Data Science Assignment #2

컴퓨터소프트웨어학부 2018062733 윤동빈

1. **Summary of your algorithm**

Decision Tree 구현을 위해 Attribute selection measure를 Information gain으로 설정했다.
즉, class label을 기준으로 주어진 database에 속한 tuples를 entropy가 낮은 집합으로 구분하기 위해 가장 높은 information gain을 갖는 attribute를 feature로 설정한다. `setCriteria()` 에서 해당 매커니즘을 재귀를 통해 구현하였으며 database의 entropy가 0일 때까지 탐색해 decision tree를 만들어 나가게 된다.

전체 entropy를 **Info(D)** = $-\sum_{i=1}^{m} p_i log_2(p_i)$,
attribute가 A인 entropy를 **Info_A(D)** = $\sum_{j=1}^{r} \frac{|D_j|}{|D|} \times$Info(D),
attribute가 A인 Information gain을 **Gain(A)** = Info(D) - Info_A(D)라 하면,
Info(D), Info_A(D), Gain(A)를 모든 tuples를 반복적으로 돌면서 얻고 feature를 최종적으로 설정한다.

2. **Detailed description of your codes**

```
# Decision Tree를 구성하는 Node
class Node:
    def __init__(self, feat):
        self.feat = feat
        self.next = []
        self.result = 'invalid'

    def append(self, newNode):
        self.next.append(newNode)

    # getter, setter ...
```

- **feat** : Node의 feature
- **next** : Node와 연결된 Node List
- **result** : Node가 leaf node일 때 갖는 class label

```
# Class label count List를 받아 entropy를 반환
def Info(values):
    sum = 0
    ret = 0.0

    for v in values:
        sum = sum + v

    for x in values:
        if x == 0 or sum == 0:
```

```
            continue

        p = x / sum
        ret = ret - p * math.log2(p)

    return ret
```

- **values** (paramter) : database의 class label count list
- 확률 p의 분자 `x` 또는 분모 `sum`이 0인 경우는 카운트하지 않도록 처리

```
# n번째 attribute에 속한 각 domain의 count List
def DomainCnt(DB, kind, n):
    ret = [0 for i in range(len(kind[n]))]
    domain_idx = {}

    i = 0
    for k in kind[n]:
        domain_idx[k] = i
        i = i + 1

    # 각 tuple에서 n-attr에 해당되는 값의 count 체크
    for tuple in DB:
        ret[domain_idx[tuple[n]]] = ret[domain_idx[tuple[n]]] + 1

    return ret
```

- **DB** (parameter) : 주어진 database
- **kind** (parameter) : DB의 각 attribute domain List
- **n** (parameter) : attribute index
- **domain_idx** : n-attribute domain index 반환용 dict

```
# Decision Tree 생성
def setCriteria(DB, attr, attr_chk, kind):
    ret = Node(-1)

    # Info(D)
    ret_cnt = DomainCnt(DB, kind, -1)
    all_info = Info(ret_cnt)

    # 종료 조건 : entropy = 0
    if all_info == 0:
        ret.setResult(DB[0][-1])
        return ret
```

```
        # Info(D_a), Gain(D_a) 구하기
        gain_list = []
        for a in range(len(attr) - 1):

            # 이미 체크된 attribute pass
            if attr_chk[a] == True:
                gain_list.append(-123456789)
                continue

            # attribute a의 값 k를 갖는 tuple 개수 카운트 및 Information Gain 구하기
            info_a = 0.0
            for k in kind[a]:

                idx = 0
                ret_domain = {}                                 # 결과값을 인덱스로 변환하는 dict
                cond_cnt = [0 for i in range(len(kind[-1]))]  # k가 속한 tuple의 각 결과 count

                for tuple in DB:
                    if tuple[a] != k:
                        continue

                    if tuple[-1] not in ret_domain:
                        ret_domain[tuple[-1]] = idx
                        cond_cnt[idx] = cond_cnt[idx] + 1
                        idx = idx + 1
                    else:
                        cond_cnt[ret_domain[tuple[-1]]] = cond_cnt[ret_domain[tuple[-1]]] + 1

                info_a_k = Info(cond_cnt)                   # attribute a의 k category info
                info_a = info_a + sum(cond_cnt) / len(DB) * info_a_k  # attribute a의 info

            gain_a = all_info - info_a  # attribute a의 Information Gain
            gain_list.append(gain_a)

        # 최대 Information Gain을 갖는 attribute를 feature로 설정
        max_gain_idx = gain_list.index(max(gain_list))
        ret.setFeat(max_gain_idx)

        # 하위 Node 생성
        dc = DomainCnt(DB, kind, -1)            # 결과값 등장 list
        max_dc_idx = dc.index(max(dc))          # 가장 많이 등장하는 label index
        max_label = list(kind[-1])[max_dc_idx]  # 가장 많이 등장하는 label

        for k, i in zip(kind[max_gain_idx], range(len(kind[max_gain_idx]))):

            # 하위 Node에 쓰일 DB 생성
            newDB = []
            for tuple in DB:
                if tuple[max_gain_idx] == k:
                    newDB.append(tuple)

            # 하위 case 없으므로 가장 많이 등장하는 label 설정
            if len(newDB) == 0:
                newNode = Node(-1)
```

```
            newNode.setResult(max_label)
            ret.append(newNode)
            continue

        # 하위 Decision Tree에 쓰일 attribute check 생성
        new_attr_chk = attr_chk.copy()
        new_attr_chk[max_gain_idx] = True

        # 현재 Node의 하위 Node 추가
        subNode = setCriteria(newDB, attr, new_attr_chk, kind)
        ret.append(subNode)

    return ret
```

