CS 486/686 Assignment 2 (115 marks)

Alice Gao

Due Date: 11:59 pm ET on Thursday, October 29, 2020

Changes

v1.1

- Changed read_data() to read_data(file_path). Added tree as a parameter to learn_dt since the function is likely recursive.
- We have provided 2 copies of the dataset. One copy is semicolon separated and the other copy is comma separated based on their file names. The unit tests assume that the dataset file is comma separated.

Instructions

- Submit the assignment in the A2 Dropbox on Learn. No late assignment will be accepted. This assignment is to be done individually.
- For any programming question, we will run your code in the CS-Teaching environment (linux.student.cs.uwaterloo.ca). You should make sure that (1) the language you use is available on this environment by default. (2) your code runs in this environment. We highly recommend using Python.
- Lead TAs:
  - Wanxin Li (wanxin.li@uwaterloo.ca)
  - Alister Liao (alister.liao@uwaterloo.ca)

  The TAs' office hours will be posted on MS Teams.
- Submit two files with the following names.

- **writeup.pdf**
  - Include your name, email address and student ID in the writeup file.
  - If you hand-write your solutions, make sure your handwriting is legible and take good quality pictures. You may get a mark of 0 if we cannot read your handwriting.
- **code.zip**
  - Include your program, and a script to run your program.
  - The TA will run your program using the following command in a terminal in the environment linux.student.cs.uwaterloo.ca.

    `bash a2.sh`

    You may get a mark of 0 if we cannot run your program.

Learning goals

Constraint Satisfaction Problem

- Formulate a given problem as a constraint satisfaction problem. Define the variables, the domains, and the constraints.
- Trace the execution of the AC-3 arc consistency algorithm on a given CSP.

Decision Trees

- Determine the prediction accuracy of a decision tree on a data-set.
- Trace the execution of and implement the ID3 algorithm, which constructs a decision tree by selecting a feature for each node using the expected information gain metric.
- Extend the IDS algorithm to handle real-valued features with binary tests at each node.
- Implement cross-validation to prevent over-fitting.

1 Wooden Stick Grid Puzzle (40 marks)

We have a 4 by 4 grid and 16 wooden sticks of 4 different lengths. The lengths are 1, 2, 3, and, 4. There are 4 wooden sticks of each length. Our goal is to place the wooden sticks vertically on the grid while satisfying the following constraints:

- (All Different Constraints) In each row, the lengths of the sticks are all different. In each column, the length of the sticks are all different.
- (Viewing Constraints) There is a number on each side of a row or a column. This number describes the exact number of different sticks that a person can see if the person views the sticks from that side of the row or column. (This is a 3D puzzle.)

  For instance, suppose that the number on the left side of a row is 3 and the number on the right is 2. In this case, [1 3 4 2] is a valid placement. From the left, a person can see the sticks 1, 3, and 4 (2 is blocked by 4). From the right, a person can see the sticks 2 and 4 (3 and 1 are blocked by 4). [1 4 3 2] violates the constraint since the person on the left can see 1 and 4, and the person on the right can see 2, 3, and 4.

Complete the following tasks using Table 1.

	1	2	3	2	
1					4
3					1
3					2
2					2
	2	1	2	3	

Table 1: Wooden Stick Grid Puzzle Example 1

(a) Solve the puzzle in Table 1 by hand. Submit the final answer. In addition, describe two different strategies you used to make progress while solving the puzzle.

Here is an example of a strategy: I need to put 3 and 4 into the squares XXX and XXX. Because of the constraint XXX, it has to be that 3 is in square XXX and 4 is in square XXX.

Marking Scheme:
(5 marks)

- (2 marks) Correct solution.
- (3 marks) Provided 3 different examples of strategies.

(b) Formulate this puzzle as a constraint satisfaction problem. We have provided the variable definitions. below. Describe the domains of the variables and the constraints.

Variables:

A variable represents the sequence of numbers from left to right in a row or the sequence of numbers from top to bottom in a column.

