Artificial Intelligence Capstone : Random Forests

Project Overview

- load_data：對資料進行 preprocessing，並切割成 train_x、train_y、test_x 及 test_y 。
- Gini：計算 **Gini Impurity**，以了解隨機選中的樣本在這個 set 中分錯的機率。
- class Node：用來組成 CART 的 Node。
  - FalseNode、TrueNode：母節點 spilt 而成的 Nodes。
  - attribute、threshold：該節點 spilt 的標準。
  - leaf、depth、majorlabel
- class CART：用來組成 Random Forest 的 Classification and Regression Tree，為二元樹。
  - root：為 class Node，用來記錄 CART 的根節點，以方便之後能拜訪其他節點。
  - spiltNode：以某個 attribute 及 threshold 為標準，將 datas 切割成兩個 set。
  - buildCART：用來建立 CART。過程中，會去遍歷各種 attribute 及 threshold 的組合，並透過 Gini impurity 來選擇最好的切割方式。
  - train：判斷是否要做 attribute bagging，並呼叫 buildCART 來 train Decision Tree。
  - predict：用來預測 testcase 的 label。
  - cal_accuracy：給定一組 testcases 及對應的 labels，依序進行 predict，並計算準確率。
- class RandomForest：
  - n_estimators、CARTrees：tree 的數量，以及訓練好的 CARTs
  - max_depth、min_impurity_decrease、min_samples_split：用來限制 tree size。
  - max_features、max_samples：用來訓練 Decision Tree 的 attribute 及 sample 數量。
  - bootstrap、criterion：是否要進行 bagging、使用的 evaluation function
  - train：依序建立 n_estimator 棵 CART，並分別對這些 CART 進行 training。
  - predict：用來預測 testcase 的 label。
  - cal_accuracy：給定一組 testcases 及對應的 labels，依序進行 predict，並計算準確率。

Experiments and Results

- hyperparameter 的初始值：set ratio(8:2)、n_estimators(100)、max_depth(12)、max_features(“sqrt”)、max_samples(100)、bootstrap(true)、min_impurity_decrease(10^{-6})、min_samples_split(2)、criterion(Gini)

- 每個實驗只會調整一種 hyperparameter，其餘皆固定，而實驗結果為 10 次準確率的平均。

Relative sizes of the training and validation subsets

- 目標：分析**不同比例的 training/validation sets** 的 model 準確率 **(train 準確率 / test 準確率)**

	5:5	**6:4**	**7:3**	**8:2**	**9:1**
BreastCancer	0.678/0.659	0.605/0.650	0.691/0.690	0.674/0.654	0.674/0.(
Glass	1.000/0.706	0.998/0.722	0.995/0.730	0.989/0.751	0.981/0.7
Ionosphere	0.996/0.929	0.995/0.915	0.997/0.931	0.992/0.943	0.994/0.9
Wine	1.000/0.981	1.000/0.983	1.000/0.981	1.000/0.989	1.000/0.9

- 結果分析：當 train data 及 validation data 的比例較接近時，train 準確率會較高，但 test 準確率較低，可能是因為 train data 沒辦法確實的代表原本 data 的 distribution，使得訓練好的 model 只符合 train data。當我們逐漸將 train data 比例上升時，可以發現即使 train 準確率下降，但 test 的準確率卻上升，代表此時的 train data 較能符合原本 data 的 distribution。
- 結論：當我們在訓練 model 時，必須要取**足夠比例的 train data**，以**符合原本的 distribution**。

Number of trees in the forest

- 目標：分析**不同數量的 CART** 的 model 準確率 **(train 準確率 / test 準確率)**

	10	**20**	**50**	**100**	**200**
BreastCancer	0.652/0.671	0.666/0.663	0.672/0.675	0.689/0.675	0.690/0.(
Glass	0.949/0.733	0.973/0.714	0.983/0.758	0.990/0.774	0.994/0.7
Ionosphere	0.964/0.903	0.983/0.923	0.989/0.934	0.993/0.939	0.994/0.9
Wine	1.000/0.978	1.000/0.975	1.000/0.964	1.000/0.972	1.000/0.9

