# 期 末 報 告 -- [Isolation forest]

學號：R06525054
姓名：麥鈞程

演算法文件(抽籤編號與套件連結)

# Outline

Algorithm Introduction

Code Review

**Bonus: Model Preview

Live Demo(Using InAnalysis)

Conclusion

Reference

# Algorithm Introduction

1.一種異常檢測的方法，非監督學習

2.時間複雜度是線性的

3.擅長處理大數據和高維度的資料

4.假設異常點很少而且與眾不同

# Algorithm Introduction

- 2個階段:
- 1.build isolation tree
- 2.obtain an anomaly score (判斷異常的程度)

# Algorithm Introduction (iTree)

**Algorithm 2** : $iTree(X, e, l)$

**Inputs**: $X$ - input data, $e$ - current tree height, $l$ - height limit

**Output**: an iTree

1: **if** $e \geq l$ or $|X| \leq 1$ **then**
2: return $exNode\{Size \leftarrow |X|\}$
3: **else**
4: let $Q$ be a list of attributes in $X$
5: randomly select an attribute $q \in Q$
6: randomly select a split point $p$ from $max$ and $min$ values of attribute $q$ in $X$
7: $X_l \leftarrow filter(X, q < p)$
8: $X_r \leftarrow filter(X, q \geq p)$
9: return $inNode\{Left \leftarrow iTree(X_l, e+1, l),$
10: $Right \leftarrow iTree(X_r, e+1, l),$
11: $SplitAtt \leftarrow q,$
12: $SplitValue \leftarrow p\}$
13: **end if**

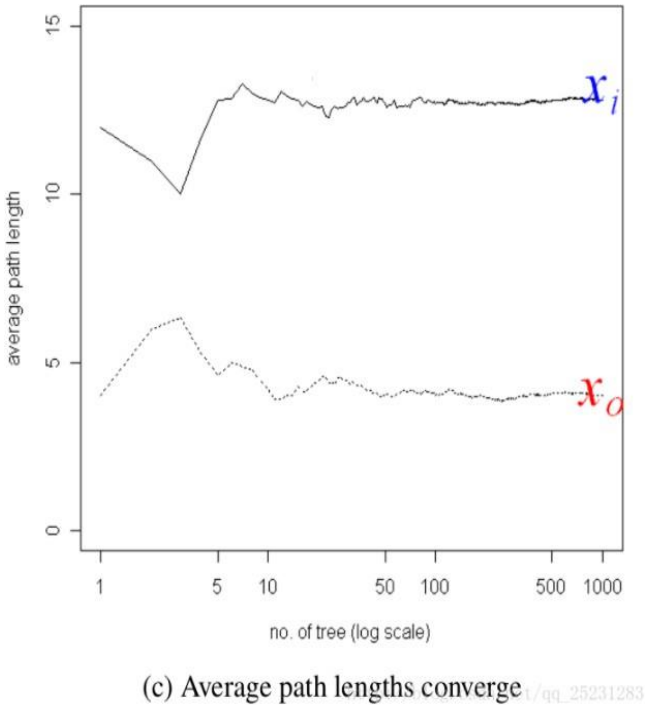


- 可能是external-node 或internal-node，隨機選擇一個屬性q，並在這個維度的最大值和最小值之間隨機選一個值p
- 如果比p大就長到右子節點，比p小就長到左子節點
- proper binary tree:每個節點可能有0或2個daughter nodes
- 重複步驟直到：

１. 每個子節點中都只有一個樣本或者多個相同的樣本

２. 樹的高度達到l

# Algorithm Introduction




(c) Average path lengths converge

- 正常點的路徑長度大於異常點的路徑長度

- 這個演算法假設異常點很少而且與眾不同(容易被孤立)

