CS771A Assignment 2

The Boys

Rajarshi Dutta
200762

Udvas Basak
201056

Shivam Pandey
200938

1 Solution 1

1.1 Theory

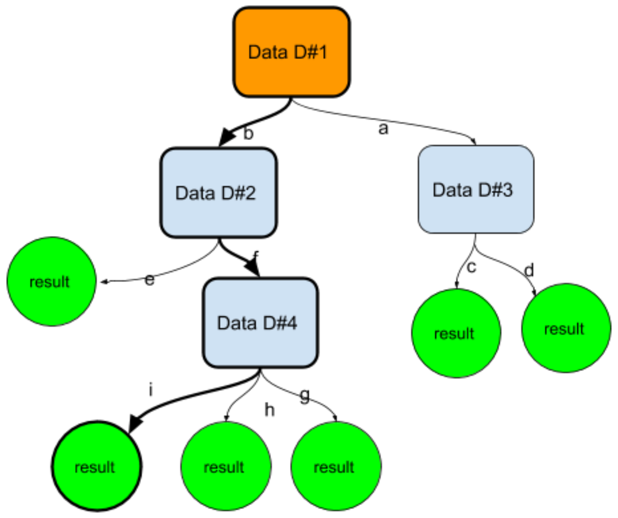

Figure 1: Illustration of a working ID3 algorithm

We have used the **Iterative Dichotomizer** decision tree algorithm which is considered one of the most robustly used algorithms for supervised machine learning classification tasks. The algorithm works by recursively partitioning the training data based on their attributes until the decision tree produces the purest splits (referred to as the **leaves**). ID3 uses a **top-down greedy approach** to build a decision tree. The tree is basically constructed from the top and the greedy approach means that at each iteration the best feature is selected to split the node.

1.2 Our approach

The decision tree constructed to solve the given **Wordle-Solvr** problem consists of the following components:

1.2.1 process_node

- For the **process_node** function, every **non-root** node present in the **ID3** tree constructed is considered and passed through the **get_entropy** function which calculates the index of the vocabulary that minimizes **entropy** the most or maximizes the **Information Gain**. This index can be used to access the most appropriate query that can be used at that step.

- All words from list **all_words** list are extracted and **reveal** function helps to uncover the mask between the **query** and the required word which is stored in **split_dict** as the key. The value of the **split_dict** contains an array of indices that returns the given mask when queried.

get_entropy

- The **get_entropy** function iterates over all of the words present in that node (possible candidate queries) and passes the array to the function **calc_entropy** which returns the entropy of the current node.
- Another dictionary is maintained which given the mask, corresponds to a number of queries (which when queried with the word provides that particular mask). These values are used to calculate the sum of weighted entropy after splitting the node.
- The difference of the weighted sum of entropies after splitting the node with the initial entropy calculated from the parent node gives us the **information gain** used to produce the best split out of all words.

$$G(S,A) = H(S) - \sum_{V \in Values(A)} \frac{S_v}{S} H(S_v) \tag{1}$$

  where, V = possible values in Attribute A,
  S = set of examples X,
  S_v = subset where X_A = V
- Information gain indirectly describes the mutual information between the attribute and the class of labels S.

calc_entropy

- This function simply calculates the **Shannon Entropy** based on the formula where p represents the probability of each class label which can be produced by the corresponding split based on their attribute.

$$E(p) = -p \cdot \log_2(p) \tag{2}$$

2 Solution 2

The entire algorithm has been implemented in the **submit.py** file.