CS 5350/6350, DS 4350: Machine Learning Spring 2023

Homework 1

Handed out: January 18, 2023
Due date: February 1, 2023

General Instructions

- You are welcome to talk to other members of the class about the homework. I am more concerned that you understand the underlying concepts. However, you should write down your own solution. Please keep the class collaboration policy in mind.
- Feel free discuss the homework with the instructor or the TAs.
- Your written solutions should be brief and clear. You need to show your work, not just the final answer, but you do *not* need to write it in gory detail. Your assignment should be **no more than 10 pages**. Every extra page will cost a point.
- Handwritten solutions will not be accepted.
- The homework is due by midnight of the due date. Please submit the homework on Canvas. You should upload two files: a report with answers to the questions below, and a compressed file (`.zip` or `.tar.gz`) containing your code.
- Some questions are marked **For 6350 students**. Students who are registered for CS 6350 should do these questions. Of course, if you are registered for CS 5350 or DS 4350, you are welcome to do the question too, but you will not get any credit for it.

Important Do not just put down an answer. We want an explanation. No points will be given for just the final answer without an explanation.

1 Decision Trees

1. In this problem, we will work with a modified version of the badges data we saw in the class. We are given names of the ML researchers along with features, and the task is to assign a positive or a negative label to their badge. There are four features:

   (a) is the length of the first name > 5 (lenFirst): Yes or No

   (b) is the length of the last name > 5 (lenLast): Yes or No

   (c) is the first letter of the first and last name same (sameFirst): Yes or No

   (d) the most frequent vowel that is not 'u' (vowel): a, e, i, o

   Training data is shown in the table 1. You can use the feature names in the parentheses as shorthand for your calculations.

Name	lenFirst	lenLast	sameFirst	vowel	Badge
Pieter Bartlett	Yes	Yes	No	e	+ve
George Berg	Yes	No	No	e	−ve
Hiroshi Motoda	Yes	Yes	No	o	+ve
Filippo Neri	Yes	No	No	i	−ve
Robert Roos	Yes	No	Yes	o	+ve
Satinder Singh	Yes	No	Yes	i	+ve
Maja Mataric	No	Yes	Yes	a	+ve
Arun Sharma	No	Yes	No	a	−ve
Michael Meystel	Yes	Yes	Yes	e	−ve

Table 1: Training data for the badges problem

(a) [5 points] How many possible functions are there to map these four features to a boolean decision? How many functions are consistent with the given training dataset?

(b) [3 points] What is the entropy of the labels in this data? Remember that when calculating entropy, the base of the logarithm should be 2.

(c) [4 points] What is the information gain of each of the features?

(d) [2 points] Which attribute will you use to construct the root of the tree using the ID3 algorithm?

(e) [8 points] Using the root that you selected in the previous question, construct a decision tree that represents the data. You do not have to use the ID3 algorithm here, you can show any tree with the chosen root.

(f) [3 points] Suppose you are given three more examples, listed in table 2. Use your decision tree to predict the label for each example. Also report the accuracy of the classifier that you have learned.

Name	lenFirst	lenLast	sameFirst	vowel	Badge
Clare Congdon	No	Yes	Yes	o	−ve
Fernando Pereira	Yes	Yes	No	e	+ve
Richard Caruana	Yes	Yes	No	a	−ve

Table 2: Test data for the badges problem

2. [10 points] Recall that in the ID3 algorithm, we want to identify the best attribute that splits the examples that are relatively pure in one label. Aside from entropy (also called *Shannon Entropy*), which we saw in class and you used in the previous question, we will now use a different measure for defining the entropy called *Rényi entropy*.

$$H_\alpha(S) = \frac{1}{1-\alpha} \log \left(\sum_i p_i^\alpha \right)$$

Table 3:

Feature	Information Gain (using Collision entropy)
lenFirst	
lenLast	
sameFirst	
vowel	

Note that in the limit $\alpha \to 1$, we get the Shannon entropy. We will use the Rényi entropy with $\alpha = 2$, also called the Collision entropy, given by the following equation:

$$H_2(S) = -\log\left(\sum_{i \in labels} p_i^2\right)$$

where, p_i is the fraction of examples in the set that are labeled with the i-th label.

