CS 5350/6350: Machine Learining Fall 2017

Homework 1

Handed out: 29 August, 2017
Due date: 12 September, 2017

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.
- 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.

1 Decision Trees

1. [6 points] 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) $(x_1 \wedge x_2) \vee (x_1 \wedge x_3)$

   (b) $(x_1 \wedge x_2)$ xor x_3

   (c) $\neg A \vee \neg B \vee \neg C \vee \neg D$

2. [24 points] In this problem, we are going to predict whether aliens will invade the earth using a decision tree classifier. The training data is given in Table 1. Build a decision tree to decide whether an alien civilization will invade the earth. There are four features:

   (a) **Superior Technology** (*Yes or No*) means whether or not the alien's technology is superior to earth's.

(b) **Environment** (*Yes or No*) means whether the environment on earth is suitable for this alien race.

(c) **Human** (*Dont Care, Like, Hate*) describes how the aliens feel about human-beings.

(d) **Distance** ($1, 2, 3, 4$ *lightyears*) describes how far is the aliens are from earth.

Technology	Environment	Human	Distance	Invade?
No	Yes	Not Care	1	Yes
No	Yes	Like	3	No
No	No	Not Care	4	Yes
Yes	Yes	Like	3	Yes
Yes	No	Like	1	No
No	Yes	Not Care	2	Yes
No	No	Hate	4	No
No	Yes	Not Care	3	Yes
Yes	No	Like	4	No

Table 1: Training data for the alien invasion 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) [1 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.

Technology	Environment	Human	Distance	Invade?
Yes	Yes	Like	2	No
No	No	Hate	3	No
Yes	Yes	Lkie	4	Yes

Table 2: Test data for alien invasion problem

3. [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. Apart from entropy, which you used in the previous question, there are other methods to measure impurity.

   We will now develop another heuristic for learning decision trees instead of ID3. If, at some node, we stopped growing the tree and assign the majority label of the remaining examples at that node, then the empirical error on the training set at that node will be

   $$MajorityError = 1 - \max_i p_i$$

   where, p_i is the fraction of examples that are labeled with the i^{th} label. Notice that $MajorityError$ can be thought of as a measure of impurity just like entropy.

   (a) [6 points] Using the $MajorityError$ measure, calculate the information gain for the four features respectively. Use 3 significant digits.

   (b) [4 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 majority error) lead to the same tree?

2 Linear Classifier

In the questions in this section, we have four features x_1, x_2, x_3 and x_4 and the label is represented by o.

1. [5 points] Write a linear classifier that correctly classifies the given dataset. You dont need to run any learning algorithm here. Try to find the weights and the bias of the classifier using the definition of linear separators.

x_1	x_2	x_3	x_4	o
1	0	1	1	1
0	1	0	1	1
0	0	1	0	-1

2. [5 points] Suppose the dataset below is an extension of the above dataset. Check if your classifier from the previous question correctly classifies the dataset. Report its accuracy.

3. [10 points] Given the remaining missing data points of the above dataset in the table below, find a linear classifier that correctly classifies the whole dataset (all three tables together)

x1	x2	x3	x4	o
0	0	0	1	1
0	0	1	1	1
0	0	0	0	-1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1
1	1	1	0	1

x1	x2	x3	x4	o
0	1	0	0	-1
0	1	1	0	-1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	1	0	1	1

3 Experiments

In this question you will be implementing a decision tree learner. You will experiment with the decision tree hyperparameters using cross-validation.

There is a secret computer science conference that only a selected group of computer scientists will be invited. Fortunately, we have a secret agent that has access to the guest list. But, she only has part of it. The accessible name list is in the **Dataset/training.data** file. Your job is to use this training data to learn a decision tree classifier that can predict if the names in the **Dataset/test.data** file will be invited to the secret conference. We suggest some features for the dataset, but you need to extract the features yourself. You are also welcome to add your own features.

You may use any programming language for your implementation. However, the graders should be able to execute your code on the CADE machines.

Cross-Validation

The depth of the tree is a hyper-parameter to the decision tree algorithm that helps reduce overfitting. You will see later in the semester that many machine learning algorithm (SVM, logistic-regression etc) have some hyper-parameters as their input. One way to determine a proper value for the hyper-parameter is to use a technique called cross-validation.