# Algorithm Introduction (iForest)

**Algorithm 1** : $iForest(X, t, \psi)$

**Inputs**: $X$ - input data, $t$ - number of trees, $\psi$ - sub-sampling size

**Output**: a set of $t$ *iTrees*

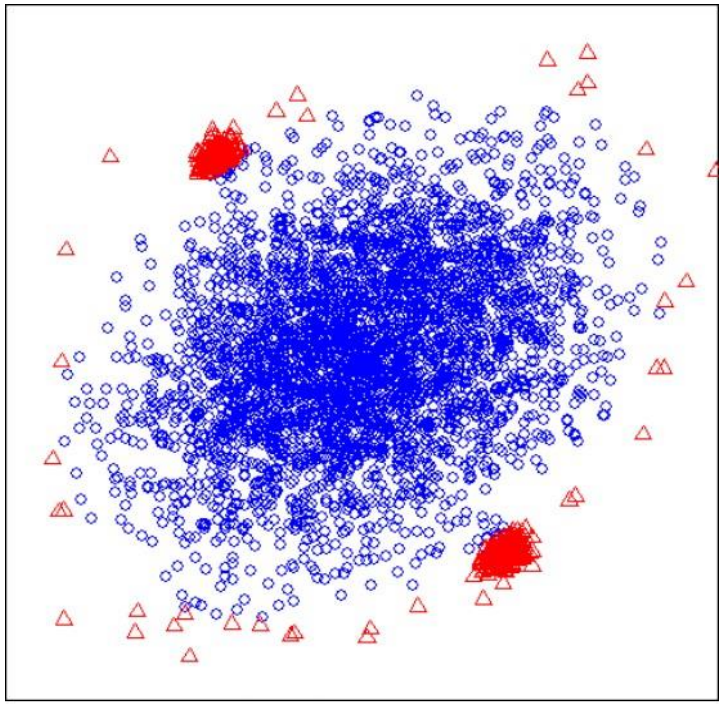
1: **Initialize** $Forest$
2: set height limit $l = ceiling(\log_2 \psi)$
3: **for** $i = 1$ to $t$ **do**
4: $X' \leftarrow sample(X, \psi)$
5: $Forest \leftarrow Forest \cup iTree(X', 0, l)$
6: **end for**
7: **return** $Forest$

●Sub-sampling可以緩解2個問題:

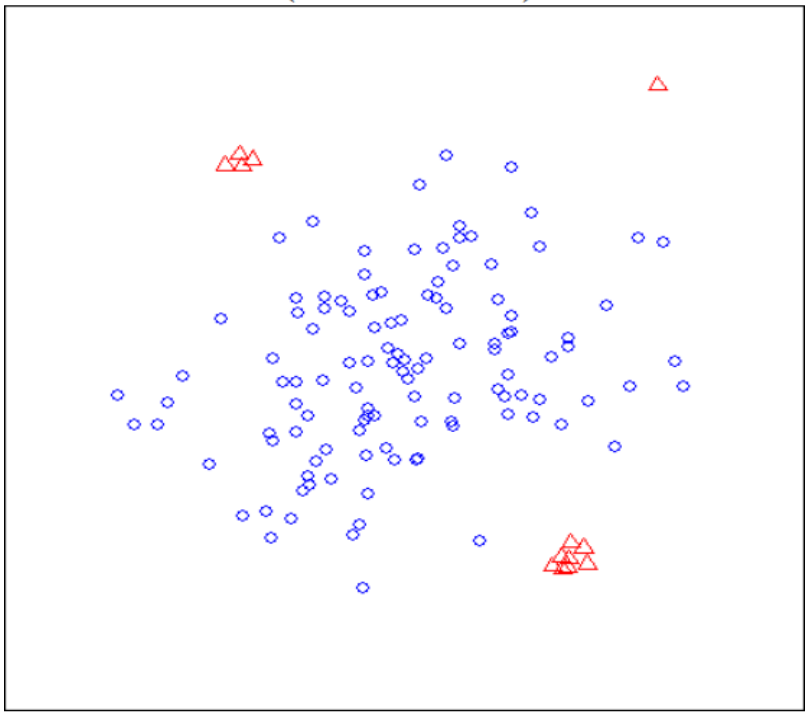
●1. Swamping:因為正常點和異常點靠太近，造成誤判。

●2.Masking:異常點聚集成一堆，變成也要很多次分割，導致被判定為正常點

# Algorithm Introduction



(a) Original sample



(b) Sub-sample

# Algorithm Introduction(obtain an anomaly score)

$$s(x,n) = 2^{-\frac{E(h(x))}{c(n)}},$$

- n:dataset裡的instance總個數
- x:要計算的instance
- E(h(x)):資料x在多棵iTree的路徑長度的平均值
- h(x):從根節點走到葉節點經過的邊數

