24-2 ML HW1

Decision Tree 구현

컴퓨터공학부
202246109
김기현

목적

Information Gain이 적용된 Decision tree 구현

데이터셋 설명

1. playtennis.csv (D1 ~ D14)

Outlook	Temperature	Humidity	Wind	Play Tennis
Rainy	Hot	High	FALSE	No
Rainy	Hot	High	TRUE	No
Overcast	Hot	High	FALSE	Yes
Sunny	Mild	High	FALSE	Yes
Sunny	Cool	Normal	FALSE	Yes
Sunny	Cool	Normal	TRUE	No
Overcast	Cool	Normal	TRUE	Yes
Rainy	Mild	High	FALSE	No
Rainy	Cool	Normal	FALSE	Yes
Sunny	Mild	Normal	FALSE	Yes
Rainy	Mild	Normal	TRUE	Yes
Overcast	Mild	High	TRUE	Yes
Overcast	Hot	Normal	FALSE	Yes
Sunny	Mild	High	TRUE	No

Feature: Outlook, Temperature, Wind

2. playtennis2.csv (D1 ~ D212)

Playtennis.csv 에서 확장된 데이터

Decision Tree란?

Decison Tree는 분할 정복 개념과 밀접한 관련이 있으며 질문과 추측의 집합을 트리 형식으로 작성 할 수 있습니다.

Information Gain란?

데이터의 분할로부터 얻어지는 불순함의 정도를 측정해 얻는 이득입니다. 불순함이 줄어들수록 정보 이득이 커지며 학습모델은 정보 이득이 가장 큰 feature를 선택하여 분할합니다.

$$Gain(D, A) \equiv Entropy(D) - \sum_{v \in Values(A)} \frac{|Dv|}{|D|} Entropy(Dv)$$

$$Entropy(D) \equiv E_p\left[\log \frac{1}{p_i}\right] = \sum_{i=1}^{c} - p_i \log p_i$$

Information Gain 구현 코드

```python
# 엔트로피 계산 log2로 계산!!
def entropy(data):
    label_counts = Counter()    # 레이블 수 계산
    for row in data:
        label_counts[row[-1]] += 1

    entropy = 0
    for count in label_counts.values():
        p = count / len(data)   # 레이블의 비율
        entropy -= p * math.log2(p)
        # print(entropy)
    return entropy
```

$$\sum_{i=1}^{c} - p_i \log p_i$$

각 feature들의 결과값의 비율을 구해 엔트로피를 계산합니다.

```python
def information_gain(data, feature_index):
    total_entropy = entropy(data)   # 전체 엔트로피
    feature_values = []

    for row in data:
        feature_values.append(row[feature index])   # 특정 feature의 값들

    value_counts = Counter(feature_value
    feature_entropy = 0

    for value, count in value_counts.items(): # 특정 feature의 값들에 대한 엔트로피
        subset = []
        for row in data: # 특정 feature의 값들에 대한 데이터
            # print(row[0])
            if row[feature_index] == value:
                subset.append(row)
        # 특정 feature의 값들에 대한 엔트로피
        feature_entropy += (count / len(data)) * entropy(subset)

    # print(f"Gain({header[feature_index]}): {total_entropy - feature_entropy:.3f}")
    return total_entropy - feature_entropy
```

$$Entropy(D)$$

$$\sum_{v \in Values(A)} \frac{|Dv|}{|D|} Entropy(Dv)$$

데이터를 feature 기준으로 분할한 후 각각의 subset에 대해 엔트로피를 계산한 후 이를 기반으로 Gain을 얻습니다.

decision_tree_train 구현 코드

```
# 결정 기준
...
class Node:
    def __init__(self, feature, branch):
        self.feature = feature
        self.branch = branch

# 예측 값
...
class Leaf:
    def __init__(self, value):
        self.value = value
def decision_tree_train(data, features):
    labels = []
    for row in data:
        labels.append(row[-1])

    most_common_label = Counter(labels).most_common(1)[0][0] # 가장 많은 레이블 -> 예측값

    if len(set(labels)) == 1:
        return Leaf(most_common_label) # 모든 레이블이 같을 때
    if not features:
        return Leaf(most_common_label) # 사용할 feature가 없을 때

    best_feature, best_feature_index = select_best_feature(data, features)
    groups = split_data(data, best_feature_index)   # best feature로 데이터 분할
    remaining_features = features - {best_feature} # 사용한 feature 제외 (나머지 feature)

    branch = {}
    for value, subset in groups.items():
        branch[value] = decision_tree_train(subset, remaining_features) # 재귀적으로 트리 생성

    return Node(best_feature, branch)
```

데이터가 모두 동일한 레이블을 가질 경우 해당 레이블을 리프 노드로 반환합니다.

