Bayesian Learning [*]

Anca Ralescu

Valentine Day 2012
revised Oct 2012
revised Sept 2015

TENTATIVE LIST OF TOPICS

- Bayes Theorem
- *Maximum a posteriori Probability* (MAP) hypotheses
- *Maximum likelihood*(ML) hypotheses
- MAP learners
- Minimum description length principle
- Bayes optimal classifier
- Naive Bayes learner
- Example: Learning over text data
- Bayesian belief networks
- Expectation Maximization algorithm

Two Roles for Bayesian Methods

Provide practical learning algorithms

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

Provide useful conceptual framework

- Provides "gold standard" for evaluating other learning algorithms
- Additional insight into Occam's razor

Bayes Theorem

The basic result underlying the Bayesian approach is the the Bayes Theorem.

[*]based on Tom Mitchell's slides

Idea

Let X and Y denote two events in the same probability space. Adopt a graphical representation of the conditional probability

$$X \longrightarrow Y \text{ is used to model } P(Y|X) \text{ the conditional probability of } Y \text{ given } X$$

Under certain conditions reverse the arrow $X \longrightarrow Y$ to infer something about X given knowledge of Y. This is then expressed as

$$Y \longrightarrow X \text{ or } P(X|Y)$$

This reversal of the arrow is stated by the result known as Bayes Theorem.

How do we use this idea in the learning framework?

As usual we denote D the set of examples for a concept C, and h a (possible) hypothesis about this concept.

Bayes Theorem states:

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

where

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

Choosing Hypotheses

Recall, that one of the important issues in machine learning (concept learning) is that of choosing a "correct hypothesis". But what do we **really** mean by a correct hypothesis?

We have seen that the more correct a hypothesis is on the training set, the more that hypothesis is in danger of over-fitting the training set, and therefore to perform very poorly on the test set.

Thus one way to avoid this problem is, as I already mentioned, to give up performance on the training set in order to improve performance on the test set.

But this introduces an element of uncertainty, which often (but not always) we model using probability theory.

Equation (1) allows us to express the goal of obtaining the best hypothesis as a simple optimization problem as follows:

We want the **most probable hypothesis** given the training data, or, using the terminology from the Bayes theorem, we want the *Maximum a posteriori* hypothesis h_{MAP}.

We define h_{MAP} as that hypothesis which maximizes the posterior probability over all possible hypotheses. That is

$$\begin{aligned} h_{MAP} &= \arg\max_{h\in H} P(h|D) \\ &= \arg\max_{h\in H} \frac{P(D|h)P(h)}{P(D)} \\ &= \arg\max_{h\in H} P(D|h)P(h) \qquad (2) \end{aligned}$$

If assume a uniform distribution on the hypothesis space, that is, $\forall h \in H$, $P(h) = \frac{1}{|H|}$, then (2) can be further simplified to

$$h_{MAP} = \arg\max_{h\in H} P(D|h) \qquad (3)$$

The quantity $P(D|h)$ in (3) is called *likelihood*, and therefore the solution to (3) is referred to as the *Maximum Likelihood Hypothesis*(ML), or,

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

Note that in fact, since D is usually discrete, that is $D = \{x_1, \ldots, x_n\}$, $P(D|h)$ really means $P(x_1, \ldots, x_n|h)$, that is the *joint probability of* $x_1, \ldots, x_n$.

The Bayesian approach is very common in diagnostic problems. Let us look at a very simple, and common example.

EXAMPLE 1 *(Bayes Theorem for diagnosis)*
A patient is suspected of a very dangerous disease. The 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 type of disease.*

The question that the patient is very anxious to have answered is this: Do I have the disease?

Unfortunately, no medical knowledge can answer that question, except qualifying the answer by a certainty/uncertainty.

