CS4700 Fall 2011: Foundations of Artificial Intelligence

Homework #4

Due Date: Friday Nov 4 on CMS (PDF) and in class (hardcopy)
Thereafter 10% off each 24H period late until homework is graded on CMS
Your graded homework can be picked up directly from the TAs.

Submission: Submit on CMS and paper copies at the beginning of class or to TA directly. Please include your **name**, **NetID**, the **CMS submission date and time**, **slack days** used and remaining. Code in 9pt courier single spacing with any blank lines removed and function names highlighted. Your assignments should reflect your individual work – ok to discuss strategies but do not compare your answers with anyone else.

Question 1: Two thumbs up (20%)

You have three friends: Mike, David and Jennifer. You believe that hanging out with them can put you in different mood. You collected some data and want to see if it is possible to predict your own feelings with those data by building a decision tree. Table below shows your past experience with different combination of your friends and your resulting feeling for each. “Yes” means present, “No” means not present.

Mood	Mike	David	Jennifer
Happy	Yes	Yes	Yes
Happy	LeftHalfway	Yes	Yes
Sad	Yes	No	No
Sad	No	Yes	No
Happy	Yes	No	Yes
Sad	No	Yes	Yes

1) Show your calculation of entropy and information gain of each attribute at each decision node. In case of attribute ties, always prefer the attribute listed first in the table. For class label majority vote ties, always prefer the class happy mood.
2) After learning the mood decision tree, one of your friends said that when building the tree, you should stop splitting whenever all candidates’ attributes have zero information gain. Do you agree? If so, why? If you don’t agree, please design a dataset with A1, A2 as binary attributes and fill in the table below with an example where information gain is 0 but splitting is still necessary.

A1	A2	Decision Class
		0
		0
		1
		1

Question 2: The Curse of Dimensionality. (30%)

While *k*-NN is simple and has interesting theoretical properties, it does have significant problems, among them that it suffers from the "curse of dimensionality" [1], as adding (potentially spurious) features serves to confuse the classifier. You will explore this effect on a very simple problem!

A. Consider inputs distributed uniformly in a *d*-dimension hypercube of length 1. Suppose we define the neighborhood around point x to be a hypercube around *x* of Length *L*. A neighborhood is significant if it captures a fraction *r* of the unit volume. It can be assumed that the point and the neighborhood are fully inside the unit hypercube without loss of generality. (Volume of a hypercube of length *L* is L^d where *d* is the dimension).
   a. Find L in terms of *r* and *d* for a significant neighborhood.
   b. What is value of *L* for r =0.05 and d = 1, 2, 5, 10, 50, 100, 500, 1000 (in other words you can see how large *L* we need in order to capture 5% of data in different dimension compare the obtained neighborhood with our data space which is a hypercube of Length 1)

B. Suppose you have a binary classification problem $y \in \{-1, +1\}$ with *N* attributes in the vector ***x*** where $x_i \in \{-1, +1\}$. For all examples ***(x**, y)*, suppose that the first 3 attributes of **x** always hold the class, that is, x_1=y, x_2=y, x_3=y and that the other attributes of a vector have no importance in determining the class. In this setting, suppose we have a training set consisting of two examples: x_{-1} = (−1,−1,−1, . . . ,−1), and x_{+1}=(+1, +1, +1, . . . , +1). Further suppose that we have a single example **x** that we classify with 1-nearest neighbor: the first 3 attributes x_1 , x_2 and x_3 hold the true class, but all other attributes of the vector are uniformly randomly chosen between {−1, +1}, so if N = 53, the first 3 attributes hold the class, and the other 50 attributes are random. For odd values of N (i.e., N = 3, 5, 7, 9, . . .), write out the probability that the random example **x** is classified correctly. Write this probability as a formula (hint: use the binomial distribution). Also write out the numeric value of this probability for N = 3, 5, 9, 19, 33, and 53.

[1] http://en.wikipedia.org/wiki/Curse_of_dimensionality

Question 3: Tumor Classification (50%)

Classifying cancerous tumors as malignant or benign is an important task when deciding on further treatment. In this problem, we will use two methods to automatically classify tumors based on their size, shape, and other features.

