Machine Learning
Assignment 1: Decision Trees

GROUP 14
ATHARVA NAIK (18CS10067)
AAYUSHI VIDYANTA (18CS30003)

What is a decision tree?

Decision tree learning is a method for approximating discrete-valued target functions, in which the learned function is represented by a decision tree. Decision trees classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance. Each node in the tree specifies a test of some attribute of the instance, and each branch descending from that node corresponds to one of the possible values for this attribute.

In general, decision trees represent a disjunction of conjunctions of constraints on the attribute values of instances. Each path from the tree root to a leaf corresponds to a conjunction of attribute tests, and the tree itself to a disjunction of these conjunctions.

The Decision Tree Algorithm

CART($Examples$, $Target\_attribute$, $Attributes$)

$Examples$ are the training examples. $Target\_attribute$ is the attribute whose value is to be predicted by the tree. $Attributes$ is a list of other attributes that may be tested by the learned decision tree. Returns a decision tree that correctly classifies the given $Examples$.

- Create a $Root$ node for the tree
- If all $Examples$ are positive, Return the single-node tree $Root$, with label $=+$
- If all $Examples$ are negative, Return the single-node tree $Root$, with label $=-$
- If $Attributes$ is empty, Return the single-node tree $Root$, with label $=$ most common value of $Target\_attribute$ in $Examples$
- Otherwise Begin
    - $A \leftarrow$ the attribute from $Attributes$ that best* classifies $Examples$
    - The decision attribute for $Root \leftarrow A$
    - For each possible value, v_i, of A,
        - Add a new tree branch below $Root$, corresponding to the test $A = v_i$
        - Let $Examples_{v_i}$ be the subset of $Examples$ that have value v_i for A
        - If $Examples_{v_i}$ is empty
            - Then below this new branch add a leaf node with label $=$ most common value of $Target\_attribute$ in $Examples$.

- Else below this new branch add the subtree
  ID3($Examples_{v_i}$, $Target\_attribute$, $Attributes - \{A\}$)
- End
- Return $Root$

---

*The best attribute is the one with lowest average ***gini index*** averaged over all it's children

$Gain(S, A)$ of an attribute A ,relative to a collection of examples S

$$AvgG(S, A) \equiv \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Gini(S_v)$$

If the target attribute can take on c different values, then the entropy of S relative to this c-wise classification is defined as

$$Gini(S) \equiv 1 - \sum_{i=1}^{c} p_i^2$$

Pruning

The validation set can be used to prevent overfitting by an approach, called *reduced-error pruning* (Quinlan 1987), is to consider each of the decision nodes in the tree to be candidates for pruning. Pruning a decision node consists of removing the subtree rooted at that node, making it a leaf node, and assigning it the most common classification of the training examples affiliated with that node.
Nodes are removed only if the resulting pruned tree performs no worse than the original over the validation set. This has the effect that any leaf node added due to coincidental regularities in the training set is likely to be pruned because these same coincidences are unlikely to occur in the validation set. Nodes are pruned iteratively, always choosing the node whose removal most increases the decision tree accuracy over the validation set. Pruning of nodes continues until further pruning is harmful (i.e., decreases accuracy of the tree over the validation set).

Rule Post Pruning

Rule post-pruning involves the following steps:

1. Infer the decision tree from the training set, growing the tree until the training data is fit as well as possible and allowing overfitting to occur.
2. Convert the learned tree into an equivalent set of rules by creating one rule for each path from the root node to a leaf node.
3. Prune (generalize) each rule by removing any preconditions that result in improving its estimated accuracy.
4. Sort the pruned rules by their estimated accuracy, and consider them in this sequence when classifying subsequent instances.

About the Dataset

DATASET : EEG Steady-State Visual Evoked Potential Signals Data Set
Download link:
https://archive.ics.uci.edu/ml/datasets/EEG+Steady-State+Visual+Evoked+Potential+Signals

There are 5 different tests performed on 30 subjects:

- SB1 - Five Box Visual Test 1
- SB2 - Five Box Visual Test 2
- SB3 - Five Box Visual Test 3
  (There are three different Five Box tests, these are not repetitions of the same test)
- SV1 - Visual Image Search
- SM1 - Motor Images (Hand Shake Experiment)

Attribute information: In this dataset, the tests are visual experiments. The time series .csv files contain 16 attributes, of which the last 14 are the signals coming from the electrodes. The first two are the time-domain and a signal called 'interpolated' which is normally 0. Each file represents a test performed on a particular subject. Each subject undergoes different tests which are provided in .csv format as follows: suppose you have a .csv whose name is A001SB1_1 This means the data corresponds to group A (only Group A is provided at present), subject 001, Test SB1 (Five Box Visual Test), and first experiment (_1).

Implementation Details

GROWING THE TREE

We carry out the tree building recursively, and for the root of every subtree, if the instances are impure, we determine the best attribute and corresponding threshold for diving the instances as follows:
We use the ***gini index*** to determine the best attribute and threshold for the split.
Gini index g for a probability distribution D is: $g = 1 - \sum_{s \epsilon D} s^2$

1. If we have reached the maximum depth we take the majority class among the instances as the predicted class. Also if the data is pure we create a leaf node and terminate that branch
2. For each attribute we sort the data according to that attribute and assume the candidate thresholds to be the average of the members in all n-1 (n is the number of datapoints) adjacent pairs, and determine the pair that leads to best split for that attribute, by minimizing the ***gini index***.
3. We do this over all attributes to get the best possible attribute (by comparing the best thresholds for each), and the best threshold for that attribute becomes the selected threshold.