Moreover, to provide that answer, several pieces of information must be taken in consideration, as follows:

- *How likely is for* **that** *patient to have the disease*
- *How likely is that disease* **in general**
- *How reliable is the test administered*

Assume that we have the following:

$$\begin{array}{rr} P(disease) = 0.008 & P(\neg disease) = 0.992 \\ P(+|disease) = 0.98 & P(-|disease) = 0.02 \\ P(+|\neg disease) = 0.03 & P(-|\neg disease) = 0.97 \end{array}$$

Using the Bayes Theorem, one can obtain $P(disease)$, and we will see just how this is done.

Basic Formulas for Probabilities

- *Product Rule*: probability $P(A \wedge B)$ of a conjunction/intersection 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/union 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

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

- Probability of the complement of an event (this follows from the theorem of total probabilities)

$$P(\overline{A}) = 1 - P(AP$$

Brute Force MAP Hypothesis Learner

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

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

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 $VS_{H,D}$)

Questions

- What would Bayes rule produce as the MAP hypothesis?
- **Does $FindS$ output a MAP hypothesis ?**

To answer these questions we need to make assumptions:

- Assume fixed set of instances $\langle x_1, \ldots, x_m \rangle$
- Assume D is the set of classifications $D = \langle c(x_1), \ldots, c(x_m) \rangle$

Then, we can proceed as follows:

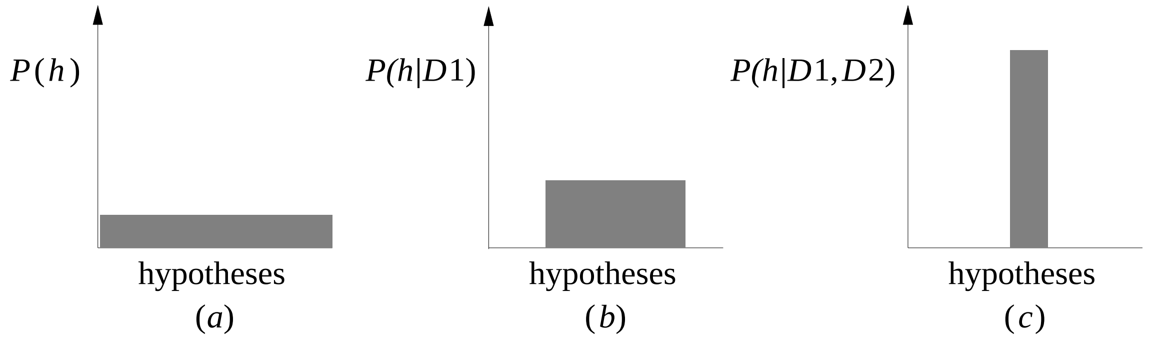


Figure 1: Evolution of Posterior probabilities

1. Choose $P(D|h)$ such that:
   - $P(D|h) = 1$ if h consistent with D
   - $P(D|h) = 0$ otherwise
2. Choose $P(h)$ to be *uniform* distribution
   - $P(h) = \frac{1}{|H|}$ for all h in H

Then,

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

Characterizing Learning Algorithms by Equivalent MAP Learners

Figure (1) illustrates the evolution of the posterior probabilities as data is presented.

- Every hypothesis consistent with D is a MAP hypothesis.
- Define *consistent learners*: a learner that outputs only a consistent hypothesis.
- **IF**
  1. the probability distribution on H uniform, i.e. $p(h) = c, \forall h \in H$, and
  2. D is noise-free (no conflicting data), and
  3. D is deterministic, that is, $P(D|h) = 1$ if D and h are consistent, otherwise $P(D|h) = 0$
- **THEN every consistent learner outputs a MAP hypothesis** .

EXAMPLE 2 *Find-S outputs a MAP hypothesis, even though it* **does not explicitly manipulate probabilities.**

Other distributions for $P(D|h)$ and $P(h)$ for which the same is true: any distributions that favor more specific hypotheses: $P(h_1) \geq P(h_2)$ if h_1 is more specific *than h_2.*

Therefore, by identifying $P(D|h)$ and $P(h)$ under which a learner outputs the MAP hypothesis, the Bayesian framework is one way to characterize the *behavior of learning algorithms*, even when no explicit use is made of these probabilities.