As usual we have a training set and a test set. Our goal is to discover good hyper-parameters using the training set. To do so, you can put aside some of the training data aside, and when training is finished, you can test the resulting classifier on the held out data. This allows you to get an idea of how well the particular choice of hyper-parameters does. However, since you did not train on your whole dataset you may have introduced a statistical bias in the classifier. To correct for this, you will need to 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 you must train n different classifiers. Of course, this is not practical for the data set you will use in this problem, so you 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 you want to evaluate how good a particular hyper-parameter is. You split the training data into k parts. Now, you will train your model on k 1 parts with the chosen hyper-parameter and evaluate the trained model on the remaining part. You should repeat this k times, choosing a different part for evaluation each time. This will give you k values of accuracy. Their average cross-validation accuracy gives you an idea of how good this choice of the hyper-parameter is. To find the best value of the hyper-parameter, you will need to repeat this procedure for different choices of the hyper-parameter. Once you find the best value of the hyper-parameter, use the value to retrain you classifier using the entire training set.

1. [25 points] **Implementation**

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

   (a) [8 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 alien data from Table 1. Discuss what approaches or choices you had to make during this implementation.

   (b) [4 points] Suggest at least 4 other features you could have extracted from this dataset.

   (c) [2 points] Report the error of your decision tree on the **Dataset/training.data** file.

   (d) [5 points] Report the error of your decision tree on the **Dataset/test.data** file.

   (e) [1 points] Report the maximum depth of your decision tree.

2. [20 points] **Limiting Depth**

   In this section, you will be using 4-fold cross-validation in order to limit the depth of your decision tree, effectively pruning the tree to avoid overfitting. You will be using the 4 cross-validation files for this section, titled Dataset/CVSplits/training 0X.data where X is a number between 0 and 4 (inclusive)

   (a) [10 points] Run 4-fold cross-validation using the specified files. Experiment with depths in the set $\{1, 2, 3, 4, 5, 10, 15, 20\}$, reporting the cross-validation accuracy and standard deviation for each depth. Explicity specify which depth should be chosen as the best, and explain why.

(b) [5 points] Using the depth with the greatest cross-validation accuracy from your experiments: train your decision tree on the **Dataset/training.data** file. Report the accuracy of your decision tree on the **Dataset/test.data** file.

(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?

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 should produce similar output to what you include in your report.

   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.

4 Decision Lists (For CS 6350 students)

A decision list is an ordered sequence of if-then-else statements. The sequence of if-then-else conditions are tested in order, and the answer associated to the first satisfied condition is output. Figure 1 shows an example of a decision list. At the root of the tree, we check whether x_1 is true *and* x_3 is false. If so, then the label is 0. Otherwise, we move to the next node below. Finally, if none of the checks succeed, then the default label (here 1 at the bottom of the list) is used. The specific decision list here is called a 2-decision list because no node has more two conditions to check.

Recall from class that a general decision tree can represent non-linear decision boundaries. In this question, we are concerned with a 1-decision list. Every condition in the 1-decision list is either a Boolean variable (such as x_1) or a negated Boolean variable (such as $\neg x_2$). Show that 1-decision lists are linearly separable functions.

(Hint: The easiest way to show this is to find a weight vector and a bias that will make the same predictions as a 1 decision list. That is, if the features that are used to construct the decision list are $\mathbf{x} = (x_1, x_2, \cdots, x_n)$, then find $\mathbf{w}$ and b such that the decision list will return 1 if, and only if, $\mathbf{w}^T\mathbf{x} \geq b$.)

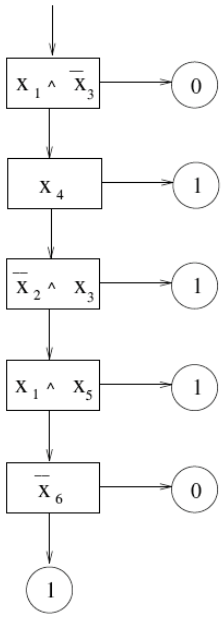

Figure 1: A 2-decision list.