c(n):n個資料建構的二元樹的平均路徑長度，在這裡用來做歸一化

# Algorithm Introduction(obtain an anomaly score)

- (a) if instances return $s$ very close to 1, then they are definitely anomalies,
- (b) if instances have $s$ much smaller than 0.5, then they are quite safe to be regarded as normal instances, and
- (c) if all the instances return $s \approx 0.5$, then the entire sample does not really have any distinct anomaly.

# Algorithm Introduction(obtain an anomaly score)

**Algorithm 3** : $PathLength(x, T, e)$

**Inputs** : $x$ - an instance, $T$ - an iTree, $e$ - current path length; to be initialized to zero when first called

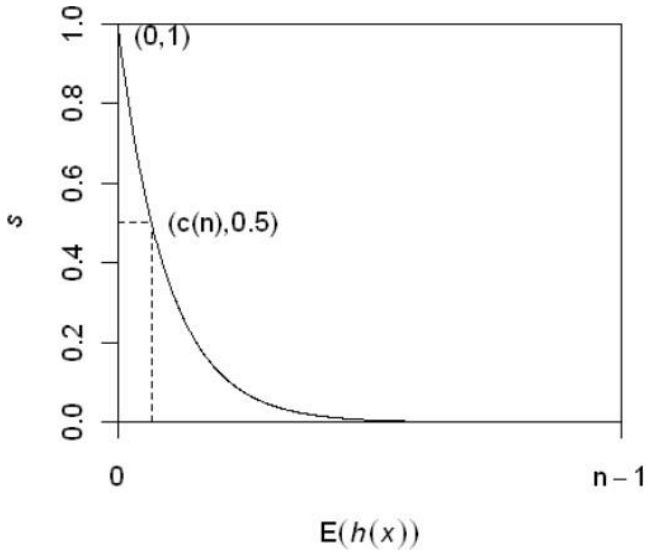
**Output**: path length of $x$

1: **if** $T$ is an external node **then**
2: return $e + c(T.size)$ $\{c(.)$ is defined in Equation 1$\}$
3: **end if**
4: $a \leftarrow T.splitAtt$
5: **if** $x_a < T.splitValue$ **then**
6: return $PathLength(x, T.left, e + 1)$
7: **else** $\{x_a \geq T.splitValue\}$
8: return $PathLength(x, T.right, e + 1)$
9: **end if**

