module-12-programming-only

April 21, 2019

1 Module 12 - Programming Assignment

1.1 Directions

There are general instructions on Blackboard and in the Syllabus for Programming Assignments. This Notebook also has instructions specific to this assignment. Read all the instructions carefully and make sure you understand them. Please ask questions on the discussion boards or email me at `EN605.645@gmail.com` if you do not understand something.

You must follow the directions *exactly* or you will get a 0 on the assignment.

Please submit your file, `<jhed_id>.py` to Blackboard when you are done.

Add whatever additional imports you require here. Stick with the standard libraries and those required by the class. The import gives you access to these functions: http://ipython.org/ipython-doc/stable/api/generated/IPython.core.display.html (Copy this link) Which, among other things, will permit you to display HTML as the result of evaluated code (see HTML() or display_html()).

1.2 Naive Bayes Classifier

For this assignment you will be implementing and evaluating a Naive Bayes Classifier with the same data from last week:

http://archive.ics.uci.edu/ml/datasets/Mushroom

(You should have downloaded it).

You'll first need to calculate all of the necessary probabilities (don't forget to use +1 smoothing) using a `train` function. You'll then need to have a `classify` function that takes your probabilities, a List of instances (possibly a list of 1) and returns a List of Dicts. Each Dict has a key for every possible class label and the associated *normalized* probability. For example, if we have given the `classify` function a list of 2 observations, we would get the following back:

```
[{"e": 0.98, "p": 0.02}, {"e": 0.34, "p": 0.66}]
```

when calculating the error rate of your classifier, you should pick the class label with the highest probability; you can write a simple function that takes the Dict and returns that class label.

As a reminder, the Naive Bayes Classifier generates the *unnormalized* probabilities from the numerator of Bayes Rule:

$$P(C|A) \propto P(A|C)P(C)$$

where C is the class and A are the attributes (data). Since the normalizer of Bayes Rule is the *sum* of all possible numerators and you have to calculate them all, the normalizer is just the sum of the probabilities.

You'll also need an `evaluate` function as before. You should use the *error_rate* again.

With +1 smoothing, the Naive Bayes Classifier has quite a different way of handling missing values.

Again, you must implement the following functions:

`cross_validate` takes the data and performs 10 fold cross validation.

`train` takes training_data and returns the probabilities as a data structure. If some kind of ADT seems reasonable to you, then you can create one but you don't really need one. Nested Dicts will work just fine.

`classify` takes the probabilities produced from the function above and applies it to labeled data (like the test set) or unlabeled data (like some new data).

`evaluate` takes a data set with labels (like the training set or test set) and the classification result and calculates the classification error rate:

$$error\_rate = \frac{errors}{n}$$

```
def train(training_data):
   # returns a NBC probability structure
```

and `classify` takes probabilities and a List of instances (possibly just one) and returns the classifications:

```
def classify(probabilities, test_data):
    # returns a list of classifications
```

and `evaluate` takes the actual classifications and the predicted classes and returns the classification error rate:

```
def evaluate(actual, predicted):
    # returns an error rate
```

You must apply 10 fold cross validation to your data set. You will treat each fold as a test set, using the combined remainder as the training set. You should print out the error rate for each fold and then an average error rate for the entire cross validation process.

This is all that is required for this assignment. I'm leaving more of the particulars up to you but you can definitely use the last module as a guide.

1.3 Results

You will get results that are different from last week's. Why? What about the Mushroom problem is likely to lead to very different results for a Decision Tree versus Naive Bayes? Specifically, how do their representations embody different assumptions?