Bayesian Methods

Not all hypotheses are created equal (even if they are all consistent with the training data)

Might have reasons (domain information) to favor some hypotheses over others a priori

Bayesian methods work with probabilities, and have two main roles:

1. Provide practical learning algorithms:
   - Naive Bayes learning
   - Bayesian belief network learning
   - Combine prior knowledge (prior probabilities) with observed data
   - Requires prior probabilities

2. Provides useful conceptual framework
   - Provides "gold standard" for evaluating other learning algorithms
   - Additional insight into Occam’s razor

Outline

- Bayes Theorem
- MAP, ML hypotheses
- MAP learners
- Minimum description length principle
- Bayes optimal classifier/Gibbs algorithm
- Naive Bayes classifier
- Bayesian belief networks

Bayes Theorem

In general, an identity for conditional probabilities

For our work, we want to know the probability that a particular \( h \in H \) is the correct hypothesis given that we have seen training data \( D \) (examples and labels). Bayes theorem lets us do this.

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

- \( P(h) \) = prior probability of hypothesis \( h \) (might include domain information)
- \( P(D) \) = probability of training data \( D \)
- \( P(h \mid D) \) = probability of \( h \) given \( D \)
- \( P(D \mid h) \) = probability of \( D \) given \( h \)

Note \( P(h \mid D) \) increases with \( P(D \mid h) \) and \( P(h) \) and decreases with \( P(D) \)

Choosing Hypotheses

Maximizing a posteriori hypothesis \( h_{MAP} \):

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

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

Generally want the most probable hypothesis given the training data

If assume \( P(h_i) = P(h_j) \) for all \( i, j \), then can further simplify, and choose the maximum likelihood (ML) hypothesis

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

Bayes Theorem Example

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, 0.008 of the entire population have this cancer.

\[
P(\text{cancer}) = \quad P(\neg\text{cancer}) =
\]

\[
P(+) = P(\neg) =
\]

Now consider new patient for whom the test is positive. What is our diagnosis?

\[
P(+ \mid \text{cancer})P(\text{cancer}) = \quad P(+ \mid \neg\text{cancer})P(\neg\text{cancer}) =
\]

So \( h_{MAP} = \)
Basic Formulas for Probabilities

- **Product Rule**: probability \( P(A \land B) \) of a conjunction of two events \( A \) and \( B \):
  \[
P(A \land B) = P(A \mid B) P(B) = P(B \mid A) P(A)
  \]

- **Sum Rule**: probability of a disjunction of two events \( A \) and \( B \):
  \[
P(A \lor B) = P(A) + P(B) - P(A \land B)
  \]

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

**Problem**: what if \( H \) exponentially or infinitely large?

Relation to Concept Learning

Consider our usual concept learning task: instance space \( X \), hypothesis space \( H \), training examples \( D \)

Consider the Find-S learning algorithm (outputs most specific hypothesis from the version space \( V_{SH,D} \))

What would brute-force MAP learner output as MAP hypothesis?

Does Find-S output a MAP hypothesis?

Characterizing Learning Algorithms by Equivalent MAP Learners

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

- \( d_i = f(x_i) + \epsilon_i \)
- \( \epsilon_i \) is random variable (noise) drawn independently for each \( x_i \) according to some Gaussian distribution with mean \( \mu_0 = 0 \)

Then the maximum likelihood hypothesis \( h_{ML} \) is the one that minimizes the sum of squared errors, e.g. a linear unit trained with GD/EG:

\[
h_{ML} = \arg \min_{h \in H} \sum_{i=1}^{m} (d_i - h(x_i))^2
\]
Thus have Bayesian justification for minimizing squared error (under certain assumptions)

Learning to Predict Probabilities
Consider predicting survival probability from patient data

Training examples \( \langle x_i, d_i \rangle \), where \( d_i \) is 1 or 0 (assume label is [or appears] probabilistically generated)

Want to train neural network to output the probability that \( x_i \) has label 1, not the label itself

Using approach similar to previous slide (p. 169), can show

\[
\min_{h} \sum_{i=1}^{m} \frac{1}{2} \log \frac{1}{2} \left( \frac{1}{2} \left( \frac{d_i - h(x_i)}{\sigma} \right)^2 \right)
\]

For single sigmoid unit, use update rule

\[
w_j \leftarrow w_j + \eta \sum_{i=1}^{m} (d_i - h(x_i)) x_{ij}
\]

to find \( h_{ML} \) (can also derive EG rule)

Minimum Description Length Principle

Bayesian Justification

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

\[
= \arg \max_{h \in H} \log P(D \mid h) + \log P(h)
\]

\[
= \arg \min_{h \in H} -\log P(D \mid h) - \log P(h)
\]

Interesting fact from information theory: The optimal (shortest expected coding length) code for an event with probability is \( -\log P(h) \) bits.

