UNIVERSITY OF SASKATCHEWAN

Department of Computer Science
176 Thorvaldson Building
110 Science Place, Saskatoon, SK, S7N 5C9, Canada
Telephine: (306) 966-4886, Facimile: (306) 966-4884

Assignment 10

Decision Trees

Date Due: Thursday April 6, 5:00pm **Total Marks: 11**

General Instructions

- **This assignment is individual work.** You may discuss questions and problems with anyone, but the work you hand in for this assignment must be your own work.
- If you intend to use resources not supplied by the instructor, please request permission prior to starting. In any case, you must provide an attribution for any external resource (library, API, etc) you use. **You should not use any resource that substantially solves the problems in this assignment.** If there's any doubt, ask! No harm can come from asking, even if my answer is no.
- Each question indicates what to hand in.
- **Assignments must be submitted to Canvas.**

Question 1 (6 points):

Purpose: To understand decision trees

For this question, your task is to implement **feature ordering** for building a **decision tree**.

You have been provided with a codebase that can build a decision tree for simple **binary** classification problems (i.e. where the prediction label for each data point is simply `yes` or `no`). As given, the code can both build the decision tree from a data file and then classify data points using that tree.

However, currently when building the tree, the features are selected in an arbitrary order. We can improve the tree by selecting features in order of their **information gain**. Your job is to implement the `get_best_feature()` method in the provided code (along with any helper methods that you think you need). Refer to section 19.3 of AIMA and the lecture slide notes for "Decision Trees', especially slides 12 and 13.

You'll notice that the `get_best_feature()` method is called in the `__build_rec()` method, on line 243. You can toggle that method call to call either the 'old' version (that returns an arbitrary feature) or your new version (which should return the best feature), thus resulting in a different tree.

Testing Data

Some simple data has been provided to you to help verify that your decision tree is still working both before and after your modifications.

- `xordata.txt`: Data for a xor-gate. You should get the same tree (or at least the same shape of tree) both before and after doing feature ordering.
- `anddata.txt`: Data for an and-gate. You should get a different tree than for xor, but again feature ordering shouldn't matter.
- `robots.txt`: For this data, feature ordering should result in a different tree. A quick look at the data should make it obvious as to why.

The last data file, `gamedata.txt`, is more complex and is primarily intended for the next question.

What to Hand In

- Your updated code in a file called `DecitionTree.py`

Evaluation

- 6 marks: The feature ordering is correctly and cleanly implemented

Question 2 (5 points):

Purpose: To explore a data set with a decision tree

You have been provided with a binary classification data set in `gamedata.txt`. It contains information about the video game collection of a certain individual. Information about each game has been summarized in terms of five features with the following categorical values.

- **Theme**: fantasy, scifi, historical, modernday, superhero, or kaiju
- **Genre**: platformer, strategy, tactical, action, rpg, arpg, deckbuilder, shooter, or datingsim
- **Gameplay**: realtime or turnbased
- **Perspective**: side, topdown, 1stperson or 3rdperson
- **Graphics**: 2d or 3d

The last column is the **classification label**, indicating whether or not the individual really loved the game or not.

Use this data to build a **decision tree** and then use that tree to help answer the following questions.

- Build the decision tree using **arbitrary** feature ordering (i.e. no calculation of information gain for each feature). What is the total **size** of the resulting decision tree?
- Build the decision tree using **information feature ordering**. What is the total **size** of the resulting tree now? If the size is different, what is responsible for the change?
- Construct the features for two **NEW** games that are NOT in the existing data set for which you think the decision tree will answer "yes". Verify that it does so by calling the `classify()` method of the decision tree.
- Construct the features for two **NEW** games that are NOT in the existing data set for which you think the decision tree will "no". Again, verify it.
- Did printing out and looking at the decision tree help you construct the examples above? If so, how? Briefly explain any reasoning you used.

What to Hand In

Your data together with your written answer in a file called `a10q2.pdf` (or .doc, .txt).

Evaluation

- 1 mark: Tree built and size reported for arbitrary-feature tree
- 1 mark: Size reported for ordered-feature tree, along with meaningful analysis
- 1 mark: 2 unique games are suggested that will yield "yes"
- 1 mark: 2 unique games are suggested that will yield "no"
- 1 mark: Discussion of how to pick examples shows insight.