Data Mining (EECS 4412)

Bayesian Classification

Parke Godfrey

EECS

Lassonde School of Engineering

York University

Thanks to

Professor Aijun An

for creation & use of these slides.

Outline

1. Introduction
2. Bayes Theorem
3. Naïve Bayes Classifier
4. Bayesian Belief Networks

Introduction

- **Goal**:
  - Determine the most probable hypothesis (class)
  - E.g, Given new instance x, what is its most probable classification?
- **Probabilistic learning & prediction**:
  - Estimate explicit probabilities for all hypotheses (classes)
  - Predict multiple hypotheses, weighted by their probabilities
  - Can combine prior knowledge (such as prior probabilities, probability distributions, causal relationships between variables in belief networks) with observed data

Introduction (*Cont'd*)

- **Incremental learning**:
  - Each training example can incrementally increase/decrease the probability that a hypothesis is correct.

- **flexible** in handling *inconsistency*

- **Provides a Standard**:

  - provides a standard of *optimal* decision making against which other methods can be measured

Bayes Theorem

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

- P (h) = *prior* probability of hypothesis h

- P (x) = probability that example x is observed
- P (h | x) = *posterior* probability of h given x

- P (x | h) = *conditional* probability of x given h

  (often called the *likelihood* of h given x)

Finding Maximum *a posteriori* Hypothesis

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

- **Goal**: Find the most probable hypothesis h from a set H of *candidate* hypotheses, given an example x.
- The most probable hypothesis is called *maximum a posteriori (MAP)* hypothesis h_{MAP}:

$$h_{MAP}(x) = \arg\max_{h \in H} P(h \mid x)$$

$$= \arg\max_{h \in H} \frac{P(x \mid h)P(h)}{P(x)}$$

(*P(x)* is constant for all hypotheses)

$$= \arg\max_{h \in H} P(x \mid h)P(h)$$

- If assume P (h_i) = P (h_j) (classes are *equally* likely), then can further simplify, and choose the *Maximum likelihood* (ML) hypothesis:

$$h_{ML}(x) = \arg\max_{h \ \in H} P(x \mid h \ )$$

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,
  - The test returns 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.

$$P(cancer) = \qquad P(\neg cancer) =$$

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

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

Our goal is to find the maximum between:

$$P(cancer \mid +) \text{ and } P(\neg cancer \mid +)$$

Learning Probabilities from Data

- Suppose we do not know the probabilities used in the example in the last slide.
- But we are given a set of data.
- In order to conduct the reasoning, i.e., to find the MAP hypothesis h_{MAP}, we can estimate the probabilities used in the reasoning from the data.
- Suppose there are k possible hypotheses (i.e., classes):

$$h_1, h_2, \ldots. h_k$$

- We need to estimate:
  - $P(h_1), P(h_2), \ldots, P(h_k)$,
  - $P(\mathrm{x}|h_1), P(\mathrm{x}|h_2), \ldots, P(\mathrm{x}|h_k)$ for each possible instance x,

in order to find:

$$h_{MAP}(x) = \arg\max_{h_i \in H} P(x \mid h_i) P(h_i)$$

Practical Problem with Finding MAP Hypothesis

- Suppose instance x is described by attributes values $\langle \boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_n} \rangle$ and there is a set C of classes: $c_1, c_2, \ldots c_m$.

$$c_{MAP}(x) = \arg\max_{c_j \in C} P(c_j \mid x_1, x_2, \ldots, x_n)$$

$$= \arg\max_{c_j \in C} \frac{P(x_1, x_2, \ldots, x_n \mid c_j) P(c_j)}{P(x_1, x_2, \ldots, x_n)}$$

$$= \arg\max_{c_j \in C} P(x_1, x_2, \ldots, x_n \mid c_j) P(c_j)$$

- Given data set with many attributes, it is infeasible to estimate $P(x_1, x_2, \ldots, x_n \mid c_j)$ for all possible x values, *unless* we have a *very*, *very large* set of training data. It is also *computationally expensive*.

Naïve Bayes Classifier

- Naïve assumption: values of attributes are conditionally independent given a class

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

which gives:

$$c_{NB}(x) = \arg\max_{c_j \in C} P(x_1, x_2, \ldots, x_n \mid c_j) P(c_j)$$

$$= \arg\max_{c_j \in C} P(c_j) \prod_i P(x_i \mid c_j)$$

- Probabilities can be estimated from the training data.