Information Gain을 기준으로 best feature를 선택합니다.

선택된 feature를 기준으로 데이터를 분할 후, 각 subset에 대해 다시 학습을 진행하여 트리를 확장합니다.

코드 실행 방법

```
python3 202246109_ML_hw1.py playtennis.csv
```

Output: If then 형식으로 표현된 학습된 트리 구조

```
python 202246109_ML_hw1.py playtennis.csv
if Outlook is Rainy
        if Humidity is High
                then Play Tennis = No
        if Humidity is Normal
                then Play Tennis = Yes
if Outlook is Overcast
        then Play Tennis = Yes
if Outlook is Sunny
        if Wind is FALSE
                then Play Tennis = Yes
        if Wind is TRUE
                then Play Tennis = No
```

```
python 202246109_ML_hw1.py playtennis2.csv
if Outlook is Rainy
        if Humidity is High
                then Play Tennis = No
        if Humidity is Normal
                if Temperature is Cool
                        then Play Tennis = Yes
                if Temperature is Mild
                        then Play Tennis = Yes
                if Temperature is Hot
                        then Play Tennis = No
if Outlook is Overcast
        then Play Tennis = Yes
if Outlook is Sunny
        if Humidity is High
                if Temperature is Mild
                        if Wind is FALSE
                                then Play Tennis = Yes
                        if Wind is TRUE
                                then Play Tennis = No
                if Temperature is Cool
                        then Play Tennis = Yes
                if Temperature is Hot
                        then Play Tennis = No
        if Humidity is Normal
                if Temperature is Cool
                        if Wind is FALSE
                                then Play Tennis = Yes
                        if Wind is TRUE
                                then Play Tennis = No
                if Temperature is Mild
                        then Play Tennis = Yes
                if Temperature is Hot
                        then Play Tennis = Yes
```

데이터의 분할

데이터를 특성대로 나누고 분할된 데이터들의 subset들을 만드는 부분이 복잡했습니다. 이를 해결하기위해 데이터를 나누는 split_data 함수를 별도로 구현하고, 각 특성 값에 따른 부분집합을 정확하게 추출하는 로직을 작성했습니다. 디버깅을 통해 특성별로 데이터가 정확하게 분할되는지 확인한 후, 트리의 각 가지가 올바르게 확장되도록 했습니다.

Information Gain 계산

엔트로피 계산의 구현 자체는 어렵진 않았는데 데이터를 여러 feature로 분할 후 부분집합들의 엔트로피를 구해서 gain을 얻는게 복잡했습니다. 이를 해결하기 위해 엔트로피 계산과 gain을 계산하는 함수를 쪼개서 구현했고, 각 단계마다 출력을 확인하면서 디버깅을 했습니다.

decision tree 학습 구현

Algorithm 1 DecisionTreeTrain(*data*, *remaining features*)

```
guess ← most frequent answer in data          // default answer for this data
if the labels in data are unambiguous then
    return Leaf(guess)                        // base case: no need to split further
else if remaining features is empty then
    return Leaf(guess)                        // base case: cannot split further
else                                          // we need to query more features
    for all f ∈ remaining features do
        NO ← the subset of data on which f=no
        YES ← the subset of data on which f=yes
        score[f] ← # of majority vote answers in NO
                 + # of majority vote answers in YES
                                  // the accuracy we would get if we only queried on f
    end for
    f ← the feature with maximal score(f)
    NO ← the subset of data on which f=no
    YES ← the subset of data on which f=yes
    left ← DecisionTreeTrain(NO, remaining features \ {f})
    right ← DecisionTreeTrain(YES, remaining features \ {f})
    return Node(f, left, right)
end if
```

처음엔 교재의 pseudo code를 참고하여 decision_tree_train을 구현하려 했지만 playtennis.csv 와 같이 여러 특성들이 있는 케이스에서 이진 분할을 가정한 pseudo code가 적합하지 않아서 헤맸습니다.

이를 해결하기 위해 이진 분할 대신 각 특성 값에 대해 분할할 수 있는 split_data함수를 구현했고 특성 값에 따라 재귀적으로 학습하는 구조로 변형했습니다.