- c(T.size)是一個修正值，表示在一棵用 T.size 個資料建的二元樹的平均路徑長度

- e表示x 從 iTree 的根節點到葉節點過程中經過的邊的數目

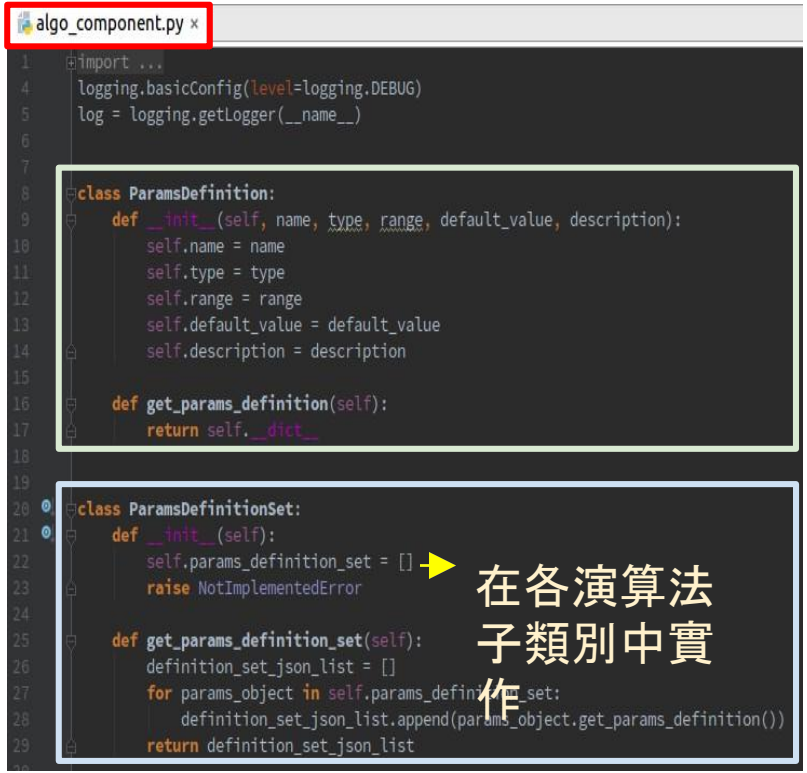
# Algorithm Introduction




- E(h(x))越小，s分數越高(異常程度越高)
- E(h(x))越大，s分數越低(越可能是正常點)
- 補:這個演算法就算訓練資料沒有異常的資料也可以跑

# Code Review

簡單說明程式架構，參考如下



# Code Review(2)

```
class ParamsDefinitionSet(alc.ParamsDefinitionSet):
    def _init_(self):
        self.params_definition_set =\
            [
                alc.ParamsDefinition(name='n_estimators', type='int', range='', default_value='100', description=''),
                alc.ParamsDefinition(name='max_samples', type='int', range='', default_value='auto', description=''),
                alc.ParamsDefinition(name='random_state', type='int', range='', default_value='1', description=''),
                alc.ParamsDefinition(name='contamination', type='float', range='0,1', default_value='0.1', description=''),
            ]
```

- n_estimators:iTree的數量
- max_samples:採樣的大小 預設:265
- contamination:異常點的比例
- random_state:隨機種子

# Code Review(3)

```
mode = input.algo_control.mode
data = input.algo_data.data
if mode == 'training':
    try:

        model=IsolationForest(
                n_estimators=control_params["n_estimators"],
                max_samples=control_params["max_samples"],
                random_state=control_params["random_state"],
                contamination=control_params["contamination"]
        )

        model.fit(data)
        algo_output = alc.AlgoParam(algo_control={'mode': 'training', 'control_params': ''},
                                    algo_data={'data': data, 'label': None},
                                    algo_model={'model_params': model.get_params(), 'model_instance': model})
```

- 開始訓練

# Code Review(4)

```
#測試有沒有1 或 -1 以外的值，有就代表有錯
def judge_predict_result(predict_result):
    n = True
    for i in range(0, len(predict_result)):
        if (predict_result[i] != 1 and predict_result[i] != -1):
            n = False
    return n
```