Let R_i denotes the variable for the i-th row from the top. $i \in \{1, 2, 3, 4\}$ R_1 denotes the top row. Let C_j denotes the variable for the j-th column from the left. $j \in \{1, 2, 3, 4\}$. C_1 denotes the leftmost column.

Marking Scheme:

(8 marks)

- (2 marks) Domains
- (6 marks) Constraints

(c) Process all the unary constraints. For each unary constraint on a variable, remove all the values in the variable's domain that violate the unary constraint. Afterwards, write down the updated domains of all the variables.

- $R_1 \in \{$
- $R_2 \in \{$
- $R_3 \in \{$
- $R_4 \in \{$
- $C_1 \in \{$
- $C_2 \in \{$
- $C_3 \in \{$
- $C_4 \in \{$

Marking Scheme:

(8 marks)

- (4 marks) The domains for $R_1 \dots R_4$.
- (4 marks) The domains for $C_1 \dots C_4$.

(d) Execute the AC-3 algorithm for 8 steps as described below.

For each step, we have provided the arc to be removed from the set S. list the values removed from the domain of the variable and whether we need to add any arcs back to S. We have completed step 1 as an example.

After the 8 steps, indicate whether the algorithm can terminate or whether we need to continue executing the algorithm. If the algorithm can terminate, describe the outcome of the algorithm execution.

(1) Remove $\langle R_4, (R_4, C_1)\rangle$ from S.
Remove 2413, 1423, 2143 from the domain of R_4.
Add nothing back to S.

(2) Remove $\langle R_4, (R_4, C_2)\rangle$ from S.

(3) Remove $\langle C_4, (R_1, C_4)\rangle$ from S.

(4) Remove $\langle C_3, (R_4, C_3)\rangle$ from S.

(5) Remove $\langle R_3, (R_3, C_4)\rangle$ from S.

(6) Remove $\langle R_2, (R_2, C_3)\rangle$ from S.

(7) Remove $\langle C_1, (R_2, C_1)\rangle$ from S.

(8) Remove $\langle C_2, (R_2, C_2)\rangle$ from S.

Marking Scheme:
(15 marks)

- (2 marks) for each of the 7 steps.
- (1 mark) for the result after 8 steps.

(e) We tackled the same problem with two methods: (1) solving it by hand, and (2) solving it by executing the AC-3 algorithm. How are these two approaches similar or different? Discuss your observations and thoughts in a few sentences.

Marking Scheme:
(2 marks)

- (2 marks) provide a reasonable discussion.

2 Decision Trees (75 marks)

You will implement the decision tree learner algorithm to predict the quality of a red wine based on results of based on physicochemical tests. The data-set is adapted from the Wine Quality Data Set in the UCI Machine Learning Repository. We are only using the data set on red wine.

The data-set has real-valued features. When generating a decision tree, use only **binary tests** at each node. At each node in the decision tree, choose a feature and a split point for the feature and the node should test whether a feature has a value greater than the split point or not. At each node, choose the feature and the split point for the feature using the expected information gain metric. Along a path from the root node to a leaf node, we may test a real-valued feature multiple times with different split points.

Please complete the following tasks.

(a) Write a program to learn a decision tree (named **tree-full**) on the entire data-set using the decision tree learner algorithm discussed in lectures 6 and 7. For each real-valued feature, use the procedure described in lecture 7 slide 44 to choose a split point for the feature.

We have broken down this problem into several parts for you. This is to help you build your program incrementally and debug the individual helper functions before you test the entire program. The following steps are recommendations only. You do not have to follow them.

If you like, implement the following functions in your program.

(1) Our implementation uses the anytree package in Python to store the decision tree and to plot it.

(2) **read_data(file_path)** reads in the data.

(3) **get_splits(examples, feature)**: Given a set of examples and a feature, get all the possible split point values for the feature.

(4) **calc_entropy(examples)**: Calculate the entropy of the probability distribution represented by the examples. If you are using Python, you may want to use the Counter object to represent the distribution.

