

CSE5230/DMS/2004/5

Data Mining - CSE5230

Classifiers 1

Bayesian Classification and Bayesian Networks

CSE5230 - Data Mining, 2004 Lecture 5.1

Lecture Outline

- ◆ What is Classification?
- ◆ Measuring Classifier Performance
- ◆ Bayesian Classifiers
- ◆ Bayes Theorem
- ◆ Naïve Bayesian Classification
 - ❖ Example application: spam filtering
- ◆ Bayesian Belief Networks
- ◆ Training Bayesian Belief Networks
- ◆ Why use Bayesian Classifiers?
- ◆ Example Software: Netica

CSE5230 - Data Mining, 2004 Lecture 5.2

Lecture Objectives

- ◆ By the end of this lecture you should be able to:
 - ❖ Explain what classification is
 - ❖ Give examples of several different types of classification algorithms and models used in data mining
 - ❖ State some basic performance measures that sophisticated classifiers should be able to beat if they are to be considered useful
 - ❖ Explain how Bayes rule relates the probabilities of evidence and class membership hypotheses
 - ❖ Explain why the Naïve Bayes classifier is called “naïve” – i.e. the assumption on which it is based
 - ❖ Explain how Bayesian networks can produce classifiers that are not “naïve”, and can exploit expert knowledge about relationships between variables

What is classification?

- ◆ A **classifier** assigns items to **classes**
 - ❖ In data mining, the items are typically records from a database
 - ❖ The classes are defined by the person doing the data mining
 - » each class has a **class label**, or name
 - this is the major difference between classification and clustering: clusters are not predefined, and are not labelled
- ◆ A classifier “decides” which class an item belongs in on the basis of the values of its attributes
- ◆ Often the class label corresponds to the value of one of the items’ attributes
 - ❖ The aim is to create a classifier that can predict the value of this attribute when the other attributes are known for a new data item
 - ❖ the attributes are divided into **input attributes** and the **output attribute** (or class attribute)

Learning Classifiers from Data

- ◆ Classifiers are typically “learnt” from a **training data set**
 - ❖ “learning” means determining the values of the parameters of the model
 - » some learning algorithms also determine the structure of the model
 - ❖ This is often called *training* the classifier
- ◆ The training data consists of many examples
 - ❖ each example is made up of a set of input attribute values and the desired output attribute value for that input
- ◆ This is an example of *supervised learning*
- ◆ Once a classifier is learnt, it should be tested on data not used during training – a **test data set**
 - ❖ Techniques exist for making use of the same data set for both training and tests, e.g. **cross-validation**

Classifiers used in Data Mining

- ◆ There are many different classifiers and classification models used in data mining
 - ❖ most have their origins in machine learning and artificial intelligence research
 - ❖ Some come from classic statistics
- ◆ Examples of classification models and classifiers:
 - ❖ Discriminant Analysis (<http://www.statsoftinc.com/textbook/stdiscan.html>)
 - ❖ K-Nearest-Neighbour Classifier
 - ❖ Naïve Bayes Classifier
 - ❖ Decision Trees (CART, CHAID, ID3, C4.5, etc.)
 - ❖ Feedforward Neural Networks (typically trained using the Backpropagation algorithm)
 - ❖ Support Vector Machines
 - ❖ and more...

Measuring Classifier Performance (1)

- ◆ It is important to know if a classifier does a good job
 - ❖ a classifier does a good job if it assigns new data items to the correct classes
- ◆ Classification performance is often expressed as a “percent correct” measure
 - ❖ classification performance of 70% correct means that 70% of items in the test data set are classified correctly
- ◆ It is important to note that 100% correct performance is often impossible to achieve – even on the training data set
 - ❖ it is usually possible to get better performance on the training data by increasing the model complexity...
 - ❖ ... but this often causes worse performance on the test data. This is called **over-fitting** the data.

Measuring Classifier Performance (2)

- ◆ “Percentage correct” measures can be misleading, and must be treated with caution
 - ❖ It is important to know the **prior probabilities** of the classes
- ◆ We really want to know if the classifier is doing better than we could be assigning items to classes at random, e.g.
 - ❖ imagine that we want to classify all the chickens in a shed as either male or female. In this shed, 90% of chickens are female
 - ❖ We would get “90% correct” performance just by classifying all chickens as female, without even looking at them!
- ◆ This is the first thing to check when evaluating a classifier: is it better than chance?