Estimating Probabilities

- Estimate $P(c_j)$:

$$P(c_j) = \frac{\text{\# of training examples of class } c_j}{\text{\# of training examples}}$$

- Estimate $P(x_i|c_j)$ for each attribute value x_i of attribute A_i and each class c_j
  - If attribute A_i is categorical,

$$P(x_i \mid c_j) = \frac{\text{\# of training examples of class } c_j \text{ with } x_i \text{ for } A_i}{\text{\# of training examples of class } c_j}$$

Estimating Probabilities

- If attribute A_i is continuous, can assume normal distribution,

$$P(x_i \mid c_j) = \frac{1}{\sqrt{2\pi}\sigma_{c_j}} e^{-\frac{(x_i - \mu_{c_j})^2}{2\sigma_{c_j}^2}}$$

where μ_{c_j} and σ_{c_j} are the mean and standard deviation of the values of A_i for training examples of class c_j

$$\sigma_{c_j} = \sqrt{\frac{1}{n-1} \sum_{x_i \in c_j} \left(x_i - \mu_{c_j}\right)^2}$$

Naïve Bayes Algorithm

- Naïve Bayes Learning (*from examples*)
  - For each class c_j

$$\hat{P}(c_j) \leftarrow \text{estimate } P(c_j)$$

  - For each attribute for which x_i is a value

$$\hat{P}(x_i \mid c_j) \leftarrow \text{estimate } P(x_i \mid c_j)$$

- Classifying new instance (x)

$$c_{NB}(x) = \arg\max_{c_j \in C} \hat{P}(c_j) \prod_{x_i \in x} \hat{P}(x_i \mid c_j)$$

Example

Training dataset

Classes:

c1:buys_computer='yes'

c2:buys_computer='no'

Classify new example:

X =(age<=30, Income=medium, Student=yes Credit_rating=Fair)

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Example (*cont'd*)

- Learning:
  - Compute **P(c_i)**

    P(buy_computer="yes") = 9/14

    P(buy_computer="no") = 5/14

  - Compute **P(x_j|c_i)** for each class and each attribute value pair:

    P(age≤30 | buys_computer="yes") = 2/9 = 0.222

    P(age≤30 | buys_computer="no") = 3/5 = 0.6

    ⋮

    P(income="medium" | buys_computer="yes") = 4/9 = 0.444

    P(income="medium" | buys_computer="no") = 2/5 = 0.4

    ⋮

    P(student="yes" | buys_computer="yes) = 6/9 = 0.667

    P(student="yes" | buys_computer="no") = 1/5 = 0.2

    ⋮

    P(credit_rating="fair" | buys_computer="yes") =6/9 =0.667

    P(credit_rating="fair" | buys_computer="no") = 2/5 = 0.4

    ⋮

Example (*cont'd*)

- Classification: to classify:

  x = (age≤30, income=medium, student=yes, credit_rating=fair)

P(x | c_i) :

P(x|buys_computer="yes")

= P(age≤30 | buys_computer=yes)×P(income=medium | buys_computer=yes) ×

P(student=yes | buys_computer=yes) × P(credit=fair | buys_computer=yes)

= 0.222 x 0.444 x 0.667 x 0.0.667

=0.044

P(x | buys_computer="no") = 0.6 x 0.4 x 0.2 x 0.4 =0.019

P(c_i | x) ∝ P(x | c_i) * P(c_i) :

P(buys_computer="yes"|x) ∝

P(x|buys_computer="yes") * P(buys_computer="yes")=0.028

P(buys_computer="yes"|x) ∝

P(x|buys_computer="no") * P(buys_computer="no")=0.007

x belongs to class "buys_computer=yes"

Naïve Bayesian Classifier: Comments

- Advantages :
  - Easy to implement
  - Good results obtained in most of the cases
- Disadvantage

  - Assumption: class conditional independence of attributes, therefore loss of accuracy

