CS 4780/5780 Machine Learning Due September 20, 2012

Assignment 2: Decision Trees and Hypothesis Testing

Instructor: Thorsten Joachims *Total Points: 100*

Course Policy:
Read all the instructions below carefully before you start working on the assignment, and before you make a submission
- Assignments are due at the beginning of class on the due date in hard copy.
- Write your NetIDs with submission date and time on the first page of the hard copy.
- Late assignments lose 1% of the cumulated final homework grade for each 24 hours
- No assignment will be accepted after the solution is publicized (about 4 days after due)
- The submission time of whatever you submit last (hard copy or CMS) is counted as the official submission timeand will determine the late penalty
- Upload only the code you wrote to CMS. Do not upload any additional libraries. Provide a README file with your submission that includes a list of all libraries and instructions needed to run your code.
- Attach a hard copy of your code to the submission.
- Late assignments can be submitted in class, in office hours, or directly to TAs.
- All sources of material must be cited. Assignment solutions will be made available along with the graded homework solutions. The University Academic Code of Conduct will be strictly enforced, including running cheating detection software.
- No more than one submission per group.

Problem 1: Decision Tree Decision Boundaries [20 points]

(a) Consider the training set S_1 in Figure 1. If you were to train an ID3 decision tree using information gain as your splitting criterion to obtain 100% training accuracy, which of the two decision trees (*Decision Tree A* in Figure 3 or *Decision Tree B* in Figure 4) would you obtain? Justify with calculations.

(b) Given your choice of either *Decision Tree A* or *Decision Tree B*, draw it's decision boundary overlaying the training set S_1 in Figure 1.

(c) Do you think the decision tree you chose would generalize well outside this training set? Explain. Draw a decision tree that you think would generalize well (for the training set in Figure 1).

(d) Consider a training set S_2 in Figure 2. Sketch by hand (do not use a computer) a decision boundary that a decision tree might generate if it were to obtain 0 training error on that set (Note, there is not a unique solution, but your boundary must look reasonable to that of decision boundaries that could be generated by decision trees).

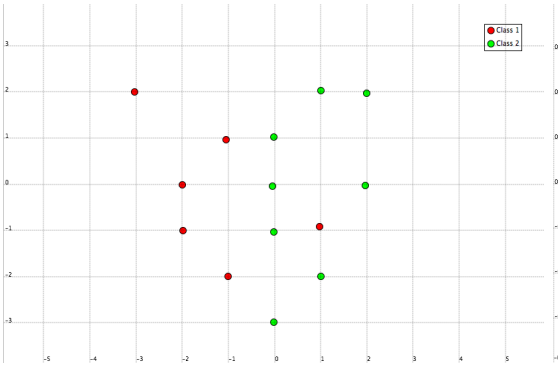

Figure 1: The training set S_1

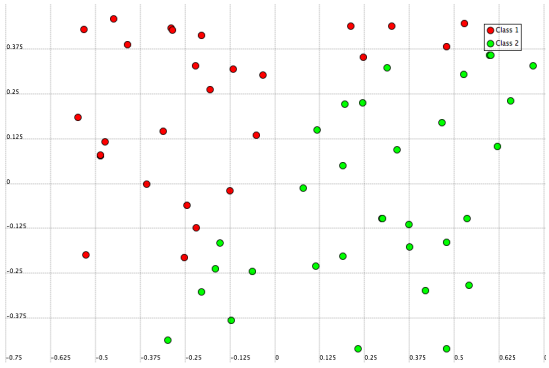

Figure 2: The training set S_2

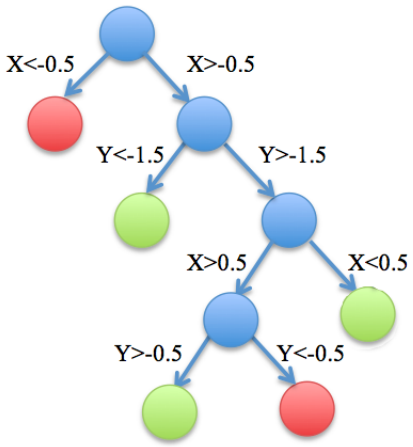

Figure 3: Decision tree A

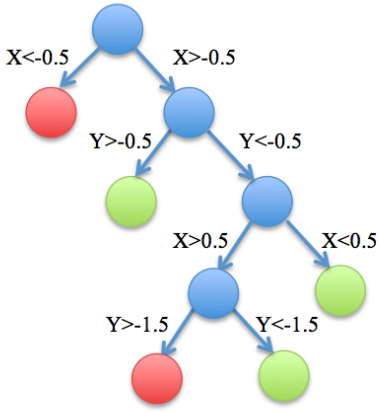

Figure 4: Decision tree B

(e) Do you think a decision tree is a good classifier to apply to a data set S_2 in Figure 2? Explain.

