CSCE 478/878 Lecture 6: Bayesian Learning and Graphical Models

Stephen Scott

(Adapted from Ethem Alpaydin and Tom Mitchell)

sscott@cse.unl.edu

Introduction

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

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

1. Provide practical learning algorithms:
    - Naïve 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
- Example
- Bayes optimal classifier
- Naïve Bayes classifier
- Example: Learning over text data
- Bayesian belief networks

Bayes Theorem

We want to know the probability that a particular label r is correct given that we have seen data D

Conditional probability: $P(r \mid D) = P(r \wedge D)/P(D)$

Bayes theorem:

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

- $P(r)$ = **prior probability** of label r (might include domain information)
- $P(D)$ = probability of data D
- $P(r \mid D)$ = **posterior probability** of r given D
- $P(D \mid r)$ = probability (aka **likelihood**) of D given r

Note: $P(r \mid D)$ increases with $P(D \mid r)$ and $P(r)$ and decreases with $P(D)$

Bayes Theorem
Example

Does a patient have cancer or not?

> *A patient takes a lab test and the result is positive. The test returns a correct positive result in 98% of the cases in which the disease is actually present, and a correct negative result in 97% of the cases in which the disease is not present. Furthermore, 0.008 of the entire population have this cancer.*

$$P(cancer) = \qquad P(\neg cancer) =$$
$$P(+ \mid cancer) = \qquad P(- \mid cancer) =$$
$$P(+ \mid \neg cancer) = \qquad P(- \mid \neg cancer) =$$

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

$P(+ \mid cancer)P(cancer) =$

$P(+ \mid \neg cancer)P(\neg cancer) =$

So diagnosis is

Basic Formulas for Probabilities

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

$$P(A \wedge B) = P(A \mid B)P(B) = P(B \mid A)P(A)$$

- **Sum Rule**: probability of a 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

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

Bayes Optimal Classifier

Bayes rule lets us get a handle on the most **probable** label for an instance

Bayes optimal classification of instance $\mathbf{x}$:

$$\operatorname*{argmax}_{r_j \in R} P(r_j \mid \mathbf{x})$$

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

On average, no other classifier using same prior information and same hypothesis space can outperform Bayes optimal!

$\Rightarrow$ Gold standard for classification

Let instance $\mathbf{x}$ be described by attributes $\langle x_1, x_2, \ldots, x_n \rangle$

Then, most probable label of $\mathbf{x}$ is:

$$\begin{aligned} r^* &= \operatorname*{argmax}_{r_j \in R} P(r_j \mid x_1, x_2, \ldots, x_n) \\ &= \operatorname*{argmax}_{r_j \in R} \frac{P(x_1, x_2, \ldots, x_n \mid r_j)\, P(r_j)}{P(x_1, x_2, \ldots, x_n)} \\ &= \operatorname*{argmax}_{r_j \in R} P(x_1, x_2, \ldots, x_n \mid r_j)\, P(r_j) \end{aligned}$$

In other words, if we can estimate $P(r_j)$ and $P(x_1, x_2, \ldots, x_n \mid r_j)$ for all possibilities, then we can give a Bayes optimal prediction of the label of $\mathbf{x}$ for all $\mathbf{x}$

- How do we estimate $P(r_j)$ from training data?
- What about $P(x_1, x_2, \ldots, x_n \mid r_j)$?

Naïve Bayes Classifier

Problem: Estimating $P(r_j)$ easily done, but there are exponentially many combinations of values of $x_1, \ldots, x_n$

E.g., if we want to estimate

$$P(Sunny, Hot, High, Weak \mid PlayTennis = No)$$

from the data, need to count among the "No" labeled instances how many exactly match $\mathbf{x}$ (few or none)

Naïve Bayes assumption:

$$P(x_1, x_2, \ldots, x_n \mid r_j) = \prod_i P(x_i \mid r_j)$$

so **naïve Bayes classifier**:

$$r_{NB} = \operatorname*{argmax}_{r_j \in R} P(r_j) \prod_i P(x_i \mid r_j)$$

Now have only polynomial number of probs to estimate

Naïve Bayes Classifier
(cont'd)

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

Naïve Bayes Algorithm

Naïve_Bayes_Learn

- For each target value r_j
  1. $\hat{P}(r_j) \leftarrow$ estimate $P(r_j)$ = fraction of exs with r_j
  2. For each attribute value v_{ik} of each attrib $x_i \in \mathbf{x}$
     - $\hat{P}(v_{ik} \mid r_j) \leftarrow$ estimate $P(v_{ik} \mid r_j)$ = fraction of r_j-labeled instances with v_{ik}

Classify_New_Instance($\mathbf{x}$)

$$r_{NB} = \operatorname*{argmax}_{r_j \in R} \hat{P}(r_j) \prod_{x_i \in \mathbf{x}} \hat{P}(x_i \mid r_j)$$

