CS 5350/6350: Machine Learining Fall 2015

Homework 1

Handed out: Sep 3, 2015
Due date: Sep 17, 2015

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 ask questions about 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) $x_1 \vee (x_2 \wedge x_3)$
   (b) x_1 xor x_2
   (c) 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) [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.

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 Nearest Neighbors

The nearest neighbors algorithm partitions the space of examples into regions corresponding to different labels. In two dimensions, the decision boundaries can be represented as a Voronoi diagram, which shows regions of the plane associated with each label.

For this part, you will be drawing the decision boundaries for simple datasets.

1. [5 points] Using the Euclidean distance measure between points, show a Voronoi map corresponding to the nearest neighbor classification of the following four points. (That is, draw a diagram that shows how the nearest neighbor classification of the following four points partitions the two dimensional plane.)

Label	x	y
A	1	1
A	1	-1
B	-1	-1
C	2	-2

2. [5 points] Using the city-block distance measure, show a Voronoi map corresponding to the nearest neighbor classification of the following three points.

   (Recall that the city-block measure/Manhattan distance/L_1 distance/taxicab metric between two points $\mathbf{x}$, $\mathbf{y}$ in the n dimensional space $\Re^n$ is defined as $\sum_{i=1}^{n} |x_i - y_i|$.)

Label	x	y
A	1	1
B	-1	-1
C	2	-2

3. [5 points] Can you design a *tiny* training data set such that nearest-neighbor classification using Euclidean distance and Manhattan distance will give you different results? To answer this question, you can assume the data are in two-dimensional plane. Each data point is specified using its Cartesian coordinate, just like in previous part of this problem. Show your training set and one test example so that the two distances will give your conflict results on that test example.

3 Experiment

In this question, you will implement decision tree learners and the K-nearest neighbors algorithms. Also you will learn to select the proper K value in your K-nearest neighbor algorithm using a technique called cross-validation.

This problem uses the Tic-Tac-Toe Endgame Data set from the UCI machine learning repository. Each data point has 9 features indicating the 9 locations on the Tic-Tac-Toe game board. The feature can have one of the three values: x, o, and b (for blank). The label is positive or negative, indicating whether player x wins or not.

Your goal is to use various learning algorithms on the training data to train a predictor and see how well it does on the test data.

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 value K is a hyper-parameter to the K nearest neighbor algorithm. 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. Some of the training data is put aside, and when training is finished, the resulting classifier is tested on the held out data. This allows you get get an idea of how well the particular choice of hyper-parameters does. Since you did not train on your whole dataset you may have introduced a bias. To correct for this, you will need to train many classifiers with different subsets of the training data removed.

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 to identify the hyper-parameter K (Don't confuse k with K, they are different). 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.

Data files

You can find the data on the assignments page of the class website in a file called `tic-tac-toe.zip`. It consists of six training files (used for 6-fold cross-validation in KNN), and one test file, which you will use for training and testing respectively.

1. Implement a decision tree data structure. (Remember that the decision tree need not be a binary tree!)

2. Implement the ID3 learning algorithm for your decision tree implementation. For debugging your implementation, you can use the previous toy examples from the homework like the restaurant data from Table 1. Report the accuracy of your decision tree on the test data. *Important*: you need to combine all training examples in all six training files when train your decision tree. Don't just train your tree using only one training file.

3. Implement a K Nearest Neighbor classifier for a general K. Note that your features are only categorical. So you have to make choices about how to measure distances between them. For example, you could consider using the Hamming distance between the feature representations.

4. Implement cross-validation. Run 6-fold cross-validation ($k = 6$) to select best K value from $K \in \{1, 2, 3, 4, 5\}$. The training data are already separated into six folds for you, each fold is a separate file. Report the average cross-validation accuracy for each choice of K. Use the best value of K to retrain your classifier on the entire train data set. Report its accuracy on the test data.

What to hand in for this problem

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. The points for this question will be split as follows: 20 points for the decision tree part, 20 points for the K-NN part and 15 points for your report.

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