( H \)
- target function \( c \in H \)
- deterministic, noise-free data
- etc. (see above)

FindS will output a MAP hypothesis, even though it does not explicitly use probabilities in learning.

Bayesian interpretation of inductive bias: use Bayes theorem, define restrictions on \( P(h) \) and \( P(D|h) \)

Characterizing Learning Algorithms by Equivalent MAP Learners

Inductive system

Training examples \( D \)

Hypothesis space \( H \)

Candidate Elimination Algorithm

Output hypotheses

Equivalent Bayesian inference system

Training examples \( D \)

Hypothesis space \( H \)

\( P(h) \) uniform

\( P(D|h) = 0 \) if inconsistent,
\( = 1 \) if consistent

Prior assumptions made explicit

Output hypotheses

Learning A Real Valued Function

\( f \)

\( h_{ML} \)
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:

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

Recall that we treat each probability $p(D | h)$ as if $h = f$

Maximize natural log to give simpler expression . . .

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

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

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

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

Learning A Real Valued Function

Learning to Predict Probabilities

Consider predicting survival probability from patient data . . .

e.g. for same set of symptoms, 92% survive but 8% do not.

Sources of uncertainty: unobservable features, inherent randomness in biological processes.

Target function $f(x)$ whose output is a probabilistic function of input.

Training examples $\langle x_i, d_i \rangle$, where $d_i$ is 1 or 0. 1 indicates survival and 0 otherwise.

Want to train neural network to output a probability given $x_i$ (rather than a 0 or 1).

$$f' : X \rightarrow [0, 1]$$ such that $f'(x) = P(f(x) = 1)$
Learning to Predict Probabilities

Set up probability of data given hypothesis as

\[ P(D|h) = \prod_{i=1}^{m} P(x_i, d_i|h) \]

Then can show

\[ h_{ML} = \arg \max_{h \in H} \sum_{i=1}^{m} d_i \ln h(x_i) + (1 - d_i) \ln(1 - h(x_i)) \]

Probability that flipping each of \( m \) distinct coins will produce the data \( \langle d_1, \ldots, d_m \rangle \), assuming each coin \( x_i \) has probability \( h(x_i) \) of producing a heads.

Minimum Description Length Principle

Once again, the MAP hypothesis

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

Which is equivalent to

\[ h_{MAP} = \arg \max_{h \in H} \log_2 P(D|h) + \log_2 P(h) \]

Or

\[ h_{MAP} = \arg \min_{h \in H} -\log_2 P(D|h) - \log_2 P(h) \]

From information theory:

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

Minimum Description Length Principle

Interestingly, this is an expression about a quantity of \( \text{bits} \).

The maximum likelihood hypothesis is to be obtained in the neural network by gradient ascent.

Can derive a weight update rule for a sigmoid unit:

\[ w_{jk} \leftarrow w_{jk} + \Delta w_{jk} \]

where

\[ \Delta w_{jk} = \eta \sum_{i=1}^{m} (d_i - h(x_i)) x_{ijk} \]

Similar to derivation of gradient ascent for minimizing sum of squared error.
Minimum Description Length Principle

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

Note well: assumes optimal encodings, when the priors and likelihoods are known. In practice, this is difficult, and makes a difference.

Occam’s razor: prefer the shortest hypothesis

MDL: prefer the hypothesis $h$ that minimizes

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

where $L_C(x)$ is the description length of $x$ under optimal encoding $C$

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!

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
  - i.e., prefer the hypothesis that minimizes

$$\text{length}(h) + \text{length(misclassifications})$$
Most Probable Classification of New Instances

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

- What’s most probable classification of \( x \)?

Bayes Optimal Classifier

Bayes optimal classification:

\[
\arg \max_{v_j \in V} \sum_{h_i \in H} P(v_j|h_i)P(h_i|D)
\]

Example:

\[
P(h_1|D) = 0.4, \quad P(-|h_1) = 0, \quad P(+|h_1) = 1 \\
P(h_2|D) = 0.3, \quad P(-|h_2) = 1, \quad P(+|h_2) = 0 \\
P(h_3|D) = 0.3, \quad P(-|h_3) = 1, \quad P(+|h_3) = 0
\]

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:

\[
E[error_{Gibbs}] \leq 2E[error_{BayesOptimal}]
\]
**Gibbs Classifier**

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

**Naive Bayes Classifier**

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

When to use

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

Successful applications:

- Diagnosis
- Classifying text documents

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

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

$$ v_{MAP} = \arg\max_{v_j \in V} P(v_j|a_1, a_2 \ldots a_n) $$

Naive Bayes assumption:

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

- Attributes are statistically independent (given the class value)
  - which means knowledge about the value of a particular attribute tells us nothing about the value of another attribute (if the class is known)

which gives

**Naive Bayes classifier**:

$$ v_{NB} = \arg\max_{v_j \in V} P(v_j) \prod_i P(a_i|v_j) $$
Naive Bayes 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)

$v_{NB} = \arg\max_{v_j \in V} \hat{P}(v_j) \prod_{a_i \in x} \hat{P}(a_i|v_j)$

Naive Bayes Example

Consider *PlayTennis* again . . .

