Machine Learning 2016 Fall
Final Project Report

Topic:

Cyber Security Attack Defender

- total submissions: 38 times
- highest score: 0.96179

Team Name:

NTU_b02902012_龍が我が敵を喰らう！！！

Teammates:

b02902096 王浩恩 b0292086 白謹瑞 b02902012 林子翔 b02902004 危存愷

Work Division:

組員	負責部分
王浩恩	Gradient Boost/xgboost
白謹瑞	Random Forest
林子翔	Dimensional Reduction
危存愷	Neural Network

所有組員都高度參與final project的完成，並且平均分配report的撰寫。

Preprocessing and Feature engineering:

1.Label Data:

先從training_attack_types.txt將attack的分類建成可對照的dictionary，並加入normal這個種類。所以總共會有五種分類：

normal	0
DOS	1
R2L	2
U2R	3
probing	4

每一筆data的feature共有42維，其中最後一維是connection的type。所以我們先把它抓出來，一一對照dictionary後存成label。

2.處理字串：

接著處理data前面的41維，如果是數字就先放著，但有的feature分類是英文字，像

tcp、http等。將英文feature取出來，計算共有幾種，再將各種類編號，用編號數字填回原本的位置，這樣所有features就都會是數字，之後把這些數字轉為one-hot形式，然後接回去原本的training data.

如此一來就建好了training data和label data，接下來就把這些data套用到各種不同的model。

Neural Network:

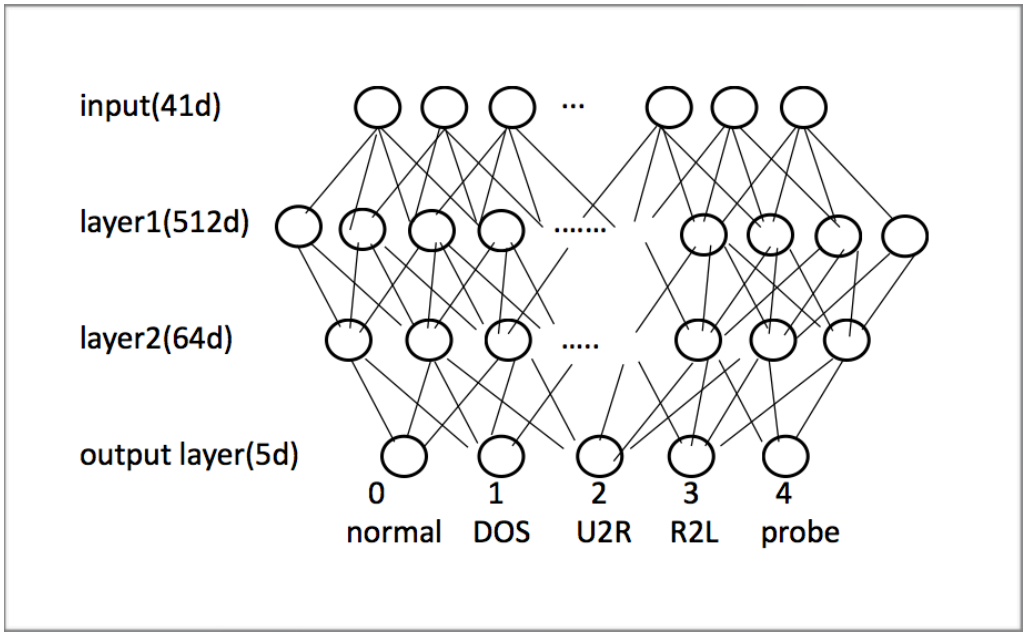
- 使用套件：Keras

我們首先嘗試的是NN，在結構上使用了兩層hidden layers，tanh activation和adam optimizer，詳細的參數如下。

```
model = Sequential()
model.add(Dense(output_dim=512, input_dim=41, init='uniform', activation='tanh'))
model.add(Dense(output_dim=64, init='uniform', activation='tanh'))
model.add(Dense(output_dim=5, init='uniform'))
model.add(Activation('softmax'))
```