Naïve Bayes Example

Training Examples (Mitchell, Table 3.2):

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Instance $\mathbf{x}$ to classify:

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

Naïve Bayes Example (cont'd)

Assign label $r_{NB} = \operatorname{argmax}_{r_j \in R} P(r_j) \prod_i P(x_i \mid r_j)$

$$P(y) \cdot P(sun \mid y) \cdot P(cool \mid y) \cdot P(high \mid y) \cdot P(strong \mid y)$$

$$= (9/14) \cdot (2/9) \cdot (3/9) \cdot (3/9) \cdot (3/9) = 0.0053$$

$$P(n)\ P(sun \mid n)\ P(cool \mid n)\ P(high \mid n)\ P(strong \mid n)$$

$$= (5/14) \cdot (3/5) \cdot (1/5) \cdot (4/5) \cdot (3/5) = 0.0206$$

So $v_{NB} = n$

Naïve Bayes Subtleties

Conditional independence assumption is often violated, i.e.,

$$P(x_1, x_2, \ldots, x_n \mid r_j) \neq \prod_i P(x_i \mid r_j) \quad ,$$

but it works surprisingly well anyway. Note that we don't need estimated posteriors $\hat{P}(r_j \mid \mathbf{x})$ to be correct; need only that

$$\operatorname*{argmax}_{r_j \in R} \hat{P}(r_j) \prod_i \hat{P}(x_i \mid r_j) = \operatorname*{argmax}_{r_j \in R} P(r_j) P(x_1, \ldots, x_n \mid r_j)$$

Sufficient conditions given in Domingos & Pazzani [1996]

Naïve Bayes Subtleties
(cont'd)

What if none of the training instances with target value r_j have attribute value v_{ik}? Then

$$\hat{P}(v_{ik} \mid r_j) = 0, \text{ and } \hat{P}(r_j) \prod_i \hat{P}(v_{ik} \mid r_j) = 0$$

Typical solution is to use m-**estimate**:

$$\hat{P}(v_{ik} \mid r_j) \leftarrow \frac{n_c + mp}{n + m}$$

where

- n is number of training examples for which $r = r_j$,
- n_c number of examples for which $r = r_j$ and $x_i = v_{ik}$
- p is **prior estimate** for $\hat{P}(v_{ik} \mid r_j)$
- m is weight given to prior (i.e., number of "virtual" examples)
- Sometimes called **pseudocounts**

Naïve Bayes Application: Text Classification

- Target concept $Spam? : Document \rightarrow \{+, -\}$
- Each document is a vector of words (one attribute per word position), e.g., $x_1 =$ "each", $x_2 =$ "document", etc.
- Naïve Bayes very effective despite obvious violation of conditional independence assumption ($\Rightarrow$ words in an email are independent of those around them)
- Set $P(+) =$ fraction of training emails that are spam, $P(-) = 1 - P(+)$
- To simplify matters, we will assume **position independence**, i.e., we only model the words in spam/not spam, not their position
  - $\Rightarrow$ For every word w in our vocabulary, $P(w \mid +) =$ probability that w appears in any position of $+$-labeled training emails (factoring in prior m-estimate)

Naïve Bayes Application: Text Classification

Pseudocode [Mitchell]

LEARN_NAIVE_BAYES_TEXT(*Examples*, *V*)

Examples is a set of text documents along with their target values. V is the set of all possible target values. This function learns the probability terms $P(w_k|v_j)$, describing the probability that a randomly drawn word from a document in class v_j will be the English word w_k. It also learns the class prior probabilities $P(v_j)$.

1. collect all words, punctuation, and other tokens that occur in Examples

- $Vocabulary \leftarrow$ the set of all distinct words and other tokens occurring in any text document from $Examples$

2. calculate the required $P(v_j)$ and $P(w_k|v_j)$ probability terms

- For each target value v_j in V do
  - $docs_j \leftarrow$ the subset of documents from $Examples$ 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 distinct word positions in $Text_j$
  - 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*)

Return the estimated target value for the document Doc. a_i denotes the word found in the ith position within Doc.

