Machine Learning Final Report

洪仲言 B00201015
蔡佳文 B00201025

January 21, 2015

1 Algorithm

1.1 Linear Model

1. *Logistic Regression*
2. *Ridge Regression*
3. *Linear SVM*

1.2 NonLinear Model

1. *SVM*
2. *Random Forest*

2 Preprocess

2.1 Image

1. *Resize*
   因為我們看見大家寫的字都東倒西歪的，如果直接將抓下來的資料拿來訓練一定很慘烈。
   所以我們用了兩種方式進行處理。
   (a) **將圖片用一個四邊形逼近，以減少太多空白的部分。只留下四邊形後，接著放大成** 122*105
   (b) **將圖片用一個四邊形逼近，以減少太多空白的部分。把四邊形移到圖中心。為什麼會想這樣做呢？因為擔心放大後失去字的結構。可能『龍』這個字會因為放大，整團擠在一起。**

2. *HOG*
   «Histograms of Oriented Gradients for Human Detection» 是在 2005 年 CVPR 上發表的
   想用這個方法的原因是：就如同論文想要找到人在圖片裡麵和其他物體的互動（車子之類的），那他可是用梯度的方法找到人和車子。那我們是文字，我們拿到的資料裡面只有字還有一堆空白處，所以我們如果成順利找到字，並且將文字和空白分離這樣也許可以解決大家同樣的字在 122*105 裡面，出現在不同位子的情況。

3. *Scan Line*
   因為我們想處理的資料是中文字，如同大家所知道的，中文字是以字根或筆劃所構成，我們便嘗試以此為出發點之一。然而對字根的辨別太過困難，幾乎等同於對於較小的資料作中文字辨識，相當困難，故我們選用筆劃作為特徵。一個中文字的筆劃有許多方向，且多是接近直線，如「大」字就有三個方向的略為彎曲筆劃。我們就以圖中出現的各個方向的直線線段作為特徵。例如 $L =$ 圖片 $\cup$ 某直線線段, 則基於此直線線段的特徵為 $f(L) = \frac{\sqrt{\sum len(l_i)^2}}{len(L)}$, 其中 l_i 為直線線段上的連續白色子線段 (即有與字重疊的部份)。我們希望能透過筆劃與筆劃之間的關聯，使的機器學習到「字」的樣子，進而在結果上有好的表現。

2.2 Class

1. **合併**
   因為在 track0 上面，『一』判斷成『壹』也算是得分，反之亦然。所以我想說可不可以在 class 上面降維，把所有 class > 21 的都減掉 10。但是結果似乎不太理想，可能令 model 感到疑惑了。

3 Bagging and Blending

3.1 Bagging

1. 對 linear svm 進行 Bagging 100 參數 C: 1
   效果不錯 0.28 $\rightarrow$ 0.267

2. 對 kernel svm 進行 Bagging 100 參數 kernel: rbf, C: 100,γ: 0.1
   實驗進行到一半……收到網管的信 memory leaks QQ ，雖然比賽很重要，但是跟實驗室學長姐的感情也很重要所以忍痛放棄這個實驗。

3. 對 ramdom forest 進行 Bagging 100 參數 870 顆樹
   Random Forest 本身就是一個 Bagging 的演算法了，想說試試看會不會發生什麼怪

異的事情，但是跑了好久還沒跑完，所以在四天後放棄這個探險。

4. 對 LogisticRegression 和 RidgeRegression 進行 Bagging 100 參數各為 c=10 $\alpha = 10$

3.2 Blending

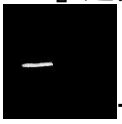
1. 對進行公平投票的方法，看哪個過半數，如果都沒有的話，隨機選一個答案
   <*SVM:* kernel=rbf C=125 $\gamma = 0.12$>
   <*Random Forest:* tree=870 max_feature=sqrt>
   <*SVM linear:* c=1 bagging=100>
   成果有進步 0.24 (三個臭皮匠勝過一個臭皮匠)

2. 對進行公平投票的方法，看哪個過半數，如果都沒有的話，選一個較多的，如果有一樣票數最多的，在隨機選一個
   <*SVM:* kernel=rbf C=125 $\gamma = 0.12$>
   <*Random Forest:* tree=870 max_feature=sqrt>
   <*SVM linear:* c=1 bagging=100>
   <*Logistic regression:* c=1>
   <*Ridge regression:* c=10>
   成果有進步 0.25 但是相較於上一個 Blending 竟然退步了，可能是因為人多口雜，好的事情被淹沒了

