Machine Learning (2016 Fall) HW1 Report

PM 2.5 Prediction by Linear Regression

Student : 沈怡廷 r05943001

Kaggle_ID : r05943001_Panda

A. Linear Regression Function by Gradient Decent

```
iteration_time = 2000
G_w            = np.zeros((2, len(select_list), 9))
G_w_s          = np.zeros((2, len(select_list), 9))
G_b            = np.zeros(2)
smooth         = 1e-8

for it in range(iteration_time):
  print it
  w_d   = np.zeros((2, len(select_list), 9))
  w_d_s = np.zeros((2, len(select_list), 9))
  b_d   = np.zeros(2)

  err = np.empty((12, 471))
  # traverse all data
  for i in range(12):
    for j in range(471):

      # check if rain
      b_hasRain = False
      if np.count_nonzero(data[i,j:j+9,10]) != 0:
        b_hasRain = True

      # prediction answer
        # order 0
      y_pred  = b[0] * (b_hasRain) + b[1] * (not b_hasRain)
        # order 1
      y_pred += np.sum(     b_hasRain *w[0]*((data[i,j:j+9,select_list] - mean[select_list,None])/std[select_list,None]))
      y_pred += np.sum((not b_hasRain)*w[1]*((data[i,j:j+9,select_list] - mean[select_list,None])/std[select_list,None]))
        # order 2
      y_pred += np.sum(     b_hasRain *w_s[0]*((data[i,j:j+9,select_list]**2 - mean_s[select_list,None])/std_s[select_list,None]))
      y_pred += np.sum((not b_hasRain)*w_s[1]*((data[i,j:j+9,select_list]**2 - mean_s[select_list,None])/std_s[select_list,None]))

      err[i][j] = (data[i][j+9][9] - y_pred)**2

      # gradient
        # order 0
      b_d[0] += (     b_hasRain)*(-2)*(data[i][j+9][9] - y_pred)
      b_d[1] += (not b_hasRain)*(-2)*(data[i][j+9][9] - y_pred)
        # order 1
      w_d[0] += (     b_hasRain)*(-2)*(data[i][j+9][9] - y_pred)*((data[i,j:j+9,select_list] - mean[select_list,None])/std[select_list,None])
      w_d[1] += (not b_hasRain)*(-2)*(data[i][j+9][9] - y_pred)*((data[i,j:j+9,select_list] - mean[select_list,None])/std[select_list,None])
        # order 2
      w_d_s[0] += (     b_hasRain)*(-2)*(data[i][j+9][9] - y_pred)*((data[i,j:j+9,select_list]**2 - mean_s[select_list,None])/std_s[select_list,None])
      w_d_s[1] += (not b_hasRain)*(-2)*(data[i][j+9][9] - y_pred)*((data[i,j:j+9,select_list]**2 - mean_s[select_list,None])/std_s[select_list,None])

  print (np.average(err))**0.5

  # update
  G_w   += w_d**2
  G_w_s += w_d_s**2
  G_b   += b_d**2
  w   = w   - (rate   /np.sqrt(G_w  +smooth))*w_d
  w_s = w_s - (rate_s /np.sqrt(G_w_s+smooth))*w_d_s
  b   = b   - (rate_b /np.sqrt(G_b  +smooth))*b_d

print w
print w_s
print b
```

B. Description of my best method on Kaggle

我選擇了 7 項 features (CO, NMHC, NO, NO2, O3, PM2.5, SO2)來 training，並對這些 feature 先進行 feature scaling。每次 iteration (2000 次)讓程式看過所有測試資料(12*471)後再更新參數 (利用 AdaGrad)。我將要預測的 PM2.5

model 成上述 7 項 features 一次方與二次方的 weighted sum (加上 bias)，同時將參數分為過去 9 小時有下雨與沒下雨 2 組，故總共有 7(/feature) *9(/hour) *2(/Rain or not)+2(/bias)=128 個 weight 需要決定。

C. Discussion on regularization

加入 regularization 的目的在於尋找較 smooth 的 function，避免 model 定的太過複雜，使得結果 overfitting 於 training data。

基於 Kaggle 上最好的版本，我有嘗試過加入 regularization，但無論是在 training error 上，或是 Kaggle public set 的 testing error 都會變差，可能原因包含 lambda 值調整的不正確，或者 model 還不夠複雜，導致加入 regularization 後 function 過於 smooth，效果變差。

Ex:

	Training error	Public set error on Kaggle
No regularization	5.64124873983	5.62701
Regularization with lambda = 0.001	5.64722272679	5.67550

D. Discussion on learning rate

```
47 rate           = np.zeros((2, len(select_list), 9)) + 1
48 rate[:,5,:]    = 10
49 rate_s         = np.zeros((2, len(select_list), 9)) + 0.1
50 rate_s[:,5,:]  = 1
51 rate_b         = np.zeros(2) + 10
```

Learning 方面，我有測試過許多方法，包括最原始單一的 learning rate (調整大小)，或者將二次方的 learning rate 調小，但 Kaggle 上面試出來最好的方法如上圖所示：我將 PM2.5 的資料之 learning rate 調大 (一次方=10，二次方=1)，其餘六種 features 的 learning rate 調小 (一次方=1，二次方=0.1)，理由是認為 PM2.5 與要預測的答案相關性應該較大。

E. Others things to be improved

a. Validation set

上週上課老師曾經有提過 validation set 的概念，如此可避免選擇 model 時被 public set 的結果誤導。我有測試過將每個月最後 20 筆資料保留起來做 validation，但發現最終 training 結果變差，故最終還是決定將所有資料拿來 training，以後應該避免如此。

b. Model selection

由於時間關係，並沒有測試其他更複雜的 model，如利用風向或風速等資訊來分類，或許就是與 Kaggle 排名前面的同學差別，需要再提升。