- 結果分析：當 forest 中的 trees 數量越多時，train 的準確率都會漸漸上升後，最終趨近於平緩；而 test 的準確率大多數也都是呈現上升，最終趨緩（有些樹的數量上升，準確率下降的情況，我認為是因為 test 的資料量不多，導致起伏較大）。
- 結論：**當 Tree 的數量增加時，一開始準確率會上升較多，而最終會趨緩於一個範圍內**。其原因為透過 **major vote** 可以排除一些極端值，使預測的穩定度上升。

Parameters used during tree induction, such as how many attributes to consider at each node splitting (By tree baggging and attribute bagging)

- 目標：分析**不同數量的 attribute** 的 model 準確率 **(train 準確率 / test 準確率)**
	“log2”	**“sqrt”**	**All**
BreastCancer	0.643/0.679	0.686/0.672	0.421/0.393
Glass	0.992/0.756	0.990/0.747	0.988/0.719
Ionosphere	0.991/0.934	0.996/0.929	0.994/0.919
Wine	1.000/0.986	1.000/0.986	1.000/0.956

  - 結果分析：log2 和 sqrt 的準確率相差不大的原因為這次的 attribute 數量皆不多，使得 log2 和 sqrt 得到的數量差異不大。然而使用全部的 attribute 時，會導致部分分類效果不佳的 attribute 被 model 考慮，且可能會產生 **overfitting**，使得準確率下降。
  - 結論：當 **attribute 的數量上升**時，**準確率也會上升**；但**超過一定的數量後，不必要的 attribute 會被考慮，overfitting 也可能發生，導致 test 的準確率下滑**。
- 目標：分析**不同數量的 training samples** 的 model 準確率 **(train 準確率 / test 準確率)**
	10	**20**	**50**	**100**	**All**
BreastCancer	0.496/0.521	0.420/0.449	0.613/0.640	0.698/0.679	0.68
Glass	0.648/0.612	0.740/0.644	0.900/0.716	0.989/0.770	0.99
Ionosphere	0.901/0.853	0.906/0.900	0.977/0.917	0.993/0.927	0.99
Wine	0.982/0.961	0.985/0.992	0.999/0.983	1.000/0.981	1.00

  - 結果分析：將 sample 數量上升後，可以發現兩者的準確率都會逐漸上升，並趨近於平緩。但從 Glass 這個 dataset 中可以發現，使用全部的 samples 時，train 的準確率上升，但 test 的準確率卻下降，因此發生了 **overfitting**（可能考慮了 outlier）。
  - 結論：當 **training data 的數量上升**時，**準確率也會上升**；但**超過一定的數量後，考慮到 outlier 的機率上升，使得 overfitting 可能會發生，導致 test 的準確率下降**。

Methods that limit a tree’s size

- 目標：分析**不同 max depth** 的 model 準確率 **(train 準確率 / test 準確率)**

	3	**5**	**10**	**15**	**20**
BreastCancer	0.678/0.689	0.682/0.691	0.681/0.726	0.648/0.694	0.66
Glass	0.742/0.677	0.875/0.691	0.978/0.747	0.994/0.763	0.99
Ionosphere	0.965/0.906	0.988/0.924	0.994/0.926	0.993/0.927	0.99
Wine	0.996/0.972	1.000/0.983	1.000/0.992	1.000/0.978	1.00

- 結果分析：當 max depth 很小時，能 spilt 的次數少，使得 sample 較沒辦法準確的分配到正確的 label；當 max depth 上升時，可以判斷較多次 attribute 及 threshold，使得各個 sample 都能較正確的分組。然而，從各組中都能發現，當 max depth 超過一定的深度時，會導致 **overfitting** 發生，因為會盡可能去把 train samples 分得很細。
- 結論：當 **max depth 上升**時，**準確率會上升**；但**超過一定的深度**時，會使得 **model 過度符合 training data 發展**，導致 **overfitting 發生**，**test 的準確率也就下降**。

