CS 486/686 Assignment 2 (126 marks in total)

Instructor: Alice Gao

Due Date: 6:00 pm on Tuesday, June 25 2019

Instructions

- Submit the assignment in the Dropbox labeled Assignment 2 Submissions in the Assignment 2 folder on LEARN. No late assignment will be accepted. This assignment is to be done individually.
- For any programming question, you may use the language of your choice. We highly recommend using Python. If you don't know Python, it may be worthwhile to learn it for this course.
- Lead TAs:
  - Wei Sun (w55sun@uwaterloo.ca)
  - Ji Xin (j9xin@uwaterloo.ca)
  - Ronghao (Steve) Yang (r39yang@uwaterloo.ca)

  The TAs' office hours will be posted on the course website.
- Submit two files with the following naming conventions. **We will deduct 10 marks if you do not follow these conventions.**
  - **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, a script to run your program, and a README.txt file with instructions to run the script. You may get a mark of 0 if we cannot run your program.

Learning goals

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.

Artificial Neural Networks

- Trace the execution of the back-propagation algorithm for learning a simple function.

1 Decision Trees (110 marks)

Your TA Ji Xin has implemented the decision tree. He used python and spent roughly 4.5 hours including generating the tree and plotting the tree. Take 4.5 and multiply it by 2 or 3, that would be roughly the number of hours you will spend working on this problem. Please start working on this problem as early as possible. Feel free to reach out to Ji if you have implementation specific questions. He will also have office hours on Saturday afternoons.

You will implement a decision tree learning algorithm to detect whether a room is occupied based on the temperature, humidity, light and carbon dioxide in the room. The data set is adapted from the Occupancy Detection Data Set in the UCI Machine Learning Repository.

You are provided with two data sets **occupancy_A** and **occupancy_B** on the course website. Data set A has five features: Temperature, Relative Humidity, Light, CO2, and Humidity Ratio. Data set B is identical to data set A except that the feature Light is removed.

Please complete the following tasks.

A. Write a program to learn a decision tree (denoted by T1) using the ID3 algorithm on the data set A. For each node in the tree, consider every feature and consider all possible split points for each feature using the rules described in class. Choose a feature and a split point for the feature using the expected information gain metric.

   The rules for choosing split points will be described in class on Monday, June 10. If you want to get started on this during this weekend, you can implement the following to start with.

   - Consider a real-valued feature. Sort the data set based on the values of the feature.
   - All the possible split points are the values that are midway between two consecutive but different values 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 helpers functions before you test the entire program. You do not have to follow the steps below exactly. **However, if a step requires you to implement a function, make sure your function uses the provided name.** If your program does not run or does not produce the correct solution, you may receive part marks for implementing the individual helper functions correctly.

   If you like, follow the recommended steps below to implement your program.

   (a) Implement a function **read_data(id)** which reads in the data set. The id parameter can be used to specify which dataset the function is trying to read in.

(b) Implement a function **entropy(dist)** which returns the entropy of the given probability distribution. If you are using Python, you may want to use the Counter object to represent the distribution.

(c) Implement a function **choose_feature_split(data)** which iterates through all the features and all possible split points of each feature and returns one feature and one split point for the feature that maximizes the expected information gain.

(d) Implement a few functions to operate on a node in the decision tree. In our implementation, we wrote a node class and implemented the following functions inside the class.

- **init_node(node)** stores useful initial information in the node.
- **split_node(node, data, max_depth)** produces the node based on the data points left and recursively produces the children of the current node if necessary.
- **get_decision(node, one_data_point)** returns the decision/label given a data point. If the given node is a leaf node, this function returns the decision in the leaf node. Otherwise, this function recursively calls itself on the child nodes until it reaches a decision.

(e) Implement a few functions to operate on a decision tree. In our implementation, we wrote a tree class and implemented the following functions inside the class. The tree class is essentially a wrapper of the node class with additional information stored in it.

- **init_tree(tree)** initializes the decision tree by creating the root node.
- **train_tree(tree, data, max_depth)** trains the decision tree using the provided data and up to the maximum depth if specified. In our implementation, this function calls the **split_node** function to produce the root node of the decision tree.
- **get_prediction_accuracy(tree, data)** returns the prediction accuracy of the decision tree on the provided data. In our implementation, this function calls the **get_decision** function on the root node of the tree.
- **plot_tree(tree, file_path)** produces a plot of the tree and saves the plot in the file at the provided path. If you are using Python, we recommend using the anytree python package to plot the tree.