So interpret (1):

- \( -\log P(h) \) is length of \( h \) under optimal code

- \( -\log P(D \mid h) \) is length of \( D \) given \( h \) under optimal code

\( \rightarrow \) prefer the hypothesis that minimizes

\[\text{length}(h) + \text{length(miscalssifications)}\]

Caveat: \( h_{MDL} = h_{MAP} \) doesn’t apply for arbitrary encodings (need \( P(h) \) and \( P(D \mid h) \) to be optimal); merely a guide

Bayes Optimal Classifier

Bayes optimal classification:

\[
\arg \max_{h \in V} \sum_{v \in V} P(v \mid h) P(h) \mid D
\]

where \( V \) is set of possible labels (e.g. \{+,-\})

Example:

\[
P(h_1 \mid D) = 0.4, \quad P(- \mid h_1) = 0, \quad P(+) \mid h_1 = 1
\]

\[
P(h_2 \mid D) = 0.3, \quad P(- \mid h_2) = 1, \quad P(+) \mid h_2 = 0
\]

\[
P(h_3 \mid D) = 0.3, \quad P(- \mid h_3) = 1, \quad P(+) \mid h_3 = 0
\]

therefore

\[
\sum_{h \in \{+,-\}} P(+) \mid h) P(h) \mid D = 0.4
\]

\[
\sum_{h \in \{+,-\}} P(- \mid h) P(h) \mid D = 0.6
\]

and

\[
\arg \max_{v \in V} \sum_{h \in \{+,-\}} P(v \mid h) P(h) \mid D = -
\]

On average, no other classifier using same prior and same hyp. space can outperform Bayes optimal!
Bayes optimal classifier provides best result, but can be expensive or impossible if many hypotheses. 

Gibbs algorithm: 
1. Randomly choose one hypothesis 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: 
\[
E[error_{Gibbs}] \leq 2 E[error_{Bayes Optimal}]
\]
i.e. if prior correct and \( c \in H \), then average error at most twice best possible! 

E.g. Suppose correct, uniform prior distribution over \( H \). Then: 
- Pick any hypothesis from VS with uniform probability  
- Expected error no worse than twice Bayes optimal 

Still have to be able to choose random hypothesis! 

Naive Bayes Classifier 

Along with decision trees, neural networks, nearest neighbor, SVMs, boosting, 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 

Naive Bayes Example 

Training Examples (Table 3.2): 

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Humidity</th>
<th>Temperature</th>
<th>Wind</th>
<th>PlayTennis</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D2</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D3</td>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D4</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D5</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D6</td>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D7</td>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D8</td>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>High</td>
<td>No</td>
</tr>
<tr>
<td>D9</td>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D10</td>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D11</td>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D12</td>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D13</td>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Example to classify: 
\( \{Outlook = sun, Temp = cool, Humid = high, Wind = strong\} \)

Assign label \( v_{NB} = \arg\max_{v_j \in V} P(v_j) \prod_i P(a_i | v_j) \)

\[
P(y) P(\text{sun} | y) P(\text{cool} | y) P(\text{high} | y) P(\text{strong} | y) = (9/14) \cdot (2/9) \cdot (3/9) \cdot (3/9) = 0.0053
\]

\[
P(n) P(\text{sun} | n) P(\text{cool} | n) P(\text{high} | n) P(\text{strong} | n) = (5/14) \cdot (3/5) \cdot (1/5) \cdot (4/5) \cdot (3/5) = 0.0206
\]

So \( v_{NB} = n \)

Naive Bayes Classifier 

(cont’d)