- 目標：分析**不同 min_impurity_decrease** 的 model 準確率 **(train 準確率 / test 準確率)**

	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-}
BreastCancer	0.688/0.709	0.674/0.677	0.647/0.644	0.688/0.684	0.68
Glass	0.987/0.728	0.993/0.714	0.988/0.753	0.989/0.758	0.98
Ionosphere	0.980/0.906	0.993/0.946	0.992/0.941	0.995/0.920	0.99
Wine	1.000/0.992	1.000/0.986	1.000/0.978	1.000/0.983	1.00

- 結果分析：當 min_impurity_decrease 下降到最佳值時，準確率會上升，因為較有可能找到符合的 threshold 及 attribute。但若超過最佳值後繼續下降的話，會使 CART 過度隨著 train data 發展，CART 很容易繼續 spilt，然而對於 test data 不一定會有改善，使準確率下降。
- 結論：當 **min_impurity_decrease 下降**時，**準確率會上升**；但**超過一定的標準**時，會使得 **model 過度符合 training data 發展**，導致 **overfitting 發生**，**test 的準確率也就下降**。

- 目標：分析**不同 min_samples_split** 的 model 準確率 **(train 準確率 / test 準確率)**

	15	**10**	**5**	**2**	**1**
BreastCancer	0.628/0.626	0.616/0.626	0.662/0.677	0.682/0.665	0.68
Glass	0.837/0.716	0.888/0.735	0.949/0.788	0.989/0.749	1.00
Ionosphere	0.947/0.923	0.967/0.919	0.982/0.894	0.994/0.929	0.99
Wine	1.000/0.978	1.000/0.989	1.000/0.986	1.000/0.975	1.00

- 結果分析：對於各個 dataset，當 min sample spilt 為 15 的時候，可以發現普遍準確率皆較低，因為每個 node 都還有很多的 data，可以卻沒辦法繼續細分，只能透過 major vote 選出一個代表。當 min sample spilt 往下降時，大部分的準確率都會逐漸上升；然而可以發現某些 dataset 在 min sample spilt 為 1 時，train 準確率很高，但 test 準確率卻下降，因此可以發現 min sample spilt 太小時，也會導致 CART 隨著 train data 發展。
- 結論：當 **min_samples_split 下降**時，**準確率會上升**；但當 **min_samples_split** 太小時，**model 會過度符合 training data 發展**，導致 **overfitting 發生**，**test 的準確率也就下降**。

Observations

- Extremely random forest：若我們在 spilt node 時，隨機選擇一個 attribute，會使得分群的結果不盡理想，尤其是當 attribute 的數量上升時，選中最佳的 attribute 的機率會下降。即使在層數很淺時，還能較有效的下降 impurity；然而一旦層數變深，Extremely random forest 便很難有效下降 impurity，因此準確率不一定很好。
- Comparisons of out-of-bag errors and validation-set errors：通常 OOB erros 會給我們近似於 validation-set erros 的值，使我們能估計 Randon forest 遇到新的 data 時的準確率。然而當樹的數量很少時，OOB errors 普遍會高於 validation-set errors，因為此時 forest 中樹的變異度還很高。一旦我們將樹的數量逐漸提升，OOB errors 會趨近於 validation-set errors，且兩者最終都會收斂在一個範圍內，因為此時整個 forest 的變異度已經下降，使得 OOB errors 可以用來推估 validation-set errors。

Things I learned from this project

- 透過自己研究並 implement CART 及 Random Forest，除了學習到 model 的架構外，也清楚的了解 Decision Tree 和 Random Forest 是如何去做 training 和預測結果。
- 透過調整 hyperparameter，使自己了解每種 hyperparameter 會如何去影響 model 的複雜度以及預測的準確率。因此往後訓練遇到瓶頸時，可以自己去分析哪個因素影響準確率，並試著去調整到最好的 hyperparameter 來提升準確率。