(5) **split_examples(examples, feature, split)**: Given a split value for the given feature, split the examples into two sets — one for all the examples where the feature value is below the split value and the other one for all the examples where the feature value is above the split value.

(6) **choose_split(examples, feature)**: Given a set of examples and a feature, return the split value that results in the largest expected information gain.

(7) **choose_feature(example, features)**: Given a set of examples and a set of features, choose a feature and a split value for the feature that maximizes the expected information gain

(8) **learn_dt(tree, examples, features)**: Given a set of examples and a set of features, learn a decision tree to classify the examples. This function is likely recursive.

(9) **predict(tree, example)**: Given a decision tree and an example, return the class label for the example.

(10) **get_prediction_accuracy(tree, data)**: Given a decision tree and a data-set, return the prediction accuracy of the decision tree on the data-set.

Your TA has graciously written unit tests to help you test and debug your program. They are provided in the unittests folder. The unit tests are provided as is. We do not guarantee that they are bug-free, and we do not provide any support for the code. Feel free to use them in any way that you like.

For part a, please complete the following tasks.

(1) Generate **tree-full** and save it in a file named **tree-full.png**.

(2) What is the maximum depth of **tree-full**?
Assume that the depth of the root node is 1.

(3) To evaluate the prediction accuracy of a decision tree on a data-set, we will calculate the fraction of examples that the decision tree classifies correctly.

What is the prediction accuracy of **tree-full** on the entire data-set?

A side note: In practice, we would usually build the decision tree using a training set and determine the prediction accuracy of the decision tree on a separate test set. However, in this question, we are using the same data set to build the decision tree and to calculate the prediction accuracy of the decision tree. Even if we are using the same data set for training and testing, the prediction accuracy of the decision tree is not necessarily 100%. (Think about why this is the case. You don't have to answer this in the write-up.)

Marking Scheme:

(16 marks)

- (2 marks) The TA can run your program to reproduce **tree-full** within 5 minutes. These 2 marks are all or nothing.
- (10 marks) Your decision tree **tree-full** is similar to ours.
- (2 marks) Correct maximum depth of **tree-full**.
- (2 marks) Correct prediction accuracy of **tree-full** on the data-set.

(b) The decision tree **tree-full** might have over-fitted to the data-set and may not generalize well to unseen data. In this part, you will experiment with pruning the tree using the maximum depth criterion to prevent over-fitting.

For a pre-defined value of the maximum depth, learn a decision tree up to the given maximum depth. When a node is at the maximum depth, instead of splitting the examples, make a decision using majority vote. Determine the best value of the maximum depth of the decision tree through five-fold cross-validation.

Five-Fold Cross-Validation:

In five-fold cross-validation, each data point serves double duty — as training data and validation data. Split the data into five subsets. For each maximum depth, perform five rounds of learning. In each round, $\frac{1}{5}$ of the data is held out as a validation set and the remaining data points are used as training set. Generate a decision tree with the maximum depth using only the training data. Then, calculate the prediction accuracy of the decision tree on the training data and on the validation data. (In each round, you will likely generate a different decision tree. However, we do not care about the decision trees generated. We only care about the average prediction accuracy of the trees.) After the five rounds of learning, calculate the average prediction accuracy of the decision tree on the training data and on the validation data, where the average is taken over the five rounds of learning. That is, for each maximum depth, you will produce two numbers, the average prediction accuracy on the training set and the average prediction accuracy on the validation set. Finally, choose the maximum depth with the highest average prediction accuracy on the validation set.

I did not specify the range of values for the maximum depth. Come up with a reasonable range of values yourself. The range should be large enough to include the value of the maximum depth that maximizes the prediction accuracy of the tree on the validation set.

For the implementation, in addition to the functions specified in part a, we also recommend that you implement a function **cross_validation(data, file_path)** to perform five-fold cross-validation on the provided data-set, saves a plot of the training accuracy and validation accuracy in the file at the provided path, and returns the best maximum depth of the decision tree.

