CSC 422/522: HW3

Total: 60 points

Question 1 [10 Points]

Devise a training set for use as input to the ID3 algorithm in order to produce a decision tree with a particular structure. We're providing the required target decision tree that should be induced using your training set. In other words, instead of the usual tree induction learning problem where we're given training data and a tree learner and we see what the program outputs, for this assignment you are given the desired output, and must design the input to produce it. To test your candidate training sets, you may use either Weka, Matlab or R.

Construct a training set having the appropriate format for the classifier you are using, which, when used with the classifier, comes as close as you can to outputting the given decision tree. The decision tree:

```
tear-prod-rate=(normal)
|  astigmatism=(yes)
|  |  spectacle-prescrip=(hypermetrope): none
|  |  spectacle-prescrip!=(hypermetrope): hard
|  astigmatism!=(yes)
|  |  age=(presbyopic): soft
|  |  age!=(presbyopic): none
tear-prod-rate!=(normal): none
```

"Outputting the tree" means the tree structure is the same as above, with the same decision nodes in the same places, etc. Generating just any tree, no matter how accurate, is not sufficient. There must be no missing data in your training set. In other words, each example must have values for all attributes. For each example, the training set should provide values for all the following attributes, including the attributes that do not appear in the target tree. The list of attributes is given in Table 1.

Deliverables: Turn in the following:

1. A description of how you arrived at the dataset. An answer along the lines of "I got lucky" or "this was simple trial and error" are not acceptable.
2. A description of how to load the dataset in either Matlab, Weka or R, and the toolbox or package used.
3. A text file with the dataset you generated.

Attributes	Values
age	{young, pre-presbyopic, presbyopic}
spectacle-prescrip	{myope, hypermetrope}
astigmatism	{no, yes}
tear-prod-rate	{reduced, normal}
contact-lenses (Target Class)	{soft, hard, none}

Table 1: List of attributes and the values

Question 2 [18 Points]

For this exercise, we will use the *Balloons* dataset (`http://archive.ics.uci.edu/ml/datasets/Balloons`). Specifically, we will use the **yellow-small+adult-stretch** data.
(`http://archive.ics.uci.edu/ml/machine-learning-databases/balloons/yellow-small+adult-stretch.data`).

The data has 16 entries each four attributes, all of which are nominal. Each entry belongs to one of the two classes {T, F}. Complete the following tasks.

1. [6 Points] Compute the tree using Gini index. Show all necessary calculations.
2. [6 Points] Compute the tree using ID3/ entropy computations. Show all necessary calculations.
3. [3 Points] Are the trees different? Why? If the trees are different, state which tree is better, and why. Your reasoning must be independent of the test set provided. That is, it must not be along the lines of "Tree A gives better accuracy on the test set than Tree B".
4. [3 Points] Use **yellow-small** data set as the test set and compute the confusion matrix in each case. If the trees are the same, one confusion matrix is enough, of course.
(`http://archive.ics.uci.edu/ml/machine-learning-databases/balloons/yellow-small.data`)

Question 3 [22 Points]

Classifiers such as decision trees, neural networks, naive bayes, and many others estimate the conditional probability of class membership. For standard two-class classification problems, the goal is to create a decision procedure that minimizes some notion of prediction error. There are many metrics that measure this, but the 'naive' optimal decision for all metrics in this case is:

$$\hat{y} = \begin{cases} 1 & : P(y=1|X) > 0.5 \\ 0 & : P(y=1|X) \leq 0.5 \end{cases}$$

where, $\hat{y}$ is the predicted class, '0' or '1', and $P(y=1|X)$ is the estimated probability of class membership provided by the classifier.

This is just a technical way of saying that we will predict the most likely class. This is the optimal rule if the cost of making any decision is the same or if the cost function is "symmetric". In this problem we will investigate what happens if the cost function is not symmetric. In a non-symmetric cost function, the penalty associated with misclassification can be different for false negative and false positive predictions, depending on the domain.

Pretend you have been approached by a large credit card company. They would like for you to help improve their current fraud detection scheme. They have a wealth of data on consumer transactions that they have used to develop a fraud classifier. This classifier takes as input data on the customer requesting the transaction and the specifics of the transaction such as time, location, and many other details. The classifier outputs an estimated probability that the transaction is fraudulent.

However, for the credit card company the cost of every decision is not the same. These costs are summarized in Table 2 below. If the transaction is fraudulent and is not caught, the credit card company loses the entire transaction amount. Similarly, if the company decides to decline a transaction because they think it is fraudulent but it is not, the customer may be upset and close their account or use it less in the future, costing the credit card company money.

	Prediction	Truth	Cost
1.	Predict No Fraud	Transaction Fraudulent	Transaction Amount (T_i)
2.	Predict Fraud	Transaction Legitimate	$10 + 0.05 * T_i$
3.	Fraud or No Fraud	Correct	0

Table 2: Cost of Classification

Using this setup, you will be asked to derive the optimal decision rule and compare it to the naive rule. Download the file `fraud_probs.csv`. This file contains information on 152880 transactions. The 3 columns are the transaction amount, the estimated probability of fraud from the classifier, and whether or not the transaction was later reported to be fraudulent. A '1' in the 3rd columns means there was fraud while a '0' means there was none.

1. [2 Points] Compute the cost of 'doing nothing', i.e. always predicting '0', or predicting "No Fraud" using the cost matrix given above.

2. [3 Points] Calculate the cost of using the naive rule, i.e. predicting '1' when the probability of fraud is > 0.5.

3. [4 Points] Calculate the average False Positive cost and the average False Negative cost. To compute the average false positive cost, compute what the cost would be if you made a false positive mistake for every transaction in the data set and then take the average. Repeat this procedure to determine the average false negative cost.

4. [4 Points] Now let's see if we can do better than the naive rule. Let us say for a given transaction and an estimate from the classifier that the transaction is fraudulent, p_i, we decide to predict fraud and decline the transaction. What is the *expected cost* of this decision? Expected cost is defined below:

   (probability decision is correct)*(cost if you are correct) +
   (probability decision is wrong)*(cost if you are wrong)

   Please use the established notation in answering this. This should be a function of p_i and the costs given in Table 2.

5. [4 Points] What if instead we decided to predict that the same transaction was legitimate (i.e. not fraudulent) and accept the transaction. Using the same fraud probability estimate, p_i, what is the expected cost in this case?

6. [5 Points] **[Compulsory for Grad students, Extra Credit for Under-Grad]**
   For every transaction, compute the cost for predicting fraud (1) or no fraud (0) using the expected costs from (4) and (5). For each transaction, predict the class with the lower expected cost. Compute the total cost using this procedure and compare it to the cost of the naive rule in (2).

Turn in any code you have written for answering this question. The code may be Matlab or R scripts or functions.