Remaining questions and Ideas of future investigation

- 這次實作 Random Forest 時，發現在 Breast Cancer 這個 dataset 的效果最差。我認為是因為在 Breast Cancer 的 attribute 幾乎都是 categorical，然而這次選用 threshold 以及如何去二分 categorical attribute 都是以較簡單的方式實作，導致準確度較低。希望未來能更進一步去對 categorical 的資料做前處理，以及改善 Decision Tree 的架構，來優化 Random Forest 對此種 dataset 的準確率。
- 此外，由於這次在學習的過程中，有去參考文獻中 Random Forest 的可調整參數，但這次沒有實作出所有的 hyperparameter，因此對於部分的參數會如何影響準確率仍尚未研究。在未來，希望有更多的機會來研究並實作出完整的 Random Forest 的架構。

Appendix

Reference

1. https://blog.csdn.net/weixin_40479663/article/details/84781500 (https://blog.csdn.net/weixin_40479663/article/details/84781500)
2. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html (https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html)
3. https://zh.wikipedia.org/wiki/随机森林 (https://zh.wikipedia.org/wiki/%E9%9A%8F%E6%9C%BA%E6%A3%AE%E6%9E%97)

Code

- 由於以 HackMD 書寫報告，Code 在轉成 PDF 之後沒辦法用滾輪滑動，因此有盡量以換行符來表示換行的位置，使 Code 能較清楚明瞭。
- 而以下是 HackMD 的連結：https://hackmd.io/NZBu0C6CT9ubXNvmLplktA?view (https://hackmd.io/NZBu0C6CT9ubXNvmLplktA?view)
- Code 位於下一頁

```
import numpy as np
import pandas as pd
import random
import copy
import math
import time

filename = "breast-cancer.data"
epsilon = 10 ** (-6)
nAttribute = -1

# load file and preprocess datas
def load_data(filename, partition=(8,2)):
    # read files
    origin_data = pd.read_csv(filename, sep=",", header=None).to_numpy()
    n_data = origin_data.shape[0]

    # shuffle the data
    for i in range(0,10):
        np.random.shuffle(origin_data)

    # calculate where to spilt data
    spiltpoint = round(n_data * partition[0] / (sum(partition)))
    train_x, train_y, test_x, test_y = None, None, None, None

    # we need to spilt different dataset with different way
    if filename == "glass.data":
        train_x = origin_data[0:spiltpoint, 1:-1]
        train_y = origin_data[0:spiltpoint, -1]
        test_x = origin_data[spiltpoint:n_data, 1:-1]
        test_y = origin_data[spiltpoint:n_data, -1]
    elif filename == "iris.data" or filename == "ionosphere":
        train_x = origin_data[0:spiltpoint, 0:-1]
        train_y = origin_data[0:spiltpoint, -1]
        test_x = origin_data[spiltpoint:n_data, 0:-1]
        test_y = origin_data[spiltpoint:n_data, -1]
    else:
        train_x = origin_data[0:spiltpoint, 1:]
        train_y = origin_data[0:spiltpoint, 0]
        test_x = origin_data[spiltpoint:n_data, 1:]
        test_y = origin_data[spiltpoint:n_data, 0]

    return train_x, train_y, test_x, test_y

# Compute Gini's impurity
def Gini(labels):
    _ , counts = np.unique(labels, return_counts=True)
    impurity = sum([(count/labels.shape[0]) ** 2 for count in counts])
    return 1 - impurity

class Node:
    def __init__(self, depth=0):
        self.FalseNode = None
        self.TrueNode = None
```

```
        self.leaf = True
        self.majorlabel = None
        self.attribute = None
        self.threshold = None
        self.depth = depth

class CART:
    def __init__(self):
        self.root = Node()

    # Given a threshold, spilt the data into two set by this threshold
    def spiltNode(self, datas, attribute, threshold):
        # FSetId for indices of false set; TsetId for indices of true set
        FSetID = []
        TSetID = []

        # According to the type of data, split the data in different way
        if isinstance(threshold, int) or isinstance(threshold, float):
            for i in range(0,len(datas[:,attribute])):
                if (datas[:,attribute][i] < threshold):
                    FSetID.append(i)
                else:
                    TSetID.append(i)
        else:
            for i in range(0,len(datas[:,attribute])):
                if (datas[:,attribute][i] != threshold):
                    FSetID.append(i)
                else:
                    TSetID.append(i)
        return FSetID, TSetID

    # recursively build the CART
    # stop when we cannot reduce the impurity
    def buildCART(self, curNode, datas, labels, attrilist, max_depth, \
        min_impurity_decrease=0.0, min_samples_split=2, criterion=Gini):
        """
        max_depth: The maximum depth of the tree
        min_impurity_decrease: A node will be split if this split induces a deci
        of the impurity greater than or equal to this value
        min_samples_split: The minimum number of samples required to split an
        internal node
        criterion: The function to measure the quality of a split
        """

        # check if we meet the limit
        if curNode.depth >= max_depth or labels.shape[0] <= min_samples_split:
            key , counts = np.unique(labels, return_counts=True)
            curNode.majorlabel = key[np.argmax(counts)]
            return

        # if the label is unique in this node
        if np.unique(labels).shape[0] == 1:
            curNode.majorlabel = labels[0]
            return

