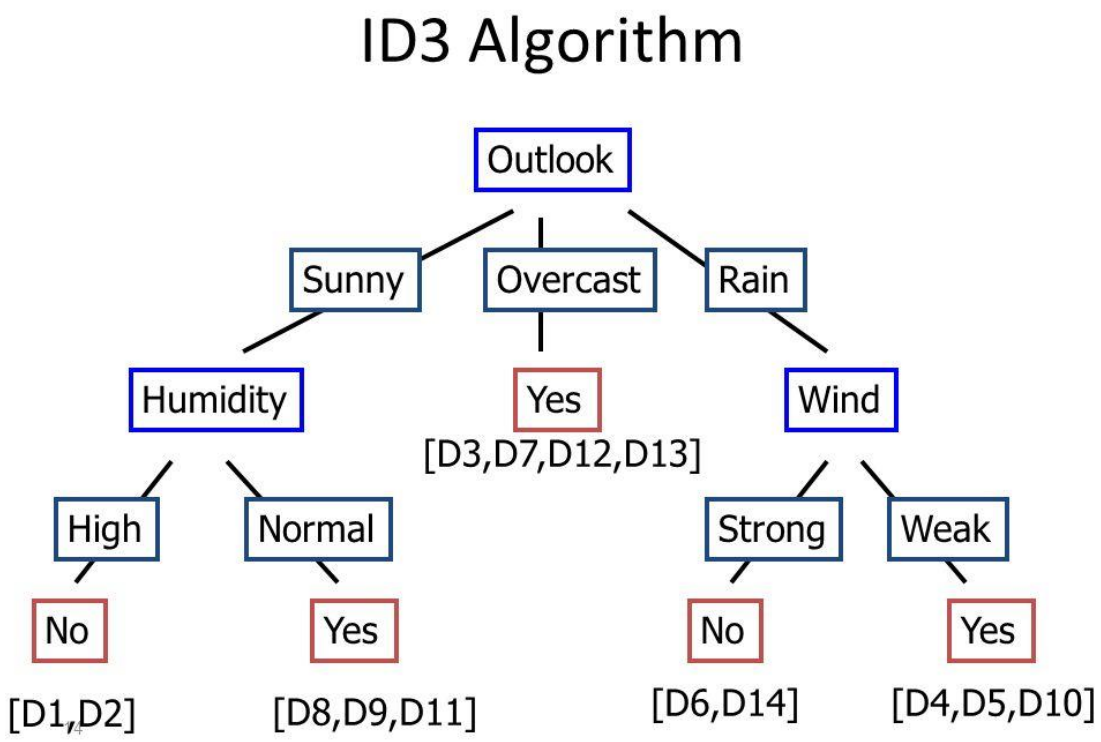
Introduction to Machine Learning Program assignment #1

Environment: Ubuntu 16.04.3 LTS
Using library: numpy、math、random
Language: Python 2.7.12

ID3 Algorithm



(圖片來源：http://mropengate.blogspot.tw/2015/06/ai-ch13-2-decision-tree.html)

Decision Tree 可以用來分類，用 information gain(越大越好)、entropy(亂度越小越好)選出最適合分類的 feature 當節點，把 training data 放上去計算，以分類、長分支，小於節點的接左子樹、大於節點的接右子樹，直到所有 data 都接上去，一棵 Decision Tree 便種好了！

我的作法與使用的 function

1. (這次的作業我有參考 github 上 Alex Yang 先生的 code—https://github.com/alxyang)
2. 用一個 class—Node 來定義節點
   裡面有 left、right、data、threshold_indices、threshold、leaf、pure、label 這些性質。
   left、right 代表那個節點有沒有左右子樹，因為大家一開始都是獨立的點，

所以預設 None。
leaf 即樹的最末枝節點，pure 代表該節點以下的資料都是同一種花種，一開始皆預設為 true，因為是獨立的資料點，還沒接到樹上，自己當然跟自己一樣。

