assignment2 - decision tree

Summary of the algorithm

structure

```
├─assignment2/
|  ├─decision_tree.py
|  ├─src.py
|  ├─dt_result.txt
|  ├─dt_result1.txt
|  ├─data/
|  |  ├─ dt_test.txt
|  |  ├─ dt_test1.txt
|  |  ├─ dt_train.txt
|  |  ├─ dt_train1.txt
|  ├─test/
|  |  ├─ dt_text.ext
|  |  ├─ dt_answer.txt
|  |  ├─ dt_answer1.txt
```

decision tree algorithm

1. 전체 DB_{train}에서 $|DB_{train}| * 2$의 dataset의 sampling with replacement해 $DB_{sample_train}, DB_{sample_val}$을 만든다.
2. DB_{sample_train}을 통해 decision tree를 생성한다.
3. DB_{sample_val}을 통해 inference한 뒤 정확도를 측정한다.
4. DB_{sample_val}에서 틀린 data sample의 가중치를 늘려 다음 sampling 시 더 잘 뽑힐 수 있도록 한다.
5. 위 과정을 `tree_num`(default=500)번 반복해 `tree_num`개 만큼의 tree를 생성한다.
6. 각 tree에서 DB_{test}를 inference한다.
7. tree의 결과와 accuracy를 weighted summation을 해 최종 결과를 생성

Detail methods

random attribute select / pre-pruning

random attribute select : attribute를 선택할 때 전체 attribute 중 1개를 random으로 제외하여 계산

pre-pruning : pruning_thr 인자(default=1e-3)를 통해 gain ratio가 해당 값보다 작은 경우 분기를 중단한다.

```
max_gain=-1
best_attr=None
for attr in np.random.choice(attribute_list,size=len(attribute_list)-1,replace=False):
  gain = self.get_information_gain(db=db,attribute=attr,ratio=self.ratio)
  if max_gain < gain:
      max_gain = gain
      best_attr = attr

if max_gain < self.pruning_thr:
  label = self.get_label(db)
  return Node(db=None,attribute_list=None,item=item,isleaf=True,label=label)
```

boosting

train data마다 틀린 횟수를 counting한 뒤 softmax 함수로 확률로 변환한 뒤 각 데이터를 sampling

```
total_idx = np.arange(db.shape[0])
idx_count = np.zeros_like(total_idx)

forest=[]
for _ in range(tree_num):
    # idx_count에 softmax 함수를 적용해 data sampling
    train_idx = np.random.choice(total_idx,size=db.shape[0]*2,replace=True,p=softmax(idx_count))
    val_idx = np.setdiff1d(total_idx,train_idx)

    train_db = db[train_idx]
    val_db = db[val_idx]

    tree = DecisionTree(db=train_db,attribute_list=attribute_list[:-1],attribute_set=attribute_set,attr2idx=attr2idx,ratio=True,pruning_thr=pruning_thr)

    label = val_db[:,-1]
    pred = np.array(tree.inference(val_db[:,:-1]))

    # 틀린 data sample의 count를 증가
    wrong_idx = val_idx[label!=pred]
```

```
    idx_count[wrong_idx]+=1

    acc = (label==pred).sum()/label.shape[0]
    forest.append([tree,acc])
```

weighted summation

각 tree의 accuracy를 저장한 뒤 D_{test} 측정값과 weighted summation을 진행

```
"""
cat2num_vec : string category값을 number category로 변경해주는 함수
num2cat_vec : number category값을 string category로 변경해주는 함수
"""
logits=None
for i in range(len(preds)):
    if logits is None:
        logits = np.eye(len(label_category))[cat2num_vec(preds[i])] * forest[i][1]
    else:
        logits += np.eye(len(label_category))[cat2num_vec(preds[i])] * forest[i][1]
logits /=len(preds)

labels = num2cat_vec(logits.argmax(axis=1))
```

termination condition

1. 현재 노드에 더 이상의 data가 없는 경우 : parent data를 voting
2. 현재 노드에 더 이상의 attribute가 없는 경우 : 현재 data를 voting
3. 현재 노드에 모든 데이터가 같은 라벨인 경우 : 해당 라벨로 종료

```
if db.size==0:
    label = self.get_label(parent.db)
    return Node(db=None,attribute_list=None,item=item,isleaf=True,label=label)
if len(attribute_list)==0:
    label = self.get_label(db)
    return Node(db=None,attribute_list=None,item=item,isleaf=True,label=label)
if self.get_info(db)<=self.pruning_thr:
    label = self.get_label(db)
    return Node(db=None,attribute_list=None,item=item,isleaf=True,label=label)
```

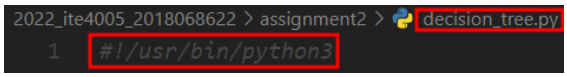
accuracy

$case1 : 5/5$

$case2 : 320.4 \pm 1.019/346$

Execute Guideline

1. `decision_tree.py`의 첫 줄에 python 실행 위치를 적어줍니다.



2. 현재 유저의 `decision_tree.py` 실행권한을 허용합니다.

```
chmod 755 decision_tree.py
```