

COMP3420: Advanced Databases and Data Mining

Classification and prediction:
Bayesian classification and
evaluating classifier accuracy

Lecture outline

- Bayesian classification
- Bayesian theorem
- Naïve Bayesian classifier
- Naïve Bayesian classifier example
- Zero-probability problem
- Comments on naïve Bayesian classifier
- Bayesian belief networks
- Measuring classifier accuracy
- Evaluating the accuracy of a classifier

Bayesian classification: Why?

- A statistical classifier: performs *probabilistic prediction*, i.e., predicts class membership probabilities
- Foundation: Based on *Bayes' Theorem*
- Performance: A simple Bayesian classifier, called *naïve Bayesian classifier*, has comparable performance with decision tree and selected neural network classifiers
- Works incremental: Each training example (training record) can incrementally increase/decrease the probability that a hypothesis is correct — prior knowledge can be combined with observed data
- Standard: even when Bayesian methods are computationally intractable, they can provide a standard of optimal decision making against which other methods can be measured

Bayesian theorem: Basics

- Let \mathbf{X} be a data sample (“*evidence*”): class label is unknown
- Let H be a *hypothesis* that \mathbf{X} belongs to class \mathbf{C}
- Classification is to determine $P(H|\mathbf{X})$, the probability that the hypothesis holds given the observed data sample \mathbf{X}
- $P(H)$ (*prior*, or a *priori probability*): The initial probability
 - For example, \mathbf{X} will buy computer, regardless of age, income, ...
- $P(\mathbf{X})$: Probability that sample data is observed
- $P(\mathbf{X}|H)$: The probability of observing the sample \mathbf{X} , given that the hypothesis holds
 - For example, given that \mathbf{X} will buy computer, the probability that \mathbf{X} is of age 31..40, has medium income, etc.

Bayesian theorem

- Given training data \mathbf{X} , the *posteriori probability of a hypothesis* H , $P(H|\mathbf{X})$, follows the Bayes Theorem

$$P(H|\mathbf{X}) = \frac{P(\mathbf{X}|H)P(H)}{P(\mathbf{X})}$$

- Predicts data sample \mathbf{X} belongs to class C_j if probability $P(C_j|\mathbf{X})$ is the highest among all the $P(C_k|\mathbf{X})$ for all the classes k
- Practical difficulty: requires initial knowledge of many probabilities, and significant computational costs

Towards naïve Bayesian classifier

- Let \mathbf{D} be a training set of tuples and their associated class labels, and each tuple is represented by an n -dimensional attribute vector $\mathbf{X} = (x_1, x_2, \dots, x_n)$

- Suppose there are m classes: C_1, C_2, \dots, C_m

- Classification is to derive the maximum posteriori, i.e., the maximal $P(C_i|\mathbf{X})$

- This can be derived from Bayes' theorem

$$P(C_i|\mathbf{X}) = \frac{P(\mathbf{X}|C_i)P(C_i)}{P(\mathbf{X})}$$

- Since $P(\mathbf{X})$ is constant for all classes, only

needs to be maximised

$$P(C_i|\mathbf{X}) = P(\mathbf{X}|C_i)P(C_i)$$

Derivation of naïve Bayesian classifier

- A simplified assumption: attributes are conditionally independent (i.e., no dependence relation between attributes):

$$P(\mathbf{X}|C_i) = \prod_{k=1}^n P(x_k|C_i) = P(x_1|C_i) \times P(x_2|C_i) \times \dots \times P(x_n|C_i)$$

- This greatly reduces the computation cost: Only counts the class distribution
- If attribute A_k is categorical, $P(x_k|C_i)$ is the number of tuples in C_i that have value x_k for A_k divided by $|C_{i,D}|$ (number of tuples of C_i in D)
- If A_k is a continuous-valued attribute, $P(x_k|C_i)$ is usually computed based on Gaussian (normal) distribution

• Formulas 6.13 and 6.14 in text book, page 312

Naïve Bayesian classifier: Training data set

Classes:

C1: buys_computer = "yes"

C2: buys_computer = "no"

Data sample (test record):

$\mathbf{X} = (\text{age} \leq 30, \text{income} = \text{"medium"}, \text{student} = \text{"yes"}, \text{credit_rating} = \text{"fair"})$