<table>
<thead>
<tr>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Windy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunny</td>
<td>Yes</td>
<td>Yes</td>
<td>False</td>
</tr>
<tr>
<td>Overcast</td>
<td>Yes</td>
<td>No</td>
<td>True</td>
</tr>
<tr>
<td>Rainy</td>
<td>No</td>
<td>No</td>
<td>False</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Windy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunny</td>
<td>Yes</td>
<td>Yes</td>
<td>False</td>
</tr>
<tr>
<td>Overcast</td>
<td>Yes</td>
<td>No</td>
<td>True</td>
</tr>
<tr>
<td>Rainy</td>
<td>No</td>
<td>No</td>
<td>False</td>
</tr>
</tbody>
</table>

Say we have the new instance:

$(Outlk = \text{sun}, Temp = \text{cool}, Humid = \text{high}, Wind = \text{true})$

We want to compute:

$v_{NB} = \arg\max_{v_j \in \{\text{"yes"}, \text{"no"}\}} P(v_j) \prod_{a_i \in x} P(a_i|v_j)$

Naive Bayes Example

So we first calculate the likelihood of the two classes, “yes” and “no”

for “yes” $= P(y) \times P(\text{sun}|y) \times P(\text{cool}|y) \times P(\text{high}|y) \times P(\text{true}|y)$

$0.0053 = \frac{9}{14} \times \frac{2}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{3}{9}$

for “no” $= P(n) \times P(\text{sun}|n) \times P(\text{cool}|n) \times P(\text{high}|n) \times P(\text{true}|n)$

$0.0206 = \frac{5}{14} \times \frac{3}{5} \times \frac{1}{5} \times \frac{4}{5} \times \frac{3}{5}$
Naive Bayes Example

Then convert to a probability by normalisation

\[ P(\text{"yes"}) = \frac{0.0053}{0.0053 + 0.0206} = 0.205 \]
\[ P(\text{"no"}) = \frac{0.0206}{0.0053 + 0.0206} = 0.795 \]

The Naive Bayes classification is "no".

---

Naive Bayes: Subtleties

1. Conditional independence assumption is often violated

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

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

\[ \arg\max_{v_j \in V} \hat{P}(v_j) \prod_i \hat{P}(a_i|v_j) = \arg\max_{v_j \in V} P(v_j)P(a_1 \ldots , a_n|v_j) \]

i.e. maximum probability is assigned to correct class

• see [Domingos & Pazzani, 1996] for analysis
• Naive Bayes posteriors often unrealistically close to 1 or 0
• adding too many redundant attributes will cause problems (e.g. identical attributes)

Naive Bayes: “zero-frequency” problem

2. what if none of the training instances with target value \( v_j \) have attribute value \( a_i \)? Then

\[ \hat{P}(a_i|v_j) = 0, \text{ and...} \]
\[ \hat{P}(v_j) \prod_i \hat{P}(a_i|v_j) = 0 \]

Pseudo-counts add 1 to each count (a version of the Laplace Estimator)

---

Naive Bayes: “zero-frequency” problem

• In some cases adding a constant different from 1 might be more appropriate

• Example: attribute outlook for class yes

\[
\begin{align*}
\text{Sunny} & \quad \text{Overcast} & \quad \text{Rainy} \\
\frac{2+\mu}{9+\mu} & \quad \frac{4+\mu}{9+\mu} & \quad \frac{3+\mu}{9+\mu}
\end{align*}
\]

• Weights don’t need to be equal (if they sum to 1) – a form of prior

\[
\begin{align*}
\text{Sunny} & \quad \text{Overcast} & \quad \text{Rainy} \\
\frac{2+\mu p_1}{9+\mu} & \quad \frac{4+\mu p_2}{9+\mu} & \quad \frac{3+\mu p_3}{9+\mu}
\end{align*}
\]
Naive Bayes: “zero-frequency” problem

This generalisation is a Bayesian estimate for \( \hat{P}(a_i|v_j) \)

\[
\hat{P}(a_i|v_j) \leftarrow \frac{n_c + mp}{n + m}
\]

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 prior estimate for \( \hat{P}(a_i|v_j) \)
- \( m \) is weight given to prior (i.e. number of “virtual” examples)

This is called the \( m \)-estimate of probability.

