

## CS 374 Assignment #2

### Bayesian Decision Theory and the Naive Bayes Classifier

Due the week of September 19, 2005

This week's topics are Bayesian Decision Theory and the Naive Bayes Classifier. The Naive Bayes Classifier is a simple probabilistic classifier that gets its name (i.e., "naive") from the fact that it makes a very strong assumption: that there are no dependencies among the attributes of an example. Despite its simplicity, the Naive Bayes Classifier often proves to be quite effective in practice.

This week you will have the opportunity to explore the Naive Bayes Classifier through a variety of different types of exercises. These involve:

- calculating probabilities,
- proving lower and upper bounds on errors of Bayesian Classifiers,
- implementing and testing a Naive Bayes Classifier of your own.

## 1 Bayesian Decision Theory

### 1.1 Reading

Please read Chapter 3: Bayesian Decision Theory (up to page 55) in Alpaydin. This chapter gives an introduction to probabilistic reasoning. In particular, it introduces Bayes' Rule in the context of classification. Bayes' Rule allows us to compute the probability of an example belonging to one of several classes, thereby allowing us to select the classification that is most appropriate.

In certain contexts, we will see that it is best to choose the class with highest probability. In other cases, we might combine this with other information, such as the cost of making a mistake in classification.

This chapter describes Bayesian Networks, which represent the relationships between the classes and attributes of the examples we would like to analyze. We will see how these structures enable us to easily compute the probabilities of class membership, given values for the attributes.

We will then turn to the Naive Bayes Classifier, which assumes a very simple ("naive") structure.

### 1.2 Exercises

Please do the following exercises on page 58 of Alpaydin:

- Exercise 3.
- Exercise 4
- Exercise 5.
- Exercise 6.

## 2 Implementing a Naive Bayes Classifier

One of the best ways to understand an algorithm is, of course, to implement it. So your next task will involve implementing a Naive Bayes Classifier. This type of classifier requires a great deal of information about the domain of examples to be classified, including:

- prior probabilities of the classes;
- likelihoods, i.e., conditional probabilities of the attribute values given the classes.

This is where the “learning” comes in. This information must be computed from sample data.

### 2.1 Reading

Begin by reading Section 3.4.2 on page 59 of Mitchell (attached). This describes a specific classification domain. (Those of you who took Artificial Intelligence will recognize this as the “Golf” domain.)

Next please read Section 4.2 in Witten and Frank. This section is on Naive Bayes Learning and Classification, and it very nicely describes the process of learning probabilities from examples (training data) and then using these probabilities to classify new examples.

### 2.2 Exercise

Please implement a Naive Bayes Learner and Classifier. For this exercise, you should assume that all attributes have nominal (i.e., discrete) values. You need not handle real-valued attributes.

Your program should begin by reading a file called *domainName.arff*, where *domainName* is the name of the domain of application. I will provide you with three such files:

- *weather.nominal.arff* - the weather domain described in Mitchell and also in Witten and Frank;
- *contact-lenses.arff* - given a description of a patient, determine the type of contact lenses that should be prescribed;
- *soybean.arff* - given a description of a soybean plant, determine the type of disease it has.

You can copy these files from

```
~andrea/shared/cs374/domains
```

Note that this is a standard format of files that we will be experimenting with later in the semester.

Each file begins with a preamble that describes the domain. This is followed by a section that specifies the attributes and the possible values for each attribute. The final attribute listed is the class to be predicted. Each attribute is on a separate line and has the following form:

```
@attribute outlook {sunny, overcast, rainy}
```

It begins with the string “@attribute”. This is followed by the attribute name. The possible attribute values are listed in curly braces, separated by commas. Please be careful when reading in this information. In particular, the word “attribute” is sometimes uppercase and at other times lowercase. Attribute values might be separated by commas only, or they may be separated by commas and blanks. Look at all of the data files to get a sense of the variety.

Each file also contains data – i.e., specific examples and their associated classifications. The data part of the file begins with the line

```
@data
```

Each line of data has the form:

```
sunny,hot,high,FALSE,no
```

This gives the values for each of the attributes defined earlier in the file. The final value is always the class value. Again, be careful about format. Sometimes examples are separated by commas only; sometimes they are separated by commas and blanks.

There are times when the values for certain attributes are unknown. These are indicated by “?”. While there might be missing attribute values, you can assume that all class values are fully specified.

Once you have read in all of the data, you should divide it into a training set of examples and a test set of examples. If the number of examples in the complete data collection is less than 100, then both the training and test sets should contain all of the examples. If the number of examples is 100 or more, then the first 75% should be used for training, with the remaining 25% used for testing.

Use the training data to calculate the prior probabilities and likelihoods that are necessary for classification of new examples. You should implement the Laplace estimator as described in Witten and Frank. This is especially important for the soybean domain.

Once the training is complete, you should classify the test examples. The output of your program should include:

- For each test example:
  - the example itself
  - the target class
  - the class assigned by your program
- Counts of the number of examples classified correctly and incorrectly.

*You may implement your program in any language, but it must compile and run on the Unix machines in our lab.* Please turn in paper copy as well as electronic copy. Instructions for turning in the electronic copy will follow later.

### 3 Error of Bayesian Classifiers

Finally, you will switch gears and do a bit of theoretical work involving lower and upper bounds on the error of the Bayes decision rule. This will also take you out of the realm of nominal-valued data and into real-valued data.

#### 3.1 Reading

Please read Section 2.1 of Duda, Hart, and Stork (attached). The first part of this section repeats some of what you learned from the reading in Alpaydin.

#### 3.2 Exercises

(Note that the following is problem 1 on page 65 of Duda, Hart, and Stork.)

In the two-category case, under the Bayes decision rule the conditional error is given by Eq. 7. Even if the posterior densities are continuous, this form of the conditional error virtually always leads to a discontinuous integrand when calculating the full error by Eq. 5.

- Show that for arbitrary densities, we can replace Eq. 7 by  $P(\text{error}|x) = 2P(\omega_1|x)P(\omega_2|x)$  in the integral and get an upper bound on the full error.
- Show that if we use  $P(\text{error}|x) = \alpha P(\omega_1|x)P(\omega_2|x)$  for  $\alpha < 2$ , then we are not guaranteed that the integral gives an upper bound on the error.
- Analogously, show that we can use instead  $P(\text{error}|x) = P(\omega_1|x)P(\omega_2|x)$  and get a lower bound on the full error.
- Show that if we use  $P(\text{error}|x) = \beta P(\omega_1|x)P(\omega_2|x)$  for  $\beta > 1$ , then we are not guaranteed that the integral gives a lower bound on the error.