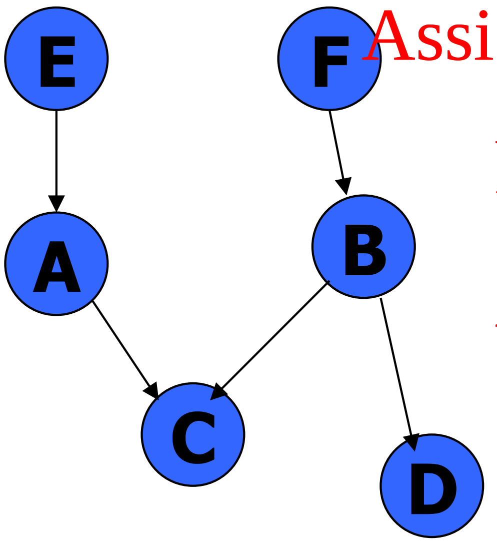
  - Practically, dependencies exist among attributes
    - For example, *headache* and *body temperature* are dependent attributes for *flu* dataset.
  - Dependencies among these cannot be modeled by Naïve Bayesian Classifier
- How to deal with these dependencies?
  - Bayesian Belief Networks

Bayesian Belief Networks

- Naive Bayes assumption of conditional independence is too restrictive.
- But it's intractable without such assumptions...
- Bayesian Belief networks provide an intermediate approach which
  - allows dependencies among attributes
  - but assumes conditional independence among subsets of attributes.

Bayesian Belief Networks

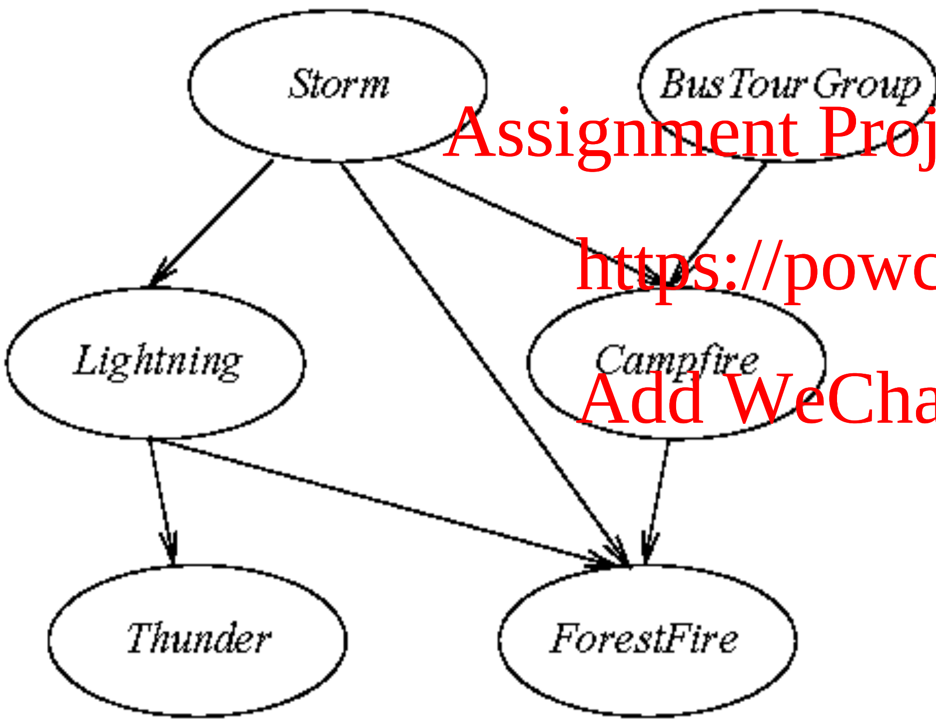
- A graphical model of causal relationships. Two components:
  - A *directed acyclic graph* (DAG): represents dependency among variables (attributes)
    - ***Nodes***: variables (including class attribute)
    - ***Links***: dependencies (e.g., A dependes on E)
    - ***Parents***: immediate predecessors. E.g., A,B are the parents of C. B is the parent of D
    - ***Descendant***: X is a descendant of Y if there is a direct path from Y to X.
    - ***Conditional Independency***:
      - Assume: each variable is conditionally independent of its nondescendants given its parents.
      - Definition: X is *conditionally independent* of Y given Z iff P(X | Y, Z)=P(X | Z)
      - E.g.: C is conditional independent of D given A and B. Thus, P(C | A, B, D)=P(C|A, B)
    - ***Acyclic***: has no loops or cycles



  - A *conditional probability table* (CPT) for each variable X: specifies the conditional probability distribution P(X | Parents(X)).

