Graphical Models Lab (Lab 3)

The naïve Bayes model is a very simple example of a graphical model, as we saw in class. Let's see whether relaxing the strong assumptions of the naïve Bayes model, obtaining a slightly less restrictive graphical model, results in improved classification performance. Fitting a more complex model will require more data, so we will try this approach on a large data set. The data set we will use is adapted from the "Adults" data set on the UCI machine learning data repository. This data set includes demographic information for tens of thousands of individuals, as well as whether or not each individual's income exceeds 50K. We will attempt to predict whether income exceeds 50K based on the demographic variables. Frequently in marketing you do not have the exact information that you would like to have on customers, for instance their income. Often other information such as demographics must be used as surrogates.

Read in the adult.data.preproc training data set on Blackboard using the R read.table command. There is an argument to the read.table function that allows you to set the variable (column) names in the resulting data.frame object; do this, setting the names to: "age", "workclass", "education", "marital", "race", "sex", and "inc". The data values in the file are separated by a comma and then a space. The read.table function allows you to specify the separator, using the "sep" argument; however, it must be a single character. Use 'sep = "," ' (just a comma) but be aware that the non-numeric predictors will all be read in with a space before the named value. For instance, calling

```
> levels( as.factor( incomeTrain$inc ) )
```

gives the two values " >50K" and " <=50K" (notice the space at the beginning of these strings).

There are some missing values in the data sets, encoded by a question mark. Accordingly set the argument 'na.strings = " ?" ' for the read.table function, noticing that there is a space before the question mark and inside the quotes. Check that the missing data has been read in correctly by calling

```
> summary( incomeTrain$workclass )
```

where incomeTrain is the training data set. This should give counts for each possible value of workclass, including a count of the NAs. There should be no question-mark values for this predictor.

Read in the test data set from Blackboard in the same manner as for the training data; the data set is called "adult.test.preproc". Notice that the age variable in the training and test data sets takes values TRUE and FALSE; this is because it has been split according to being above or below the median age.

Your professor has once again been nice enough to alter the naïve Bayes prediction function for you. This time, the function allows you to alter the probability threshold at which the outcome is predicted to be T vs. F (in this case, "+" vs. "-"). The code is available in the naiveBayesThreshold.R file on Blackboard. The default threshold is 0.5. In order to change the threshold, add the argument e.g. "threshold = 0.6" when you call

the prediction function on the test data. Train the model using the new nb.train before you predict using nb.predict. For now, use a threshold of 0.5 (the default).

What is your overall error rate? What are your false positive and false negative rates? Consider " >50K" to be a positive and " <=50K" to be a negative. Report the answers in homework 3.

It turns out that the two predictor variables "age" and "marital" are highly dependent, conditional on the outcome variable. We will relax the naïve Bayes assumption that this pair of predictors is independent, conditional on the outcome variable. Create a variable that encodes both the predictors together. In other words, this new variable (call it incomeTrain$ageMarital) has one value for each pair of values for the two predictor variables. Since marital has 3 values and age has 2 values, then incomeTrain$ageMarital should take 6 different values. The code to do this is:

```
> incomeTrain$ageMarital =
  1*( (incomeTrain$age == TRUE) & (incomeTrain$marital == " Divorced") ) +
  2*( (incomeTrain$age == TRUE) & (incomeTrain$marital == " Married/Widowed") ) +
  3*( (incomeTrain$age == TRUE) & (incomeTrain$marital == " Never-married") ) +
  4*( (incomeTrain$age == FALSE) & (incomeTrain$marital == " Divorced") ) +
  5*( (incomeTrain$age == FALSE) & (incomeTrain$marital == " Married/Widowed") ) +
  6*( (incomeTrain$age == FALSE) & (incomeTrain$marital == " Never-married") )
> incomeTrain$ageMarital = as.factor( incomeTrain$ageMarital )
```

Notice the space at the beginning of " Divorced", etc. Check that your new variable is correct by calling

```
> summary( incomeTrain$ageMarital )
```

and comparing it to

```
> table( incomeTrain$age, incomeTrain$marital )
```

The output of these two calls will not be formatted in the same way but the counts should be the same.

You will also need to create a new variable ageMarital for the test data set in the same way. You will then need to create new training and test data sets that do not have the "marital" and "age" predictors but do have the new ageMarital predictor. Remember that for the purposes of the nb.train and nb.predict functions the last column in the new train and test data sets must be the outcome variable.

Call nb.train on the new training data, and nb.predict on the new test data. **Did your overall error rate improve? Did your false positive and false negative rates change?**

Graphical Models Homework (HW 3)

1. Report your answers to the questions in bold from lab. A few sentences should be plenty.

2. Recall that during one Wednesday lecture we fit naïve Bayes to the credit risk data. The object that was returned from the call to nb.train ("creditTrained") contains the estimated marginal distribution of the outcome variable and the estimated conditional distribution of each predictor variable given the outcome.

Estimate the probability that an individual is a good credit risk, if V1 = b and V2 = 75 and all other predictors are unknown. Compute by hand using the estimated distributions from creditTrained. Show your work, and report what quantities in your calculations were obtained from creditTrained.

3. This question is also regarding the credit data analysis. The new nb.predict function allows you to alter the probability threshold at which the outcome is predicted to be T vs. F (in this case, "+" vs. "-"). Try different values of the threshold (between 0 and 1). The nb.predict function does not actually know that "+" means a positive and "-" means a negative, so depending on your test data set it might get it backwards. The threshold argument is interpreted relative to what nb.predict thinks is a positive. So you may find that when you set threshold = 0 you get false positive and true positive rates of 0 instead of 1 as we learned in class. This will not make a difference for the ROC curve in question 4, but it may affect some of the answers that you get for this question; don't worry about it.

How does changing the threshold change the error rate? How does it change what types of errors are made (i.e. the false positive and false negative rates)?

4. Regarding question 3, obtain the true positive and false positive rates at 5 or more values of the threshold (between 0 and 1 inclusive), and generate an ROC curve. Choose the threshold values so that you get a fairly smooth curve. Label the axes appropriately, and make sure that your x- and y-axis ranges are appropriate for the ROC plot. An ROC plot is always a line plot (a curve), not a scatter plot!

5. Create ROC curves for the two classifiers that you created in Lab 3 for the income data set. Plot the two ROC curves on the same plot, using different colors. Label the x and y axis appropriately, and choose the threshold values so that you get fairly smooth curves. Using the ROC curves, is one of the classifiers universally better than the other? How can you tell? Hint: in order to get the two curves on the same plot, plot the first using e.g.

```
> plot( x = c( 0, 0.05, 0.1, 0.15, 1 ), y = c( 0, 0.4, 0.5, 0.6, 1), type = "l" )
```

then plot the second using e.g.
> lines(x = c(0, 0.07, 0.13, 0.18, 1), y = c(0, 0.4, 0.51, 0.62, 1), col = “red”)
Of course, you will need to replace these “x” and “y” values with vectors of appropriate false-positive and true-positive rates from your classifiers.