Project 05: Machine Learning (100 points, Due Fri Dec 15 before midnight)

1. Clickstream Mining with Decision Trees [50 points]

The project is based on a task posed in KDD Cup 2000. It involves mining click-stream data collected from Gazelle.com, which sells legware products. Your task is to determine: Given a set of page views, will the visitor view another page on the site or will he leave?

The data set given to you is in a different form than the original. In particular it has discretized numerical values obtained by partitioning them into 5 intervals of equal frequency. This way, we get a data set where for each feature, we have a finite set of values. These values are mapped to integers, so that the data is easier for you to handle

Dataset

The data set can be downloaded from here.

You have 5 files in .csv format

1. `trainfeat.csv`: Contains 40000 examples, each with 274 features in the form of a 40000 x 274 matrix.
2. `trainlabs.csv`: Contains the labels (class) for each training example (did the visitor view another page?)
3. `testfeat.csv`: Contains 25000 examples, each with 274 features in the form of a 25000 x 274 matrix.
4. `testlabs.csv`: Contains the labels (class) for each testing example.
5. `featnames.csv`: Contains the "names" of the features. These might useful if you need to know what the features represent.

The format of the files is not complicated, just rows of integers separated by empty spaces.

Stopping Criterion: Chi-squared criterion

What about split stopping? Suppose that the attribute that we want to split on is irrelevant. Then we would expect the positive and negative examples (labels 1 and 0) to be distributed according to a specific distribution. Suppose that splitting on attribute T, will produce sets `{T_i} where i = 1 to m`

Let p, n denote the number of positive and negative examples that we have in our dataset (not the whole set, the remaining one that we work on at this node). Let (N is the total number of examples in the current dataset):

$$p_i^{'} = p\frac{|T_i|}{N}$$

$$n_i^{'} = n\frac{|T_i|}{N}$$

be the expected number of positives and negatives in each partition, if the attribute is irrelevant to the class. Then the statistic of interest is:

$$S = \sum_{i=1}^{m} \left(\frac{\left(p_i' - p_i\right)^2}{p_i'} + \frac{\left(n_i' - n_i\right)^2}{n_i'} \right)$$

where `p_i, n_i` are the positives and negatives in partition `T_i`. The main idea is that S is distributed (if the class is irrelevant) according to a chi-squared distribution with `m-1` degrees of freedom.

Now we must compute the p-value. This is the probability of observing a value X at least as extreme as S coming from the distribution. To find that, we compute `P(X >= S)`. The test is passed if the p-value is smaller than some threshold. Remember, the smallest that probability is, the more unlikely it is that S comes from a chi-squared distribution and consequently, that T is irrelevant to the class.

Your Task:

Code (45 points)

Implement the ID3 decision tree learner on Python. Your program should use the chi-squared split stopping criterion with the p-value threshold given as a parameter. Use your implementation with the threshold for the criterion set to 0.05, 0.01 and 1. Remember, 1 corresponds to growing the full tree.

Report (5 points)

1. For each value of threshold, what is your tree's accuracy and size (size equals number of internal nodes and leaves)? What do you observe? If all your accuracies are low, tell us what you have tried to improve the accuracies and what you suspect is failing.

2. Explain which options work well and why?

Your submission file should be named `'q1_classifier.py'` and it should run as follows:

```
Usage: python q1_classifier.py -p <pvalue> -f1 <train_dataset> -f2 <test_dataset> -o <output_file> -t <decision_tree>
```

Where:

- `<train_dataset>` and `<test_dataset>` are .csv files
- `<output_file>` is a csv file containing the predicted labels for the test dataset (same format as the original `testlabs.csv`)
- `<decision_tree>` is your decision tree [To be posted soon with a sample autograder]
- `<pvalue>` is the p-value threshold as described above

Note:

1. You may find the function `scipy.stats.chisquare` helpful
2. You should implement the `ID3` algorithm yourself
3. You may use libraries such as `pandas` for parsing the data. You can also use standard libraries such as `scipy` and `numpy`

---

2. Spam Filter [50 points]

The dataset we will be using is a subset of 2005 TREC Public Spam Corpus. It contains a `training set` and a `test set`. Both files use the same format: each line represents the space-delimited properties of an email, with the first one being the email ID, the second one being whether it is a spam or ham (non-spam), and the rest are words and their occurrence numbers in this email. In preprocessing, non-word characters have been removed, and features selected similar to what Mehran Sahami did in his original paper using Naive Bayes to classify spams.

Dataset

The data set can be downloaded from `here`.

Your Task:

Code (45 points)

Implement the Naive Bayes algorithm to classify spam.

Report (5 points)

Use your algorithm to learn from the training set and report accuracy on the test set. Try various smoothing parameters for the Naive Bayes learner.

Which parameters work best?

Extra Credit (10 points)

Features selected makes learning much easier, but it also throws out useful information. For example, exclamation mark (!) often occurs in spams. Even the format of email sender matters: in the case when an email address appears in the address book, a typical email client will replace it with the contact name, which means that the email is unlikely to be a spam (unless, of course, you are a friend of the spammer!). Sahami's paper talked about a few such features he had used in his classifier. For extra credit, you can play with the original files and come up with useful features that improve your classifier. `Index.zip` contains the list of the files used in train/test.

Your submission file should be named `'q2_classifier.py'` and it should run as follows:

```
Usage: python q2_classifier.py -f1 <train_dataset> -f2 <test_dataset> -o <output_file>
```

Where:

- `<train_dataset>` and `<test_dataset>` contain space delimited properties of an email
- `<output_file>` is a csv file containing the predicted labels for the test dataset

---

What to Submit: You should submit a **zip** file containing:

- Source code `q1_classifer.py, q2_classifier.py` (**Python 2.x**) with good documentation.
- A Report (PDF) with the required explanations for each problem.

All Project submissions must be made through Blackboard.

Academic Dishonesty: We will be checking your code against other submissions in the class for logical redundancy. If you copy someone else's code and submit it with minor changes, we will know. These cheat detectors are quite hard to fool, so please don't try. We trust you all to submit your own work only; *please* don't let us down. If you do, we will pursue the strongest consequences available to us.

Getting Help: Feel free to use the Piazza discussion board to discuss or get clarifications on homework-related issues. If you find yourself stuck on something, go to office hours or email the TAs for help. We want these projects to be rewarding and instructional, not frustrating and demoralizing. But, we don't know when or how to help unless you ask.