3.

```
289     file = open('iris.csv','r')
290     data = []
291     # loading iris dataset
292     for idx ,line in enumerate(file):
293         line = line.strip("\r\n")
294         data.append([float(element) for element in line.split(',')[:-1]])
295         data[idx].append(line.split(',')[-1])
```

main 裡的 Line289-295 一開始先把資料用 list 的形式吃進來，因為資料是按照花種(Iris-setosa→Iris-versicolor→Iris-virginica)排好的，所以我們先使用 shuffle 打亂順序(要 include random)，以免樹都是往某一個方向長。

4. 接著 ID3 Algorithm 內有用到 find impure leaf、calculate threshold、spilt 這些方法：
首先介紹 find impure leaf (node)，這是一個 recursive function，會從樹頂一直往樹的分支找，找出不純的地方，當左或右子樹分類不同時，便回傳那時候的節點(curr_node)。
把 curr_node 餵進 calc_threshold 的 function 計算 threshold 值跟最好的 feature_index(得出結果是花瓣/花萼的長/寬之一)，再把 threshold 值跟 feature_index 放到 spilt 的 function 以切割出資料的左右。最後把左、右子樹分別接上 ID3_Tree 的左、右，我們就種完一棵大樹了！
以下將介紹 calc_threshold 與 spilt 的 function：

5. calc_threshold(data)

```
68  #choose the best feature from sepal/petal length/width
69  def calc_threshold(data):
70      best_feature_index = -1
71      best_entropy = float('inf')
72      best_thres = float('inf')
73
74      for i in range(len(data[0][:-1])):
75          (entropy, thres) = calc_lowest_entropy(data, i)
76          if entropy < best_entropy:
77              best_entropy = entropy
78              best_feature_index = i
79              best_thres = thres
80      return(best_thres, best_feature_index)
81
```

```
82  #specific feature_index's entropy
83  def calc_lowest_entropy(data, feature_index):
84      sort = sorted(data, key=lambda tup: tup[feature_index])#sort by row
85      #print 'type(sort[0][0])',type(sort[0][0])
86      best_entropy = float('inf')
87      best_thres = float('inf')
88      curr_entropy = float('inf')
89      curr_thres = float('inf')
90
91      for i in range(0,len(data)):
92          if i < len(data)-1:
93              curr_thres = ( (sort[i][feature_index])+(sort[i+1][feature_index]) )/2.0
94
95          (left, right) = split(sort, curr_thres, feature_index)
96          curr_entropy = calc_entropy(left)*float(len(left))/float(len(data))\
97              + calc_entropy(right)*float(len(right))/float(len(data))
98
99          if curr_entropy < best_entropy:
100             best_entropy = curr_entropy
101             best_thres = curr_thres
102     return(best_entropy, best_thres)
```

這個 function 會選出並回傳(best_thres, best_feature_index)。

Line82-102 其中會 call 另外一個 calc_lowest_entropy(data, feature_index)的 function，該 function 會先將 data 依照 feature 的值重新排序，每一個資料點跟資料點相加除以 2 當作 threshold 值做切割，我們便能分別計算左右子樹的 entropy，從所有切割結果中挑一個 entropy 最小的，得出特定 feature 的(best_entropy, best_thres)並回傳進剛剛 calc_threshold 的 function 內。

