CS534 — Implementation Assignment 3 — Due 11:59PM Nov 24, 2019

General instructions.

1. The following languages are acceptable: Java, C/C++, Python, Matlab

2. You can work in a team of up to 3 people. Each team will only need to submit one copy of the source code and report. You need to explicitly state each member's contribution in percentages (a rough estimate).

3. Your source code and report will be submitted through TEACH

4. You need to submit a readme file that contains the programming language version you use (e.g. Python 3.6) and the command to run your code (e.g. python main.py).

5. Please make sure that you can be run code remotely on the server (i.e. babylon01) especially if you develop your code using c/c++ under visual studio.

6. Be sure to answer all the questions in your report. You will be graded based on your code as well as the report. In particular, **the clarity and quality of the report will be worth 10 pts**. So please write your report in clear and concise manner. Clearly label your figures, legends, and tables.

7. In your report, the results should always be accompanied by discussions of the results. Do the results follow your expectation? Any surprises? What kind of explanation can you provide?

Decision Tree Ensemble for Mushroom Classification (total points: 80 pts + 10 report pts + 10 result pts)

In this assignment we will work on the Mushroom Classification task to classify between edible and poisonous based on a set of discrete features related to each species of mushroom. The goal in this assignment is to develop variations of the **decision tree** algorithm and ensemble methods.

Data. The data for this assignment is taken directly from the commonly used UCI Machine Learning Repository. Here is a short description of each train and validation split:

(a) **Train Set (pa3_train.csv):** Includes 4874 rows (samples). Each sample contains the class (poisonous or edible) with 22 categorical features (split into one-hot vectors for total of 117 features) related to the mushroom's properties. For a data dictionary, see here: `https://archive.ics.uci.edu/ml/datasets/mushroom`

(b) **Validation Set (pa3_val.csv):** Includes 1625 rows. Each row obeys the same format given for the train set. This set will be used to see the performance of the models.

(c) **Validation Set (pa3_test.csv):** Includes 1625 rows. This is to ensure your model generalizes. We omit the class values and will make sure the accuracy is sufficient when grading.

Important Guidelines. For all parts of this assignment:

(a) We assigned labels already for the class **1 to poisonous (p) and 0 to edible (e)**. Be sure they aren't modified when making predictions, as we'll use this for the ground-truth.

(b) The data is already pre-processed for you, but you should be familiar with creating one-hot vectors out of discrete features. The csv headers will contain the particular feature/value combination you'll be splitting on. Notice that there are some where values were missing (a "?") which we treat as a possible value itself.

(c) Please do not add bias to the features.

Part 1 (20 pts) : Decision Tree (DT). For this part we are interested in using a decision tree with below configuration:

- The DT uses gini-index to measure the uncertainty. Specifically if we have a node split from training list A to two left and right list AL and AR as depicted in figure 1 then

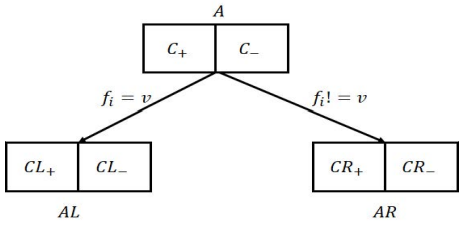

Figure 1: Split according to feature f_i testing against value v

the benefit of split for feature f_i against value v is computed as:

$$B = U(A) - p_l U(AL) - p_r U(AR) \tag{1}$$

Where U is the uncertainty measure. For this assignment, we will use gini-index, which is computed for a list such as AL as follows:

$$U(AL) = 1 - p_+^2 - p_-^2 = 1 - (\frac{CL_+}{CL_+ + CL_-})^2 - (\frac{CL_-}{CL_+ + CL_-})^2 \tag{2}$$

and p_l and p_r are the probabilities for each split which is given by

$$p_l = \frac{CL_+ + CL_-}{C_+ + C_-} \tag{3}$$

- The features are categorical with more than 2 possible values in most cases. So a feature might be tested multiple times against different values in the tree. However, when creating one-hots, we essentially make each combination of feature/value a feature itself. Note that this data does contain missing features. For this assignment, we will use a simple strategy of treating "missing" which is denoted as "?" in the data, as another possible value for the categorical feature. As mentioned earlier, this is reflected in the pre-processing (headers) that we have done.

