機器學習 作業一

Machine Learning HW1

R05943040 電子一 林家禾

1.(1%) Linear regression function by Gradient Descent.

```
rate = 0.1     # learning rates
lamb = 0       # regularization coefficient, 0 for no regularization
power_num = 1
w_num = 18*9*power_num
b = 0
b_table = np.zeros(iter_num)
w = np.zeros(w_num)
w_table = np.zeros([w_num,iter_num])
rmse_table = np.zeros(iter_num)
ada_b = 0
ada_g = np.zeros(w_num)
```

 # interation 外層迴圈

```
for ite in range(iter_num):       # iter_num：iteration times=10000
    grad_b = 0
    grad_w = np.zeros(w_num)
```

 # 累加 N 次求出

$\frac{\partial L}{\partial w}$、$\frac{\partial L}{\partial b}$

```
    for n in range(sample_num):   # sample_num = 5751
        train_this = train[0+18*n:162+18*n]
        ans_this = train[171+18*n]
```

#用現有**w**、**b**求出預測值$\mathbf{y} = \mathbf{b} + \sum_{i=0} \boldsymbol{w_i} \times \boldsymbol{x_i^n}$

```
        y = b
        for power in range(1,power_num+1):
            w_this = w[0+162*(power-1):162+162*(power-1)]
            entry_this = entry[0+162*(power-1):162+162*(power-1)]
            y += np.sum( entry_this * w_this * (train_this**power) )
```

 # 算出$\widehat{y_n} - (\boldsymbol{b} + \sum_{i=0} \boldsymbol{w_i} \times \boldsymbol{x_i^n})(-\boldsymbol{x^n})$ **& adagrad 累加**

```
        temp = 2*( ans_this-y ) * ( -1 )
        grad_b += temp
        for power in range(1,power_num+1):
            w_this = w[0+162*(power-1):162+162*(power-1)]
            entry_this = entry[0+162*(power-1):162+162*(power-1)]
            grad_w[0+162*(power-1):162+162*(power-1)] += ( entry_this * temp *
(train_this**power) + 2*lamb*w_this )
    ada_b += grad_b**2
    for i in range(w_num):
        if(entry[i]==0):
            ada_g[i] = 1
        else:
            ada_g[i] += grad_w[i]**2

# gradient descent
    b = b - rate * grad_b / (ada_b**0.5)
    b_table[ite] = b
    for i in range(w_num):
        w[i] = w[i] - rate * grad_w[i] / (ada_g[i]**0.5)
        w_table[i,ite] = w[i]
```

2.(1%) Describe your method. 因為我們沒限制你該怎麼做，所以請詳述方法 ex: 怎麼取 training feature (X,y).

(1) Training data 挑選

第一天 0~8 時、1~9 時、…、16~24 時、17~25 時(次日 1 時)、…

第二天 0~8 時、1~9 時、…

以此類推，共 5751 筆 training data

(2) Feature (X,y)取法：

最大 feature 數為 18x9 個，選取時依天數和不同化合物剃除部分：

天數：固定化合物，用 RMSE 衡量用前 9 天、前 8 天、…、前 1 天的 feature 預測的準度，以前七天的 error 最小

化合物：固定天數，每次只選取一種化合物進行 training，視 RMSE 越小代表相關性越高，最後選取 CH4、NO、NO2、NOx、PM2.5、SO2、THC 七種

3.(1%) Discussion on regularization.

經實測後正規化對於 error 並無實質上的幫助，在同樣條件下正規化系數越大 error 會越大，因此最後的版本並不使用正規化的技巧

探其原因，這可能是因為正確的模型遠比我們想像中複雜，還有很多因素沒有考慮到，因此 training data 對現在的模型來說過於不規律，才導致硬是平滑造成反效果

4.(1%) Discussion on learning rate.

經實測後學習率會與選取的 feature 數目、模型次方項等等因素有關，因此每調整一次其他變因，最佳的學習率極可能有些微的變化。

若不使用 adagrad，學習率有一明顯上界，在調大至界線前 error 可以漸小，只要超過該值 train 時就會失效使 error 不斷增加。以自己的情況最佳學習率大概在 10^(-8)上下

若使用 adagrad，學習率在某值為可得最小 error，變大或變小都會使 error 增加。以自己的情況最佳學習率大概在 0.1 上下

5.(1%) TA depend on your other discussion and detail.

(1) scaling

實測後結果反而較差，推測是因為各 feature 原本的平均值與標準差較大，將各參數 normalize 後造成對各參數的調整較為敏感反而震盪

(2) feature 二次項

實測後結果反而較差，實作上僅對關聯性較大的 feature 進行二次項修正，但結果都不如預期，推測應該是有更多的因素沒有考慮進來，增加二次項造成 overfitting