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

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

Bayesian Belief Networks

- Sometimes naïve 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

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

$$P(X \mid Y, Z) = P(X \mid Z)$$

Example: $Thunder$ is conditionally independent of $Rain$, given $Lightning$

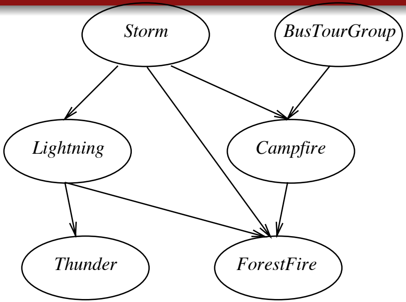
$$P(Thunder \mid Rain, Lightning) = P(Thunder \mid Lightning)$$

Naïve Bayes uses conditional independence and product rule to justify

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

Bayesian Belief Networks

Definition

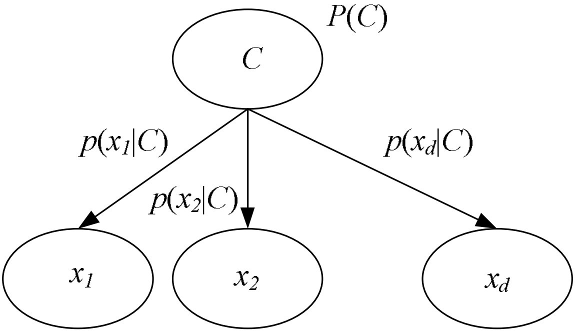


	S,B	$S,\neg B$	$\neg S,B$	$\neg S,\neg B$
C	0.4	0.1	0.8	0.2
$\neg C$	0.6	0.9	0.2	0.8

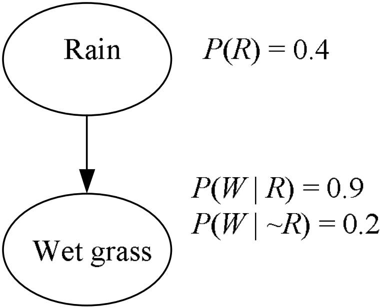
Network (directed acyclic graph) represents a set of conditional independence assertions:

- **Each node is asserted to be conditionally independent of its nondescendants, given its immediate predecessors**
- E.g., Given *Storm* and *BusTourGroup*, *Campfire* is CI of *Lightning* and *Thunder*

Since each node is conditionally independent of its nondescendants given its immediate predecessors, what model does this represent, given that C is class and x_is are attributes?

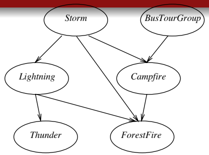


Can think of edges in a Bayes net as representing a **causal relationship** between nodes



E.g., rain causes wet grass

Probability of wet grass depends on whether there is rain



	S,B	$S,\neg B$	$\neg S,B$	$\neg S,\neg B$
C	0.4	0.1	0.8	0.2
$\neg C$	0.6	0.9	0.2	0.8

Campfire

Represents joint probability distribution over variables $\langle Y_1, \ldots, Y_n \rangle$, e.g., $P(Storm, BusTourGroup, \ldots, ForestFire)$

- In general, for y_i = value of Y_i

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

  where $Parents(Y_i)$ denotes immediate predecessors of Y_i in graph
- E.g., $P(S, B, C, \neg L, \neg T, \neg F) =$

$$P(S) \cdot P(B) \cdot \underbrace{P(C \mid B, S)}_{0.4} \cdot P(\neg L \mid S) \cdot P(\neg T \mid \neg L) \cdot P(\neg F \mid S, \neg L, \neg C)$$

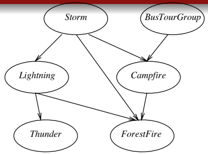
We sometimes call Bayes nets **generative** (vs **discriminative) models** since they can be used to generate instances $\langle Y_1, \ldots, Y_n \rangle$ according to joint distribution

Can use for classification

- Label r to predict is one of the variables, represented by a node
- If we can determine the most likely value of r given the rest of the nodes, can predict label
- One idea: Go through all possible values of r, and compute joint distribution (previous slide) with that value and other attribute values, then return one that maximizes

Bayesian Belief Networks

Predicting Most Likely Label (cont'd)



	S,B	$S,\neg B$	$\neg S,B$	$\neg S,\neg B$
C	0.4	0.1	0.8	0.2
$\neg C$	0.6	0.9	0.2	0.8

Campfire

E.g., if $Storm$ (S) is the label to predict, and we are given values of B, C, $\neg L$, $\neg T$, and $\neg F$, can use formula to compute $P(S,B,C,\neg L,\neg T,\neg F)$ and $P(\neg S,B,C,\neg L,\neg T,\neg F)$, then predict more likely one

Easily handles unspecified attribute values

Issue: Takes time exponential in number of values of unspecified attributes

More efficient approach: **Pearl's message passing algorithm** for chains and trees and polytrees (at most one path between any pair of nodes)

Learning of Bayesian Belief 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 all variables observed, then it's as easy as training a naïve Bayes classifier:

- Initialize CPTs with pseudocounts
- If, e.g., a training instance has set S, B, and $\neg C$, then increment that count in C's table
- Probability estimates come from normalizing counts

	S, B	$S, \neg B$	$\neg S, B$	$\neg S, \neg B$
C	4	1	8	2
$\neg C$	6	10	2	8

Learning of Bayesian Belief 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 $P(D \mid h)$, i.e., **maximum likelihood** 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)

- 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