Measuring Classifier Performance (3)

- ◆ Whenever you are presented with a measure of classification performance, you should consider whether it is doing better than chance
- ◆ A measure exists that indicates how much better than chance performance actually is. This is known as Cohen's κ -statistic (kappa statistic) [Dun1989]:

$$\kappa = \frac{\text{observed frequency of agreement} - \text{expected frequency of agreement}}{1 - \text{expected frequency of agreement}}$$

where frequencies are in [0,1].

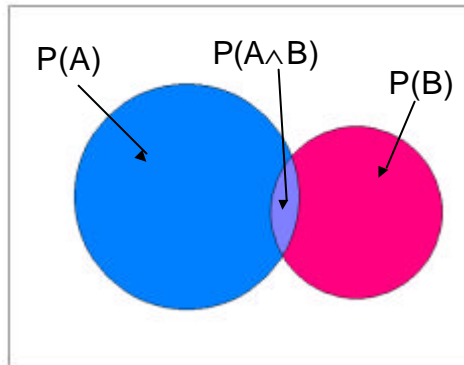
- ◆ There are also other, more sophisticated measures of classifier performance. Good data mining packages, such as Weka, report several

Bayesian Classifiers

- ◆ Bayesian Classifiers are **statistical** classifiers
 - ❖ based on **Bayes Theorem** (see following slides)
- ◆ They can **predict the probability** that a data item is a member of a particular class
- ◆ Perhaps the simplest Bayesian Classifier is known as the **Naïve Bayesian Classifier**
 - ❖ based on an independence assumption (which is usually incorrect)
 - ❖ performance is still often comparable to Decision Trees and Neural Network classifiers
 - ❖ First introduced as a "straw man" against which the performance of more sophisticated classifiers could be compared [CIN1989]
 - » This is the second thing you should check when evaluating classifier performance: is it better than Naïve Bayes?

Bayes Theorem - 1

- ◆ Consider the Venn diagram at right. The area of the rectangle is 1, and the area of each region gives the probability of the event(s) associated with that region
- ◆ $P(A|B)$ means “the probability of observing event A *given that* event B has already been observed”, i.e.
 - ❖ how much of the time that we see B do we also see A? (i.e. the ratio of the purple region to the magenta region)



$$P(A|B) = P(A \cap B) / P(B), \text{ and also } P(B|A) = P(A \cap B) / P(A), \text{ therefore}$$

$$P(A|B) = P(B|A)P(A) / P(B)$$

(Bayes formula for two events)

Bayes Theorem - 2

More formally,

- ◆ Let X be the data (evidence)
- ◆ Let H be a hypothesis that X belongs to class C
- ◆ In classification problems we wish to determine the **probability** that H holds given the observed data X
- ◆ i.e. we seek $P(H|X)$, which is known as the **posterior probability** of H conditioned on X
 - ❖ e.g. The probability that X is a kangaroo given that X jumps and is nocturnal

Bayes Theorem - 3

- ◆ P(H) is the **prior probability**
 - ❖ i.e. the probability that any given data is a kangaroo regardless of its method of locomotion or night time behaviour - i.e. before we know anything about X
- ◆ Similarly, P(X|H) is the **posterior probability** of X conditioned on H
 - ❖ i.e. the probability that X is a jumper and is nocturnal given that we know X is a kangaroo
- ◆ Bayes Theorem (from earlier slide) is then

$$P(H | X) = \frac{P(X | H)P(H)}{P(X)} \quad \text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{evidence}}$$

Naïve Bayesian Classification - 1

- ◆ Assumes that the effect of an attribute value on a given class is independent of the values of other attributes. This assumption is known as *class conditional independence*
 - ❖ This makes the calculations involved easier, but makes a simplistic assumption - hence the term "naïve"
- ◆ Can you think of a real-life example where the class conditional independence assumption would break down?

