데이터 과학

L14.1: Decision Tree Practice

Kookmin University

모듈 불러오기

- 사용할 모듈 import 하기

```
import torch
import requests
import matplotlib.pyplot as plt
import random
from collections import Counter
```

데이터 준비

- Iris 데이터 불러오기
- train, test 데이터 분리

```python
# iris 데이터 다운로드
iris_url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
r = requests.get(iris_url)
open('iris.data', 'wb').write(r.content)

# 데이터 분리
vectors = []
answers = []
with open('iris.data', 'r') as f:
   for line in f:
       if len(line.strip()) != 0:
           items = line.strip().split(",")
           vectors.append([float(i) for i in items[:4]])
           answers.append(items[4])

zipped = list(zip(vectors, answers))
random.shuffle(zipped)
train_size = int(len(vectors) * 0.8)
train_x, train_y = zip(*zipped[:train_size])
test_x, test_y = zip(*zipped[train_size:])
```

Gini Impurity

- Gini Impurity 계산

```
def gini_score(items):
    counter = Counter(items)
    total_count = sum(counter.values())
    return 1 - sum((c / total_count) ** 2 for c in counter.values())
```

Dealing with Numeric Data

- Numeric data에서 나뉘는 지점 찾기

```python
def find_split_point_of_a_field(pairs):
   S = sorted(pairs, key=lambda x: x[0])

   gini_min = 99
   split_point = 0
   for i in range(1, len(S)):
       if S[i-1][0] != S[i][0]:
           prop = i/len(S)
           gini = gini_score([s[1] for s in S[:i]]) * prop + gini_score([s[1] for s in S[i:]]) * (1 - prop)
           if gini < gini_min:
               gini_min = gini
               split_point = (S[i-1][0] + S[i][0])/2

   return split_point, gini_min
```

Split Data

- 여러 필드 중 가장 Gini Impurity를 낮추는 필드를 선택하여 데이터 분리

```
def split_data(X, Y):
    num_fields = len(X[0])

    gini_min, sp_min, fid_min = 99, -1, -1
    for fid in range(num_fields):
        sp, gini = find_split_point_of_a_field(zip([item[fid] for item in X], Y))
        if gini < gini_min:
            gini_min, sp_min, fid_min = gini, sp, fid

    ret = {}
    ret["sp"], ret["gini"], ret["fid"] = sp_min, gini_min, fid_min
    ret["left"] = tuple(zip(*[(x, y) for x, y in zip(X, Y) if x[fid_min] < sp_min]))
    ret["right"] = tuple(zip(*[(x, y) for x, y in zip(X, Y) if x[fid_min] >= sp_min]))
    return ret
```

Decision Tree

- 재귀 함수로 구현

```
def decision_tree(X, Y, threshold):

    original_gini = gini_score(Y)
    node = split_data(X, Y)
    counter = Counter(Y)

    if original_gini < node['gini'] + threshold:
        ans, cnt = counter.most_common(1)[0]
        return ans, cnt / sum(counter.values())
    else:
        node['left'] = decision_tree(node['left'][0], node['left'][1], threshold)
        node['right'] = decision_tree(node['right'][0], node['right'][1], threshold)
        return node
```

Predict

```
def predict(x, tree):
   if 'fid' not in tree:
       return tree

   if x[tree['fid']] < tree['sp']:
       return predict(x, tree['left'])
   else:
       return predict(x, tree['right'])
```

테스트 해보기

- 잘 맞히는지 확인해보기

```python
tree = decision_tree(train_x, train_y, 0)

success = 0

for x, y in zip(test_x, test_y):
   p = predict(x, tree)
   if p[0] == y:
       success += 1

print("accuracy: ", success / len(test_x))
```

```
accuracy:  0.9333333333333333
```

Questions?