For characterization of Find-S and CandElim in terms of the Bayesian framework we use probabilities with values 0 or 1 only (capturing, deterministic aspect and noise-free data).

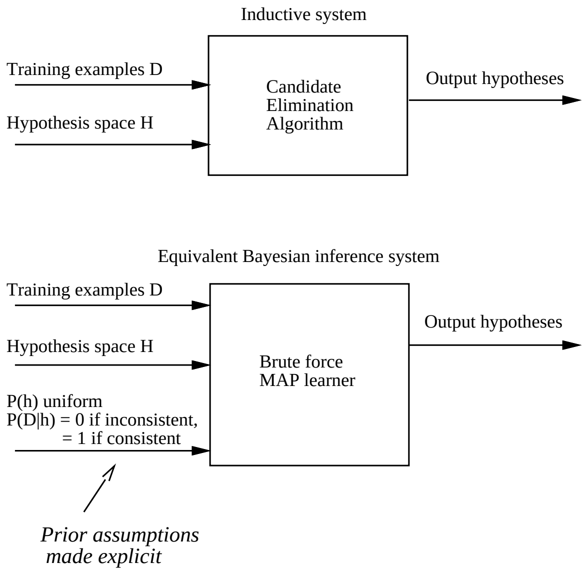


Figure 2: Version Space versus Equivalent MAP Learners

Learning A Real Valued Function

Consider any real-valued target function f. Under certain conditions, the ML hypothesis coincides with the Least Squared Errors hypothesis.

More precisely, we have the following result:

PROPOSITION 1 *Let the 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:

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

Proof:

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

Maximize natural log of this instead: this is a very common device in optimization.

The product of values in $[0, 1]$ is also in $[0, 1]$ and smaller that all of the factors in the product.

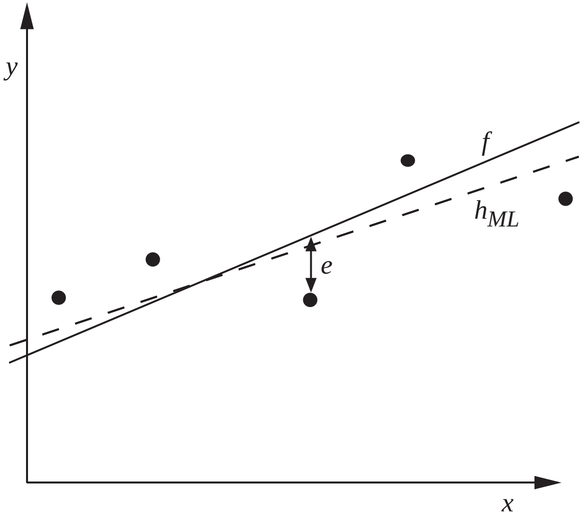


Figure 3: Learning a Linear Function

So, as we multiply many such values the result will be very close to 0.

So, we take the log of the product to obtain a sum of logs.

Since *log* is a monotone function, the optimization problems are equivalent.

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

Learning to Predict Probabilities

We want to learn a function

$$f : X \to \{0, 1\}$$

and we let

$$p_0(x) = Prob(f(x) = 0), \quad p_1(x) = P(f(x) = 1) = 1 - p_0(x)$$

For example,

$$X = \{x;\ x \textit{ is a patient with symptoms}\}$$

and we let, for $x \in X$

$$f(x) = \begin{cases} 1 & \text{if } x \text{ survives} \\ 0 & \text{otherwise} \end{cases}$$

We want to learn

$$p(x) = P(f(x) = 1)$$

Training examples

$$\langle x_i, d_i \rangle$$

where d_i is 1 or 0. For each $x \in X$, let $F_1(x)$ denote the frequency of 1s and $F_0(x)$ the frequency of 0s.

State the problem as a MAP problem:

- What is $P(D|h)$? $D = \{(x_1, d_1), \ldots, (x_m, d_m)\}$, where $d_i = f(x_i) = 0$ or 1.

Here x_i, d_i are random variables, x_i and h are independent.