```

```
110         # initialize
111         parent_imp = criterion(labels)
112         max_impurity_decrease = 0.0
113         best_attribute, best_threshold = None, None
114 
115         # reset min_impurity_decrease
116         min_impurity_decrease = max(min_impurity_decrease, epsilon)
117 
118         # find the best attribute for reducing most impurity
119         for curAtt in attrilist:
120 
121             # collect all possible threshold
122             threslist = np.unique(datas[:,curAtt])
123             if isinstance(datas[0][curAtt], int) or \
124             isinstance(datas[0][curAtt], float):
125                 threslist = [(threslist[i-1]+threslist[i])/2 \
126                 for i in range(1,len(threslist))]
127 
128             # find the best threshold for reducing most impurity
129             for threshold in threslist:
130                 FSetID, TSetID = self.spiltNode(datas, curAtt, threshold)
131                 ratio = len(FSetID) / labels.shape[0]
132                 impurity_decrease = parent_imp - \
133                     ratio * criterion(labels[FSetID]) - \
134                         (1 - ratio) * criterion(labels[TSetID])
135 
136                 # we get better attribute and threshold to spilt the node
137                 if impurity_decrease > max_impurity_decrease:
138                     max_impurity_decrease = impurity_decrease
139                     best_attribute, best_threshold = curAtt, threshold
140 
141         # the reduction of impurity is more than limit
142         if max_impurity_decrease > min_impurity_decrease:
143             Best_FSetID, Best_TSetID = \
144                 self.spiltNode(datas, best_attribute, best_threshold)
145             curNode.FalseNode, curNode.TrueNode = \
146                 Node(curNode.depth+1), Node(curNode.depth+1)
147             curNode.attribute, curNode.threshold = \
148                 best_attribute, best_threshold
149             curNode.leaf = False  # we can spilt more
150 
151             self.buildCART(curNode.FalseNode, datas[Best_FSetID], \
152             labels[Best_FSetID], attrilist, max_depth, min_impurity_decrease, \
153             min_samples_split, criterion)
154             self.buildCART(curNode.TrueNode, datas[Best_TSetID], \
155             labels[Best_TSetID], attrilist, max_depth, min_impurity_decrease, \
156             min_samples_split, criterion)
157 
158         # the reduction cannot meet the limit,
159         # so this node is leaf and should compute majorlabel
160         else:
161             key , counts = np.unique(labels, return_counts=True)
162             curNode.majorlabel = key[np.argmax(counts)]
163 
164     def train(self, datas, labels, max_depth=None, max_features="auto", \
```

```
    bootstrap=True, min_impurity_decrease=0.0, min_samples_split=2, criterion=G
        if max_depth == None:
            max_depth = 10 ** 9

        # Attribute Bagging
        if bootstrap == True:
            attributeID = \
            np.random.choice(a=nAttribute, size=max_features, replace=False)
        else:
            attributeID = [i for i in range(0, nAttribute)]

        self.buildCART(self.root, datas, labels, attributeID, max_depth, \
             min_impurity_decrease, min_samples_split, criterion)

    def predict(self, testcase):
        curNode = self.root
        while (not curNode.leaf):
            if isinstance(curNode.threshold, int) or \
            isinstance(curNode.threshold, float):
                if testcase[curNode.attribute] < curNode.threshold:
                    curNode = curNode.FalseNode
                else:
                    curNode = curNode.TrueNode
            else:
                if testcase[curNode.attribute] == curNode.threshold:
                    curNode = curNode.FalseNode
                else:
                    curNode = curNode.TrueNode
        return curNode.majorlabel

    def cal_accuracy(self, testcases, labels):
