Machine Learning Assignment 1 Report

李漪莛 B03202017 物理三

Data Selecting

1. Train data

   將所有資料以每 10 小時為一筆，為了得到最多組訓練數據，因此每相隔 1 小時當成一筆數據；並排除前一個月的 20 號到下個月的 1 號，中間不連續的 9 小時的數據。

   則共有 5652 筆資料。12 個月，每月 20 天，每天 24 小時，再扣除每月最後 9 筆不連續的數據 → 12 x（24 x 20 – 9）= 5652。

2. Cross validation data

   測試時將每月最後 120 筆（共 1440 筆，約佔總資料量的 25%），分隔開當作 cross validation data，供評估用；待訓練好後再把這幾筆資料合併回原本的 train data。

   原因：最終測試時是用每個月的前 20 天訓練的參數去預測後 10 天的資料，因此我採用此方法，利用前 15 天訓練好的參數，去計算第 16~20 天的 Cross validation error（以下稱 CV error）。

Feature Set Selecting

1. 取出每個月其中一天的資料當樣本，共 240 筆，放入 excel。

   將這些資料也以每 10 小時為單位，所有第 10 小時的 PM2.5 值當 y 軸，所有第 9 小時的其他參數當 x 軸，計算所有 x-y 圖的相關係數 R，若|R| > 0.3，視為有效的 feature，加入 feature set。（右表顯示所有參數的 R）

   此時有效的 feature 有：（1, 3, 4, 6, 7, 8, 9, 12, 13, 14, 15, 16）

2. PM2.5：經測試後發現時間全取最佳，從第 1 小時到第 9 小時均加入 feature set。
3. 其他參數：經測試後只取前四小時，亦即第 6 小時到第 9 小時，效果最佳。

   推測原因：查到的資料顯示 PM2.5 的計算步驟包含空氣中某些氣體分子在前四小時的量。

4. 風向和風速：在 0 度到 360 度不連續，若針對風向用 cos 換算，或分組換算（例如：把 330~30

1	AMB_TEMP	0.791
2	CH4	0
3	CO	0.665
4	NMHC	0.714
5	NO	0.185
6	NO2	0.647
7	NOx	0.607
8	O3	0.746
9	PM10	0.846
10	PM2.5	x
11	RAINFALL	x
12	RH	-0.520
13	SO2	0.456
14	THC	0.753
15	WD_HR	0.538
16	WIND_DIREC	0.490
17	WIND_SPEED	0.076
18	WS_HR	-0.033

度當成一組），效果仍不佳，因此把風向這個特徵剔除，並加入風速效果較好。

5. 降雨量：初始資料若無降雨則非浮點數，一般降雨值約為 0.1~1 的數量級，原先測試把沒降雨的值調成 -100 至 -1 等不等的數，但訓練結果都不佳，因此移除。
6. 綜合 1~5 的結果：（1, 3, 4, 6, 7, 8, 9, 12, 13, 14, 17, 18）
7. 加入自己設定的一階及二階微分近似，CV error 可從 6.07 降到 5.84。
   一階近似：hr10（第十小時的 PM2.5）≅ 2*hr9 – hr8
   二階近似：hr10（第十小時的 PM2.5）≅ 3*hr9 – 3*hr8 + hr7
8. 最終 feature set 有 60 維，包括第一維的 bias，第 1 到第 9 小時的 PM2.5，第 6 到第 9 小時、各 12 組的其他特徵，兩個自己近似的值 → 1+9+48+2 = 60。

Linear regression function by Gradient Descent

1. Method

   實際使用 gradient descent 發現收斂速度不夠快，因此我結合了兩種方式

   - 前 1/2 的 iterator：使用 stochastic，可使前半段收斂快
   - 後 1/2 的 iterator：使用 Adam，可後半段達不到誤差最低點的情況。