Naïve Bayesian Classification - 2

- ◆ Consider each data instance to be an n -dimensional vector of attribute values (i.e. features):

$$X = (x_1, x_2, \dots, x_n)$$

- ◆ Given m classes C_1, C_2, \dots, C_m , a data instance X is assigned to the class for which it has the greatest posterior probability, conditioned on X , i.e. X is assigned to C_i if and only if

$$P(C_i | X) > P(C_j | X) \quad \forall j \text{ s.t. } 1 \leq j \leq m, j \neq i$$

Naïve Bayesian Classification - 3

- ◆ According to Bayes Theorem:

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

- ◆ Since $P(X)$ is constant for all classes, only the numerator $P(X|C_i)P(C_i)$ needs to be maximized
- ◆ If the class probabilities $P(C_i)$ are not known, they can be assumed to be equal, so that we need only maximize $P(X|C_i)$
- ◆ Alternately (and preferably) we can estimate the $P(C_i)$ from the proportions in some training sample

Naïve Bayesian Classification - 4

- ◆ It is can be very expensive to compute the $P(X|C_i)$
 - ❖ if each component x_k can have one of c values, there are c^n possible values of X to consider
- ◆ Consequently, the (naïve) assumption of *class conditional independence* is often made, giving

$$P(X | C_i) = \prod_{k=1}^n P(x_k | C_i)$$

- ◆ The $P(x_1|C_i), \dots, P(x_n|C_i)$ can be estimated from a training sample

(using the proportions if the variable is categorical; using a normal distribution and the calculated mean and standard deviation of each class if it is continuous)

Naïve Bayesian Classification - 5

- ◆ Fully computed Bayesian classifiers are provably optimal
 - ❖ i.e. under the assumptions given, no other classifier can give better performance
- ◆ In practice, assumptions are made to simplify calculations (e.g. class conditional independence), so optimal performance is not achieved
 - ❖ sub-optimal performance is due to inaccuracies in the assumptions made
- ◆ Nevertheless, the performance of the Naïve Bayes Classifier is often comparable to that decision trees and neural networks [p. 299, HaK2000], and has been shown to be optimal under conditions somewhat broader than class conditional independence [DoP1996]

Application: Spam Filtering (1)

- ◆ You are all almost certainly aware of the problem of “spam”, or junk email
 - ❖ Almost every email user receives unwanted, unsolicited email every day:
 - » Advertising (often offensive, e.g. pornographic)
 - » Get-rich-quick schemes
 - » Attempts to defraud (e.g. the Nigerian 419 scam)
- ◆ Spam exists because sending email is extremely cheap, and vast lists of email addresses harvested from the internet are easily available
- ◆ Spam is a big problem. It costs users time, causes stress, and costs money (the download cost)

Application: Spam Filtering (2)

- ◆ There are several approaches to stopping spam:
 - ❖ Black lists
 - » banned sites and/or emailers
 - ❖ White lists
 - » allowed sites and/or emailers
 - ❖ Filtering
 - » deciding whether or not an email is spam based on its content
- ◆ “Bayesian filtering” for spam has got a lot of press recently, e.g.
 - ❖ “How to spot and stop spam”, BBC News, 26/5/2003
<http://news.bbc.co.uk/2/hi/technology/3014029.stm>
 - ❖ “Sorting the ham from the spam”, Sydney Morning Herald, 24/6/2003
<http://www.smh.com.au/articles/2003/06/23/1056220528960.html>
- ◆ The “Bayesian filtering” they are talking about is actually Naïve Bayes Classification

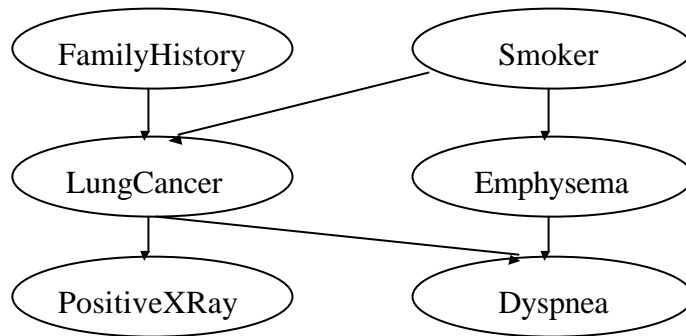
Application: Spam Filtering (3)

- ◆ Spam filtering is really a classification problem
 - ❖ Each email needs to be classified as either spam or not spam (“ham”)
- ◆ To do classification, we need to choose a classifier model (e.g. neural network, decision tree, naïve Bayes) and features
- ◆ For spam filtering, the features can be
 - ❖ Words
 - ❖ combinations of (consecutive) words
 - ❖ words tagged with positional information
 - » e.g. body of email, subject line, etc.
- ◆ Early Bayesian spam filters achieved good accuracy:
 - ❖ Pantel and Lim [PaL1998]: 98% true positive, 1.16% false positive
- ◆ More recent ones (with improved features) do even better
 - ❖ Graham [Gra2003]: 99.75% true positive, 0.06% false positive
 - ❖ This is good enough for use in production systems (e.g. Mozilla) – it’s moving out of the lab and into products