3. 對進行公平投票的方法，看哪個過半數，如果都沒有的話，選一個較多的，如果有一樣票數最多的，在隨機選一個
   <*SVM:* kernel=rbf C=125 $\gamma = 0.12$>
   <*SVM:* kernel=rbf C=100 $\gamma = 0.12$>
   <*Random Forest:* tree=870 max_feature=sqrt>
   <*SVM linear:* c=1 bagging=100>
   <*SVM linear:* c=1 >
   <*Logistic regression:* c=1>
   <*Ridge regression:* c=10>
   成果沒進步 0.29 但是相較於上一個 Blending 竟然退步了，果然太多臭皮匠會讓事情變糟

4 Best Result

經過一連串的時候發現還是原始的資料搭配使用 HOG 的效果最好。反而我自己 Resize 後在使用 HOG 的效果沒那麼好，可能是因為我的 Resize 用壞原本的資料形狀了。
而在原始資料上做 Random Forest 是表現最好的，不做 HOG 時可以到 0.65 其他演算法都在 0.7 0.8 遊蕩。Overfitting 非常嚴重，我認為是因為在為處理的 data 非常雜亂。光看『一』這個字:


一個寫在正中間

 一個寫到天上去了
這樣會讓演算法學不好，因為對他們來說一個像素是一個 feature 所以如果演算法學到在 100 到 110 之間有值是『一』。那寫在不同位子的『一』可能會讓演算法誤判
但是一做 HOG 之後，都用向量表示，這樣子雜訊就變少，而 SVM 展現自己的價值，馬上變成最強的演算法了。
而 Scan Line 的部份，則因負責生的人寫的太慢了，導致能用來測試的時間很少。不過在結果上大多是贏過 hog 的，或許是因為他的 feature 是特別對文字產生的？hog 有些特性在文字上比較無法體現，像是對圖片處理顏色、光影變化等，在文字中就會無法使用。
目前 SVM 與 Scan Line 最匹配，效果最好，與 hog 類似。不過他跟 random forest 就相當不合拍，其 E_out 也大幅度的超越了 logistic regression 。
或許我們可以稱為「不同的 feature 對不同的 learner 適應性可能有顯著差異」。

4.1 For each algorithm with HOG

algorithm/with hog	c	gamma	kernel	bagging	tree	e_out
SVM	100	0.12	rbf			0.26
SVM	125	0.12	rbf			0.26
LinearSVM	10					0.28
LinearSVM	10			100		0.27
LogisticRegression	10					0.32
LogisticRegression	10			100		0.29
RidgeRegression	alpha=10					0.44
RidgeRegression	alpha=10			100		0.38
RandomForest					870	0.28

4.2 For Scan Line

algorithm/with scan line	c	gamma	tree	e_out
SVM	8	0.007		0.187
RandomForest			870	0.27
LogisticRegression	10			0.215

4.3 For each blending

Blending	e_out
RF+SVM_c=125+LinearSVM	0.24
RF+SVM_c=125+LinearSVM+LogisticReg+RidgeReg	0.25
RF+SVM_c=100+SVM_c=125+LinearSVM+LinearSVM_bag +LogisticReg+RidgeReg	0.29

4.4 Best Approach

So for both of track0 and track1: the best approach is Blending: SVM + RandomForest + LinearSVM
這個 Blending 方法的優缺點 (我相信如果是搭配 Scan Line 的 feature 應該效果會更好。目前的實驗結果是搭配 HOG 的)

1. **優點：使用** Blending **的方法，可以達到三個臭皮匠勝過一個諸葛亮的境界，因為各自的演算法有他自己的盲點，多一點不一樣的意見可能會讓演算法表現得更好。對於最佳的演算法（**SVM**）進行每一筆的預測：錯誤的地方也許會被其他演算法給糾正，而如果是對的地方那其他演算法也能如願支持的話那更容易是對的，所以這個** Blending **的方法是會讓成果更進步的。**

2. **缺點：就是你必須先** Train **出三個** Model**，這將會耗掉非常多的時間。而且公平投票這個方法是否真的是正確的呢？是不是比較強大的** Model **應該要有比較大的分數呢？難道多數就是對的？也許有方法可以調出更好的權重，票票等值也許是有問題的。**

5 工作分配

1. **洪仲言：**HOG, Bagging, Blending, Random Forest, SVM, LinearSVM

2. **蔡佳文：**Scan Line, Resize, LogisticRegression, RidgeRegression