Claim: In general,

$$P(x_i, d_i|h) = P(d_i|h, x_i)P(x_i|h)$$

Proof:

$$rhs = P(d_i|h, x_i)P(x_i|h) = \frac{P(d, h, x)}{P(h, x)} \times \frac{P(x, h)}{P(h)}$$

$$lfs = P(x_i, d_i|h) \equiv \frac{P(x, h, d)}{P(h)}$$

Now, independence of x and h means that $P(x|h) = P(x)$ (easy to show).

Therefore,

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

Now, $P(d_i = 1|h, x_i) = h(x_i)$, so,

$$\begin{aligned} P(d_i|h, x_i) &= h(x_i) \; if \; d_i = 1 \\ &= 1 - h(x_i) \; if \; d_i = 0 \end{aligned} \qquad (4)$$

which can be rewritten as

$$P(d_i|h, x_i) = (h(x_i))^{d_i}(1 - h(x_i))^{(1-d_i)}$$

and thus,

$$P(D|h_i) = \prod_{i=1}^{m} h(x_i)^{d_i}(1 - h(x_i))^{(1-d_i)}P(x_i)$$

Now, h_{MLE} is the hypothesis such that

$$P(D|h_{MLE}) = max_{h_i} P(D|h_i).....$$

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

Analogy with entropy: *negative cross entropy*

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:

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

- Given new instance x,

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

- 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:

$$\begin{array}{lll} P(h_1|D) = 0.4, & P(-|h_1) = 0, & P(+|h_1) = 1 \\ P(h_2|D) = 0.3, & P(-|h_2) = 1, & P(+|h_2) = 0 \\ P(h_3|D) = 0.3, & P(-|h_3) = 1, & P(+|h_3) = 0 \end{array}$$

therefore

$$\begin{aligned} \sum_{h_i \in H} P(+|h_i)P(h_i|D) &= 0.4 \\ \sum_{h_i \in H} P(-|h_i)P(h_i|D) &= 0.6 \end{aligned}$$

and

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

Gibbs Classifier

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

Gibbs algorithm

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

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

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

Suppose correct, uniform prior distribution over H, then

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

Naive Bayes Classifier

Along with decision trees, neural networks, nearest nbr, 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:

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

Naive Bayes assumption:

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

which gives

$$\textbf{Naive Bayes classifier: } v_{NB} = \operatorname*{argmax}_{v_j \in V} P(v_j) \prod_i P(a_i | v_j)$$

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($x$)
```

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

Naive Bayes: Example

Consider *PlayTennis* again, and new instance

$$\langle Outlk = sun, Temp = cool, Humid = high, Wind = strong \rangle$$

Want to compute:

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

$$P(y)\ P(sun|y)\ P(cool|y)\ P(high|y)\ P(strong|y) = .005$$

$$P(n)\ P(sun|n)\ P(cool|n)\ P(high|n)\ P(strong|n) = .021$$

$$\rightarrow v_{NB} = n$$

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

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

   - see [Domingos & Pazzani, 1996] for analysis
   - Naive Bayes posteriors often unrealistically close to 1 or 0

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

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

   $$\hat{P}(v_j) \prod_i \hat{P}(a_i|v_j) = 0$$

   Typical solution is 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)

Learning to Classify Text

Why?

- Learn which news articles are of interest
- Learn to classify web pages by topic

Naive Bayes is among most effective algorithms
What attributes shall we use to represent text documents??
Target concept $Interesting?: Document \rightarrow \{+, -\}$

1. Represent each document by vector of words
   - one attribute per word position in document
2. Learning: Use training examples to estimate
   - $P(+)$
   - $P(-)$
   - $P(doc|+)$
   - $P(doc|-)$

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