Bayesian Belief Networks - 1

- ◆ Problem with the naïve Bayesian classifier: dependencies **do** exist between attributes
- ◆ **Bayesian Belief Networks** (BBNs) allow for the specification of the *joint conditional probability distributions*: the class conditional dependencies can be defined between subsets of attributes
 - ❖ i.e. we can make use of prior knowledge
- ◆ A BBN consists of two components. The first is a *directed acyclic graph* where
 - ❖ each node represents an variable; variables may correspond to actual data attributes or to “hidden variables”
 - ❖ each arc represents a probabilistic dependence
 - ❖ each variable is conditionally independent of its non-descendants, given its parents

Bayesian Belief Networks - 2



- ◆ A simple BBN (from [HaK2000]). Nodes have binary values. Arcs allow a representation of causal knowledge

CSE5230 - Data Mining, 2004

Lecture 5.23

Bayesian Belief Networks - 3

- ◆ The second component of a BBN is a *conditional probability table (CPT)* for each variable Z , which gives the conditional distribution $P(Z|Parents(Z))$
 - ❖ i.e. the conditional probability of each value of Z for each possible combination of values of its parents
- ◆ e.g. for for node LungCancer we may have

$$P(\text{LungCancer} = \text{"True"} \mid \text{FamilyHistory} = \text{"True"} \wedge \text{Smoker} = \text{"True"}) = 0.8$$

$$P(\text{LungCancer} = \text{"False"} \mid \text{FamilyHistory} = \text{"False"} \wedge \text{Smoker} = \text{"False"}) = 0.9$$

...

- ◆ The joint probability of any tuple (z_1, \dots, z_n) corresponding to variables Z_1, \dots, Z_n is

$$P(z_1, \dots, z_n) = \prod_{i=1}^n P(z_i \mid Parents(Z_i))$$

CSE5230 - Data Mining, 2004

Lecture 5.24

Bayesian Belief Networks - 4

- ◆ A node within the BBN can be selected as an **output node**
 - ❖ output nodes represent class label attributes
 - ❖ there may be more than one output node
- ◆ The classification process, rather than returning a single class label (as many classifiers, e.g. decision trees, do) can return a probability distribution for the class labels
 - ❖ i.e. an estimate of the probability that the data instance belongs to each class
- ◆ A Machine learning algorithm is needed to find the CPTs, and possibly the network structure

Training BBNs - 1

- ◆ If the network structure is known and all the variables are observable then training the network simply requires the calculation of Conditional Probability Table (as in naïve Bayesian classification)
- ◆ When the network structure is given but some of the variables are *hidden* (variables believed to influence but not observable) a **gradient descent method** can be used to train the BBN based on the training data. The aim is to learn the values of the CPT entries

Training BBNs - 2

- ◆ Let S be a set of s training examples X_1, \dots, X_s
- ◆ Let w_{ijk} be a CPT entry for the variable $Y_i = y_{ij}$ having parents $U_i = u_{ik}$
 - ❖ e.g. from our example, Y_i may be LungCancer, y_{ij} its value "True", U_i lists the parents of Y_i , e.g. {FamilyHistory, Smoker}, and u_{ik} lists the values of the parent nodes, e.g. {"True", "True"}
- ◆ The w_{ijk} are analogous to weights in a neural network, and can be optimized using gradient descent (the same learning technique as backpropagation is based on). See [HaK2000] for details
- ◆ An important advance in the training of BBNs was the development of *Markov Chain Monte Carlo* methods [Nea1993]

Training BBNs - 3

- ◆ Algorithms also exist for learning the *network structure* from the training data given observable variables (this is a discrete optimization problem)
- ◆ In this sense they are an unsupervised technique for discovery of knowledge
- ◆ A tutorial on Bayesian AI, including Bayesian networks, is available at <http://www.csse.monash.edu.au/~korb/bai/bai.html>

Why use Bayesian Classifiers?