PRINTING THE TREE

We use the python ***pydot*** module to print the tree. The tree is represented by
graph = pydot.Dot(graph_type='graph')
The tree is passed as a dict of dict, and for each node, each corresponding edge is added by
edge = pydot.Edge(parent_name, child_name)

graph.add_edge(edge)

RESULTS

10 seeded splits of 60:20:20 (train:val:test) are used (generated by shuffling). We use the train and test set to find the best depth, with the following results.

Depth value	Accuracy averaged over 10 splits
2	0.16785714285714287
3	0.175
4	0.18214285714285713
5	0.16071428571428573
6	0.1714285714285714
7	0.17857142857142855
8	0.17857142857142855
9	0.17142857142857143
10	0.1714285714285714
11	0.16785714285714284
12	0.16785714285714284
13	0.16785714285714284
14	0.16785714285714284
15	0.16785714285714284
16	0.16785714285714284

For the splits tested 4 was marginally the best, with accuracy averaged over 10 splits being 0.18214285714285713. The best split (for a tree of depth 4) and the corresponding tree are picked. The best tree has accuracy of 0.2857142857142857 on the test set.

PRUNING THE TREE

For pruning we use reduced error pruning, pruning the tree in a top down manner, collapsing the nodes when the validation accuracy improves (i.e. validation set accuracy is used as an estimate of the test set accuracy).

RESULTS AFTER PRUNING

For the best tree of depth = 4

(In this case pruning doesn't improve test accuracy, as tree is very small and dataset only has 28 points, so we cover a non optimal case where pruning has significant performance gains (depth=5))

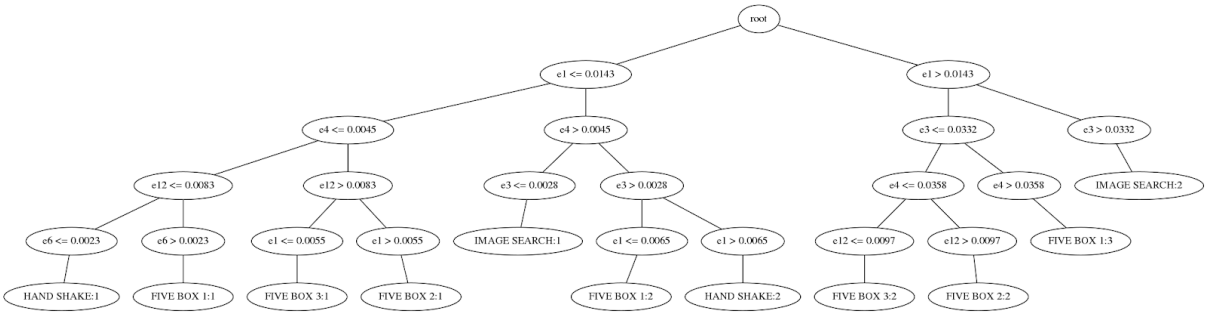
Validation accuracy before pruning = 0.17857142857142858
Test accuracy before pruning = 0.2857142857142857
Validation accuracy after pruning = 0.21428571428571427
Test accuracy after pruning = 0.2857142857142857

For example case of depth = 5

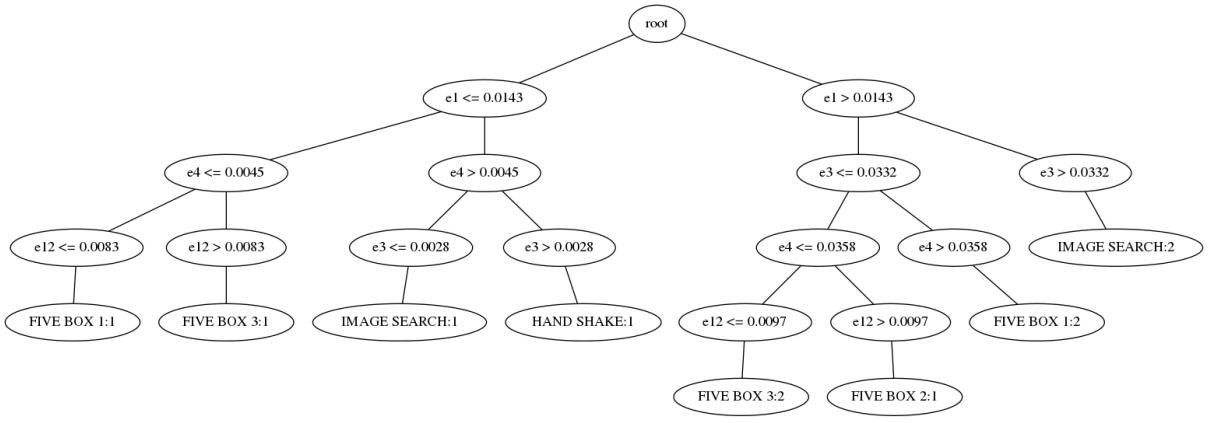
Validation accuracy after pruning = 0.21428571428571427
Test accuracy before pruning = 0.17857142857142858
Validation accuracy after pruning = 0.2857142857142857
Test accuracy after pruning = 0.25

FIGURES:

The unpruned decision tree



The pruned decision tree



The accuracy vs depth values