Introduction to AI Spring 2019 Programming Assignment #2 Report

0516072 洪立宇

- ## Introduction:

用 CART binary tree 和 Gini-impurity function 實作一個 decision tree，透過 Gini-impurity 來找出下一層所將要使用的 attribute，直到所有 attribute 都分完為止，此外我們整合多個 decision tree ，並且跑每一個 decision tree 的 prediction，並且選最多數的 prediction 去建一個 random forest。所用的 Dataset: cross200.txt, ellipse100.txt, glass.txt, iris.txt。

- ## Experiment & Result:

  - Relative sizes of the training and validation subsets (change the training ratio):
    - Ratio=0.8 (0.8training and 0.2 validation)

```
Train-Test Ratio:  0.8
Number of tree in forest:  5
Tree depth limit:  Unlimited


Ellipse 100 :  0.65
Iris :  0.967741935483871
Cross 200 :  0.55
Glass :  0.7674418604651163
```

    - Ratio=0.7(0.7training and 0.3 validation)

```
Train-Test Ratio:  0.7
Number of tree in forest:  5
Tree depth limit:  Unlimited


Ellipse 100 :  0.7666666666666667
Iris :  0.9347826086956522
Cross 200 :  0.5166666666666667
Glass :  0.7076923076923077
```

- Ratio=0.5(0.5training and 0.5 validation)

```
Train-Test Ratio:  0.5
Number of tree in forest:  5
Tree depth limit:  Unlimited


Ellipse 100 :  0.6
Iris :  0.9473684210526315
Cross 200 :  0.44
Glass :  0.8037383177570093
```

- Number of trees in the forest:
  - Number of tree in forest: 5

```
Train-Test Ratio:  0.8
Number of tree in forest:  5
Tree depth limit:  Unlimited


Ellipse 100 :  0.55
Iris :  0.9354838709677419
Cross 200 :  0.7
Glass :  0.7674418604651163
```

  - Number of tree in forest: 20

```
Train-Test Ratio:  0.8
Number of tree in forest:  20
Tree depth limit:  Unlimited


Ellipse 100 :  0.65
Iris :  0.9354838709677419
Cross 200 :  0.575
Glass :  0.7906976744186046
```

◆ Number of tree in forest: 100

```
Train-Test Ratio:  0.8
Number of tree in forest:  100
Tree depth limit:  Unlimited


Ellipse 100 :  0.5
Cross 200 :  0.5
Glass :  0.7209302325581395
```

■ Limit a tree's size:

◆ Tree depth: unlimited

```
Train-Test Ratio:  0.8
Number of tree in forest:  5
Extreme random:  True
Tree depth limit:  Unlimited


Ellipse 100 :  0.75
Iris :  0.8709677419354839
Cross 200 :  0.55
Glass :  0.8604651162790697
```

◆ Tree depth:50

```
Train-Test Ratio:  0.8
Number of tree in forest:  5
Tree depth limit:  50


Ellipse 100 :  0.85
Iris :  0.8709677419354839
Cross 200 :  0.7
Glass :  0.8372093023255814
```

◆ Tree depth:30

```
Train-Test Ratio:  0.8
Number of tree in forest:  5
Tree depth limit:  30


Ellipse 100 :  0.7
Iris :  0.967741935483871
Cross 200 :  0.525
Glass :  0.6976744186046512
```

- Observations:

從上面的實驗和結果，我們可以觀察出當 ratio 越大的時候大部分的 dataset 的 Accuracy 都是上升的。並且在 tree 增加的時候 accuracy 也會增加，但是可以觀察的到的是當 tree num=100 的時候 accuracy 反而降低因為 overfitting 的原因。從 tree 的深度來看，我們可以觀察到深度越深，我們的 accuracy 還有表現就越好。

- Things I Have Learned:
  - 練習了用 CART 來實作 decision tree 和 random forest
  - 學習如何在 training 中使用 validation
  - 更了解 random forest 演算法的細節以及原理
  - 觀察到 overfitting 在何者情況下會發生，並試著不要發生 overfitting
  - 如何隨機分 training data, validation 還有 testing data
- Remaining Questions:

- 有些 dataset 的 accuracy 還是不太高 ，bad performance，可能是因為 data 和 attribute 關聯性不高，或者是 training data 太少的原因
- 要如何更好的隨機 split dataset，或許在 training 上會有更好的幫助

Future Investigation:

- 嘗試 cross-validation
- 嘗試不同的 learning model 來看看會不會有更好的效果
- 用其他的 dataset 來看它們的表現如何，並思考甚麼樣的 dataset 會比較適合 random forest 的 model

Implemented Code

- 計算 Gini Impurity 還透 impurity 計算它的 information gain 來選出 Threshold

```python
#calculate the Gini Impurity for both left and right node and sum them up
def calGiniImpurity(left, right):
    left_times = calTimes([data[len(left[0])-1] for data in left])
    right_time = calTimes([data[len(right[0])-1] for data in right])
    left_gini = 1
    right_gini = 1
    for key, value in left_times.items():
        left_gini -= (value / len(left)) ** 2
    for key, value in right_time.items():
        right_gini -= (value / len(right)) ** 2
    return left_gini + right_gini

def selectThreshold(data_set, attr):
    impurity = 2 #since max impurity=1
    is_leaf = False #if impurity=0 then the node is leaf
    left_data_set = []
    right_data_set = []
    for i in range(0, len(data_set)-1):
        tmp_threshold = (data_set[i][attr] + data_set[i+1][attr]) / 2
        tmp_left = []
        tmp_right = []
        for data in data_set:
            if data[attr] < tmp_threshold:
                tmp_left.append(data)
            else:
                tmp_right.append(data)
        tmp_impurity = calGiniImpurity(tmp_left, tmp_right)
        if tmp_impurity < impurity: ##
            impurity = tmp_impurity
            left_data_set = tmp_left
            right_data_set = tmp_right
            threshold = tmp_threshold

    if impurity == 0:
        is_leaf = True
        threshold = vote([data[len(data_set[0])-1] for data in left_data_set] + [data[len(data_set[0])-1] for data in right_data_set]) ###
    return threshold, left_data_set, right_data_set, is_leaf
```

- 透過給的樹 depth 和 dataset 中的 data 和選出的 attribute 來建樹

```python
def buildTree(node, data_set, attr_left, depth = 0):
    if len(attr_left) == 0 or depth >= TREE_DEPTH_LIMIT or len(data_set) <= 1:
        node.is_leaf = True
        node.data = vote([data[len(data_set[0])-1] for data in data_set])
        return
    if EXTREME_RANDOM:
        selected_attr = attr_left[random.randint(0, len(attr_left)-1)] # randint (a,b) select a integer from a and b
    else:
        selected_attr = attr[0]
    attr_left.remove(selected_attr)
    node.attr = selected_attr
    node.data, left_data_set, right_data_set, node.is_leaf = selectThreshold(data_set, selected_attr)

    if not node.is_leaf:
        node.left = Node()
        node.right = Node()
        if len(left_data_set) == 0:
            node.left = node.right
        else:
            left_thread = threading.Thread(target=buildTree,
                                           args=(node.left, left_data_set, copy.deepcopy(attr_left), depth + 1,))
            left_thread.start()
        if len(right_data_set) == 0:
            node.right == node.left
        else:
            right_thread = threading.Thread(target = buildTree, args = (node.right, right_data_set, copy.deepcopy(attr_left), depth + 1, ))
            right_thread.start()
        try:
            left_thread.join()
        except UnboundLocalError:
            pass
        try:
            right_thread.join()
        except UnboundLocalError:
            pass
```

- Predict 每一個跑過 decision tree 的 data，並且計算最後的 accuracy，以及 validation

```python
def predict(root, data, predictions):
    if root.is_leaf:
        predictions.append(root.data)
        return
    if data[root.attr] < root.data:
        predict(root.left, data, predictions)
    else:
        predict(root.right, data, predictions)

def predictForest(roots, data, accuracy):
    predictions = []
    threads = []
    for i in range(0, len(roots)):
        threads.append(threading.Thread(target=predict, args=(roots[i], data, predictions,)))
        threads[i].start()
    for thread in threads:
        thread.join()
    if vote(predictions) == data[len(data)-1]:
        accuracy['correct'] += 1

def validate(roots, data_set):
    accuracy = {"total": 0, "correct": 0}
    threads = []
    for i in range(0, len(data_set)):
        accuracy['total'] += 1
        threads.append(threading.Thread(target=predictForest, args=(roots, data_set[i], accuracy, )))
        threads[i].start()
    for thread in threads:
        thread.join()

    return float(accuracy['correct'])/float(accuracy['total'])
```