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

Most probable value of \( f(x) \) is: 
\[
v_{MAP} = \arg\max_{v_j \in V} P(v_j | a_1, a_2, \ldots, a_n) \\
= \arg\max_{v_j \in V} \frac{P(a_1, a_2, \ldots, a_n | v_j) P(v_j)}{P(a_1, a_2, \ldots, a_n)} \\
= \arg\max_{v_j \in V} P(a_1, a_2, \ldots, a_n | v_j) P(v_j)
\]

Problem with estimating probs from training data: estimating \( P(v_j) \) easily done by counting, but there are exponentially \( (n^k) \) many combos of values \( a_1, \ldots, a_n \), so can’t get estimates for most combos 

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

so naive Bayes classifier: 
\[
v_{NB} = \arg\max_{v_j \in V} P(v_j) \prod_i P(a_i | v_j)
\]

Now have only polynomial number of probs to estimate 

Naive Bayes Subtleties 

- Conditional independence assumption is often violated, i.e. 
\[
P(a_1, a_2, \ldots, a_n | v_j) \neq \prod_i P(a_i | v_j)
\]

...but it works surprisingly well anyway. Note don’t need estimated posteriors \( P(v_j | x) \) to be correct; need only that 
\[
\arg\max_{v_j \in V} P(v_j) \prod_i P(a_i | v_j) = \arg\max_{v_j \in V} P(v_j) P(a_1, a_2, \ldots, a_n | v_j)
\]

Sufficient conditions given in [Domingos & Pazzani, 1996]
Naive Bayes

Application: Learning to Classify Text

• Target concept Interesting? : Document → (+, −)
  (can you also use NB as a ranker?)

• Each document is a vector of words (i.e. one attribute per word position), e.g. \( a_1 = "our" \), \( a_2 = "approach" \), etc.

• Naive Bayes very effective despite obvious violation of conditional independence assumption

• See Section 6.10 for more detail

Bayesian Belief Networks

• Sometimes naive Bayes assumption of conditional independence too restrictive

• But inferring probabilities is intractable without some such assumptions

• Bayesian belief networks (also called Bayes Nets) describe conditional independence among subsets of variables

• Allows combining prior knowledge about dependencies among variables with observed training data

Bayesian Belief Network
(cont’d)

Represents joint probability distribution over all network variables \( Y_1, \ldots, Y_n \), e.g.
\[
P(\text{Storm}, \text{BusTourGroup}, \ldots, \text{ForestFire})
\]

• In general, for \( y_i = \text{value of } Y_i \)
\[
P(y_1, \ldots, y_n) = \prod_{i=1}^{n} P(y_i \mid \text{Parents}(Y_i))
\]
where \( \text{Parents}(Y_i) \) denotes immediate predecessors of \( Y_i \) in graph

• E.g. \( P(S, B, C, \neg L, \neg T, \neg F) = P(S)P(B)P(C \mid B, S)P(\neg L \mid S)P(\neg T \mid \neg L)P(\neg F \mid S, \neg L, \neg C) \)

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
\[
P(x \mid y, z) = P(x \mid z)
\]
more compactly, we write
\[
P(X \mid Y, Z) = P(X \mid Z)
\]

Example: Thunder is conditionally independent of Rain, given Lightning
\[
P(\text{Thunder} \mid \text{Rain}, \text{Lightning}) = P(\text{Thunder} \mid \text{Lightning})
\]

Naive Bayes uses conditional independence and product rule (slide 7) to justify
\[
P(X, Y \mid Z) = P(X \mid Y, Z)P(Y \mid Z) = P(X \mid Z)P(Y \mid Z)
\]
Inference in Bayesian Networks

Want to infer probabilities of values of one or more network variables (attributes), given observed values of others, i.e. want the probability distribution of a subset of variables given values of a subset of the others

- Bayes net contains all information needed for this inference: can simply brute force try all combinations of values of the unknown variables
- Of course, this takes time exponential in number of unknowns
- 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

We know how to use Bayesian Networks, but how do we learn one?

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 all variables observed, then it’s as easy as training a naïve Bayes classifier (just count occurrences as before)

Learning of Bayesian Networks (cont’d)

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 \( P(D|h) \), i.e. search for ML hypothesis
- Can also use EM (expectation maximization) algorithm
  - Use observations of variables to predict their values in cases when they’re not observed
  - EM has many other applications, e.g. hidden Markov models (HMMs) used for e.g. biological sequence analysis

Bayesian Belief Networks

Summary

- 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
- More effective inference methods