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

Naïve Bayesian classifier: An example

- $P(C_i)$: $P(\text{buys_computer} = \text{"yes"}) = 9/14 = 0.643$
 $P(\text{buys_computer} = \text{"no"}) = 5/14 = 0.357$
 - Compute $P(X|C_i)$ for each class:
 - $P(\text{age} = \text{"<=30"} | \text{buys_computer} = \text{"yes"}) = 2/9 = 0.222$
 - $P(\text{age} = \text{"<= 30"} | \text{buys_computer} = \text{"no"}) = 3/5 = 0.6$
 - $P(\text{income} = \text{"medium"} | \text{buys_computer} = \text{"yes"}) = 4/9 = 0.444$
 - $P(\text{income} = \text{"medium"} | \text{buys_computer} = \text{"no"}) = 2/5 = 0.4$
 - $P(\text{student} = \text{"yes"} | \text{buys_computer} = \text{"yes"}) = 6/9 = 0.667$
 - $P(\text{student} = \text{"yes"} | \text{buys_computer} = \text{"no"}) = 1/5 = 0.2$
 - $P(\text{credit_rating} = \text{"fair"} | \text{buys_computer} = \text{"yes"}) = 6/9 = 0.667$
 - $P(\text{credit_rating} = \text{"fair"} | \text{buys_computer} = \text{"no"}) = 2/5 = 0.4$
 - $\mathbf{X} = (\text{age} \leq \text{"30"}, \text{income} = \text{"medium"}, \text{student} = \text{"yes"}, \text{credit_rating} = \text{"fair"})$
 - $P(X|C_i)$: $P(X|\text{buys_computer} = \text{"yes"}) = 0.222 \times 0.444 \times 0.667 \times 0.667 = 0.044$
 - $P(X|\text{buys_computer} = \text{"no"}) = 0.6 \times 0.4 \times 0.2 \times 0.4 = 0.019$
 - $P(X|C_i) \cdot P(C_i)$: $P(X|\text{buys_computer} = \text{"yes"}) \cdot P(\text{buys_computer} = \text{"yes"}) = 0.028$
 - $P(X|\text{buys_computer} = \text{"no"}) \cdot P(\text{buys_computer} = \text{"no"}) = 0.007$
- Therefore, the naive Bayesian classifier predicts \mathbf{X} belongs to class: "buys_computer = yes"

Avoiding the 0-probability problem

- Naïve Bayesian prediction requires each conditional probability to be non-zero, as otherwise the predicted probability will be zero:

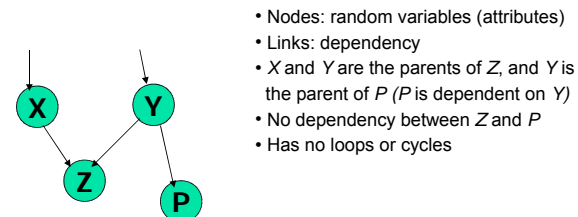
$$P(X|C_i) = \prod_{k=1}^n P(x_k|C_i)$$
- For example, suppose a data set with 1000 tuples
 - income = "low" (0), income = "medium" (990), income = "high" (10)
 - Use *Laplacian* correction (or *Laplacian* estimator)
 - Adding 1 to each case
 - $\text{Prob}(\text{income} = \text{"low"}) = 1/1003$
 - $\text{Prob}(\text{income} = \text{"medium"}) = 991/1003$
 - $\text{Prob}(\text{income} = \text{"high"}) = 11/1003$
- The "corrected" probability estimates are close to their "uncorrected" counterparts

Naïve Bayesian classifier: Some comments

- Advantages
 - Easy to implement
 - Good results obtained in most of the cases
- Disadvantages
 - Assumption: class conditional independence, therefore loss of accuracy
 - Practically, dependencies exist among variables
 - For example:
 - Hospital patients with attributes *age*, *family history*, etc.
 - Symptoms: *fever*, *cough* etc.
 - Diseases: *lung cancer*, *diabetes*, etc.
 - Dependencies among these cannot be modeled by naïve Bayesian classifier
- How to deal with these dependencies?
 - Bayesian Belief Networks

Bayesian belief networks

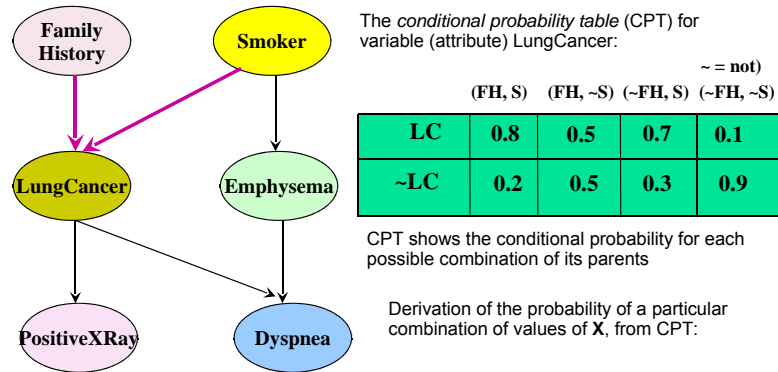
- Bayesian belief network allows a *subset* of the variables conditionally independent
- A graphical model of causal relationships
 - Represents dependency among the variables
 - Gives a specification of joint probability distribution



- Nodes: random variables (attributes)
- Links: dependency
- X and Y are the parents of Z , and Y is the parent of P (P is dependent on Y)
- No dependency between Z and P
- Has no loops or cycles

Source: Han and Kamber, DM Book, 2nd Ed. (Copyright © 2006 Elsevier Inc.)