Naive Bayes: missing values

- Training: instance is not included in frequency count for attribute value-class combination
- Classification: attribute will be omitted from calculation

Naive Bayes: numeric attributes

- Usual assumption: attributes have a normal or Gaussian probability distribution (given the class)
- The probability density function for the normal distribution is defined by two parameters:
  The sample mean \( \mu \):
  \[
  \mu = \frac{1}{n} \sum_{i=1}^{n} x_i
  \]
  The standard deviation \( \sigma \):
  \[
  \sigma = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \mu)^2
  \]

Then we have the density function \( f(x) \):

\[
f(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}
\]

Example: continuous attribute temperature with mean = 73 and standard deviation = 6.2. Density value

\[
f(\text{temperature} = 66|"yes") = \frac{1}{\sqrt{2\pi\times6.2}} e^{-\frac{(66-73)^2}{2\times6.2^2}} = 0.0340
\]

Missing values during training are not included in calculation of mean and standard deviation.
Naive Bayes: numeric attributes

Note: the normal distribution is based on the simple exponential function

\[ f(x) = e^{-|x|^m} \]

As the power \( m \) in the exponent increases, the function approaches a step function.

Where \( m = 2 \)

\[ f(x) = e^{-|x|^2} \]

and this is the basis of the normal distribution – the various constants are the result of scaling so that the integral (the area under the curve from \(-\infty \) to \(+\infty \)) is equal to 1.


Learning to Classify Text

Target concept \( \text{Interesting?} : \text{Document} \rightarrow \{+, -\} \)

1. Represent each document by vector of words
   - one attribute per word position in document

2. Learning: Use training examples to estimate
   - \( P(+) \)
   - \( P(-) \)
   - \( P(\text{doc}|+) \)
   - \( P(\text{doc}|-) \)

Naive Bayes conditional independence assumption

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

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

one more assumption: \( P(a_i = w_k|v_j) = P(a_m = w_k|v_j), \forall i, m \)

“bag of words”
Learning to Classify Text

\textsc{learn-naive-bayes-text}(Examples, V)

// collect all words and other tokens that occur in Examples
Vocabulary ← all distinct words and other tokens in Examples
// calculate the required $P(v_j)$ and $P(w_k|v_j)$ probability terms
for each target value $v_j$ in $V$ do
\hspace{1em}docs_j ← subset of Examples for which the target value is $v_j$
\hspace{1em}$P(v_j) ← \frac{|docs_j|}{|Examples|}$
\hspace{1em}$Text_j ← a single document created by concatenating all members of docs_j$
\hspace{1em}$n ← total number of words in Text_j (counting duplicate words multiple times)$
\hspace{1em}for each word $w_k$ in Vocabulary
\hspace{2em}$n_k ← number of times word $w_k$ occurs in Text_j$
\hspace{2em}$P(w_k|v_j) ← \frac{n_k+1}{n+|Vocabulary|}$

Application: 20 Newsgroups

Given: 1000 training documents from each group
Learning task: classify each new document by newsgroup it came from

\begin{align*}
\text{comp.graphics} & \quad \text{misc.forsale} \\
\text{comp.os.ms-windows.misc} & \quad \text{rec.autos} \\
\text{comp.sys.ibm.pc.hardware} & \quad \text{rec.motorcycles} \\
\text{comp.sys.mac.hardware} & \quad \text{rec.sport.baseball} \\
\text{comp.windows.x} & \quad \text{rec.sport.hockey} \\
\text{alt.atheism} & \quad \text{sci.space} \\
\text{soc.religion.christian} & \quad \text{sci.crypt} \\
\text{talk.religion.misc} & \quad \text{sci.electronics} \\
\text{talk.politics.mideast} & \quad \text{sci.med} \\
\text{talk.politics.misc} & \quad \text{} \\
\text{talk.politics.guns} & \quad \text{}
\end{align*}

Naive Bayes: 89% classification accuracy

Classification: 20 Newsgroups

\textsc{classify-naive-bayes-text}(Doc)

\begin{itemize}
\item \textit{positions} ← all word positions in Doc that contain tokens found in Vocabulary
\item Return $v_{NB}$, where
\hspace{1em}$v_{NB} = \arg\max_{v_j \in V} P(v_j) \prod_{i \in \text{positions}} P(a_i|v_j)$
\end{itemize}

Article from rec.sport.hockey

Path: cantaloupe.srv.cs.cmu.edu!das-news.harvard.edu!logicse!uwm.edu
From: xxx@yyy.zzz.edu (John Doe)
Subject: Re: This year’s biggest and worst (opinion)...
Date: 5 Apr 93 09:53:39 GMT