To Reduce Randomness for Marking:

To make marking easier, please follow the instructions below to reduce the randomness in your solutions.

- Do not shuffle the data set. (In practice, it is a good idea to shuffle the data set. For this assignment, however, not shuffling will ensure that your results match ours.)
- For cross validation, use these ranges for the five folds: (0, 318), (319, 637), (638, 956), (957, 1275), and (1276, end).
- When there are examples with different labels at a leaf node, we need to determine a decision using majority vote. If there is a tie for majority voting, always return the label with the smallest value.

If your results are different from ours, we may deduct marks at our discretion. However, you will get most marks if your results satisfy the following requirements.

- Your cross-validation curves have reasonable shapes.
- Your answers are consistent with your results.

For part b, please complete the following tasks.

(1) Plot the results of the five-fold cross-validation. Save the plot in a file named **cv-max-depth.png**. There should be two curves on your plot.
   - The average prediction accuracy on the training set with respect to the maximum depth.
   - The average prediction accuracy on the validation set with respect to the maximum depth.

(2) Based on the results of the cross validation, what is the best value of the maximum depth of the tree?

(3) Using the best value for the maximum depth, generate a decision tree (denoted by **tree-max-depth**) using the entire data-set. Save the tree in a file named **tree-max-depth.png**.

(4) What is the prediction accuracy of the tree **tree-max-depth** on the entire data-set?

Marking Scheme:

(28 marks)

- (4 marks) The TA can run your program to reproduce the cross validation results in **cv-max-depth.png** (2 marks), and the tree in **tree-max-depth.png** (2 marks) within 5 minutes. These marks are all or nothing.
- (10 marks) Your cross validation plot **cv-max-depth** is similar to ours.
- (2 marks) The correct value of the best maximum depth of the decision tree based on cross validation.
- (10 marks) Your tree **tree-max-depth** is similar to ours.
- (2 marks) Correct prediction accuracy of **tree-max-depth** on the data-set.

(c) In this part, you will experiment with post-pruning the tree by using the minimum information gain criterion.

For post-pruning, grow a full tree first. Define a value for the minimum information gain. Next, keep track of all the nodes that only has leaf nodes as its descendants. For each node that only has leaf nodes as its descendants, if the expected information gain

at the node is less than the pre-defined value, than delete the children of this node and convert this node to a leaf node with majority decision.

Modify your decision tree learner algorithm to determine the best value for the minimum information gain for the post-pruned decision tree through five-fold cross-validation.

For part c, please complete the following tasks.

(1) Plot the results of the five-fold cross-validation. Save the plot in a file named **cv-min-info-gain.png**. There should be two curves on your plot.
   - The average prediction accuracy on the training set with respect to the minimum information gain.
   - The average prediction accuracy on the validation set with respect to the minimum information gain.

(2) Based on the results of the cross-validation, what is best value of the minimum information gain pruning criterion?

(3) Using the best value for the minimum expected information gain, generate a decision tree (denoted by **tree-min-info-gain**) using the entire data-set. Save **tree-min-info-gain** in a file named **tree-min-info-gain.png**.

(4) What is the prediction accuracy of the decision tree **tree-min-info-gain** on the entire data-set?

Marking Scheme:

(28 marks)

- (4 marks) The TA can run your program to reproduce the cross validation result in **cv-min-info-gain.png (2 marks)**, and the tree in **tree-min-info-gain.png** (2 marks) within 5 minutes. These marks are all or nothing.
- (10 marks) Your cross validation plot **cv-min-info-gain** is similar to ours.
- (2 marks) The correct best value for the minimum expected information gain at a node based on cross validation.
- (10 marks) Your tree **tree-min-info-gain** is similar to ours.
- (2 marks) Correct prediction accuracy of the decision tree **tree-min-info-gain** on the data-set.

(d) Given your results, how would you generate a decision tree for the provided data-set? Explain your strategies in a few sentences.

Marking Scheme:

(3 marks)

A reasonable answer and explanation