```
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 $V$ do
    – $docs_j \leftarrow$ subset of $Examples$ for which the target value is $v_j$
    – $P(v_j) \leftarrow \frac{|docs_j|}{|Examples|}$
    – $Text_j \leftarrow$ a single document created by concatenating all members of $docs_j$
    – $n \leftarrow$ total number of words in $Text_j$ (counting duplicate words multiple times)
    – for each word $w_k$ in $Vocabulary$
        * $n_k \leftarrow$ number of times word $w_k$ occurs in $Text_j$
        * $P(w_k|v_j) \leftarrow \frac{n_k+1}{n+|Vocabulary|}$
```

CLASSIFY_NAIVE_BAYES_TEXT(Doc)

- $positions \leftarrow$ all word positions in Doc that contain tokens found in $Vocabulary$
- Return v_{NB}, where

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

Twenty NewsGroups

Given 1000 training documents from each group
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

Bayesian Belief Networks (also called Bayes Nets or Graphical Models)

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

Conditional Independence

DEFINITION 1 *(Conditional Independence) 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

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

or, more compactly, we write

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

EXAMPLE 3 *Thunder is conditionally independent of Rain, given Lightning*

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

Naive Bayes uses conditional independence to justify the following equation which very useful computationally:

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

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

Represents joint probability distribution over all variables

- e.g., $P(Storm, BusTourGroup, \ldots, ForestFire)$
- in general,
$$P(y_1, \ldots, y_n) = \prod_{i=1}^{n} P(y_i | Parents(Y_i))$$
where $Parents(Y_i)$ denotes immediate predecessors of Y_i in graph
- so, joint distribution is fully defined by graph, plus the $P(y_i | Parents(Y_i))$

Inference in Bayesian Networks

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

- Bayes net contains all information needed for this inference
- If only one variable with unknown value, easy to infer it
- In general case, problem is NP hard

In practice, can succeed in many cases

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

Learning of Bayesian Networks

Several variants of this learning task

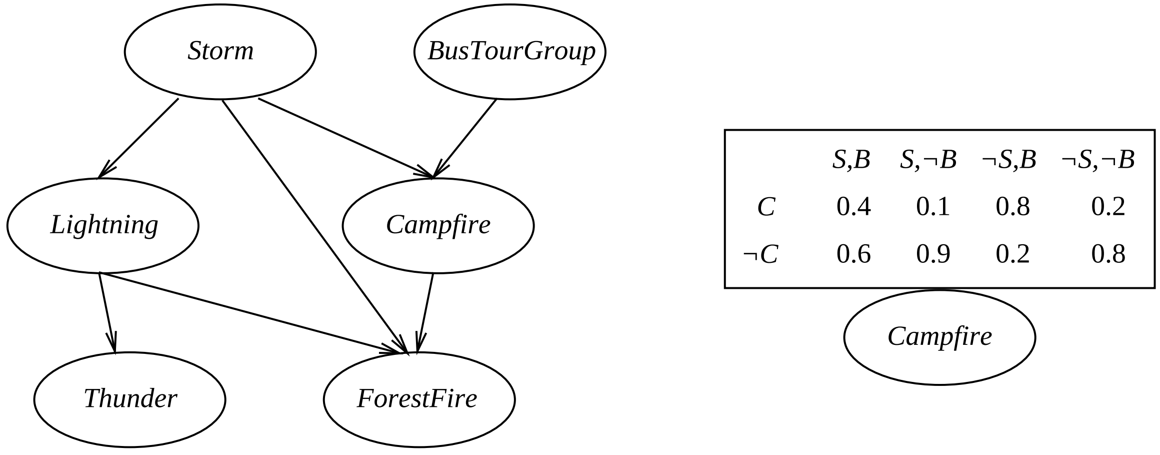
- Network structure might be *known* or *unknown*
- Training examples might provide values of *all* network variables, or just *some*

If structure known and observe all variables

- Then it's easy as training a Naive Bayes classifier

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 (not covered yet)
- In fact, can learn network conditional probability tables using gradient ascent (not covered yet)!
- Converge to network h that (locally) maximizes $P(D|h)$



	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

Figure 4: Bayesian Belief Network

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

- **Idea in Gradient Ascent**: Need to maximize a quantity. Use gradient - the vector of its partial derivatives with respect to its variables and adjust these in the direction of the gradient.
- The Gradient Ascent Training for Bayesian networks to obtain the following formula for gradient ascent:
  1. update all w_{ijk} using training data D
  $$w_{ijk} \leftarrow w_{ijk} + \eta \sum_{d \in D} \frac{P_h(y_{ij}, u_{ik} | d)}{w_{ijk}}$$
  2. then, renormalize the w_{ijk} to assure
     - $\sum_j w_{ijk} = 1$
     - $0 \leq w_{ijk} \leq 1$
- Russell shows that this converges to a **locally** optimal solution: **locally maximum likelihood hypothesis for the conditional probabilities of the Bayesian network**.