此model的validation可以到99.98%，kaggle test可以到95%以上。

- NN示意圖：



Experiment and discussion：

- Finding:

用NN model做classification，會發現output出來的結果幾乎沒有2和3這兩個classes。回頭檢查train data後，發現原本data的分類數量就極為不平均，導致數量太少的class根本沒辦法train到。

- Solution 1:

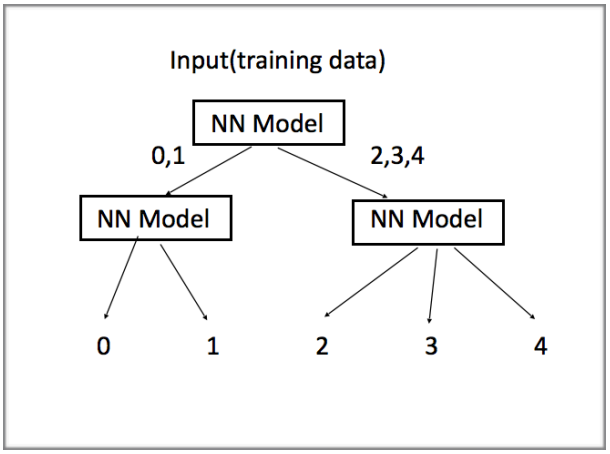
所以首先，我們做了data duplication，將數量較少的data複製好幾倍，來稍微平衡train data裡分類的分布。接著用同樣的model重新train，效果就有明顯提升，原本不會出現的2、3就有出現在predict出來的結果裡。Validation可以達到99.99%，kaggle test也上升到約96%，但這樣做還是無法突破strong baseline。

- Solution 2 (strong baseline):

為了讓數量較少的classes分得更準確，接下來我們把原本的model切成兩階段分類(架構參照下圖)。第一階段先用一個NN model將原本就佔大多數的0、1和較少數的2、3、4做二分，第二階段再用兩個NN model分別細分出最後共5個classes。這樣做出來的validation基本就是100%，kaggle test也達到96.11%，順利達到strong baseline。

- Solution 2的結構示意圖：

兩階段classification:



- Solution 3: ensemble

最後有嘗試將多個NN model ensemble起來，performance會再上升一些，但還是沒有Gradient Boost來的好。

- Performance of NN:

solutions	kaggle score
solution 1	0.96101
solution 2	0.96102
solution 3	0.96109

Random Forest:

- 使用套件:sklearn
- model description:

random forest model 是一個包含多個decision tree的classifier，分類的結果將由每個decision tree個別輸出的類別的眾數而定。 而我們選用sklearn中RandomforestClassifier來實作，若training data總數為N，則對於random forest中每棵樹而言，隨機且有放回的從training data中的抽取N個訓練樣本，作為該樹的訓練集，這是bootstrap方法，也是sklearn的random forest classifier的預設方法。每次樹進行分裂時，從全部41個特徵中隨機選擇m

個特徵，決策樹上每個節點的決定都是基於這些特徵確定的，根據這m個特徵，計算其最佳的分裂方式，每棵樹也以最大程度進行生長，沒有剪枝的動作。

Experiment & Discussion:

在我們初版的random forest我們只調整了tree的數量，設n_estimators=100，最後得到預測結果的準確率為0.959~0.96附近。

之後我們發現，因為在training data中各個class的資料數量並不一致，因此我們也在建立classifier的時候調整了class_weight的參數希望能減少資料數量落差帶來的影響，調整的方式參考文件上的方法，每個class的權重分配依照該class資料的數量倒數做比例分配。

另外，因為觀察到training set資料的特性，我們發現類別2的攻擊除了資料量少之外，在預測時也較少被歸類在類別2，因此只要預測結果中類別2的機率不為零時，我們會將預測結果判定成類別2。加入class_weight以及手動對類別2的預測做調整之後，準確率提升至0.961~0.962，並無十分顯著差異。

