Assignment 1: Apriori Algorithm

2019094511 김준표

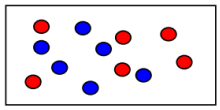
1. Summary of algorithm

해당 과제에서는 gain ratio 방식을 이용하여 decision tree를 생성했다.

Gain ratio 방식:

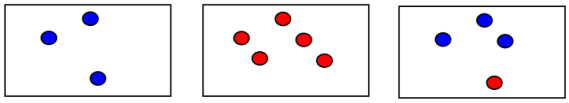
이는 특정 attribute에 대해 information gain을 구한다. Information gain은 분류 전 집단에서 entropy를 구하고, 기준 attribute에 대해 분류 후 entropy를 구한 뒤 둘의 차이를 반환한 값이다.

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$



$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

$$Gain(A) = Info(D) - Info_A(D)$$

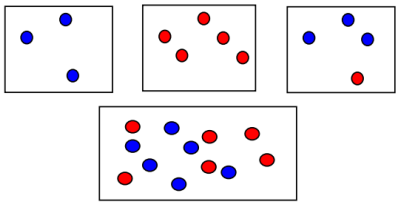


Gain ratio은 위의 방식으로 구한 information gain에서 attribute마다 가지는 value set의 요소 수가 달라 발생하는 bias를 보정하기 위한 방식이다. 이는 이전에 구한 gain 값을 split info값으로 나눠 반환한다.

- **Gain Ratio** takes the number and size of branches into account when choosing a feature

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right)$$

$$Gain(A) = Info(D) - Info_A(D)$$



$$\boldsymbol{GainRatio}(A) = Gain(A) \,/\, SplitInfo_A(D)$$

2. Code description

main 함수에서는 input file로부터 받은 train file과 test file을 data frame에 넣어준다. 이때, csv 형식으로 읽고 쓰도록 한다. 이후 decision tree를 생성하고 분류를 실시한 뒤 output으로 result file을 작성한다.

```python
def main():
    train_file = sys.argv[1]
    test_file = sys.argv[2]
    result_file = sys.argv[3]

    train_df = pd.read_csv(train_file, sep='\t')
    test_df = pd.read_csv(test_file, sep='\t')

    dt = DecisionTree(train_df)

    df = dt.classify(test_df)
    df.to_csv(result_file, index=False, sep='\t')


if __name__ == "__main__":
    main()

```

Node class는 decision tree의 노드를 저장하는 구조체로 해당 노드의 attribute, leaf 여부, label, childe node를 저장한다.

```python
class Node:
    def __init__(self, attr, isLeaf = False, label = None):
        self.attr = attr        # DB의 카테고리 (ex: attr 1, attr 2, ... , attr n)
        self.isleaf = isLeaf   # 노드 속성 (ex: leaf node/internal node)
        self.label = label      # 데이터의 최종 분류 기준 (ex: yes/no)
        self.child = dict()
```

DecisionTree class는 tree를 초기화하고, gain ratio를 계산하고, 이를 바탕으로 decision tree를 생성하고, data frame을 받아와 label을 분류해주는 메소드로 이루어져있다.

```python
    '''
    Decision tree을 초기화한다.
    '''
    def __init__(self, df):
        # Data frame의 마지막 column에 label 정보가 담겨있음
        self.label = df.columns[-1]
        # Data frame의 attribute와 attribute의 value set들 추출하기
        self.attrs = {attr: df[attr].unique() for attr in df.columns if attr!=self.label}
        # Decision tree의 root 노드
        self.root = self.create_dt(df)
```

Decision tree는 label (최종 결정), attribute와 value set 집합의 딕셔너리, root 노드를 갖는다.

```python
    '''
    Data frame의 label 값들을 분류하여 각 값이 나올 확률을 구한다.
    '''
    def probability(self, df):
        values = df[self.label]
        total_cnt = len(values)
        return [float(value)/float(total_cnt) for value in values.value_counts()]

    '''
    확률을 받아와 entropy를 구한다.
    '''
    def entropy(self, prob):
        return sum(-1.0 * (p *math.log(p + 1e-15,2)) for p in prob if p)
```

entropy 계산은 알고리즘 소개의 공식을 따른다. 계산을 위해 data frame의 label (최종 결정)값을 분류하여 각 label이 나올 확률을 계산하는 함수를 추가해준다.

```
    '''
    특정 attribute를 인자로 받아 해당 attribute의 information gain을 구한다.
    '''
    def information_gain(self, df, attr):
        info_D = self.entropy(self.probability(df))
        info_A = 0.0
        for a in self.attrs[attr]:
            # 속성이 a인 row들을 뽑음 (ex: attr(income)이 a (high, medium, low)인 애들을 각각 뽑음)
            tmp_df = df[df[attr] == a]
            # 각각에 대해 entropy를 계산하고 가중치를 곱해줌
            info_A += (len(tmp_df) / len(df)) * self.entropy(self.probability(tmp_df))

        gain = info_D - info_A
        return gain

    '''
    Split information 값을 계산한다.
    '''
    def split_info(self, df, attr):
        splitInfo = 0.0
        for a in self.attrs[attr]:
            # 속성이 a인 row들을 뽑음
            tmp_df = df[df[attr] == a]
            p = len(tmp_df) / len(df)
            splitInfo += (-1.0) * p * (math.log(p + 1e-15,2))

        return splitInfo

    '''
    Information gain과 split information을 통해 gain ratio를 구한다.
    '''
    def gain_ratio(self, df, attr):
        gain = self.information_gain(df, attr)
        splitInfo = self.split_info(df, attr)

        return gain / splitInfo
```