        ncase = len(labels)
        correct = \
        [int(self.predict(testcases[i]) == labels[i]) for i in range(0,ncase)]

        return float(sum(correct)) / ncase

class RandomForest:
    def __init__(self, n_estimators=100, max_depth=None, max_features="auto", \
    max_samples=None, bootstrap=True, min_impurity_decrease=0.0, \
    min_samples_split=2, criterion=Gini):
        self.CARTrees = []
        self.n_estimators = n_estimators
        self.max_depth = max_depth
        self.max_features = max_features
        self.max_samples = max_samples
        self.bootstrap = bootstrap
        self.min_impurity_decrease = min_impurity_decrease
        self.min_samples_split = min_samples_split
        self.criterion = criterion

    def train(self, datas, labels):
        # According to different input, give the corresponding number of featur
        if self.max_features == "auto" or self.max_features == "sqrt":
            self.max_features = math.sqrt(nAttribute)
```

```
220             elif self.max_features == "log2":
221                 self.max_features = math.log2(nAttribute)
222             elif type(self.max_features) == float:
223                 self.max_features = self.max_features * nAttribute
224             elif self.max_features == None:
225                 self.max_features = nAttribute
226 
227             if self.max_samples == None:
228                 self.max_samples = datas.shape[0]
229             elif type(self.max_samples) == float:
230                 self.max_samples = self.max_samples * datas.shape[0]
231 
232             # ensure that the number of  samples and features is less than
233             # or equal to the original size
234             self.max_samples = min(round(self.max_samples), datas.shape[0])
235             self.max_features = min(round(self.max_features), nAttribute)
236 
237             for i in range(0, self.n_estimators):
238                 tree = CART()
239                 if self.bootstrap == True:
240                     sampleID = np.random.choice(a=datas.shape[0], \
241                     size=self.max_samples, replace=False)
242                 else:
243                     sampleID = [i for i in range(0, datas.shape[0])]
244 
245                 tree.train(datas[sampleID], labels[sampleID], self.max_depth, \
246                 self.max_features, self.bootstrap, self.min_impurity_decrease, \
247                 self.min_samples_split, self.criterion)
248                 self.CARTrees.append(tree)
249 
250         def predict(self, testcase):
251             predict_labels = \
252             np.array([tree.predict(testcase) for tree in self.CARTrees])
253             key , counts = np.unique(predict_labels, return_counts=True)
254             return key[np.argmax(counts)]
255 
256         def cal_accuracy(self, testcases, labels):
257             ncase = len(labels)
258             correct = \
259             [int(self.predict(testcases[i]) == labels[i]) for i in range(0,ncase)]
260             return float(sum(correct)) / ncase
261 
262     if __name__ == "__main__":
263         train_x, train_y, test_x, test_y = load_data(filename)
264         nAttribute = train_x.shape[1]
265         attrilist = [i for i in range(0, nAttribute)]
266 
267         a = time.time()
268         RF = RandomForest(n_estimators=50, max_depth=12, max_features="sqrt", \
269         max_samples=100, bootstrap=True, min_impurity_decrease=epsilon, \
270         min_samples_split=2)
271         RF.train(train_x, train_y)
272         train_accuracy = RF.cal_accuracy(train_x, train_y)
273         test_accuracy = RF.cal_accuracy(test_x, test_y)
274         b = time.time()
275         print("Train Accuracy:", train_accuracy, '\n', end ='')
```

```
275     print( Train Accuracy: , train_accuracy,  \n , end=  )
276     print("Test Accuracy:", test_accuracy, '\n', end='')
277     print("Used Time:", b-a, "s")
```