Performance:

n_estimator	class_weight	kaggle score
100	x	0.95959
200	x	0.96065
100	training set	0.96011
200	training set	0.96021

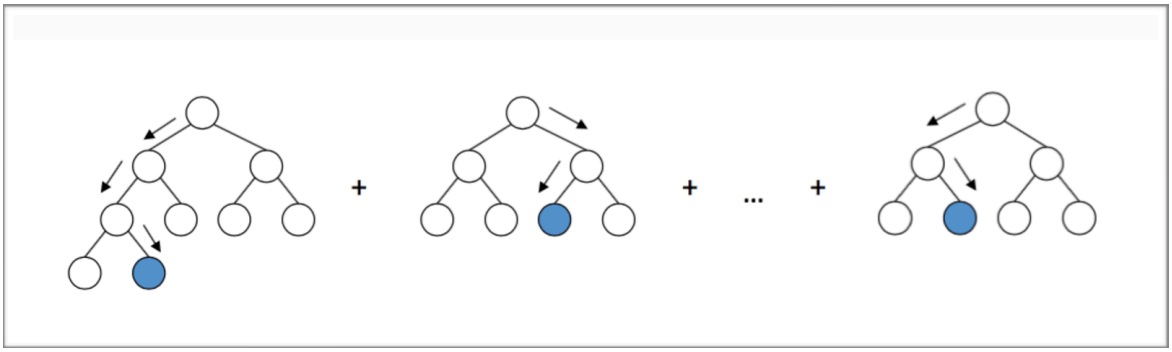
Boosting 系列：

在嘗試NN與random forest以後，我們轉而嘗試boosting method，在這次的Task中我們使用了以下兩種Boosting的方法/套件:

- *Gradient Boost Classifier from sklearn*
- *Xg-boost*

兩者都有良好的效果且皆超過了strong baseline，以下會分別介紹這兩個方法。

1.Gradient Boost:



- 使用套件：sklearn
- 簡介：

因為寫final時老師還沒教到ensemble，所以就自己先研究了，最一開始先從Boosting研究起，我認為Boosting的核心概念就像是『三個臭皮匠，勝過諸葛亮』，我們可以把多個小的classifier（weak learner)，合併起來變成一個大的classifier（strong learner)，每個weak learner的貢獻可能不大，但是只要他們binary classify的錯誤率 < 50%，也就是比亂猜好，ensemble起來就可以變成一個strong learner，而且理論上錯誤率可以降到0%，但實際上很難達成。sklearn的Gradient Boost中的classifier預設是regression tree，要minimize的objective function是 mean squared error。

在Gradient Boosting演算法中，是每一輪拿一個learner加進來原本的ensemble裡面，那選這個learner的原則就是，要讓這個learner盡量cover到target function跟目前model的預測分數的相減——也就是要train一個新的learner去重建出residual，如下圖所示。f 是在Function Space F中的一個function，F是所有可能的樹的集合。

$$\begin{aligned}\text{obj}^{(t)} &= \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{i=1}^{t} \Omega(f_i) \\ &= \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) + constant\end{aligned}$$

2.Xgboost:

使用套件：Xgboost

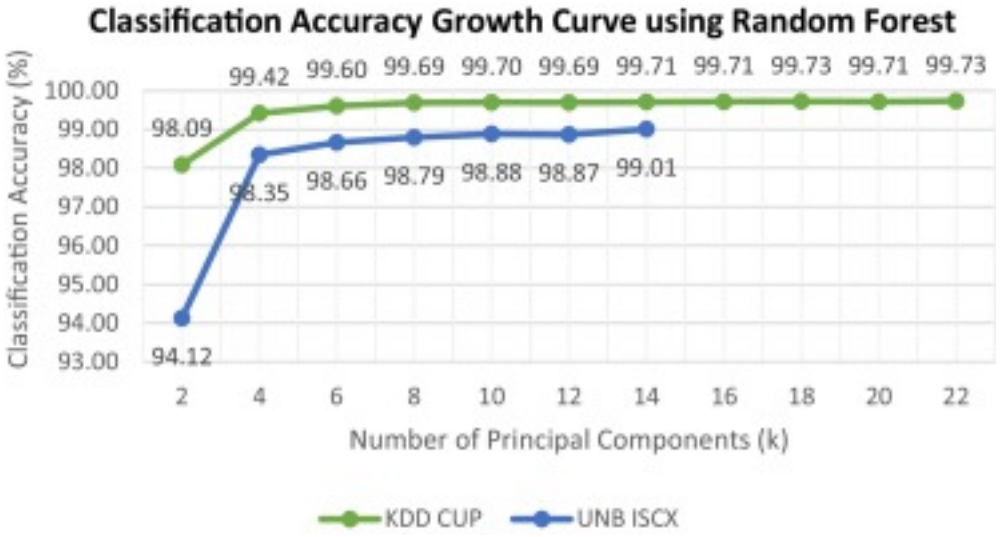
簡介：Xgboost是一套奠基於Gradient Boosting的套件，並且實作了許多optimization，其目標是提供一個scalable, portable, accurate的程式庫。

Performance：皆突破Strong Baseline

Model	n_estimator	other_parameters	Kaggle score
Gradient Boost	100	default	0.96163
Gradient Boost	200	default	0.96179
Xgboost	100	max_depth = 6	0.96147
Xgboost	200	max_depth = 6	0.96165

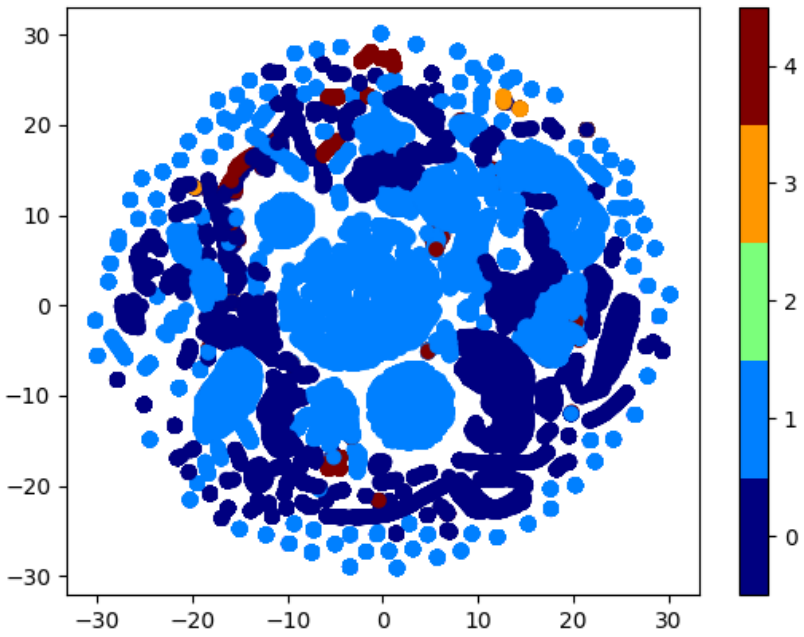
TruncatedSVD：

後來我們嘗試往這個dataset去找資料，查到了一篇關於這個dataset的論文，發現他們用PCA降低維度。原本的dataset的維度是41維，他們把維度降到8~12維就有非常好的效果，因此我們除了三種model之外，我們也嘗試了sklearn的TruncatedSVD。試著把三種不同的model都加上這個LSA的方法看看會不會有在準確率上有改進。我們加上了svd把維度降到8～12維，結果在每種model上都比較差，大概在0.94~0.95之間。

