CS434 — Implementation Assignment 3 — Due May 6th 11:59PM, 2016

General instruction.

1. The following languages are acceptable: Java, C/C++, Matlab, Python and R.
2. You can work in team of up to 3 people. Each team will only need to submit one copy of the source code and report.
3. You need to submit your source code (self contained, well documented and with clear instruction for how to run) and a report on canvas. On your report you should clearly indicate your team members.
4. Be sure to answer all the questions in your report. Your report should be typed, submitted in the pdf format. You will be graded based on both your code as well as the report. In particular, **the clarity and quality of the report will be worth 10% of the pts**. So please write your report in clear and concise manner. Clearly label your figures, legends, and tables.

Part I: Model selection for KNN

The task is to implement the K-nearest neighbor algorithm. In particular, you need to:

- (20 pts) Perform model selection using leave-one-out cross-validation to select the best K for the given learning task using the provided training data. Please consider the following range of K values: 1, 3, 5, $\cdots$, 15. (This is a suggested range, feel free to explore more possible K values). What is your choice of K?
- (10 pts) For each possible value of K, please compute the following: 1) the training error (measured as the number of mistakes) 2) the leave-one-out cross-validation error on the training set; and 3) the number of errors on the provided test data. Plot these errors as a function of K.
- (5pts) Discuss what you observe in terms of the relationship between these three different measure of errors.

  The dataset provided for Part I constitutes of 30 features and is a matrix of $N \times 31$ dimension. The first column of the test and train data is the true class label of the samples and the remaining columns are the features. This dataset belongs to the Wisconsin Diagnostic Breast Cancer dataset and the classes are the diagnosis (+1= malignant and -1= benign).

Part II: Decision tree

For this part, you will implement the top-down greedy induction algorithm for learning decision trees. In particular, given a set of training examples your implementation should perform the following:

- (10 pts) learn a decision stump, i.e. a decision tree with only a single test. To build a decision stump, simply apply the top down decision tree induction algorithm to select the root test and then stop and label each of the branches with its majority class label.
- (25 pts) learn a fully grown decision tree.

Please use the information gain as the selection criterion for building the decision tree. Test your implementation on the monk data set that is provided to you. The original monk dataset is a synthetically generated dataset that was created for a comparison study on different machine learning algorithms. For those of you who are interested, this dataset (and two other related data sets), and the comparison results are described in the following paper: The MONKs Problem A performance comparison of different learning algorithms Accessible at `http://robots.stanford.edu/papers/thrun.MONK.html`. The particular dataset provided for this implementation assignment is the a revised monk 1 dataset. There are 6 features:

- x_1: head_shape $\in$ round, square, octagon
- x_2: body_shape $\in$ round, square, octagon
- x_3: is_smiling $\in$ yes, no
- x_4: holding $\in$ sword, balloon, flag
- x_5: jacket_color $\in$ red, yellow, green, blue
- x_6: has_tie $\in$ yes, no

Note that the provided data use different numerical values to represent the different feature values. For example, feature 1 in the data has three different values 1, 2 and 3, representing round, square, and octagon respectively. For this assignment, please only use binary split. That is for features with more than two possible values, you will test it against a particular value. For example, $x_1 = square$.

For this particular data set, the class label is a concept defined by the following rule: $(has\_tie = yes)\&(head\_shape != body\_shape)$ That is if $x_6 = 1$ and $x_1 \neq x_2$, then $y = 1$ (positive), otherwise $y = 0$ (negative).

In your report, please provide

(a) Both the learned stump and the learned decision tree. To help grading easier, please provide for each selected test its information gain.

(b) The training and testing error rate of the learned stump and full decision tree.

Finally, please answer the following questions:

(a) (5 pts) Given the formula for generating the class label, please provide a (as compact as possible) decision tree that will correctly classify all training examples.

(b) (5 pts) Given a training set that is randomly generated based on this concept, do you expect in general the top-down greedy algorithm to be able to learn this optimal tree? Why?

Note that while you can implement both the decision stump and the decision tree as a single function, I strongly suggest you to create a separate function that learns decision stump directly. This is because the decision stump function will be used as a base function for future assignments.