- 檢測預測結果有沒有1或-1以外的值
- 如果有就是false，代表有誤

# Code Review(5)

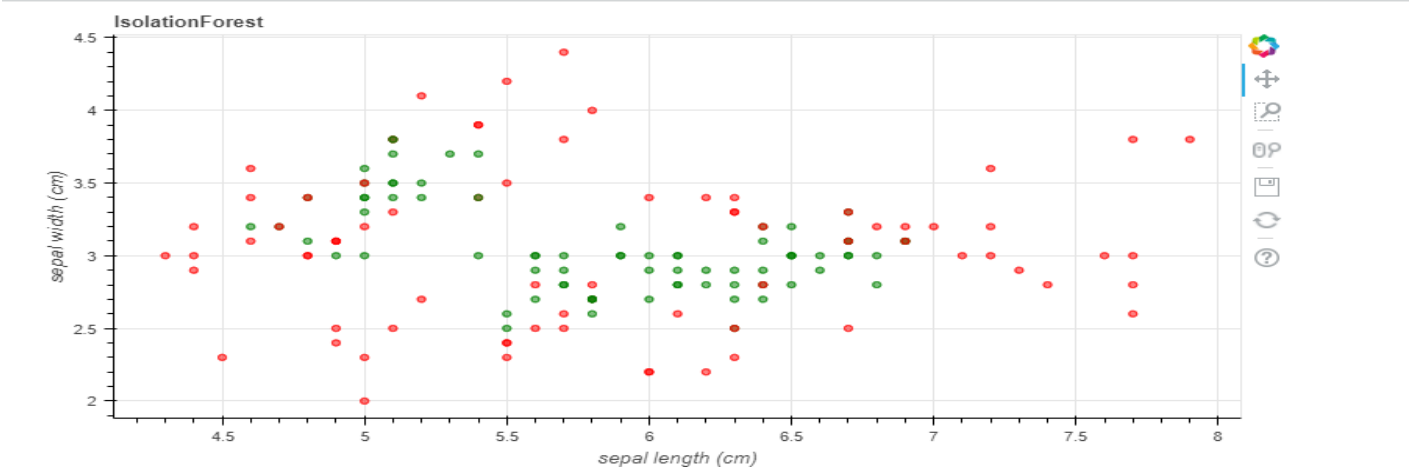
```
#correct test
def test_iscorrect_isolation_forest(self):
    data = load_iris()
    arg_dict = {
        "n_estimators": 100,
        "max_samples": 'auto',
        "random_state": 1,
        "contamination": 0.5
    }
    iris_data = pd.DataFrame(data.data, columns=data.feature_names)
    iris_label = data.target
    algo_input = alc.AlgoParam(algo_control={'mode': 'training', 'control_params': arg_dict},
                               algo_data={'data': iris_data, 'label': iris_label},
                               algo_model={'model_params': None, 'model_instance': None})
    in_algo = AlgoUtils.algo_factory('Isolation-Forest')
    algo_output = in_algo.do_algo(algo_input)
    model = algo_output.algo_model.model_instance
    predict_result = model.predict(iris_data)
    judge=judge_predict_result(predict_result)
    self.assertTrue(judge is True)
```

- 把預測結果放入函式中

# **Bonus: Model Preview

呈現模型圖結果，並簡單說明實作方法或使用的套件

實作方法:正常點標成綠色，異常點標成紅色 (使用bokeh)



# Live demo

報告當天使用InAnalysis完成live demo：http://ntuesoe.com:8008/

步驟：

1. 上傳期中專案使用的訓練資料
2. 點選自己新增的演算法訓練
3. 口頭講述訓練結果、預測結果

$$F_1 = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}.$$

# Live demo(和期中專案比較)

One-class-svm : nu:0.05 , gamma:0.05 , kernel:rbf , degree:2

Step 2: Show Prediction Results

| | Abnormal(predict) | Normal(predict) |
|---|---|---|
| Abnormal(actual) | 447 | 60 |
| Normal(actual) | 28 | 464 |

isolation_forest : max_samples:auto , contamination:0.1 , random_state:1 , n_estimators:100

Prediction Results: Confusion Matrix

| true/predict | -1 | 1 |
|---|---|---|
| -1 | 468 | 24 |
| 1 | 169 | 338 |

# Conclusion

經過了這學期的練習，我學會了新的機器學習演算法( isolation forest 和 one class svm )，也了解了什麼是單元測試，也大致對機器學習如何用在處理異常偵測的問題上有了一點概念，另外老師在上課中講到的一些機器學習的演算法跟應用的範圍也讓我對機器學習的認知變得更廣泛，知道了機器學習的應用範圍，讓我對於這領域有了更強的學習動力，也讓我了解到了自己哪裡需要加強。

# Reference

参考

http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html#sklearn.ensemble.IsolationForest

http://blog.csdn.net/qq_25231283/article/details/77987717

https://www.jianshu.com/p/d20e4e6a4b0a