What to submit:

In your writeup.pdf, answer the following questions.

- What is the maximum depth of the decision tree generated? The depth of the root node is 1.
- What is the prediction accuracy of the decision tree on data set A?
- Based on the tree you generated, what is the most informative feature in this data set? Why? Discuss your thoughts in no more than 1 paragraph.

In your code.zip, save the decision tree T1 in a file named **t1.pdf**.

In your code.zip, submit your program for generating the decision tree T1. If it is complicated to run your program, be sure to write a script to run your program and to generate the decision tree. In your README file, provide clear instructions for running your program. If you want the TA to run your program in a particular environment and use a particular version of a programming language, be sure to provide enough information on these. The TA will not modify your source code to run your program.

Marking Scheme: (30 marks)

- (2 marks) Maximum depth of the tree
- (2 marks) Prediction accuracy of the tree on data set A
- (6 marks) Discussion of the most informative feature
- (10 marks) The TA can run your program to produce the decision tree T1 in under 5 minutes and the produced tree is the same as the tree in **t1.pdf** you provided.
- (10 marks) Your decision tree T1 is similar to our decision tree T1 except for any differences caused by noise in the data.

B. The decision tree T1 should have achieved pretty good accuracy on data set A. However, this does not mean that it will generalize well to unseen data. In this part, you will experiment with generating a smaller decision tree using cross-validation.

Implement a modified version of the ID3 algorithm which takes as input a maximum depth. You will use this modified ID3 algorithm and five-fold cross-validation to determine the best maximum depth of the decision tree.

In five-fold cross-validation, each data point serves double duty — as training data and validation data. First, randomly shuffle all the data points. Next, split the data into five equal 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. (Note that, in each round, you will likely generate a different decision tree. However, we do not care about all the different 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.

Note that I did not specify the range of values for the maximum depth. Experiment with different ranges and come up with one that is reasonable. You will want the range to be large enough so that you can find the maximum depth that maximizes the prediction accuracy on the validation set.

Having determined the best maximum depth for our decision tree, you will generate a decision tree (denoted by T2) with the best maximum depth using the entire data set A.

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.

What to submit:

In your writeup.pdf, include the following items.

- Plot the results of the five-fold cross-validation. That is, plot the average prediction accuracy on the training set (on the y-axis) and on the validation set (on the y-axis) with respect to the maximum depth of the decision tree (on the x-axis). Please put both plots on the same graph.
- What is the maximum depth of the decision tree that maximizes the average prediction accuracy on the validation set?
- What is the prediction accuracy of the decision tree **T2** on data set A?
- Compare the decision trees T1 (from part A) and T2 (from this part). What are their similarities and differences? Which decision tree would you prefer and why? Discuss your observations in no more than 2 paragraphs.

In your code.zip, save the decision tree T2 in a file named **t2.pdf**.

In your code.zip, include the program you used to determine the best maximum depth of the decision tree using five-fold cross validation and to generate the decision tree with the best maximum depth. Provide clear instructions for running your program. See more details in the last paragraph of part A. In particular, provide clear instructions for the TA to run your program and to produce

- the plot of the results of five-fold cross-validation and
- the decision tree T2.

Marking Scheme: (50 marks)

- (10 marks) The TA can run your program to reproduce the plot of the results of the five-fold cross-validation in under 5 minutes and the produced plot is the same as the one provided in your writeup.pdf.
- (10 marks) Plot of the results of the five-fold cross-validation.

  5 marks for the plot of the average prediction accuracy on the training set with respect to the maximum depth.

5 marks for the plot of the average prediction accuracy on the validation set with respect to the maximum depth.

- (2 marks) Maximum depth of the decision tree
- (2 marks) Prediction accuracy of the decision tree T2 on data set A
- (6 marks) Discussion of a comparison between T1 and T2.
- (10 marks) The TA can run your program to reproduce the decision tree T2 in under 5 minutes and the produced tree is the same as the tree in **t2.pdf** you provided.
- (10 marks) Your decision tree T2 is similar to our decision tree T2. These 10 marks will be awarded based on the TA's discretion.

C. We've constructed data set B, which is identical to data set A except that the feature Light is removed. Repeat part B for data set B. First, figure out the best maximum depth of the tree using five-fold cross-validation. Next, train a decision tree (denoted T3) with the best maximum depth on the entire data set B.