- **attr** (parameter) : DB attribute List
- **attr_chk** (parameter) : attribute의 feature 사용 여부
- **all_info** (parameter) : Info(D)
- feature의 label `k`에서 조회되는 tuple이 없어 class label이 정해지지 않는 경우
  현재 DB에서 가장 많이 등장하는 class label로 설정 (예외 처리)

```
# Decision Tree 통해 각 tuple 결과 구하기
def SearchResult(node, kind, tuple):
    # 종료 조건 : leaf node
    if node.feat == -1:
        return node.getResult()

    # 조건에 맞춰 탐색하면서 최종 결과 얻기
    for k, i in zip(kind[node.getFeat()], range(len(kind[node.getFeat()]))):

        if k == tuple[node.getFeat()]:
            ret = SearchResult(node.getNext(i), kind, tuple)
            return ret

    return 'invalid'
```

- **node** (parameter) : 현재 Decision Tree의 root node
- **tuple** (parameter) : test.txt의 tuple

```
# input : One command line
input_train = sys.argv[1]
input_test = sys.argv[2]
output_name = sys.argv[3]

f = open(input_train, 'r')
attr = f.readline().split()  # attribute
attr_chk = [False for i in range(len(attr))]  # 사용된 attribute 체크
DB = []  # input의 tuples
kind = []  # 각 attribute
```

```
# input data 파싱해 DB, kind 업데이트
while True:

    line = f.readline()
    if not line:
        break

    category_list = line.split()
    category_idx = {}
    category_cnt = 0

    for i, category in zip(range(len(category_list)), category_list):
        if len(kind) != len(attr):
            category_set = set()
            category_set.add(category)
            kind.append(category_set)
        else:
            kind[i].add(category)

    DB.append(category_list)

f.close()

# decision tree 설정
dt_root = setCriteria(DB, attr, attr_chk, kind)

# 위에서 얻은 decision tree로 test
f = open(input_test, 'r')
test_attr = f.readline().split()  # attribute
TDB = []  # test DB
test_result = []  # 각 tuple output

# test 파일을 파싱해 TDB 업데이트
while True:

    line = f.readline()
    if not line:
        break

    TDB.append(line.split())

f.close()

# 위에서 구한 Decision Tree 통해 결과 얻기
for tuple in TDB:
    test_result.append(SearchResult(dt_root, kind, tuple))

# 얻은 결과 작성
f = open(output_name, 'w')

# attributes 작성
attrs = ''
for a, i in zip(attr, range(len(attr))):
    if i == (len(attr) - 1):
```

```
            attrs = attrs + a
        else:
            attrs = attrs + a + '\t'
    f.write(attrs + '\n')

    # tuple 및 test 결과 작성
    for tuple, i in zip(TDB, range(len(TDB))):

        tuple_line = ''

        for v, j in zip(tuple, range(len(tuple))):
            if j == (len(tuple) - 1):
                tuple_line = tuple_line + v + '\t' + test_result[i]
            else:
                tuple_line = tuple_line + v + '\t'

        f.write(tuple_line + '\n')

    f.close()
```

- input 처리 및 output 작성
- Decision tree 생성 및 test

3. **Instructions for compiling your source codes**

```
C:₩Users₩동비니₩Desktop₩PythonWorkspace>python dt.py dt_train1.txt dt_test1.txt dt_result1.txt

C:₩Users₩동비니₩Desktop₩PythonWorkspace>
```

`python dt.py dt_train1.txt dt_test1.txt dt_result1.txt`

4. **Any other specification of your implementation and testing**

- Entropy를 구할 때 확률 `p` 가 0이 되는 경우 제외
- `setCriteria()` 에서 training data를 받아 decision tree를 생성할 때,
  class label이 결정되지 않는 case가 발생하는 경우 class label을 현재(conditional) DB에서
  가장 많이 등장하는 class label로 설정

```
C:₩Users₩동비니₩Desktop₩PythonWorkspace>python dt.py dt_train.txt dt_test.txt dt_result.txt

C:₩Users₩동비니₩Desktop₩PythonWorkspace>dt_test.exe dt_answer.txt dt_result.txt
5 / 5

C:₩Users₩동비니₩Desktop₩PythonWorkspace>python dt.py dt_train1.txt dt_test1.txt dt_result1.txt

C:₩Users₩동비니₩Desktop₩PythonWorkspace>dt_test.exe dt_answer1.txt dt_result1.txt
320 / 346
```

- `dt_test.exe` 를 활용한 test 결과