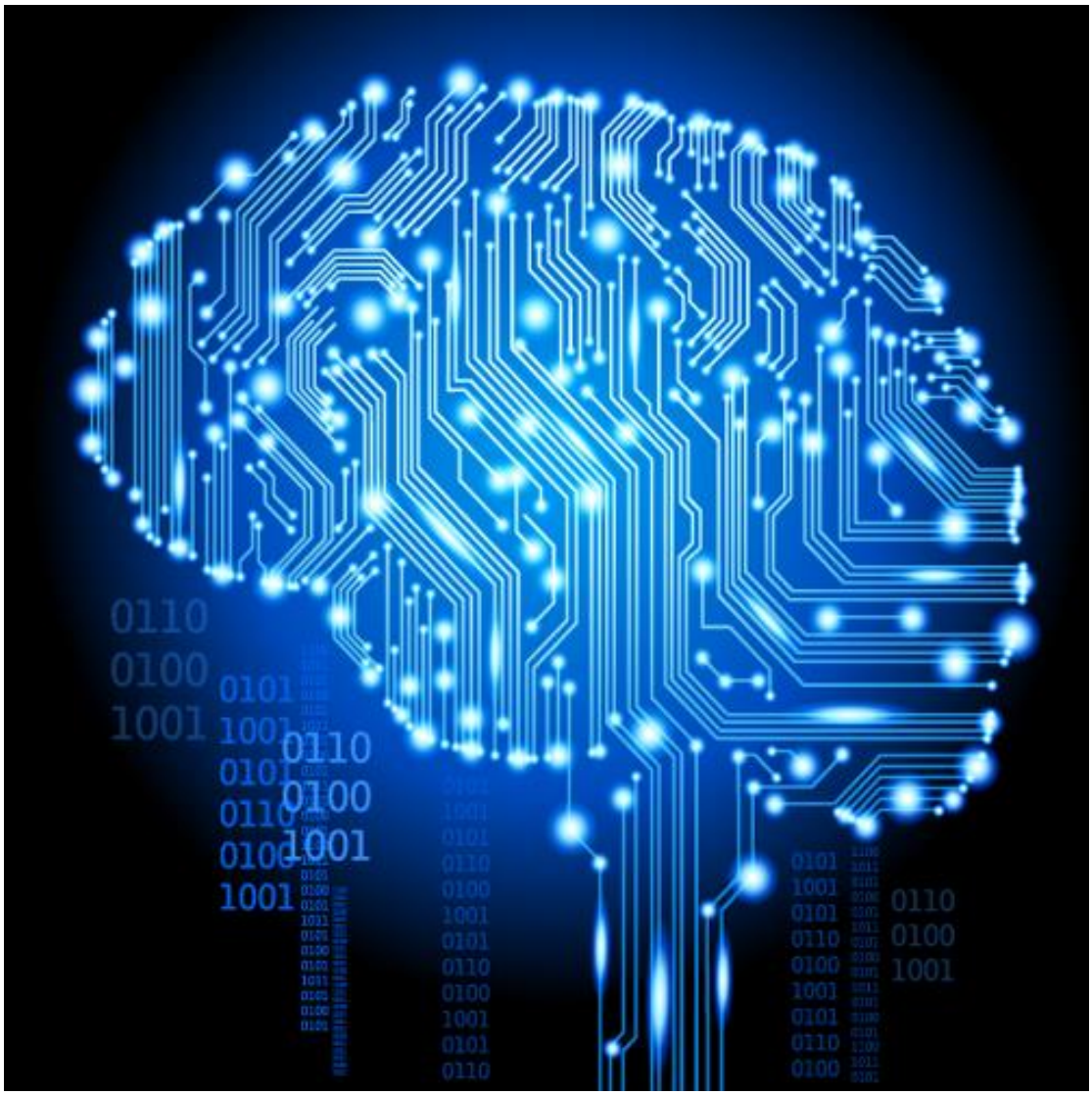
COMP3009 Machine Learning

Computer Based Coursework Manual – Autumn 2021

Assessed coursework 2: Decision Trees



Assignment 2: Decision Trees

In this assignment, as a group, you will implement decision trees algorithm for both regression and classification. The submission deadline is **25th Nov, 2021 at 4 pm.** This assignment will require you to train and test on both your regression and classification datasets. If your classification dataset is multi-class, convert it to a binary classification problem choosing one class as the positive class. If your regression problem has multiple regression variables, choose one to predict.