Problem 2: Decision Trees II [30 points]

In many real-world scenarios, we would like incorporate real-valued variables into our learning problem. In this problem we will investigate two different scenarios: one where the attribute is real-valued, and in another, where the target variable is real-valued.

(a) *Real-valued attributes.* Given N training examples, we would like to predict a

boolean target variable $y = \{yes, no\}$. We already know that if the attributes are discrete, we can construct a decision tree by considering each of the values that an attribute can take, and calculating information gain if we were to split on that attribute. Typically the number of values an attribute can take does not scale with the training set, and thus this calculation does not pose a big computational hurdle. The situation is very different if an attribute we are considering is real-valued (e.g. temperature), where you are unlikely to see the same attribute value twice. A naive way to find the splitting value of the real-valued attribute is to sort the training examples along that attribute and consider every possible decision of the form ($x \leq t$ and $x > t$) where t is every possible value of the attribute in the training set and x is a value of the attribute of a test point. We can then consider the information gain for every decision of that form, and choose the splitting threshold t that maximizes it. *Show that we don't actually have to consider every value of t for splitting, but only those that lie on the target value boundary (i.e. for which the target value y changes from yes to no and vice versa, e.g. as in Figure 5, where the boundaries are marked with darker borders). NOTE: You do not need a formal proof, but a mathematical line of reasoning beyond an 'example' needs to justify your answer*

Class	+	+	+	-	-	-	+	+	+	+	+	-	-	-
Attribute	0.4	0.8	1.2	1.5	1.9	2.5	3.1	3.3	5.8	5.9	6.1	7.0	7.5	8.1

Figure 5: Real-valued attribute example

(b) *Real-valued target (Regression).* Decision trees that make real-valued predictions are also known as regression trees. Consider that you now want to make a prediction about tomorrow's temperature. In trying to construct a decision tree with the methodology we learned, we will immediately run into a problem of calculating entropy (try it!). But entropy is just one of the ways of computing 'impurity' of a split. For discrete valued attributes and targets some of the other heuristics are Gini-index $\left(1 - \sum(p_+^2 + p_-^2)\right)$, and misclassification error $(1 - max\{p_+, p_-\})$ where p_+ and p_- are class conditional probabilities. All of these heuristics share a common characteristic - they are zero when the class distribution is 'pure' (just one class remains). But none of the three methods will work for continuous target variables. *Suggest a splitting criterion for continuous target variables and show that it too measures 'impurity' of the class conditional distribution. You do not need to write a formal proof, but you must show clear and logic reasoning justifying why your method will work in building a regression tree*

(c) *Test your regression tree method* Now build a regression tree using your suggested metric for splitting to fit the following function $y(x) = \exp(-x^2)$ on an interval $x \in (-5, 5)$ (x is a real-valued attribute, and y is a real-valued target). Constrain

your tree to depth of two with binary splits. What are the splitting thresholds for each node in the tree, and what do you think is a reasonable way to compute the tree output at each leaf? Plot the thresholds (as vertical lines) over the graph $y(x)$. *NOTE: depending on your splitting method, there is no single correct answer. But the result should look reasonable to you*

Problem 3: Decision Tree and Hypothesis Testing [50 points]

If you have ever been to the 2nd floor of Upson, you may have seen QR codes posted all over the walls. QR codes like those can be used to localize a robot precisely in an indoor setting. Another approach for indoor localization is to use WIFI. Given a unique identifier (MAC address) of a Red Rover router and its signal strength, we can figure out approximately where we are within a building. Unfortunately simple geometry doesn't work well, because signal strength is often a highly non-linear function of distance with dead-spots and all sorts of distortions. That, however, makes it a perfect problem for machine learning. In this problem we will investigate two approaches (Decision Tree and KNN) to predicting a user's location within the first floor of Duffield given a WIFI signature (an associative vector of signal strengths and routers) and compare their performance.