2. Code

lost = 2×[sum((y - dot(x ・w))^2) + λ ×sum(w[1:-1]^2)]

if t<iterator/2: #在前 1/2 的 iterator，採用 Stochastic

$for\ i\ in\ range(100)$ # 把資料分成 100 組，每次用其中一組調整 weight 的值

$start = 56 \times (i\%100)$ # start, finish 使每次都取總資料量的 1/100（56 筆）左右

$if\ \ i\ \neq 99: finish = 56 \times (i\%100 + 1)$

$else: finish = x.shape[0]$ # 若最後一次（99 次）就取到最後一筆資料

$g = 2 \times \left(sum\left(-(y[s:f,:] - \ x[s:f,:], w)\right) \times x[s:f,:]\right) + \ \lambda \times sum(i_{matrix} \cdot w)$

i_matrix 是單位矩陣，但在 (0, 0) = 0，可避免計算到 bias 的 regularization

$w -= g \times \eta$ # 調整 weight

else: # 後半使用 Adam

$g = 2 \times [\ sum((-x \times (y - x \cdot w)) + \ \lambda \times sum(i_{matrix} \cdot w)]$ # sum：把內積完的值加起來

$m = \ \beta 1 \times m + (1 - \beta 1) \times g$

$v \ = \ \beta 2 \times v + (1 - \beta 2) \times g^2$

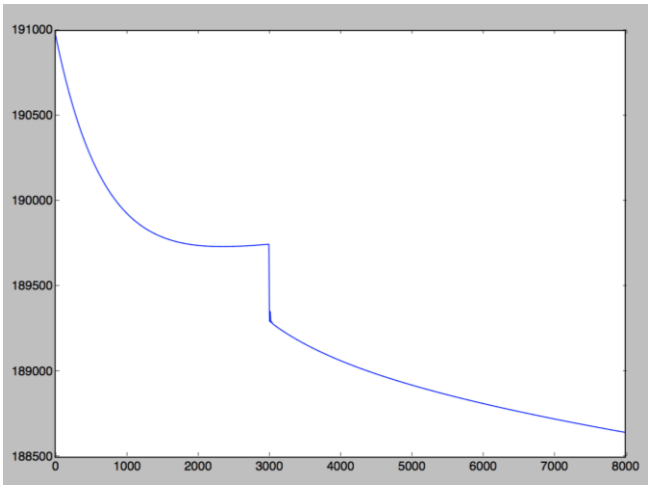
$\dot{m} = \ {}^{m}/_{(1 - \beta 1^t)}$

$\dot{v} \ = \ {}^{v}/_{(1 - \beta 2^t)}$

$weight = \ \eta\_adam \times \dot{m} \div (\ \dot{v}^{0.5} + 10^{-8})$ # 調整 weight，此處的η_adam，跟 Stochastic 採用的大小不同，但也是自己測試得到的。

Learning rate（η）

1. 前半段為使收斂夠快，因此調了一個洽不會發散的最大的 η = 3E-6。
   後半段的 η = 3E-7 較小，且每次還會除以與 t 相關的函數，使模型保證能在小小震盪中盡可能穩定的找到最低點。
2. 右圖為 CV error 對 iterator 作圖，中間不連續處洽為 Stochastic 轉成 Adam 方法時的交界處。
   圖中只顯示整個訓練過程的後 8/10，避免最前面的 error 過大，影響到座標軸縱軸的尺度，並且可明顯看出 Adam 方法與 Stochastic 相比，收斂得較慢但較穩定。
3. 利用此方法通常在 iterator 數量為 5000 時就能接近 error 最小值。



Regularization

iterator = 5000 時，針對不同的 lambda、三組不同的 weight 初始值得到的 cross validation error。									
lambda	0.01	0.03	0.1	0.2	0.3	0.6	1	3	10
w1	4.318	4.290	4.291	4.303	4.311	4.324	4.331	4.340	4.346
w2	4.323	4.291	4.293	4.304	4.312	4.325	4.330	4.340	4.345
w3	4.318	4.288	4.290	4.302	4.311	4.324	4.331	4.341	4.346
average	4.320	4.290	4.291	4.303	4.311	4.324	4.331	4.340	4.346

雖然統計顯示 lambda = 0.03 時，CV error 有最小值，但是過小的 lambda 可能導致 overfitting，則在 kaggle 上的 test error 反而不佳，因此最終在 kaggle 上採用的兩筆 lambda = 0.03 及 1。

參考資料

- Adam 方法：https://arxiv.org/pdf/1412.6980.pdf