Learning the Structure of the Bayesian Network

Very difficult (some research is being done):

- greedy search among network structures on a cost function that trades off structure complexity for accuracy;
- Constraint-based approaches.

More on Learning Bayes Nets

EM algorithm can also be used. Repeatedly:

1. Calculate probabilities of unobserved variables, assuming h
2. Calculate new w_{ijk} to maximize $E[\ln P(D|h)]$ where D now includes both observed and (calculated probabilities of) unobserved variables

When structure unknown...

- Algorithms use greedy search to add/substract edges and nodes
- Active research topic

Summary: Bayesian Belief Networks

- Combine prior knowledge with observed data
- Impact of prior knowledge (when correct!) is to lower the sample complexity
- Active research area
  - Extend from boolean to real-valued variables
  - Parameterized distributions instead of tables
  - Extend to first-order instead of propositional systems
  - More effective inference methods
  - ...

Expectation Maximization (EM)

Originally proposed by Arthur Dempster (1977).

Dempster also proposed a new framework for probability: interval value probabilities, lower- and upper- probability model.

This lead to anothe model for uncertainty and the development of belief functions, and a mathematical theory of evidence (Glenn Shaefer, mid 1980s).

Dempster-Shaeffer Rule of combination of evidence.

When to use:

- Data is only partially observable: **some attributes values are sometimes known (observable) sometimes are NOT known (unobservable)**
- Unsupervised clustering (target value unobservable)
- Supervised learning (some instance attributes unobservable)

Some uses:

- Train Bayesian Belief Networks
- Unsupervised clustering (AUTOCLASS)
- Learning Hidden Markov Models

Generating Data from Mixture of k Gaussians

Each instance x generated by

1. Choosing one of the k Gaussians with uniform probability
2. Generating an instance at random according to that Gaussian

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 $\langle \mu_1, \ldots, \mu_k \rangle$. Note that it is easy to estimate each of these means from samples of size m drawn from the respective distributions

$$\mu_{ML} = argmin_{\mu} \sum_{i=1}^{m} (x_i - \mu)^2 \tag{5}$$

  which is solved for

$$\mu_{ML} = \frac{1}{m} \sum_{i=1}^{m} x_i \tag{6}$$

  However, in the problem on hand we have a mixture of k different Normal dostributions, and we cannot observe which instance was generated from which distribution.

- This is a problem involving **hidden variables**.

Think of full description of each instance as

$$y_i = \langle x_i, z_{i1}, z_{i2} \rangle$$

where

- $$z_{ij} = \begin{cases} 1 & \text{if } x_i \text{ is generated by } j\text{th Gaussian distribution} \\ 0 & \text{otherwise} \end{cases}$$
- x_i is observable
- z_{ij} are hidden or unobservable

In particular, assume $k = 2$. Then we have z_{i1} and z_{i2}. If each of the variables z_{i1} and z_{i2} were observed, then we could use equation 6 to estimate μ_1 and μ_2.

The EM Algorithm 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 z_{ij} is its expected value $E[z_{ij}]$ calculated in Step E. Then replace h by h'. The maximum likelihood hypothesis is given by

$$\mu_j \longleftarrow \frac{\sum E[z_{ij}] x_i}{\sum E[z_{ij}]}$$

Summary of the EM Alg.

Iterate between the following steps

1. Use current hypothesis to estimate the (means) of the unobserved variables
2. Use the unobserved variables to calculate a new. improved hypothesis

After each iteration $P(D|h)$ increases, unless it is at a local maximum.