Bayesian Learning

- Bayes Theorem
- MAP, ML hypotheses
- MAP learners
- Minimum description length principle
- Bayes optimal classifier
- Naïve Bayes learner
- Bayesian belief networks

Two Roles for Bayesian Methods

Provide practical learning algorithms:

- Naïve 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

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

- *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

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

Generally want the most probable hypothesis given the training data

Maximum a posteriori hypothesis h_{MAP}:

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

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

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

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, 0.8% of the entire population have this cancer.

P(cancer) =	P(¬cancer) =
P(+|cancer) =	P(-|cancer) =
P(+|¬cancer) =	P(-|¬cancer) =

P(cancer|+) =

P(¬cancer|+) =

Some Formulas for Probabilities

- *Product rule*: probability $P(A \wedge B)$ of a conjunction of two events *A* and *B*:

  $P(A \wedge B) = P(A|B)P(B) = P(B|A)P(A)$
- *Sum rule:* probability of disjunction of two events *A* and *B*:

  $P(A \vee B) = P(A) + P(B) - P(A \wedge B)$
- *Theorem of total probability:* if events $A_1,\ldots,A_n$ are mutually exclusive with $\sum_{i=1}^{n} P(A_i) = 1$, then

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

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 $VS_{H,D}$)

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

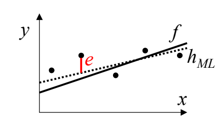
Choose $P(h)$ to be uniform distribution

- $P(h) = 1/|H|$ for all h in H

Then

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

Learning a Real Valued Function



Consider any real-valued target function f

Training examples (x_i, d_i), where d_i is noisy training value

- $d_i = f(x_i) + e_i$
- e_i is random variable (noise) drawn independently for each x_i according to some Gaussian distribution with *mean* = 0

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

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

Learning a Real Valued Function

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

Maximize natural log of this instead ...

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

Minimum Description Length Principle

Occam's razor: prefer the shortest hypothesis

MDL: prefer the hypothesis h that minimizes

$$h_{MDL} = \arg\min_{h \in H} L_{C1}(h) + L_{C2}(D \mid h)$$

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

Example:

- H = decision trees, D = training data labels
- $L_{C1}(h)$ is # bits to describe tree h
- $L_{C2}(D|h)$ is #bits to describe D given h
  - Note $L_{C2}(D|h) = 0$ if examples classified perfectly by h. Need only describe exceptions
- Hence h_{MDL} trades off tree size for training errors

Minimum Description Length Principle

$$h_{MAP} = \arg\max_{h \in H} P(D \mid h)P(h)$$
$$= \arg\max_{h \in H} \log_2 P(D \mid h) + \log_2 P(h)$$
$$= \arg\min_{h \in H} -\log_2 P(D \mid h) - \log_2 P(h) \quad (1)$$

Interesting fact from information theory:

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

So interpret (1):

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

$-\log_2 P(D|h)$ is length of D given h in optimal code

$\rightarrow$ prefer the hypothesis that minimizes

length(h)+length(misclassifications)

Bayes Optimal Classifier

Bayes optimal classification

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

Example:

$P(h_1|D)=.4,\quad P(-|h_1)=0,\quad P(+|h_1)=1$

$P(h_2|D)=.3,\quad P(-|h_2)=1,\quad P(+|h_2)=0$

$P(h_3|D)=.3,\quad P(-|h_3)=1,\quad P(+|h_3)=0$

therefore

$$\sum_{h_i \in H} P(+ \mid h_i) P(h_i \mid D) = .4$$
$$\sum_{h_i \in H} P(- \mid h_i) P(h_i \mid D) = .6$$

and

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

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

Naïve Bayes Classifier

Along with decision trees, neural networks, nearest neighbor, 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

Naïve Bayes Classifier

Assume target function $f: X \rightarrow V$, where each instance x described by attributed $(a_1, a_2, \ldots, a_n)$.

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

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

Naïve Bayes assumption:

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

which gives

Naïve Bayes classifier: $v_{NB} = \arg\max_{v_j \in V} P(v_j) \prod_i P(a_i \mid v_j)$

Naïve 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)

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

Naïve Bayes Example

Consider *CoolCar* again and new instance

(Color=Blue,Type=SUV,Doors=2,Tires=WhiteW)

Want to compute

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

P(+)*P(Blue|+)*P(SUV|+)*P(2|+)*P(WhiteW|+)=

5/14 * 1/5 * 2/5 * 4/5 * 3/5 = **0.0137**

P(-)*P(Blue|-)*P(SUV|-)*P(2|-)*P(WhiteW|-)=

9/14 * 3/9 * 4/9 * 3/9 * 3/9 = **0.0106**

Naïve Bayes Subtleties

1. Conditional independence assumption is often violated

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

- … but it works surprisingly well anyway. Note that you do not need estimated posteriors to be correct; need only that

$$\arg\max_{v_j \in V} \hat{P}(v_j)\prod_i \hat{P}(a_i \mid v_j) = \arg\max_{v_j \in V} P(v_j)P(a_1, \ldots, a_n \mid v_j)$$

- see Domingos & Pazzani (1996) for analysis
- Naïve Bayes posteriors often unrealistically close to 1 or 0

Naïve Bayes Subtleties

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

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

$$\hat{P}(v_j)\prod \hat{P}(a_i \mid v_j) = 0$$

Typical solution is Bayesian estimate for $\hat{P}(a_i \mid v_j)$

$$\hat{P}(a_i \mid v_j) \leftarrow \frac{n_c + mp}{n + m}$$

- n is number of training examples for which $v=v_j$
- n_c is number of examples for which $v=v_j$ and $a=a_i$
- p is prior estimate for $\hat{P}(a_i \mid v_j)$
- m is weight given to prior (i.e., number of “virtual” examples)

Bayesian Belief Networks

Interesting because

- Naïve Bayes assumption of conditional independence is too restrictive
- But it is intractable without some such assumptions…
- Bayesian belief networks describe conditional independence among *subsets* of variables
- allows combing prior knowledge about (in)dependence 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

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

more compactly we write

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

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

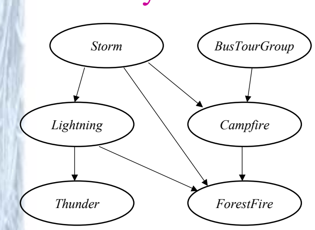
P(Thunder|Rain,Lightning)=P(Thunder|Lightning)

Naïve Bayes uses conditional ind. to justify

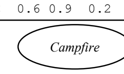
$P(X,Y|Z)=P(X|Y,Z)P(Y|Z)$

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

Bayesian Belief Network



	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



Network represents a set of conditional independence assumptions

- Each node is asserted to be conditionally independent of its nondescendants, given its immediate predecessors
- Directed acyclic graph

Bayesian Belief Network

- Represents joint probability distribution over all variables
- e.g., *P(Storm,BusTourGroup,…,ForestFire)*
- in general,

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

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
- 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 is easy as training a Naïve Bayes classifier

Learning Bayes Net

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(Yi=yij|Parents(*Y_i*)=the list* u_{ik} *of values)*

e.g., if Y_i = *Campfire*, then u_{ik} might be (*Storm=T*, *BusTourGroup=F*)

Perform gradient ascent by repeatedly

1. Update all w_{ijk} using training data D

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

2. Then renormalize the w_{ijk} to assure

$$\sum_j w_{ijk} = 1 \ , \ 0 \le w_{ijk} \le 1$$

Summary of Bayes 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