CS771A: Assignment 2

Technocrats

April 5, 2023

Group members

Aakarsh Mittal	200002
Aditya Jain	200045
Mayank Pushpjeet	200572
Sushmita	201027
Yash Gupta	201143

Methodology

We have adopted the **Decision Tree Algorithm** as our preferred **Methodology**.
To implement it, we have used **the Hangman problem** Code provided by the **course instructor** .
Using this code as a reference, we developed an **effective approach** for constructing our Decision Tree.

Functions and Classes used to Construct Tree

We implemented the Decision Tree algorithm and defined a **Tree Class** with the following functions:

1. `fit`
2. `_init_`

Similarly, in the **Class node**, we have defined the following functions:

1. `get_query`: which is a `get_query()` A method has been defined that **specifies the query that Melbo** should ask at a **given node.**
2. `get _child`. This returns **the child node** at every **tree node** leaving the **leaf nodes.**
3. `reveal`. It will match the alphabets between the **query** and the **word** and return the mask in which the places at which **alphabets are common between two alphabets** are placed in mask otherwise **' - '** is added.

Try_attr

The function we have employed is responsible for **creating different splits at various nodes** in the **decision tree**. Specifically, it utilizes **the masks generated** by the **reveal_function** to **construct a dictionary** and then generates a **list** for each **corresponding mask**. The **most suitable dictionary** is then selected based on the **reduction in entropy**.

Process_node

It selects the **best-split dictionary** based on **maximum entropy reduction** Other functions, such as **try_attr** and **entropy**, are called in this function, and it returns **best_attr** and **best_split_dict**.

Get_entropy

This uses the given **entropy formula** to compute the **entropy based on relative observation frequencies.**

$$entropy = -\sum_{x} p(x) \log p(x)$$

Here *p(x)* is a fraction of examples in a given class

Fit function

This function runs **recursively** and creates the **decision tree** using the **other functions** described above, taking **all_words,max_depth**, etc. as input, and returns us the **tree**.

Implementation

1. Our code is based on the **Decision Tree Algorithm**, where we trained the model using the **complete dictionary**.
2. To split **our dictionary**, we begin from the **root node** and move toward the **next node** based on the **size of the words.**
3. We start with a set of words ranging from **4 to 15** and then group words of **similar sizes into nodes**. Within each node, we aim to identify the word that results in the **maximum reduction in entropy** by running a nested for loop on the words of the same size. This involves finding the **word** that provides the **maximum information gain or the minimum entropy**.
4. With the help of the **reveal function**, we get the mask in which some alphabets we found by **running query** get displayed. We then make a split dictionary based on these **new masks** (example _a_,__t _etc different words get sorted)
   This method runs **recursively** until we guess the **correct word.**

To Stop Expanding
We have set the max depth, which is **15** in our case, so our **node will not split further** if the **maximum depth is reached**.

Pruning

1. In the **try_attr** function, rather than iterating over the whole **my_words_idx**, we can iterate over **60-80%** of the **indexes to reduce model size.**
2. The fit function takes **all, max_depth**, etc., as input and returns the **decision tree.**
3. To prevent **overfitting**, the maximum depth of the tree has been **set to 15 in the implementation**.

Hyperparameters

1. The **maximum depth of the decision tree** is a hyperparameter that can be tuned to improve the **model's performance.**
2. Other hyperparameters that can be tuned include the criteria for splitting nodes, such as the **maximum number of features to consider at each split**, **the minimum number of samples required to split a node, and the minimum number of samples required to be at a leaf node.**
3. These hyperparameters can be optimized using techniques such as **cross-validation** to find the best values that result in **the highest accuracy on the validation set.**