Gain ratio 역시 알고리즘을 따라 작성했다. Data frame을 받아와 특정 속성에 대해, 해당 속성의 value set을 따라 data frame을 나눠 entropy를 계산하고 이를 통해 information gain을 구하고 split info 값을 구하여 gain ratio를 구한다.

```
    '''
    Decision tree를 생성한다.
    '''
    def create_dt(self, df):
        # Data frame이 분류가 완벽하여 label이 한가지만 존재하는 경우
        # Label을 정한다.
        if df[self.label].nunique() == 1:
            label = df[self.label].unique()[0]
            return Node(None, True, label)

        # 분류 방법으로 gain ratio를 이용한다.
        result_dict = {attr: self.gain_ratio(df, attr) for attr in df.columns if attr!=self.label}
        selected_attr = sorted(result_dict.items(), key=lambda x: x[1], reverse = True)[0][0]

        # 노드를 생성하고 선택된 속성을 기준으로 split한다.
        node = Node(selected_attr, False, None)

        a = self.attrs[node.attr]
        for i in range(len(a)):
            tmp_df = df[df[node.attr].values == a[i]]
            tmp_df = tmp_df.drop(columns=node.attr,axis=1)
            if (len(tmp_df) > 0):
                node.child[a[i]] = self.create_dt(tmp_df)

        return node
```

Decision tree 생성은 data frame의 모든 attribute에 대해서 gain ratio를 적용해보고 이를 정렬하

여 gain ratio의 반환 값이 제일 커지는 attribute를 골라 해당 attribute를 기준으로 노드를 split하고 child를 생성한다.

```
    '''
    Data가 주어진 경우 해당 data의 leaf node를 찾는다.
    '''
    def find_leaf(self, data):
        node = self.root
        a = data[node.attr]
        while not node.isleaf:
            for (child, next_node) in node.child.items():
                if a in child:
                    break
            node = next_node
            if node.attr:
                a = data[node.attr]

        return node.label

    '''
    Data frame을 받아서 label을 분류해준다.
    '''
    def classify(self, df):
        label = [self.find_leaf(df.loc[i]) for i in range(len(df))]
        df[self.label] = label
        return df
```

분류 작업은 생성된 decision tree의 노드를 따라 attribute를 기준으로 값을 비교하고 leaf node까지 순회하여 최종 label을 반환한다.

3. Compiling method

실행환경:

OS: Windows 10

Python version: Python 3.8.3

실행방법:

```
PS C:\Users\pyo99\CSE\DS\2022_ite4005_2019094511\Assignment#2>
PS C:\Users\pyo99\CSE\DS\2022_ite4005_2019094511\Assignment#2> python ./dt.py dt_train.txt dt_test.txt dt_result.txt
PS C:\Users\pyo99\CSE\DS\2022_ite4005_2019094511\Assignment#2>
```

python dt.py train_file_name.txt test_file_name.txt result_file_name.txt

위의 실행 명령어를 입력하면 아래와 같이 test.txt에 대해 result.txt 파일이 생성되는 것을 확인할 수 있다.

dt_test.txt

```
2022_ite4005_2019094511 > Assignment#2 > dt_test.txt
1  age income  student credit_rating
2  <=30    low no  fair
3  <=30    medium  yes fair
4  31...40 low no  fair
5  >40 high    no  fair
6  >40 low yes excellent
7
```

dt_result.txt

```
2022_ite4005_2019094511 > Assignment#2 > dt_result.txt
1  age income  student credit_rating   Class:buys_computer
2  <=30    low no  fair    no
3  <=30    medium  yes fair    yes
4  31...40 low no  fair    yes
5  >40 high    no  fair    yes
6  >40 low yes excellent   no
7
```

4. Other specification

dt_test.exe 결과

```
PS C:\Users\pyo99\CSE\DS\2022_ite4005_2019094511\Assignment#2>
PS C:\Users\pyo99\CSE\DS\2022_ite4005_2019094511\Assignment#2> ./dt_test.exe dt_answer.txt dt_result.txt
5 / 5
PS C:\Users\pyo99\CSE\DS\2022_ite4005_2019094511\Assignment#2>
```

첫번째 예제인 buy computer에 대해서 5 / 5의 결과를 확인할 수 있었다.

```
PS C:\Users\pyo99\CSE\DS\2022_ite4005_2019094511\Assignment#2>
PS C:\Users\pyo99\CSE\DS\2022_ite4005_2019094511\Assignment#2> ./dt_test.exe dt_answer1.txt dt_result1.txt
317 / 346
PS C:\Users\pyo99\CSE\DS\2022_ite4005_2019094511\Assignment#2>
```

두번째 예제인 car evaluation에 대해서 317 / 346의 결과를 확인할 수 있었다.