Decision trees are one of the simplest and yet most successful forms of learning algorithms. A decision tree takes as input an object or situation described by a set of attributes and returns a decision – the predicted value for the input. Classification is done by learning a function with discrete outputs based on the subset of data that has reached the node. A decision tree makes its decision by performing a sequence of tests, one per node on its path. Each internal node in the tree corresponds to a test of the value of one of the properties (for a monothetic tree), and the branches from the node are labelled with the possible values of the test. Problems where each test results in a binary outcome will generate binary trees, i.e. the branching factor of the tree will be 2. Each leaf node in the tree specifies the value to be returned if that leaf is reached. Decision tree representations seem to be very natural for humans, e.g. some "How To" manuals are written entirely as a single decision tree stretching over hundreds of pages. A pseudo code of the decision tree learning algorithm for classification is depicted in Table 1.

function DECISION-TREE-LEARNING(features, labels) **returns** a decision tree
    inputs: *features*, set of N x d training examples
        labels, set of N x 1 target labels for the training examples

if all *examples* have the same label **then return** a leaf node with label = the label
else
    [*best_attribute, best_threshold]* ← CHOOSE-ATTRIBUTE(*features,targets*)
    tree ← a new decision tree with root decision attribute *best_attribute* and
        threshold *best_threshold*

    add a branch to *tree* corresponding to *best_attribute* < *best_threshold*
    {$examples_l$, $targets_l$}← {elements of *examples* with *best_attribute*
        <*best_threshold* and corresponding target labels}
    subtree ← DECISION-TREE-LEARNING($examples_l$, *targets*)
    add a branch to *tree* corresponding to *best_attribute* >= *best_threshold*
    {$examples_r$, $targets_r$}← {elements of *examples* with *best_attribute* >=
        best_threshold and corresponding target labels}
    subtree ← DECISION-TREE-LEARNING($examples_r$, *targets*)

Table 1: Pseudo code for the decision tree learning algorithm.

The function MAJORITY-VALUE just returns the majority value of the example labels for classification. You need to replace this function with another one for regression. The definition of this function is up to you. The function CHOOSE-ATTRIBUTE measures how "good" each attribute (i.e. feature) in the set is. Several methods can be applied here. You should use the ID3 algorithm, based on information theory. Suppose

the training set contains p positive and n negative examples. Then an estimate on the information contained in a correct answer is:

$$I\left(\frac{p}{p+n},\frac{n}{p+n}\right)=-\frac{p}{p+n}\log_2\frac{p}{p+n}-\frac{n}{p+n}\log_2\frac{n}{p+n}$$

Testing of any attribute A will divide the training set E into subsets $E_1,\ldots,E_\upsilon$ according to their values for A, where A can have υ distinct values (e.g. 2 for binary problems). Each subset E_i has p_i positive examples and n_i negative examples, so going along that branch, $I(\ p_i/(p_i+n_i),\ n_i/(p_i+n_i))$ bits of information will be needed to answer the question. So, on average, after testing attribute A we will need:

$$\mathrm{Re}\,mainder(A)=\sum_{i=1}^{\upsilon}\frac{p_i+n_i}{p+n}I\left(\frac{p_i}{p_i+n_i},\frac{n_i}{p_i+n_i}\right)$$

bits of information to classify the example. The information gain from the attribute test is the difference between the original information requirement and the new requirement:

$$Gain(A)=I\left(\frac{p}{p+n},\frac{n}{p+n}\right)-\mathrm{Re}\,mainder(A)$$

Finally, the attribute that is chosen is the one with the largest gain.
For more information on decision trees, see Chapter 8.1-8.3 of *Duda & Hart's "Pattern Classification"* book.

