CS 5350/6350: Machine Learning Fall 2020

Homework 1

Handed out: September 2, 2020
Due date: September 16, 2020

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, 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. Write the following Boolean functions as decision trees. (You can write your decision trees as a series of if-then-else statements, or use your favorite drawing program to draw a tree. You can use 1 to represent True and 0 to represent False.)

   (a) [2 points] x_1

   (b) [2 points] $x_1 \vee (x_2 \wedge x_3)$

   (c) [2 points] x_1 xor x_2

   (d) [4 points] The 2-of-3 function, whose value is true if at least two out of three Boolean features x_1, x_2 and x_3 are true. After you represent this function using decision tree, say whether you think using decision tree to represent m-of-n functions is a good idea.

2. In this problem we will manually build a decision tree to decide whether to wait for a table at a restaurant. Training data is given in Table 1. There are four features: Friday (Yes or No), Hungry (Yes or No), Patrons (None, Some, Full), and Type (French, Italian, Thai, Chinese).

Friday	Hungry	Patrons	Type	Wait?
No	Yes	Some	French	Yes
No	Yes	Full	Thai	No
No	No	Some	Chinese	Yes
Yes	Yes	Full	Thai	Yes
Yes	No	Full	French	No
No	Yes	Some	Italian	Yes
No	No	None	Chinese	No
No	Yes	Some	Thai	Yes
Yes	No	Full	Chinese	No

Table 1: Training data for the restaurant 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? When calculating entropy, the base of the logarithm should be base 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.

Friday	Hungry	Patrons	Type	Wait?
Yes	Yes	Full	Italian	No
No	No	None	Thai	No
Yes	Yes	Full	Chinese	Yes

Table 2: Test data for the restaurant problem

3. [**CS6350 students only**, 10 points] Recall that the ID3 algorithm identifies the best attribute to create the decision tree using the information gain heuristic. This heuristic

uses the difference between the entropy of the data and the expected entropy of the splits to identify the root attribute.

Here, we use entropy as a way quantify *impurity* of labels in the data. However, we could use other measures of impurity instead of entropy within the same definition. In this question, we will explore the use of other impurity measures.

(a) One natural way to define impurity is to measure the misclassification rate. This measures the error that we would have made if we had chosen the most frequent label. It is defined as

$$Misclassification(S) = 1 - \max_i p_i \tag{1}$$

Here, p_i is the fraction of examples that have a label i and the maximization is over all labels.

i. [2 points] Write down the definition of the information gain heuristic that uses the misclassification rate as its measure of impurity instead of entropy.

ii. [4 points] Use your new heuristic to identify the root attribute for the data in Table 1.

(b) [4 points] Another heuristic that is used to define impurity is the Gini coefficient, which is defined as

$$Gini(S) = \sum_i p_i(1 - p_i) \tag{2}$$

Use Gini coefficient to identify the root attribute for the training data in Table 1.

2 Experiments

For this part of the homework, you will be implementing different variants of the decision tree learner we saw in class. 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.

Task and Data

We want to predict whether a person makes $50K a year or not by accessing information about people such as age, gender and level of education.

We will use the `Adult` data set from the UCI Machine Learning repository to study this. The original version of `Adult` has 14 features, among which 6 are continuous and 8 are categorical. In order to make it easier to use, we will use a pre-processed version (and subset) of the original Adult data set, created by the makers of the popular LIBSVM tool. From the LIBSVM website: "In this data set, the continuous features are discretized into quantiles, and each quantile is represented by a binary feature. Also, a categorical feature with m categories is converted to m binary features."

Your task is to implement the ID3 algorithm and build a decision tree using the training files `a1a.train` available on the assignments page of the class website. [1]

The goal of the decision tree is to learn how to distinguish persons making more (labeled `+1`) or less (labeled `-1`) than $50,000 a year. We also provide the test set `a1a.test` to evaluate how your decision tree classifier is performing.

The data provided is in the LIBSVM format, where each row is a single training example. The format of the each row in the data is

```
<label> <index1>:<value1> <index2>:<value2> ...
```

Here, `<label>` denotes the label for that example. The rest of the elements of the row is a sparse vector denoting the feature vector. For example, if the original feature vector is $[0, 0, 1, 2, 0, 3]$, this would be represented as `3:1 4:2 6:3`. That is, only the non-zero entries of the feature vector are stored.

Cross-Validation

First, let us look at *cross-validation*, an important idea that will be used in all your homeworks and your project, and also when you want to use machine learning techniques on your own.

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

[1] Available at `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html`

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.

Implementation details

1. **Full trees**

   For this problem, your should use the data in `data` folder. You should train your trees on the examples in the training file `data/a1a.train` and evaluate on the test examples in `data/a1a.test`. Remember that you should not look at or use your testing file until your training is complete.

   In the first set of 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 restaurant data from Table 1. Discuss what approaches and design choices you had to make for your implementation and what data structures you used.

   (b) Report the error of your decision tree on all the examples `data/a1a.train`.

   (c) Report the error of your decision tree on the examples in `data/a1a.test`.

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

   Your report should include the following information

   (a) [3 points] The root feature that is selected by your algorithm

   (b) [3 point] Information gain for the root feature

   (c) [3 points] Maximum depth of the tree that your implementation gives

   (d) [4 points] Error on the training set

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

2. [**For CS 6350 Students**, 20 points] Update your decision tree implementation to the Gini index based information gain from earlier.

- [2 points] Describe (briefly) your implementation.
- [14 points] Report the same details as before for this new setting.
- [4 points] Does using Gini index give a different error on the test set that the original ID3 implementation? Are the resulting trees different?

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` where `X` is a number between 1 and 5 (inclusive).

   (a) [20 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) [16 points] Using the depth with the greatest cross-validation accuracy from your experiments: train your decision tree on the `data/train` file. Report the training and test errors.

   (c) [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. [**Discussion about potential fairness concerns**, 5 points, extra credit] An important socio-technical issue we are currently facing due to the prevalence of machine learning is that learning based systems can amplify societal biases. This can be harmful when learned systems make decisions that affect people's daily lives.

   As described above, the `Adult` dataset seeks to build predictors of a person's financial state using attributes like age, gender and level of education. Suppose you have a perfect decision tree classifier for the `Adult` dataset, as per test set performance. Briefly describe (in 4-5 sentences) why even such a classifier may incorporate societal biases.

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. Your code must print your results in the following order:

   (a) Most common label in the training data

   (b) Entropy of the training data

   (c) Best feature and its information gain in training data

(d) Accuracy on the training set

(e) Accuracy on the test set

(f) Average accuracies from cross-validation for each depth

(g) Best depth

(h) Accuracy on the test set using 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 compiled binary files! We will not grade binary submissions.