Homework 2 for CSI 431/531

All homeworks are individual assignments. This means: write your own solutions and do not copy code/solutions from peers or online. Should academic dishonesty be detected, the proper reporting protocols will be invoked (see Syllabus for details).

Instructions: Submit two files. One should be a write-up of all solutions and observations, as *Solution.pdf*. The second should be an archive *Code.zip* containing code and any relevant results files.

Note: individual functions will be tested by a script and potentially different train/test data. Do not change method names or parameters, simply provide implementations. Also, please, do not add additional library imports, except the standard numpy and math.log(). In other words, you should not import the decision tree implementation from scipy or other ML libraries.

1. **Decision trees (100 points total).** As part of this question you will implement and compare the Information Gain, Gini Index and CART evaluation measures for splits in decision tree construction.Let $D = (X, y), |D| = n$ be a dataset with n samples. The entropy of the dataset is defined as

$$H(D) = -\sum_{i=1}^{2} P(c_i|D) log_2 P(c_i|D),$$

where $P(c_i|D)$ is the fraction of samples in class i. A split on an attribute of the form $X_i \leq c$ partitions the dataset into two subsets D_Y and D_N based on whether samples satisfy the split predicate or not respectively. The split Entropy is the weighted average Entropy of the resulting datasets D_Y and D_X:

$$H(D_Y, D_N) = \frac{n_Y}{n} H(D_Y) + \frac{n_N}{n} H(D_N),$$

where n_Y are the number of samples in D_Y and n_N are the number of samples in D_N. The Information Gain (IG) of a split is defined as the the difference of the Entropy and the split entropy:

$$IG(D, D_Y, D_N) = H(D) - H(D_Y, D_N). \tag{1}$$

The **higher** the information gain the better.

The Gini index of a data set is defined as $G(D) = 1 - \sum_{i=1}^{2} P(c_i|D)^2$ and the Gini index of a split is defined as the weighted average of the Gini indices of the resulting partitions:

$$G(D_Y, D_N) = \frac{n_Y}{n} G(D_Y) + \frac{n_N}{n} G(D_N). \tag{2}$$

The **lower** the Gini index the better.

Finally, the CART measure of a split is defined as:

$$CART(D_Y, D_N) = 2\frac{n_Y}{n}\frac{n_N}{n} \sum_{i=1}^{2} |P(c_i|D_Y) - P(c_i|D_N)|. \tag{3}$$

The **higher** the CART the better.

You will need to fill in the implementation of the three measures in the provided Python code as part of the homework. **Note: You are not allowed to use existing implementations of the measures.**

The homework includes two data files, *train.txt* and *test.txt*. The first consists of 100 observations to use to train your classifiers; the second has 10 to test. Each file is comma-separated, and each row contains 11 values - the first 10 are attributes (a mix of numeric and categorical translated to numeric, e.g. {T,F} = {0,1}), and the final being the true class of that observation. You will need to separate attributes and class in your $load(filename)$ function.

You are also given $HW2.py$. This contains all necessary functions with additional details in the docstrings. Implement each function within the skeleton given. You are allowed to add any helper functions as desired. The functions provided are what will be graded, along with solutions written up.

(a) **[10 pts.]** Implement the $IG(D, index, value)$ function according to equation 1, where D is a dataset, $index$ is the index of an attribute and $value$ is the split value such that the split is of the form $X_i \leq value$. The function should return the value of the information gain.

(b) **[10 pts.]** Implement the $G(D, index, value)$ function according to equation 2, where D is a dataset, $index$ is the index of an attribute and $value$ is the split value such that the split is of the form $X_i \leq value$. The function should return the value of the gini index value.

(c) **[10 pts.]** Implement the $CART(D, index, value)$ function according to equation 3, where D is a dataset, $index$ is the index of an attribute and $value$ is the split value such that the split is of the form $X_i \leq value$. The function should return the value of the CART value.

(d) **[20 pts.]** Implement the function *bestsplit*$(D, criterion)$ which takes as an input a dataset D, a string value from the set $\{``IG", ``GINI", ``CART"\}$ which specifies a measure of interest. This function should return the best possible split for measure *criterion* in the form of a tuple $(i, value)$, where i is the attribute index and $value$ is the split value. The function should probe all possible values for each attribute and all attributes to form splits of the form $X_i \leq value$.

(e) **[10 pts]** Load the training data "train.txt" provided in the homework by implementing the function $load(filename)$, which should return a dataset D, and find the best possible split for each of the three criteria for the data (Hint: Note that some measures need to be minimized and some maximized, also consider numpy's loadtxt to load the dataset).

(f) **[10 pts]** Assume that you built a very simple DT based on the only best split for each criterion from the previous question. Show the three decision trees resulting from the best splits (draw the trees, split points and decision nodes). Assume that in each of the two leaves of the tree a decision is made to classify as the majority class.

(g) **[30 pts]** How many classification errors will the above three classifiers make on the testing set provided in "test.txt". A classification error occurs when a testing instance is not classified as its correct class. Note, that you need to implement the three functions $classifyIG(train, test)$, $classifyG(train, test)$, $classifyCART(train, test)$ based on the best splits you found in the previous parts. Each function should take in training data, create the best single split on a single attribute using the above defined functions, and classify each observation in the test data. The output should be a list of predicted classes for the test data (in the same order, of course). Then compare these predicted classes to the true classes of the test data.