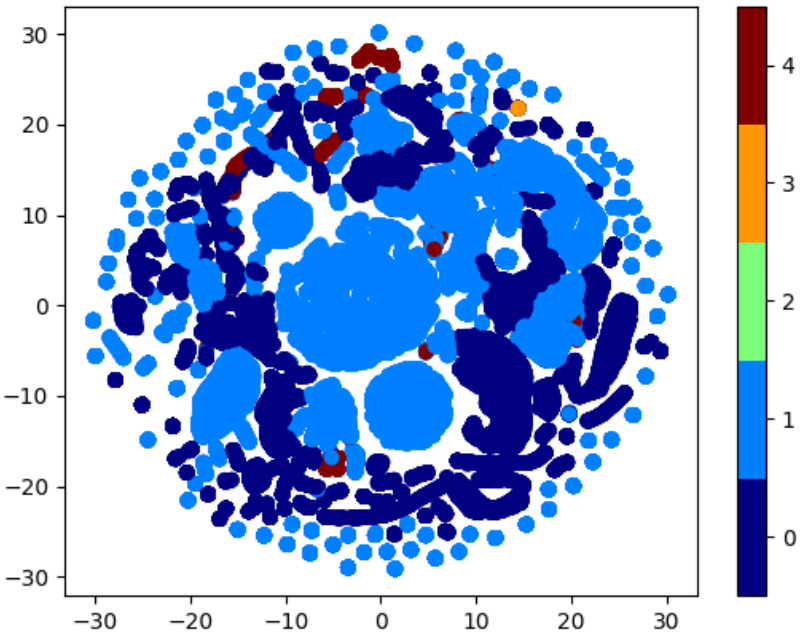


Data Visualization:

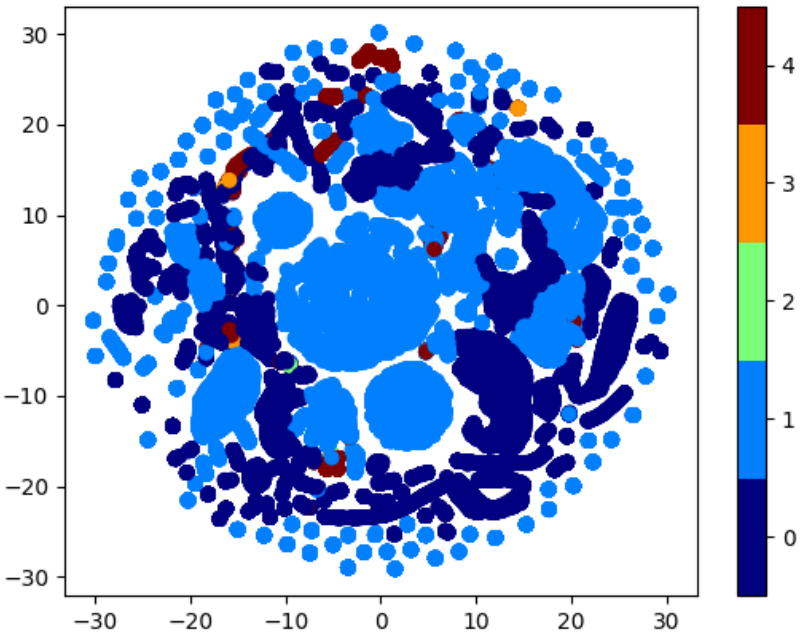
- **Neural Network**



- **Random forest:**



- **Gradient Boost:**



Observation:

以上三張圖是使用testdata取100000筆，利用tsne library畫在二維平面上，可以發現performance較佳的兩種model：NN, Gradient Boost，在左上角都可以發現橘色的點點，也就是第三種attack，在之前的觀察中有發現，第二、三種的training data非常少，prediction中也很少會有第三種，此外在Gradient Boost的圖中還看得到綠色的點，我們推斷影響最後kaggle score的因素就是預測出第二、三類型attack的多寡，以上三張visualization更能夠印證我們的推論。

Model Comparison:

以下取出各種model最佳的成績來做比較：

model	score
Gradient Boost	0.96179
Random Forest	0.96021
Neural Network	0.96109

Reference:

- Dimensionality reduction using Principal Component Analysis for network intrusion detection: http://www.sciencedirect.com/science/article/pii/S2213020916301446
- Gradient Boost: http://arogozhnikov.github.io/2016/07/05/gradient_boosting_playground.html