(a) *Implement Decision Tree* Implement an ID3 decision tree to predict a user's location y from a set {*LOUNGE, HALLWAY, ATRIUM_1, ATRIUM_2, MATTINS, PHILLIPS*} using information gain as the splitting criterion. Train the decision tree using the *wifi.train* file which follows the following format:

$<$ `location` $>$ $-$ $<$ `wifi_station index` $>$:$<$ `station_strength_in_dB value` $>$
$<$ `wifi_station index` $>$:$<$ `station_strength_in_dB value` $>$ ...

Due to the sparsity of the wifi signature, treat missing station power values as $-\infty$ when constructing your decision tree (i.e. replace missing attribute values with a large negative value).

(a) *Accuracy on training set* Measure the classification error of your decision tree on the training set *wifi.train* for each location (ratio of the number of misclassified instances belonging to class i to the total number of instances belonging to class i) as well as the total classification error across all classes (ratio of the total number of misclassified instances to the total number of instances). Do you expect 100% accuracy on the training set?

(b) *Accuracy on test set* Measure the classification error of your decision tree on the supplied test set *wifi.test* for each location as well as the total classification error across all classes (as in part (a)).

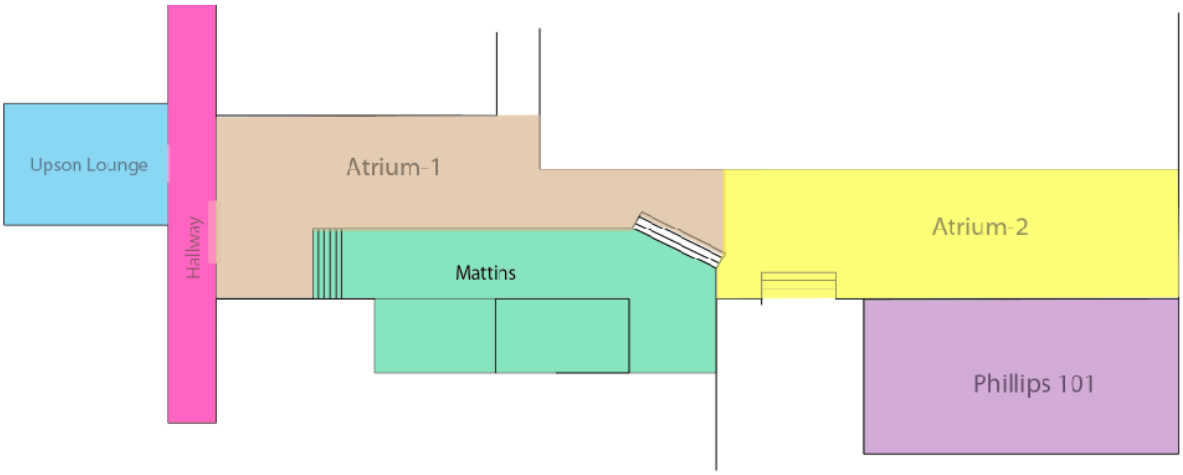

Figure 6: Duffield layout

(c) *Hypothesis Testing* Hypothesis Testing. Now we would like to compare the generalization accuracy of the ID3 decision tree you trained in the previous part to the one with early stopping at a depth of two (construct a decision tree with depth two by removing all notes from your decision tree above depth 2). Use the McNemar and explain whether you can conclude with 95% confidence whether one of the two classifiers has lower generalization error than the other.

(d) *Cross Validation* You now want to know whether the generalization accuracy of decision trees produced by ID3 is different from those produced by ID3 with early stopping at depth 2 on average for the localization problem. Concatenate training and test set we supplied, and perform 10-fold cross validation over the combined sample. Use the paired t-test to establish whether the two algorithms have different classification error at a 95% confidence level (e.g. you can ignore that the different folds are not statistically independent when you apply your test, but always keep in mind that the test is only approximate in this situation).

(e) *Effect of validation set size* We will explore a technique of validation as a method for finding an optimal parameter for a classifier. We will split the supplied training set into training and validation sets using a fixed ratio. We will then train the decision tree (as in the previous parts) on the resulting (smaller) training set, and use the validation set to pick the optimal tree depth (one that minimizes classification error). Finally we use the test set to measure the classification error of the decision tree that performed best on the validation set. We will now explore how the ratio of training to validation

set sizes affects the validation/testing errors. Because smaller training and validation sets will provide unreliable point-estimates of classification errors, repeat each experiment 10 times by splitting the test set randomly (with the same ratio for each experiment) and plot each test/validate error datapoint on the same set of axes (classification error vs. train/validate ratio). Explore train/validate ratio from 0.1 to 0.9 in increments of 0.1. What do you observe in the resulting plots?