Homework 2

Automated Learning and Data Analysis
Dr. Thomas Price

Spring 2020

Instructions

Due Date: February, 18 2020 at 11:45 PM
Total Points: 100 for CSC522; 85 for CSC422.
Submission checklist:

- Clearly list each team member's names and Unity IDs at the top of your submission.
- Your submission should be a single zip file containing a PDF of your answers, your code, and (if needed) a README file with running instructions. **Name your file**: G(homework group number)_HW(homework number), e.g. G1_HW2.
- If a question asks you to explain or justify your answer, **give a brief explanation** using your own ideas, not a reference to the textbook or an online source.
- In addition to your group submission, please also *individually* submit your Peer Evaluation form on Moodle, evaluating yours and your teammates' contributions to this homework.

Problems

1. Decision Tree Construction (17 points) [**Ge Gao**]. Create decision trees for the `hw2q1.csv` dataset by hand, as explained below, using Hunt's algorithm. Note the following:

   - In the given dataset, all of the input attributes are binary except for the first attribute, V4, which is ratio and continuous.
   - The output label has two class values: T or F.
   - In the case of ties then selecting an attribute, break ties in favor of the leftmost attribute.
   - When considering a split for the continuous attribute, identify the best value to split on (e.g. $\leq$ 15 and $>$ 15) by testing all possible split values.

   When calculating Information Gain or Gini Index, write down each step of the computation process and show your work step by step. Draw a separate tree after each attribute split, like the example in Figure 1. You can use a program (e.g. tikz with LaTeX) to draw your trees, or draw them by hand on paper and scan your results into the final pdf.

   (a) Construct the decision tree manually, using Information Gain (IG) to select the best attribute to split on, as in the ID3 algorithm. The maximum depth of your tree should be 4 (count the root node as depth 0), meaning that any node at depth 4 will automatically be a leaf node, even if it has objects with different classes.

   (b) Construct the tree manually using the Gini index. The maximum depth of the tree should be 2.

   (c) How are the trees different? As part of your explanation, give 2 examples of data objects that would be classified differently by the two trees.

   (d) Which decision tree will perform better on the training dataset (`hw2q1.csv`)? Which will perform better on a test dataset? Can we know the answer?

2. Evaluation Measures & Decision Tree Pruning (14 points) [**Ge Gao**] Given the decision tree in Figure 1, complete the following tasks:

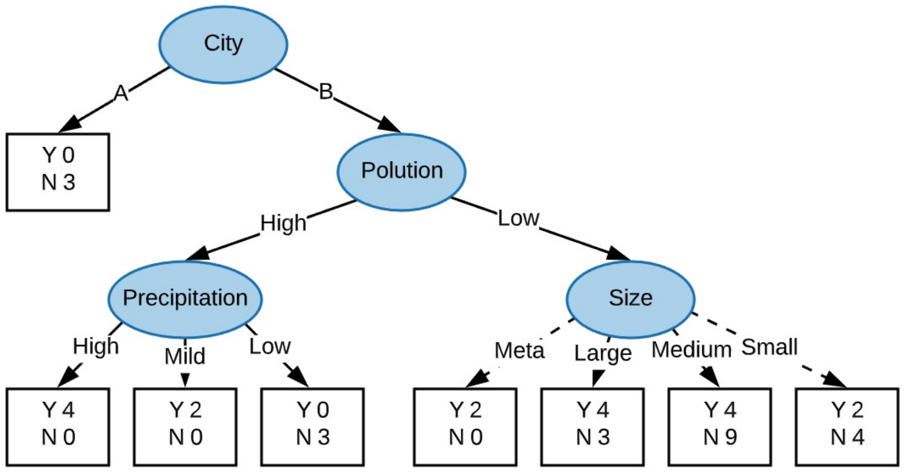

Figure 1: Decision Tree

(a) Use the decision tree above to classify the provided dataset. `hw2q2.csv`. Construct a confusion matrix and report the test Accuracy, Error Rate, Precision, Recall, and F1 score. Use "Yes" as the positive class in the confusion matrix.

(b) Calculate the optimistic training classification error before splitting and after splitting using Size, respectively. **Consider only the subtree starting with the Size node.** If we want to minimize the optimistic error rate, should the node's children be pruned?

(c) Calculate the pessimistic training errors before splitting and after splitting using Size respectively. Consider only the subtree starting with the Size node. When calculating pessimistic error, use a leaf node error penalty of 0.8. If we want to minimize the pessimistic error rate, should the node's children be pruned?