Example of CPT

- Suppose each variable is binary (contain two values: X and ¬X)



CPT table for *Campfire*

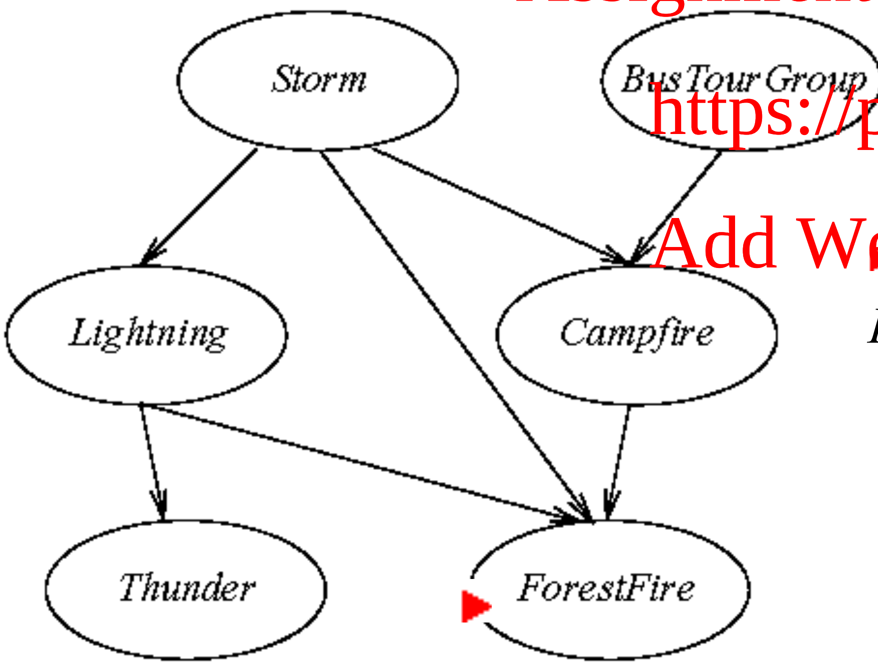
	S,B	$S,\neg B$	$\neg S,B$	$\neg S,\neg B$
C	0.4	0.1	0.8	0.2
$\neg C$	0.6	0.9	0.2	0.8

- There is a conditional probability table (CPT) for each variable

Inference Rule in Bayesian Networks

- The joint probability of any tuple ($x_1, \ldots, x_n$) corresponding to the variables or attributes ($X_1, \ldots, X_n$) is computed by

$$P(x_1,\ldots,x_n) = \prod_{i=1}^{n} P(x_i \mid Parents(X_i))$$

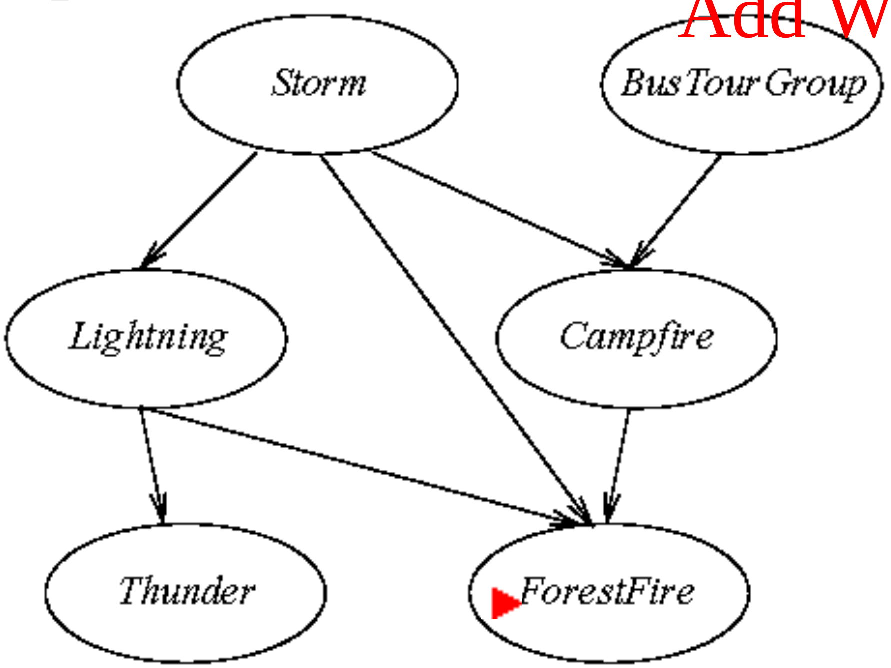


- Example:

$$P(\neg S, B, \neg L, C, \neg T, F) = P(\neg S) \times P(B) \times P(\neg L \mid \neg S) \times P(C \mid \neg S, B) \times P(\neg T \mid \neg L) \times P(F \mid \neg L, \neg S, C)$$

Inference in Bayesian Networks

- A Bayesian network can be used to infer the (probabilities of) values of one or more network variables, given observed values of others.
- Example:
  - Given Storm= 0, BusTourGroup=1, Lightning=0, Campfire=1, Thunder=0, we want to know ForestFire=?



- Compute two probabilities:

  (1) $P(F \mid \neg S, B, \neg L, C, \neg T) = P(F \mid \neg L, \neg S, C)$

  (2) $P(\neg F \mid \neg S, B, \neg L, C, \neg T) = P(\neg F \mid \neg L, \neg S, C)$

- ForestFire = True if (1) > (2)

Inference in Bayesian Networks

- Another example:
  - Given Storm=1, Campfire=0, ForestFire=1, what is the probability distribution of Thunder?
  - Compute two probabilities:

$$(1)\ P(T \mid S, \neg C, F) = P(T, L \mid S, \neg C, F) + P(T, \neg L \mid S, \neg C, F)$$

$$= P(T \mid L, S, \neg C, F) P(L \mid S, \neg C, F) + P(T \mid \neg L, S, \neg C, F) P(\neg L \mid S, \neg C, F)$$

$$= P(T \mid L) P(L \mid S, \neg C, F) + P(T \mid \neg L) P(\neg L \mid S, \neg C, F)$$

where $P(L \mid S, \neg C, F) = \dfrac{P(L, F \mid S, \neg C)}{P(F \mid S, \neg C)} = \dfrac{P(F \mid L, S, \neg C) P(L \mid S, \neg C)}{P(F, L \mid S, \neg C) + P(F, \neg L \mid S, \neg C)}$

$$= \frac{P(F \mid L, S, \neg C) P(L \mid S)}{P(F, L \mid S, \neg C) + P(F, \neg L \mid S, \neg C)} = \frac{P(F \mid L, S, \neg C) P(L \mid S)}{P(F \mid L, S, \neg C) P(L \mid S, \neg C) + P(F \mid \neg L, S, \neg C) P(\neg L \mid S, \neg C)}$$

$$= \frac{P(F \mid L, S, \neg C) P(L \mid S)}{P(F \mid L, S, \neg C) P(L \mid S) + P(F \mid \neg L, S, \neg C) P(\neg L \mid S)}$$

and similarly $P(\neg L \mid S, \neg C, F) = \dfrac{P(F \mid \neg L, S, \neg C) P(\neg L \mid S)}{P(F \mid L, S, \neg C) P(L \mid S) + P(F \mid \neg L, S, \neg C) P(\neg L \mid S)}$

(2) $P(\neg T \mid S, \neg C, F)$ can be calculated similarly.

- Thunder = True if (1) > (2)

Learning of Bayesian Networks

- Several scenarios of this learning task
  - Network structure might be *known* or *unknown*.
  - Training examples might provide values of all network variables, or just *some*.

- **Scenario 1**: If structure known and observe all variables:

  - Then it's easy as training a Naïve Bayes classifier.
  - Learn only CPTs (estimate the conditional probabilities from training data)

Learning of Bayesian Networks

- **Scenario 2**: Suppose structure known, variables partially observable
  - For example, 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* method!
- **Scenario 3**: When structure unknown
  - Use heuristic search or constraint-based technique to search through potential structures.
  - K2 algorithm

Summary: Bayesian Belief Networks

- Combine prior knowledge with observed data
- Intermediate approach that allows both dependencies and conditional independencies
- Other issues
  - Extend from categorical to real-valued variables
  - Parameterized distributions instead of tables
  - More effective inference and learning methods
  - ...

Next Class

- Neural Networks