CSE 446 (Winter 2014)
1st Assignment

1 Decision trees

1.1 Description

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. Please browse the KDD Cup **website** to understand the domain, and read the organizer's **report**. Your task is Question 1 from the KDD Cup: 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. We have discretized numerical values 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. We mapped these values to integers, so that the data is easier for you to handle. Even though we mapped features that were unavailable to integers too, you are encouraged to read (Mitchell) on what are the usual practices for missing attributes.

1.2 Datasets

Download the compressed file containing the data (**hw1data.tar.gz**) from the course website. There you will find 5 files in **.csv** format:

1. **trainfeat.csv**: Contains 40000 examples, each with 274 features in the form of a 40000×274 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 25000×274 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.

Remember that in a "real" application, you can synthesize new features from the original ones if you think that they will be useful (if you want you can try that for yourself, it is not required). The format of the files is not complicated, just rows of integers separated by empty spaces.

1.3 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 $\{T_i\}_{i=1}^m$. 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:

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

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) accoring to a chi-squared distribution with $m-1$ degrees of freedom (if you want you can work out why, not required).

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 $P(X \geq S)$. 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.

1.4 Implementation

Your task is to implement the ID3 decision tree learner, as described in Chapter 8 of Duda et al. or Chapter 3 of Mitchell. Another very good description of the algorithm can be found in the original ID3 paper, which is pretty readable (**pdf**). You may program in C/C#/C++, Java, Python or R. If you are not proficient in any of these, please contact the TAs. Your program should take the given .csv files as input. For initial debugging, it is recommended that you construct a very simple data set (e.g., based on a boolean formula) and test your program on it. Your program should use the chi-squared split stopping criterion with the p-value threshold given as a parameter.

1.5 Questions

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. Answer the following questions:

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.

3. For your best performing tree, choose a path that you think explains a large fraction of the data. Pick the first few nodes and argue whether they make sense or not (as in do they intuitively provide a lot of information).

4. **Extra Credit** (5%): Suppose you have $10 - 20$ different values for the p-value threshold. How would you decide on the best model and why?

   (a) Would you pick the model with the lowest training error? Explain why.

   (b) Would you pick the model with the lowest testing error? Explain why.

   (c) Would you do something else? Explain why.

5. **Extra Credit** (5%): Try to improve your predictive accuracy. There are many approaches you could try. You could construct new attributes from the existing ones and augment the examples with them. You could also try alternative classification techniques, or modify one that you used above. Explain.

Your report should also contain a good high level description of how your code works. Document well any complicated parts you might have. You are not required to provide code that is extremely well optimized, but you should look for any obvious, gross optimizations that are possible.

2 Rule Induction

Mitchell 10.1: Consider a sequential covering algorithm such as CN2 and a simultaneous covering algorithm such as ID3. Both algorithms are to be used to learn a target concept defined over instances represented by conjunctions of n boolean attributes. If ID3 learns a balanced decision tree of depth d, it will contain $2^d - 1$ distinct decision nodes, and therefore will have made $2^d - 1$ distinct choices while constructing its output hypothesis. How many rules will be formed if this tree is re-expressed as a disjunctive set of rules? How many preconditions will each rule possess? How many distinct choices would a sequential covering algorithm have to make to learn this same set of rules? Which system do you suspect would be more prone to overfitting if both were given the same training data?

NOTE: Submit your homework on the **dropbox**