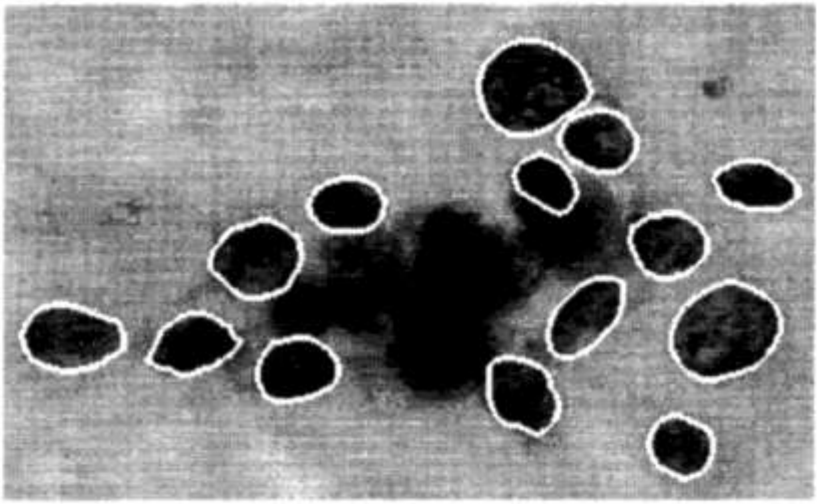

Figure 1: This is a magnified image of a malignant breast Fine Needle Aspirate (FNA). The visible cell nuclei have been outlined with the help of a curve fitting program. The system also computes various features for each nucleus, which can be used to diagnose the sample [2]

First, download the data from the course website (data originally from UCI ML Archive[3]). You should have two files: wdbc_train.dat and wdbc_test.dat (as well as wdbc_descrip.txt, which describes the data but is not necessary for this problem). As you might imagine, the two files contain the training and testing data, respectively. Note that it is critical never to use the test data at all while training!

The format of the data in each file is one row per data point. Each row contains the following fields:
ID (not used), class (0 for benign, 1 for malignant), feature 1, feature 2, ….

a) Load the data into your favorite program. How many data points are in the train file? How many in the test file? How many dimensions of features are available?

b) Implement a k-Nearest Neighbors (kNN) classifier. Provide a performance table showing your performance. Columns should be k, classification accuracy on training set, F-measure on training set, classification accuracy on test set, F-measure on test set. (In many problems, including this one, the class labels are unbalanced, and so accuracy is not necessarily the best measure of performance. The F-measure is the harmonic mean of precision and recall, and it provides a more balanced performance metric.). Create rows for k=1, 3, 5, 7, 9.

c) If you wanted to use an Artificial Neural Network (ANN) instead to classify the data, how many input neurons would you need? How many output neurons?

[2] O.L. Mangasarian, W.N. Street and W.H. Wolberg. Breast cancer diagnosis and prognosis via linear programming. Operations Research, 43(4), pages 570-577, July-August 1995.

[3] http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29

d) How would you map the network's (real-valued) output to the (binary) classification? Your classifier may output either {0,1} or {-1,1}, whichever you find easiest.

e) Given this interpretation of the output, what is the perceptron learning (weight update) rule?

f) Implement an ANN to classify the tumors as benign or malignant. Make a table as in (a) showing the performance of your classifier for five different learning rates.

g) Briefly discuss your implementation, and give the values you choose for any parameters that were necessary.

Bonus 1 (up to 20% on this problem): Improve your kNN implementation so that it reliably outperforms the best F-measure you found in part (a). Update your table in a. to show your new results. Also add a new "10 run average" row showing the average results over 10 runs for the previous best k, as well as for your new method. Briefly describe what improvements you made.

Bonus 2 (up to additional 20% on this problem): Improve your ANN implementation so that it reliably outperforms your F-measure found in part (f) Update your table in (f) to show your new results. Also add a new "10 run average" row showing the average results over 10 runs for your previous best-learning-rate ANN, as well as for your new method. Briefly describe what improvements you made.

Programming tips: Don’t create anything more complicated than you need to. This is a simple program. Implement it simply. Help your grader help you: be clear with comments and naming.