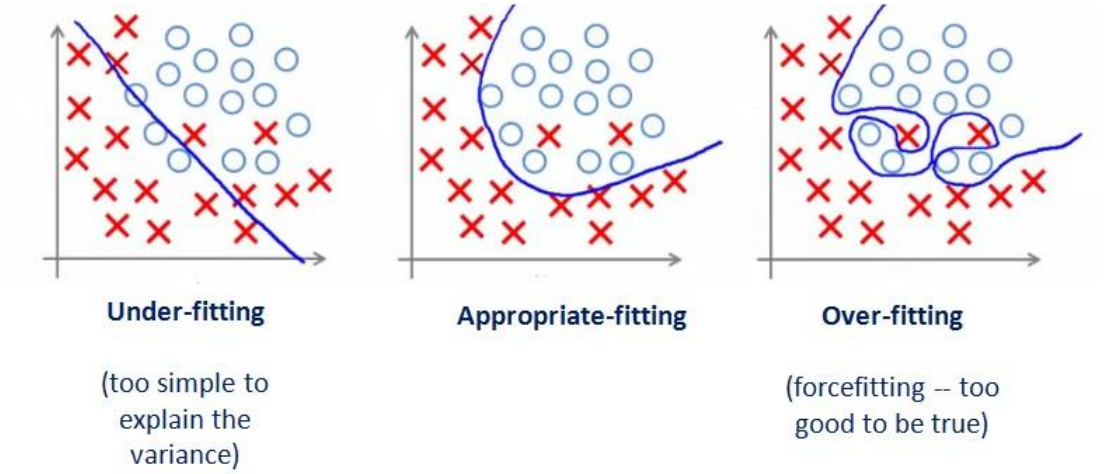
Line69-80 用 for loop 跑四次，從各個 feature 的 best entropy 中再選出所有 data 裡最猛的，就能算出我們要的(best_thres, best_feature_index)。

6. split(data, threshold, feature_index)

```
35  def split(data, threshold, feature_index):
36      left = []
37      right = []
38      for datapoint in data:
39          if datapoint[feature_index] <= threshold:
40              left.append(datapoint)
41          else:
42              right.append(datapoint)
43      return(left, right)
```

這是用來切割左、右子樹的 function，根據先前計算決定的 feature_index 切好之後會黏到該 datapoint 的左、右子樹。

Testing Result



(圖片來源：http://mropengate.blogspot.tw/2015/06/ai-ch13-2-decision-tree.html)

Iris dataset 裡只有 150 筆資料，所以我們先將他打亂，再用 K-fold validation 切割出 training set 和 testing set，testing 的結果如下圖，90 幾%的正確率已經非常高，若是追求 100%之後可能會 overfitting。

然而，因為資料數實在太少，可以看出 accuracy 差異很大，每一次 train 的結果都不一樣，即使我 test time 設 80 次取平均，accuracy 平均值最大仍可以相差到 4%。

ID3 DecisionTree:

表現好時

```
general@Si2-1080ti:~/mnist/secret$ ./run.sh
Requirement already satisfied: numpy in /usr/local/lib/python2.7/dist-packages
0.96
1.0     1.0
0.984615384615  0.92905982906
0.920659340659  0.971428571429
```

表現不好時

```
general@Si2-1080ti:~/mnist/secret$ ./run.sh
Requirement already satisfied: numpy in /usr/local/lib/python2.7/dist-packages
0.906666666667
1.0     0.928571428571
0.837056277056  0.925454545455
0.888095238095  0.925274725275
```

→可以看出總 accuracy 的誤差有 5.4%之多

Random_forest:

表現好時

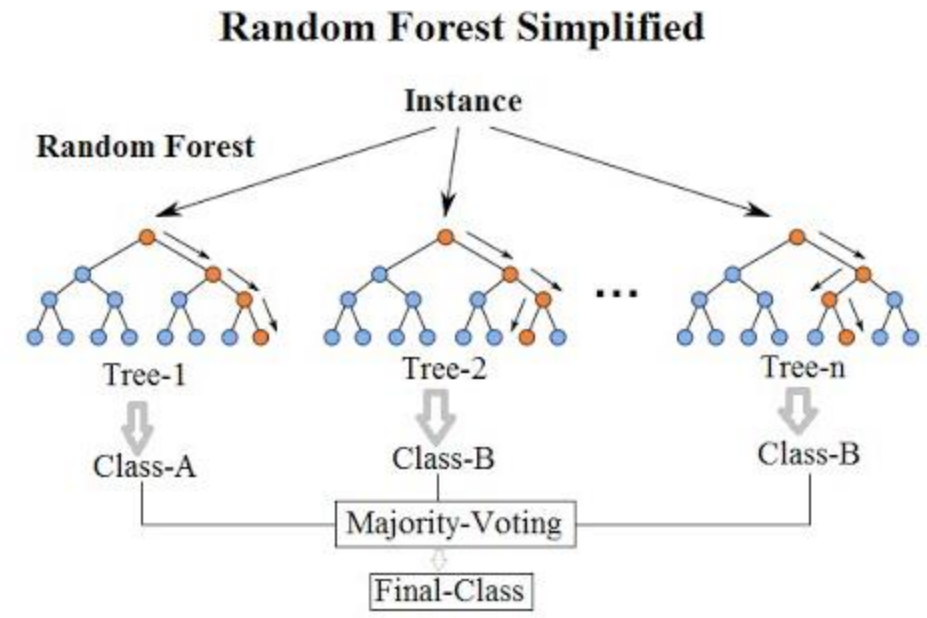
```
general@Si2-1080ti:~/mnist/secret$ ./run.sh
Requirement already satisfied: numpy in /usr/local/lib/python2.7/dist-packages
0.953333333333
1.0     1.0
0.981818181818  0.919230769231
0.894047619048  0.983333333333
```

表現不好時

```
general@Si2-1080ti:~/mnist/secret$ ./run.sh
Requirement already satisfied: numpy in /usr/local/lib/python2.7/dist-packages
0.913333333333
1.0     1.0
0.951428571429  0.929206349206
0.940170940171  0.959595959596
```

→可以看出總 accuracy 的誤差有 4%之多

Random_Forest(bonus)



(圖片來源：https://www.youtube.com/watch?v=ajTc5y3OqSQ)

用隨機的方式建構出一座森林，森林的每一棵樹都是由 ID3 Algorithm 種的，一樣用 K_fold 分開 training data 和 testing data，差異在於 predict 完結果之後，用 voting 的方式，森林裡的每一棵樹做投票，Iris-setosa、Iris-versicolor、Iris-virginica 誰的票數多，就偵測那筆 data 是該花種。

我的作法與使用的 function

和 ID3_DecisionTree 幾乎一樣，但是 Random_Forest 的 Forest 是由 training set 一次次打亂、每次取其中前 40 筆資料，餵進 ID3 種下一棵棵樹而組成。
而 prediction 的部分，則是要累加每一次預測的投票結果，根據樹們多數決來判斷 testing datapoint 的花種。
經過多次嘗試之後，我選 accuracy 相對較大較穩定的設定條件——森林裡有 15 棵樹，每棵樹 40 筆資料，黏成一座森林，即長度為 15 的 list。

◎Predict_and_vote(datapoint, Forest)

Testing 時把資料都丟到森林裡的每一棵樹裡面，跟著他們的分支走，走到底就

是 prediction 的花種，再用 array 計算每一個 label 有幾票，票多的就是我們預測的結果。

```
178 def predict_and_vote(datapoint, Forest):
179     curr_node = Forest
180     label = [0, 0, 0]
181     #print root
182     #print list(curr_node.__dict__.keys())
183     for tree in Forest:
184         curr_node = tree
185         while not(curr_node.pure):
186             threshold = curr_node.threshold
187             feature_index = curr_node.threshold_indices
188             if datapoint[feature_index] <= threshold:
189                 curr_node = curr_node.left
190             else:
191                 curr_node = curr_node.right
192         if curr_node.label == 'Iris-setosa':
193             label[0] += 1
194         elif curr_node.label == 'Iris-versicolor':
195             label[1] += 1
196         elif curr_node.label == 'Iris-virginica':
197             label[2] += 1
198
199     if label[0]>label[1] and label[0]>label[2]:
200         return 'Iris-setosa'
201     elif label[1]>label[0] and label[1]>label[2]:
202         return 'Iris-versicolor'
203     elif label[2]>label[0] and label[2]>label[1]:
204         return 'Iris-virginica'
```