Bayesian belief network: An example



$$P(\text{LC} = \text{"yes"} \mid \text{FH} = \text{"yes"}, \text{S} = \text{"yes"}) = 0.8$$

$$P(\text{LC} = \text{"no"} \mid \text{FH} = \text{"no"}, \text{S} = \text{"no"}) = 0.9$$

$$P(x_1, \dots, x_n) = \prod_{i=1}^n P(x_i \mid \text{Parents}(Y_i))$$

Source: Han and Kamber, DM Book, 2nd Ed. (Copyright © 2006 Elsevier Inc.)

Training Bayesian networks

- Several scenarios:
 - Given both the network structure and all variables observable: *learn only the CPTs*
 - If network structure known, some hidden variables: *gradient descent* (greedy hill-climbing) method, analogous to neural network learning
 - If network structure unknown, all variables observable: search through the model space to *reconstruct network topology*
 - If unknown structure, all hidden variables: No good algorithms known for this purpose
- Reference paper: David Heckerman: *Bayesian networks for data mining (1997)*

Measuring classifier accuracy

- Classification of new tuples/records results in a *confusion matrix* with positive and negative examples

		Predicted class	
		C ₁	C ₂
True class	C ₁	True positive	False negative
	C ₂	False positive	True negative

- Given m classes, Cm_{ij} (an entry in a confusion matrix) indicates the number of tuples in class i that are labeled by the classifier as class j
- Accuracy of a classifier M , $acc(M)$ is the percentage of test set tuples that are correctly classified by the model M
 - $accuracy = (true_pos + true_neg) / (all_pos + all_neg)$
- Error rate (misclassification rate) of $M = 1 - acc(M)$

Classifier accuracy measures

classes	buy_computer = yes	buy_computer = no	total	recognition(%)
buy_computer = yes	6954	46	7000	99.34
buy_computer = no	412	2588	3000	86.27
total	7366	2634	10000	95.42

- Accuracy in example here is: $(6954+2588) / 10000 = 95.42\%$
- Ideally, non-diagonal entries should be zero or close to it
- Alternative accuracy measures:
 - $sensitivity = true_pos / all_pos$ (true positive recognition rate) (also called *recall*)
 - $specificity = true_neg / all_neg$ (true negative recognition rate)
 - $precision = true_pos / (true_pos + false_pos)$ (also called *positive predictor value*)
- This model can also be used for cost-benefit analysis

Evaluating the accuracy of a classifier (1)

- **Holdout method**
 - Given data is randomly partitioned into two independent sets
 - Training set (for example, 2/3 of all records in data) for model construction
 - Test set (for example, 1/3 of all records in data) for accuracy estimation
 - Random sampling: a variation of holdout
 - Repeat holdout k times, accuracy = average of the accuracies obtained
- **Cross-validation (k -fold, where $k=10$ is most popular)**
 - Randomly partition the data into k mutually exclusive subsets, each approximately equal size
 - At i -th iteration, use D_i as test set and others as training set
 - *Leave-one-out*: k folds where k = number of tuples, for small sized data
 - *Stratified cross-validation*: folds are stratified so that class distribution in each fold is approximately the same as that in the initial data

Evaluating the accuracy of a classifier (2)

- **Bootstrap method**
 - Works well with small data sets
 - Samples the given training tuples (records) uniformly *with replacement*
 - Each time a tuple is selected, it is equally likely to be selected again and re-added to the training set
- **Several bootstrap methods, and a common one is .632 bootstrap**
 - Suppose we are given a data set of d tuples (records), which is sampled d times (with replacement) resulting in a training set of d samples
 - Records that did not make it into the training set end up forming the test set. About 63.2% of the original data will end up in the *bootstrap* (training set), and the remaining 36.8% will form the test set (since $(1 - 1/d)^d \approx e^{-1} = 0.368$)
 - Repeat the sampling procedure k times, overall accuracy of the model:

$$acc(M) = \sum_{i=1}^k (0.632 \times acc(M_i)_{testset} + 0.368 \times acc(M_i)_{trainset})$$

What now... things to do

- Start reading chapter 6 in text book (sections 6.1 to 6.4, and 6.12)