CS 771: Introduction To Machine Learning
Assignment 2

Akshat Agarwal
200081
akshatag20@iitk.ac.in

Muskan Kumari
200610
muskank20@iitk.ac.in

Navya Ratnan
200627
nratnan20@iitk.ac.in

Sejal Sahu
200911
sejals20@iitk.ac.in

Yash Goel
201142
yashgoel20@iitk.ac.in

Answer 1

Entropy Maximization
In this approach the decision tree algorithm was devloped using Entropy Maximization. This approach involves selecting a probability distribution that has the highest entropy (i.e., the most balanced tree) subject to a set of constraints.
The idea is to find the probability distribution that is the most unbiased or uncertain, given the available constraints or information. The approach involves constructing a model that assigns probabilities to each possible class given a set of features. During training, the model learns the best set of parameters that maximize the likelihood of the data (in this case, minimize the query rate) while satisfying the given constraints.
Consider the set S of n elements of C type, with n_c elements of type c, then we get its entropy as follows

$$H(S) = -\sum_{c \epsilon [C]} \frac{n_c}{n} log_2 \left(\frac{n_c}{n} \right)$$

The advantage of the maximum entropy approach is that it allows for the incorporation of prior knowledge or constraints into the model. Entropy was calculated for each possible query. The query which gives the maximum entropy was selected and its query index was used. This provided the best of queries that fetched the most balanced decision tree.

At the root node the splitting was based on the number of letters in the query word. In the subsequent levels, the criterion used for splitting the nodes was the intersection of the query and the words in the dictionary. We stop only when the minimum leaf size or the maximum depth of the decision tree has been reached.

Results obtained:

Hyperparameter K	Run time	Model Size	Query Rate
10	0.988851546600199	1311445.6	5.7736017031159275
50	2.672666549800488	1178869.4	4.860034836462163
100	4.698782605600354	1151685.2	4.649041997290498

Depth Reduction
Since entropy calculation requires us to visit each mask, it is time intensive, and thus shoots up the time taken to train the model. Hence we tried a different approach to reduce depth of the tree. we

greedily considered the tree with least number of splits at each node which is represented by length of split_dict in the code. This approach reduced time drastically and also reduced query rate better than entropy maximization approach.

Choosing similar random query words at each node
Instead of choosing random query words (to train ad the model) from the whole dictionary, we instead chose them from the "node words bucket" represented by (my_words _idx in the code), this reduced training time and model size drastically as query word selection happened from more similar words instead of the whole dictionary.

Results obtained:

Hyperparameter K	Run time	Model Size	Query Rate
10	0.4881703855990054	987149.8	4.342906909231663
50	1.4832324425995467	981510.2	4.216024772595317
100	2.250566706400059	982517.6	4.2109928391716664

Hyperparameter K
The hyperparameter K represents the number of query words passed on to the model at a time. Each query word is used as the criterion to split the node (intersection of the query word with the words in the dictionary). Increasing the value of K results in the increase of training time and a reduction of model size.

Effect of History
The use of history in the entropy maximization algorithm involves analyzing the past performance of the decision tree in order to identify the most informative features. This involves looking at the past decisions made by the decision tree, and using this information to refine the decision tree in order to make better decisions in the future. It was observed that including history in the code gave runtime 20s while with it excluded the runtime reduced considerably to 2s, i.e. up to ten times.

Lookahead
To improvise upon *maximum entropy* approach we used *lookahead*. Lookahead was implemented, while choosing the query index in 3 levels. The query with maximum entropy at the end of 3 levels was selected to be the output query, and was fed into the Merlin Bot. Then based on the response, a loop was generated which goes on henceforth.
After implementation of lookahead, it was observed that the query rate and training time decreased slightly and the model size increased, however the winrate was not affected much. Since this approach did not give much improvement in the runtime and query rate, it was excluded.

Balanced Tree Regularization
Balanced tree regularization is the technique in decision tree construction that helps to improve the generalization performance of the model and prevent overfitting to the training data. By encouraging the tree to be more balanced, the regularization term can help to limit the depth and complexity of the tree, leading to a simpler and more interpretable model.
The balanced tree regularization term used is as follows

$$R(S) = -\sum_{c\epsilon[C]} C * \left(\frac{d_c}{d}\right)^2$$

Here C denotes the hyperparameter used to adjust the weight of penalty term, d_c depth of the current node, and d is the maximum depth of the decision tree.
The balanced tree regularization term was added to the Entropy function used in the tree construction algorithm. While the entropy function was a measure of the quality of a split at a given node, and was typically based on the reduction in impurity that results from the split, by adding a penalty term to the the entropy function that depends on the imbalance of the resulting split, the algorithm was encouraged to choose splits that lead to more balanced trees.

Answer 2

1)

Our my_fit method is able to finish training around in 0.3 sec

2)

On-disk size of our learnt model is around 900000

3)

4.4 queries our model make per round on average.