Please implement below steps:

(a) Create a decision tree with maximum depth of 2 (root is at depth=0) on the train data using binary splits. It is of course possible to avoid doing one-hot vectors on discrete data when using decision trees and extending the impurity to handle more than 2 children at each node (and sometimes this actually performs better), but for simplicity we have provided it in a format set up for binary splits.

(b) Using the created decision tree, compute the train and validation accuracy. (Note: this may seem over-simplified with depth 2, but it is possible to fit this data perfectly, so we are restricting the size).

(c) Now, create trees with depths ranging from 1 to 8 (inclusive). Plot and explain the behavior of train/validation performance against the depth. At which depth does the train accuracy reaches to 100% accuracy? If your tree could not get to 100% before the depth of 8, keep on extending the tree in depth until it reaches 100% for the train accuracy.

(d) Report the depth that gives the best validation accuracy? You will probably hit a point where it will not need to be deeper, and it's best to use the simplest version.

Part 2 (30 pts) : Random Forest (Bagging). In this part we are interested in random forest which is a variation of bagging without some of its limitations. Please implement below steps:

(a) Implement a random forest with below parameters:
n : The number of trees in the forest.
m : The number of features for a tree.
d : Maximum depth of the trees in the forest.

Here is how the forest is created: The random forest is a collection of n trees. All the trees in the forest has maximum depth of d. Each tree is built on a data set of size 4874 sampled (with replacement) from the train set. In the process of building a tree of the forest, each time we try to find the best feature f_i to split, we need to first sub-sample (without replacement) m number of features from the feature set and then pick f_i with highest benefit from m sampled features.

(b) For $d = 2$, $m = 5$ and $n \in [1, 2, 5, 10, 25]$, plot the train and validation accuracy of the forest versus the number of trees in the forest n.

(c) What effect adding more tree into a forest has on the train/validation performance? Why?

(d) Repeat above experiments for $d = 2$, $n = 15$ trees, and $m \in [1, 2, 5, 10, 25, 50]$. How does m change the train/validation accuracy? Why?

(e) With your best result, run 10 trials with different random seeds (for the data/feature sampling you can use the same seed) and report the individual accuracy's, as well as the average train/validation accuracy across the 10 trials. Overall, how do you think randomness affects the performance?

Part 3 (30 pts) : AdaBoost (Boosting). For this part we are interested in applying AdaBoost to create yet another ensemble model with decision tree. Considering the AdaBoost algorithm described in the slide, please do below steps:

(a) Let the weak learner be a decision tree with **depth of only 1.** The decision tree should get a weight parameter D which is a vector of size 4874 (train size). Implement the decision tree with parameter D such that it considers D in its functionality.
(Hint: It changes the probability estimation used for computing gini-index and also for the predictions at leaves, use majority "weights" instead of majority counts).

(b) Using the decision tree with parameter D implemented above, develop the AdaBoost algorithm as described in the slide with parameter L (number of base classifiers).

(c) Report the train and validation accuracy for $L \in [1, 2, 5, 10, 15]$.

(d) Explain the behavior of AdaBoost against the parameter L.

(e) Repeat with depth of 2 and $L = 6$. How does this compare to the best train/validation performance with depth of 1 and $L = 15$?

(f) Using your best AdaBoost, generate predictions for **pa3_test.csv**. You should include only one column, which is the class prediction for the mushroom (0 or 1). Preferably without a header or index (just ensure the index isn't shuffled - same order and length as the test csv).

Submission. Your submission should include the following:
1) Your source code with a short instruction on how to run the code in a readme.txt.
2) Your report only in pdf, which begins with a general introduction section, followed by one section for each part of the assignment.
3) Predictions file for **pa3_test.csv**. We will verify the accuracy (10 pts).
4) Please note that all the files should be in one folder and compressed only by .zip.