What to submit:

In your writeup.pdf, include the following items.

- Plot the results of the five-fold cross-validation. That is, plot the average prediction accuracy on the training set (on the y-axis) and on the validation set (on the y-axis) with respect to the maximum depth of the decision tree (on the x-axis). Please put both plots on the same graph.
- What is the maximum depth of the decision tree that maximizes the average prediction accuracy on the validation set?
- What is the prediction accuracy of the decision tree with the best maximum depth on data set B?
- What are the similarities and differences between the decision trees T2 (from part B) and T3 (from this part)? How did removing the feature Light impact the shape and the size of the decision tree? Discuss your observations in no more than 2 paragraphs.

In your code.zip, save the decision tree T3 in a file named **t3.pdf**.

The TA will not be running your program for this part. If your program for part B runs, it likely runs for this part as well. =)

Marking Scheme: (30 marks)

- (10 marks) Plot of the results of the five-fold cross-validation.

  5 marks for the plot of the average prediction accuracy on the training set with respect to the maximum depth.

  5 marks for the plot of the average prediction accuracy on the validation set with respect to the maximum depth.

- (2 marks) Maximum depth of the decision tree
- (2 marks) Prediction accuracy of the decision tree T3 on data set B
- (6 marks) Discussion of a comparison between T2 and T3.
- (10 marks) Your decision tree T3 is similar to our decision tree T3. These 10 marks will be awarded based on the TA's discretion.

2 Neural Networks (16 marks)

Consider the function f defined by the following table.

x_1	x_2	y
1	1	1
2	2	0
3	2	1
2	3	1

Table 1: A simple function defined by four points

Answer the following questions.

1. Consider a perceptron with two inputs x_1 and x_2, a bias input $x_0 = 1$, three weights w_0, w_1 and w_2, and an output o. Let $o = 1$ when $\sum_{i=0}^{2}(w_i x_i) \geq 0$ and $o = 0$ when $\sum_{i=0}^{2}(w_i x_i) < 0$. Can the function f be represented by this perceptron? Why or why not? Justify your answer in no more than 1 paragraph.

   Marking Scheme: (6 marks)

   - (2 marks) Correct answer.
   - (4 marks) Correct justification.

2. We can learn this function f in Table 2 using a multi-layer feed-forward neural network shown in Figure 2 below. The activation function is the sigmoid function $y = \dfrac{1}{1 + e^{-x}}$. To determine the weights, we will use the back-propagation algorithm discussed in class.

   For this part, you will **execute one iteration of the back-propagation algorithm**. One iteration consists of performing the forward propagation and updating all the weights by summing the gradients for all four points in the data set.

   Use the learning rate $\alpha = 1$ and start with the following initial weights.

   - $w1_{01} = 0.9$, $w1_{02} = 0.4$, $w1_{11} = 0.7$, $w1_{12} = 0.3$, $w1_{21} = 0.6$, and $w1_{22} = 0.8$
   - $w2_{01} = 0.8$, $w2_{11} = 0.7$, $w2_{21} = 0.7$.

   Be sure to include the following values in your answer.

   - The values of the hidden units h_1 and h_2

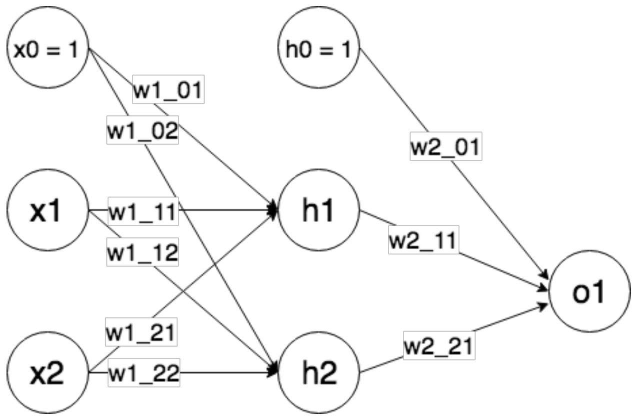

Figure 1: A neural network with one hidden layer

- The values of the output unit o_1
- The updated values of $w1$ and $w2$

Marking Scheme: (10 marks)

- (4 marks) values of h_1 and h_2
- (2 marks) values of o_1
- (4 marks) updated values of w_1 and w_2