Just like we used entropy to define information gain, we can define a new version of information gain that uses *Collision entropy* in place of the standard (Shannon) entropy.

(a) [4 points] Write down an expression that defines a new version of information gain that uses *Collision entropy* in place of entropy.

(b) [4 points] Calculate the value of your newly defined information gain from the previous question for the four features in the badges dataset from 1. Enter the information gain into a table of the form shown as Table 3.

(c) [2 points] According to your results in the last question, which attribute should be the root for the decision tree? Do these two measures (entropy and *Collision entropy*) lead to the same root feature?

2 Experiments

For this question, we will use the famous Car Evaluation Dataset from the UCI machine learning repository.[1] It involves determining if a car with given attributes (like the buying price, etc.) is acceptable or not, with four labels: *unacceptable*, *acceptable*, *good*, *very good*. Each instance has 6 features indicating different characteristics of a car. You can find definitions of each feature in `information.txt`, and `feats_description.txt` files. Your task is to implement the ID3 algorithm and build a decision tree using the training data `data/train.csv`. We also provide the test set `data/test.csv` that you will use to evaluate your decision tree classifiers.

You may use any programming language for your implementation. However, the graders should be able to execute your code on the CADE machines without having to install any libraries. A reminder that you are not allowed to use any machine learning libraries (ex:

[1] `https://archive.ics.uci.edu/ml/datasets/car+evaluation`

scikit-learn, tensorflow, pytorch, etc.), but can use libraries with mathematical functions like numpy.

Cross-Validation

The depth of the tree is a hyper-parameter to the decision tree algorithm that helps reduce overfitting. By depth, we refer to the maximum path length from the root to any leaf. That is, a tree with just a single node has depth 0, a tree with a root attribute directly leading to labels in one step has depth 1 and so on. You will see later in the semester that many machine learning algorithm (SVM, logistic-regression, etc.) require choosing hyper-parameters before training commences, and this choice can make a big difference in the performance of the learners. One way to determine a good hyper-parameter values to use a technique called *cross-validation*.

As usual, we have a training set and a test set. Our goal is to discover good hyperparameters using the training set only. Suppose we have a hyperparameter (e.g. the depth of a decision tree) and we wish to ascertain whether it is a good choice or not. To do so, we can set aside some of the training data into a subset called the *validation* set and train on the rest of the training data. When training is finished, we can test the resulting classifier on the validation data. This allows us to get an idea of how well the particular choice of hyper-parameters does.

However, since we did not train on the whole dataset, we may have introduced a statistical bias in the classifier caused by the choice of the validation set. To correct for this, we will need to repeat this process multiple times for different choices of the validation set. That is, we need train many classifiers with different subsets of the training data removed and average out the accuracy across these trials.

For problems with small data sets, a popular method is the leave-one-out approach. For each example, a classifier is trained on the rest of the data and the chosen example is then evaluated. The performance of the classifier is the average accuracy on all the examples. The downside to this method is for a data set with n examples we must train n different classifiers. Of course, this is not practical in general, so we will hold out subsets of the data many times instead.

Specifically, for this problem, you should implement k-fold cross validation.

The general approach for k-fold cross validation is the following: Suppose we want to evaluate how good a particular hyper-parameter is. We randomly split the training data into k equal sized parts. Now, we will train the model on all but one part with the chosen hyper-parameter and evaluate the trained model on the remaining part. We should repeat this k times, choosing a different part for evaluation each time. This will give we k values of accuracy. Their average cross-validation accuracy gives we an idea of how good this choice of the hyper-parameter is. To find the best value of the hyper-parameter, we will need to repeat this procedure for different choices of the hyper-parameter. Once we find the best value of the hyper-parameter, we can use the value to retrain we classifier using the entire training set.