(d) Assuming that the "Size" node is pruned, recalculate the test Error Rate using `hw2q2.csv`. Based on your evaluation using the test dataset in `hw2q2.csv`, was the original tree (with the Size node) over-fitting? Why or why not?

3. 1-NN, Evaluation, Cross Validation (12 points) [**Ge Gao**] Consider the following dataset (9 instances) with 2 continuous attributes (x_1 and x_2) and a class attribute y, shown in Table 1. For this question, we will consider a 1-Nearest-Neighbor (1-NN) classifier that uses euclidean distance.

Table 1: 1-NN

ID	x_1	x_1	**y**
1	10.0	23.0	-
2	15.0	18.5	-
3	9.0	24.5	+
4	16.5	30.0	+
5	31.0	12.0	-
6	27.5	25.0	+
7	20.0	8.0	-
8	12.0	43.0	+
9	21.0	22.5	-

(a) Calculate the distance matrix for the dataset using euclidean distance. **Tip**: You can write a simple program to do this for you, and there is an example of how to do this in the R programming portion of this homework.

(b) By hand, evaluate the 1-NN classifier, calculating the confusion matrix and testing accuracy (show your work by labeling each data object with the predicted class). **Tip**: you can scan a row or column of the distance matrix to easily find the closest neighbor. Use the following evaluation methods:

i. A holdout test dataset consisting of last 4 instances

ii. 3-fold cross-validation, using the following folds with IDs: [3,6,9], [1,4,7], [2,5,8] respectively.

iii. Leave one out cross validation (LOOCV)

(c) For a data analysis homework, you are asked to perform an experiment with a binary classification algorithm. You are given a dataset with 30 instances and a class attribute that can be either Positive or Negative. The dataset includes 15 positive and 15 negative instances. You use three different validation methods: holdout (with a random 20/10 training/test split), 5-fold cross validation (with random folds) and LOOCV. As a baseline, you compare your algorithm to a "simple majority classifier," which always predicts the majority class in the training dataset (if there is no majority, one of the classes is chosen at random). You expect the simple majority classifier to achieve approximately 50% testing accuracy, but for one of these evaluation methods you get 0% testing accuracy. Which evaluation gives this results and why?

4. BN Inference (12 points) [**Ge Gao**]. The following dataset represents clothes with four categorical attributes: Size, Color, Price, Material, and a Boolean output Class: Buy.

Size	**Color**	**Material**	**Buy?**
large	red	cotton	No
large	white	cotton	No
large	white	fiber	No
large	red	cotton	No
small	black	cotton	No
large	black	cotton	Yes
large	red	cotton	Yes
small	black	fiber	Yes
large	black	fiber	Yes
small	red	fiber	Yes

For the following problem, you may find it useful to fill in the following table (optional).

$P(Buy = Yes) =$	$P(Buy = No) =$
$P(Size = small \mid Buy = Yes) =$	$P(Size = small \mid Buy = No) =$
$P(Size = large \mid Buy = Yes) =$	$P(Size = large \mid Buy = No) =$
$P(Color = white \mid Buy = Yes) =$	$P(Color = white \mid Buy = No) =$
$P(Color = red \mid Buy = Yes) =$	$P(Color = red \mid Buy = No) =$
$P(Color = black \mid Buy = Yes) =$	$P(Color = black \mid Buy = No) =$
$P(Material = cotton \mid Buy = Yes) =$	$P(Material = cotton \mid Buy = No) =$
$P(Material = fiber \mid Buy = Yes) =$	$P(Material = fiber \mid Buy = No) =$

Using the training dataset above, how would a Naive Bayes classifier classify the following data points? Show your work.

(a) {*Size = small, Color = black, Material = cotton*}

(b) {*Size = small, Color = red, Material = fiber*}

(c) {*Size = large, Color = red, Material = cotton*}

(d) {*Size = large, Color = white, Material = fiber*}

5. R Programming Part 1: KNN and Decision Tree Classifiers (45 points) [**Yang Shi**]

R Programming Submission Instructions

Instructions:

- Make sure you clearly list each team member's names and Unity IDs at the top of your submission.
- Edit the *hw2.R* file included in the homework, and **do not change** this file name. Add this file, along with a README to the zip file mentioned in the first page.
- Carefully read the documentation for each function you need to implement, including expected input and output data types.
- To **test you code**, step through the `hw2_checker.R` file. If you update you code, make sure to run `source('./hw2.R')` again to update your function definitions. You can also check the "Source on save" option in R Studio to do this automatically on save.

