3 Decision Trees

3.1 Implementation:

The flow goes like this:

- Reading and storing the data
- Splitting the data into training and test sets.
- Creating a decision tree instance.
- Learning phase of the decision tree.
  - Assign the depth of the tree(for performance)
  - Building the decision tree
    - If the depth is above the max depth, return a leaf
    - Get the best split of the data
      - Assign initial information gain to negative maximum
      - Iterate over each index in the training set
      - Separate the left tree data and right tree data
      - Get the information gain from the data obtained.(Gini/Entropy)
      - Pick out the split with max information gain and return it.
    - create a decision node with the data obtained from best split.
    - Recursively call build tree on left and right data as splitted in best split.
- Evaluating the result on the test set using classify method.
  - If the node is leaf, return the value of the leaf.
  - Choose the tree(left/right) based on the data present in the decision node.
  - Recursively go through the choosen tree until we hit a leaf node.

The Entire code is in the file named **cs19btech11039.py** All the initial code and improved code is in the same file. Refer to note.txt for the improved code line numbers.

3.2 K-fold cross validation:

3.2.1 Splitting data Initial:

- Add into training set if index of row in set is not a multiple of 10
- Add into testing set if index of row in set is a multiple of 10
- Train and test on the data.

3.2.2 Splitting data Final:

- Loop over 0 to K(Given, K = 10) and label as rem(remainder)
- In each iteration, create a tree instance, and split the data
- Add into the training set if index of row in set doesn't leave the remainder as rem
- Add into the testing set if index of row in set leaves a remainder as rem
- Essentially, we get the required split.
- Train the model using decision tree algorithm using training set
- Test the model using test set.
- Record the accuracy and decide. Add it to the global accuracy for getting average of K accuracies.
- Average the global accuracy to get an estimate accuracy.

Below is the implementation:

```
accuracy = 0
K = 10
for rem in range(0,K):
    training_set = [x for i, x in enumerate(data) if i % K != rem]
    test_set = [x for i, x in enumerate(data) if i % K == rem]

    tree = DecisionTree()
    # Construct a tree using training set
    tree.learn(training_set)

    # Classify the test set using the tree we just constructed
    results = []
    for instance in test_set:
        result = tree.classify(instance[:-1])
        results.append(result == instance[-1])

    # Accuracy
    kaccuracy = float(results.count(True))/float(len(results))
    print(rem,"completed with accuracy ",kaccuracy)
    accuracy += kaccuracy

accuracy /= K
print("accuracy: %.4f" % accuracy)
```

The same can be found in the python files attached.

3.3 Improvements:

3.3.1 Initial Implementation Detials:

- Depth = No limit, Time taken is too much as it is over fitting.
- Average Accuracy: 0.79 (using Entropy)
- Average Accuracy: 0.82 (using Gini)
- Time taken for 1 fold: Nearly 1.5 mins
- Time taken on average for all sets: 13-14 mins

This is too much cost on a data with 5000 data points. So the improvements done are:

- Limiting the depth of the tree
- Using Gini Index

3.3.2 Depth of the tree(observed on entropy):

Initially, there is no limit on the depth of the tree.
So, as a result the model became overfitting with the data set.

So, added a condition in build tree which directs the program to build sub trees if and only if the present depth of the tree is less than the maximum depth assigned.

Now, the model does only certain number of computations and builds a minimal version of the previously built over fitting tree.

Observations:

- Depth: Assigned as number of features.
  - Average Accuracy: 0.79 (same as that of the over fitting tree) with minimal changes.
  - Time taken for 1 fold: 1 min
  - Average time taken: 8-9 Mins

  Better than the over fitting tree as there is no much difference in the final accuracy and time decreased by a considerable extent.

- Depth: Assigned as half the number of features.
  - Average Accuracy: 0.78(same as that of the over fitting tree) with very minimal changes on a scale of 0.000001.
  - Time taken for 1 fold: 42 sec
  - Average Time taken: 4-5 mins

  Better than the improved tree as there is no much difference in the final accuracy and time decreased by a considerable extent.

- Depth: Assigned as a quarter of number of features.
  - Average Accuracy: 0.78(same as that of the improved fitting tree) with very minimal changes on a scale of 0.000001.
  - Time taken for 1 fold: 25-30 sec
  - Average Time taken: 2 mins

  Better than the improved tree as there is no much difference in the final accuracy and time decreased by a considerable extent.

- Depth: Less than 1/4 of number of features
  - Time is improving.
  - But the problem is the fall in accuracy.

So finally, the over fitting model is improved by limiting the depth of the tree. Finally the best depth is recorded for the value of depth as **(numFeatures)/4**

3.3.3 Gini Index:

Using Gini Index and impurity concept increased the accuracy slightly on a scale of 0.1 which is highly preferred minimum reduce of change in cost of computation.

- Accuracy: 0.8 and above
- Time taken: Very slightly better than Entropy.

Reasons:

- **cost:**
  - The computation cost of Gini index is slightly better than that of entropy.
  - Because, the calculation of entropy involves complex logartihmic in built function but on the other hand, gini index just uses squaring of probability.
- **Feature Selection:**
  - The entropy of node can vary from 0 to 1
  - The Gini Index of node can vary from 0 to 0.5
  - The entropy has a wider range of values than gini index.
  - Given, that entropy has a wider range, the values are more discrete for Entropy than the gini index.
  - Therefore, it is better to have gini index as a criterion for information gain.

  So both the reasons played part to make Gini index criterion better than the entropy.

These are the two improvements made for the initially built decision tree.

LaTeX generated document

THE END