I can only comment on the Kings, but the most obvious candidate for pleasant surprise is Alex Zhitnik. He came highly touted as a defensive defenseman, but he's clearly much more than that. Great skater and hard shot (though wish he were more accurate). In fact, he pretty much allowed the Kings to trade away that huge defensive liability Paul Coffey. Kelly Hrudey is only the biggest disappointment if you thought he was any good to begin with. But, at best, he's only a mediocre goaltender. A better choice would be Tomas Sandstrom, though not through any fault of his own, but because some thugs in Toronto decided ...
Learning Curve for 20 Newsgroups

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

\[
P(\text{Thunder} | \text{Rain, Lightning}) = P(\text{Thunder} | \text{Lightning})
\]
A Bayesian Belief Network or Bayes Net is:

- a directed acyclic graph, plus
- a set of associated conditional probabilities

A Bayes Net represents a set of conditional independence assertions:

- Each node is conditionally independent of its nondescendants, given its immediate predecessors

A Bayes Net factors a joint probability distribution over all variables:

- e.g., $P(\text{Storm}, \text{BusTourGroup}, \ldots, \text{ForestFire})$
- in general,

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

where $\text{Parents}(Y_i)$ denotes immediate predecessors of $Y_i$ in graph

- so, joint distribution is fully defined by graph, plus the $P(y_i | \text{Parents}(Y_i))$
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

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

e.g., if $Y_i = \text{Campfire}$, then $u_{ik}$ might be $\langle \text{Storm} = T, \text{BusTourGroup} = F \rangle$
Gradient Ascent for Bayes Nets

Perform gradient ascent by repeatedly

1. update all \( w_{ijk} \) using training data \( D \)
\[
  w_{ijk} \leftarrow w_{ijk} + \eta \sum_{d \in D} P_h(y_{ij}, u_{ik}|d) \frac{1}{w_{ijk}}
\]

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/subtract 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
Finite mixtures

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

Called *finite mixtures* because there is only a finite number of *generating distributions* being represented.

---

Generating Data from Mixture of $k$ Gaussians

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 $j$th Gaussian
- $x_i$ observable
- $z_{ij}$ unobservable

---

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

$$E[z_{ij}] = \frac{p(x = x_i | \mu = \mu_j)}{\sum_{n=1}^{2} p(x = x_i | \mu = \mu_n)} = \frac{e^{-\frac{1}{2}x_i^2(x_i-\mu_j)^2}}{\sum_{n=1}^{2} e^{-\frac{1}{2}x_i^2(x_i-\mu_n)^2}}$$

M step: Calculate a new maximum likelihood hypothesis $h' = \langle \mu'_1, \mu'_2 \rangle$, assuming the value taken on by each hidden variable $z_{ij}$ is its expected value $E[z_{ij}]$ calculated above. Replace $h = \langle \mu_1, \mu_2 \rangle$ by $h' = \langle \mu'_1, \mu'_2 \rangle$.

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

i.e.

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

EM for Estimating $k$ Means

E step: Calculate probabilities for unknown parameters for each instance

M step: Estimate parameters based on the probabilities

In $k$-means the probabilities are stored as instance weights.

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 $E[\ln P(Y|h)]$

---

**Extending the mixture model**

- Using more than two distributions
- Several attributes: easy if independence assumed
- Correlated attributes: difficult
  - Modeled jointly using a bivariate normal distribution with a (symmetric) covariance matrix
  - With $n$ attributes this requires estimating $n + n(n+1)/2$ parameters

---

**EM for Estimating $k$ Means**

**Many uses:**
- Train Bayesian belief networks
- Unsupervised clustering (e.g., $k$ means)
- Hidden Markov Models

---

**Extending the mixture model**

- Nominal attributes: easy if independence assumed
- Correlated nominal attributes: difficult
  - Two correlated attributes result in $v_1 \times v_2$ parameters
- Missing values: easy
- Distributions other than the normal distribution can be used:
  - "log-normal" if predetermined minimum is given
  - "log-odds" if bounded from above and below
  - Poisson for attributes that are integer counts
- Cross-validation can be used to estimate $k$ - time consuming!
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.

$$h \leftarrow \text{argmax}_{h'} Q(h'|h)$$

On Bayesian Learning

... if the prediction of a further observation is the sole objective, a Bayesian mixture of all tenable models is hard to beat. But is this really inferring anything about the source of the data?

... Also, would we be happy with a scientist who proposed a Bayesian mixture of a countably infinite set of incompatible models for electromagnetic fields?


Summary: Bayesian Learning

- Well-founded framework for learning where the model and the outputs are characterised using probabilities
- How to get the probabilities?
  - problem of prior probabilities, also
  - think carefully about likelihoods, choice of parameters, etc.
  - Empirical Bayes (estimate from data)?
- Machine Learning techniques – algorithmically clear, extensive empirical validation, powerful models
- Bayesian Learning – shows how to set up Machine Learning methods in more formal probabilistic and statistical frameworks
  - more work needed