Tips:

- You are free to write your own functions to handle sub-tasks, but the TA will only call the functions requested.
- Your code should be clearly documented, so we can give you partial credit if the initial output is incorrect.
- The TA will have an autograder, which will first run `source(hw2.R)`, then call each of the functions requested in the homework and compare with the correct solution.
- The TA will test your solution with a *different* dataset than what you are provided, so do not hard-code any answers.

Warnings:

- Failure to follow naming conventions or programming related instructions specified here may result in your submission not being graded.
- **DO NOT** set working directory (`setwd` function) or clear memory (`rm(list=ls(all=T))`) in your code. TA(s) will do so in their own auto grader.
- **DO NOT** include `install.packages()` in your `hw2.R`. This will result in a penalty of 5 points.

You are given the following files:

- `pima-indians-diabetes.csv`: CSV file with 100 rows, 9 columns. The first 8 columns refer to features, final column (called 'Class') refers to the Class variable. You have 2 classes: {0, 1}.
- Data is from Johannes et al. [1], the 8 features are number of times pregnant, glucose concentration, blood pressure, skinfold thickness, serum insulin, body mass index, diabetes pedigree function, age. The label indicates whether the patient was diagnosed with diabetes (1) or not (0).

Please note that this exercise has sections where you need to implement methods, and sections where you can use a library. Whether you need to implement/use library is clearly stated. **Note**: Part A and E are only required for CSC 522 (bonus for CSC422). CSC422 can ignore any steps using the k-nn classifier.

(a) **Part A (522 Required / 422 Bonus): KNN Classification: (Implement)** You are tasked with implementing the KNN algorithm using the dataset provided to you. You will be implementing the KNN classifier in `hw2.R` using the `knn` function. The input and output variables, and their formats are explained in detail `knn` function documentation – read this carefully before starting.

Hint: The `sort()` function may be useful for finding the k neighbors with the least distance to a given test instance. Use `?sort` for documentation.

(b) **Part B: Decision Tree (Library):**

Implement a decision tree classifier using the `rpart` library. Use information gain (IG) to choose the best attribute. You should train a model using the provided training data and then predict the classes of the provided test data. Code for this question has to be written in the function `dtree`. Please note that by default, `rpart` will automatically tune its model parameters (e.g. `cp`)

– we assume you will leave these settings (specified with the `control` argumet) with their default values. The input, output formats, as well as allowed packages are described as comments under the function.

(c) **Part C: Cross Validation (Implementation):** Write a function for k-fold cross-validation. It should split the whole dataset into k folds, then predict the class values for each fold by training on the other $k-1$ folds. It should return a vector of predicted class values, one for each row in original the dataset. You will do this in 2 steps. First you will write a `generate_k_folds` function that randomly divides the dataset into k folds. Then you will write a `k_fold_cross_validation_prediction` function that uses these folds to make predictions. For each fold i, the classifier should be trained on all but the ith folds and used to predict class values for the i-th fold. Explanations of each function are in their respective documentation.

(d) **Part D: Evaluation (Library + implementation):** Write functions `calculate_confusion_matrix`, which that takes in the predicted class labels of type vector (factor) and ground truth labels of type vector (factor), and outputs a confusion matrix. Then write `calculate_accuracy`, `calculate_recall`, `calculate_precision`, which take in that matrix and output accuracy, precision and recall, respectively. The input, output formats, as well as allowed packages are described as comments under the function.

(e) **Part E (522 Required / 422 Bonus): Analysis:** Report your answers for PART E in your pdf. Compare the performance of the KNN Classifer using euclidean distance, KNN Classifier using cosine similarity and Decision Tree classifiers. At the minimum, we expect to see the following comparisons. You are welcome to perform a more in-depth analysis in addition to these analyses.

- Comparison in terms of overall accuracy - which classifier performed best?
- Comparison in terms of precision and recall - What does the difference mean?
- Considering the background and target of this dataset (classifying whether a patient has diabetes), which evaluation metrics are more important? Explain your choice.
- Based on your findings, are these classifiers effective on this dataset?

References

[1] R. S. Johannes, "Using the g algorithm to forecast the onset of diabetes mellitus," *Johns Hopkins APL Technical Digest*, vol. 10, pp. 262–266, 1988.