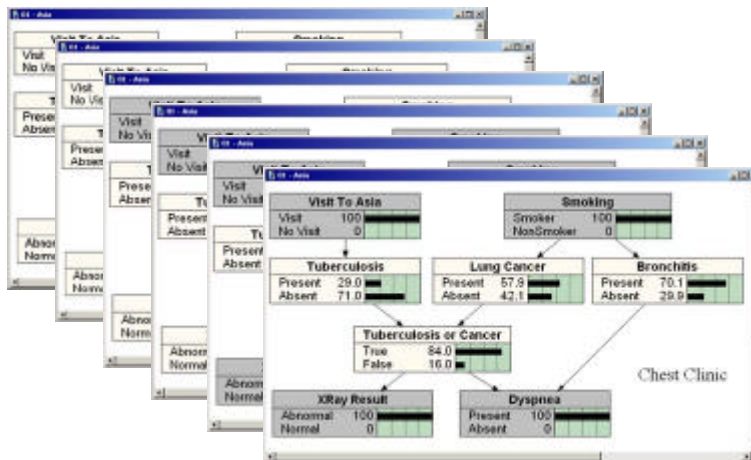
- ◆ No classification method has been found to be superior over all others in every case (i.e. a data set drawn from a particular domain of interest)
 - ❖ indeed it can be shown that no such classifier can exist (see “No Free Lunch” theorem [p. 454, DHS2000])
- ◆ Methods can be compared based on:
 - ❖ accuracy
 - ❖ interpretability of the results
 - ❖ robustness of the method with different datasets
 - ❖ training time
 - ❖ scalability
- ◆ e.g. neural networks are more computationally intensive than decision trees
- ◆ BBNs offer advantages based upon a number of these criteria (all of them in certain domains)

Example application – Netica (1)

- ◆ Netica is an Application for Belief Networks and Influence Diagrams from Norsys Software Corp. Canada
- ◆ <http://www.norsys.com/>
- ◆ Can build, learn, modify, transform and store networks and find optimal solutions using an inference engine
- ◆ A free demonstration version is available for download
- ◆ There is also a useful tutorial on Bayes Nets: http://www.norsys.com/tutorials/netica/nt_toc_A.htm

Example application – Netica (2)

- ◆ Netica Screen shots (from their tutorial):



CSE5230 - Data Mining, 2004

Lecture 5.31

References

- ◆ [CIN1989] Peter Clark and Tim Niblett, *The CN2 Induction Algorithm* Machine Learning 3(4), pp. 261-283, 1989.
- ◆ [DHS2000] Richard O. Duda, Peter E. Hart and David G. Stork, *Pattern Classification* (2nd Edn), Wiley, New York, NY, 2000
- ◆ [DoP1996] Pedro Domingos and Michael Pazzani. *Beyond independence: Conditions for the optimality of the simple Bayesian classifier*, In Proceedings of the 13th International Conference on Machine Learning, pp. 105-112, 1996.
- ◆ [Dun1989] Graham Dunn, *Design and Analysis of Reliability Studies: The Statistical Evaluation of Measurement Errors*, Edward Arnold, London, 1989.
- ◆ [Gra2003] Paul Graham, *Better Bayesian Filtering*, In Proceedings of the 2003 Spam Conference, Cambridge, MA, USA, January 17 2003
- ◆ [HaK2000] Jiawei Han and Micheline Kamber, *Data Mining: Concepts and Techniques*, The Morgan Kaufmann Series in Data Management Systems, Jim Gray, Series (Ed.), Morgan Kaufmann Publishers, August 2000
- ◆ [Nea2001] Radford Neal, *What is Bayesian Learning?*, in comp.ai.neural-nets FAQ, Part 3 of 7: Generalization, on-line resource, accessed September 2001 <http://www.faqs.org/faqs/ai-faq/neural-nets/part3/section-7.html>
- ◆ [Nea1993] Radford Neal, *Probabilistic inference using Markov chain Monte Carlo methods*. Technical Report CRG-TR-93-1, Department of Computer Science, University of Toronto, 1993
- ◆ [PaL1998] Patrick Pantel and Dekang Lin, *SpamCop: A Spam Classification & Organization Program* In AAAI Workshop on Learning for Text Categorization, Madison, Wisconsin, July 1998.
- ◆ [SDH1998] Mehran Sahami, Susan Dumais, David Heckerman and Eric Horvitz, *A Bayesian Approach to Filtering Junk E-mail*, In AAAI Workshop on Learning for Text Categorization, Madison, Wisconsin, July 1998.

CSE5230 - Data Mining, 2004

Lecture 5.32