For this problem, your should use the data in `data` folder. This folder contains two files: `train.csv` and `test.csv`. You should train your algorithm on the training file. Remember that you should not look at or use your testing file until your training is complete.

1. [6 points] **Baseline**

   First, find the most common label in the training data. What is the training and test accuracy of a classifier that always predicts this label?

2. **Implementation: Full trees**

   In the first set of decision tree experiments, run the ID3 algorithm we saw in class without any depth restrictions. (That is, there are no hyperparameters for this setting.)

   (a) [6 points] Implement the decision tree data structure and the ID3 algorithm for your decision tree (Remember that the decision tree need not be a binary tree!). For debugging your implementation, you can use the previous toy examples like the apartment data from Table 1. Discuss what approaches and design choices you had to make for your implementation and what data structures you used. Also explain the organization of your code.

   (b) Report the accuracy of your decision tree on all the examples in `data/train.csv`.

   (c) Report the accuracy of your decision tree on the examples in `data/test.csv`.

   (d) Report the maximum depth of your decision tree.

   Your report should include the following information

   (a) [2 points] The root feature that is selected by your algorithm

   (b) [2 point] Information gain for the root feature

   (c) [2 points] Maximum depth of the tree that your implementation gives

   (d) [3 points] Accuracy on the training set

   (e) [5 points] Accuracy on the test set

3. **Limiting Depth**

   Next, you will perform 5-fold cross-validation to limit the depth of your decision tree, effectively pruning the tree to avoid overfitting. We have already randomly split the training data into five splits. You should use the 5 cross-validation files for this section, titled `data/CVfolds/foldX.csv` where `X` is a number between 1 and 5 (inclusive).

   (a) [15 points] Run 5-fold cross-validation using the specified files. Experiment with depths in the set 1, 2, 3, 4, 5, reporting the average cross-validation accuracy and standard deviation for each depth. Explicitly specify which depth should be chosen as the best, and explain why.

   (b) [4 points] Explain briefly how you implemented the depth limit functionality in your code.

   (c) [15 points] Using the depth with the greatest cross-validation accuracy from your experiments: train your decision tree on the `data/train.csv` file. Report the accuracy of your decision tree on the `data/test.csv` file.

(d) [5 points] Discuss the performance of the depth limited tree as compared to the full decision tree. Do you think limiting depth is a good idea? Why?

4. [**For CS 6350 Students**, 20 points] For this part, we will work with a modified version of the car evaluation dataset. In this dataset, some of the feature values are missing and are marked with a question mark '`?`'. The training files for this can be found at `data_missing/train.csv`, and `data_missing/test.csv`.

   Your task is to first impute the missing feature values, and then train the full decision tree on that data. You should impute the missing feature using the following two criteria:

   (a) Replacing it with the most common feature value in the whole dataset

   (b) Replacing it with the most common feature value for the same label

   - [6 points] Describe briefly your implementation.
   - [14 points] Report the same details as in part 2 for these settings.

Experiment Submission Guidelines

1. The report should detail your experiments. For each step, explain in no more than a paragraph or so how your implementation works. You may provide the results for the final step as a table or a graph.

2. *Your code should run on the CADE machines.* You should include a shell script, **run.sh**, that will execute your code in the CADE environment. For each experiment, your code must print your results in the following order:

   (a) Entropy of the data

   (b) Best feature and its information gain

   (c) Cross-validation accuracies for each fold (for limiting depth setting)

   (d) Best depth (for the limiting depth setting)

   (e) Accuracy of the trained classifier on the training set (For the limiting depth setting, this would be for the tree with the best depth)

   (f) Accuracy of the trained classifier on the test set (For the limiting depth setting, this would be for the tree with the best depth)

   Without these details, you will lose points for your implementation.

   You are responsible for ensuring that the grader can execute the code using only the included script. If you are using an esoteric programming language, you should make sure that its runtime is available on CADE.

3. Please do *not* hand in binary files! We will not grade binary submissions.