For your regression tree you can still use entropy as the measure, but you will need to calculate it differently than the equations provided above for classification. It may be a good idea to assign this task to a sub-group of your team. However, again, every group member is expected to understand every aspect of the work.

Implementation

The goal of the assignment is to implement a decision tree algorithm using MATLAB, along with the accompanying functions, as described in the previous section. The inputs of the implemented algorithm should be the set of examples, their respective target labels, and a binary variable that indicates whether it's a classification problem or regression problem: True meaning classification.

You should implement the decision trees from **basic** Matlab functions. In particular, you can ***only*** use functions in the following (very long) list: https://uk.mathworks.com/help/matlab/referencelist.html

Create data

Make sure your data is loaded in the workspace using e.g. the *load* command. This function outputs an array *x*, which is an $N\times D$ array, where N is the total number of examples and D is the number of features in your dataset. You will have different datasets for your classification tree and your regression tree.

Create decision tree

The output of your code should be two decision trees: one for classification and one for regression. You are expected to follow the algorithm shown in Table 1, and implement the ID3 algorithm for the attribute selection. The resulting tree must be a MATLAB structure (*struct*) with the following fields:

- *tree.op* : a label name (string) for the corresponding node (i.e. the name of the attribute that the node is testing). It must be an empty string for a leaf node.
- *tree.kids* : a cell array which will contain the subtrees that initiate from the corresponding node. Since the resulting tree will be binary, the size of this cell array must be 1x2, where the entries will contain the left and right subtrees respectively. This must be empty for a leaf node since a leaf has no kids, i.e. tree.kids = [].
- *tree.prediction* : a label for the predicted class/regression value. This field can have the following possible values:
    - *{0, 1}*: for classification trees, or a real value for regression trees.
    - It must be empty for an internal node, since the tree returns a label only in a leaf node.
- *tree.attribute* : the attribute number tested at this node
- *tree.threshold* : the threshold for the attribute to decide which way to send data points

This tree structure is essential for the visualisation of the resulting tree by using the ***DrawDecisionTree.m*** function, which is provided. Alternatively, you can choose a different tree structure, provided that you also provide a visualisation function for this new structure.

Evaluation

Now that you know the basic concepts of decision tree learning, you can use the clean dataset provided to train your tree, and visualise it using the DrawDecisionTree function. Then, evaluate your decision trees using 10-fold cross validation. You should expect that slightly different trees will be created per each fold, since the training data that you use each time will be slightly different. Use your resulting decision trees to classify your data in your test set. In order to evaluate a tree, you may wish to write a function that takes as inputs the tree you want to evaluate, the input samples and returns the outputs of the trees.

Deliverables

For the completion of this assignment, the following have to be submitted on Moodle:

1. The code used to create the decision trees, including the loading and transforming of the data.

2. A report of up to 1,000 words containing the following:
    - The acquired decision trees for classification and regression.
    - Cross-validation classification results, that include:
        - Recall and precision rates for classification.
        - The F_1-measure derived from the recall and precision rates of the previous step for classification.
        - RMSE for regression

This implies that you will have to write a small script that splits the given dataset into training and test sets. **You must write your own cross-validation code.**

In addition, provide answers to the following questions in your report:

1. Pruning is an important issue in trees. Explain what pruning does and find the node(s) that would be pruned first for one of your learned trees. Explain the difference between the original and the pruned tree.
2. In this assignment, you trained a binary tree. Explain how you would use a single decision tree to learn to predict multiple independent binary class labels. Explain how to make decisions in leaf nodes, and how you would have to change your query search algorithm for any node.

Marking Criteria

Clarity of report: 10%

Code quality: 20%
Implementation correctness: 20%
Parameter setting: 15%
Method evaluation: 15%
Questions: 20%