ML HW1 B02902096 資工四 王浩恩

1. Linear regression function by Gradient Descent

```
gradient[0] += -(2/N) * (target_pm - innerproduct)
for k in xrange(1,len(gradient)):
    gradient[k] += -(2/N) * data[i][k] * (target_pm - innerproduct)
Coef = Coef - learning_rate * np.array(gradient)#((np.array(ada))**(1/2))
#print "Coef = " + str(Coef)
```

2. Explanation :

我的Gradient Descent取的feature是，每小時18個features然後取9個小時，所以Weight會有18 * 9 = 162個，再加上常數項的bias，總共163項。並用第十個小時的PM2.5當成計算loss function的y。

然後我採用Stochastic Gradient Descent，每一個iteration隨機選取一筆，去做gradient descent. 原本採用一般的GD，但是跑的速度十分緩慢，50000個iterations在工作站上跑了一天之類的(結果居然是我的Kaggle Best)，所以才改用SDG。另外我在數學式中多除了一個N， N是training data的數量，讓數字不會變得太大。

3. Discussion on Regularization:

lamda	0.01	0.001	0.0001	0.00001
validation	13460	1846	-1346	1061

在我的測試中固定learning rate 跟 iterations 的情況下，validation的誤差隨著lamda減少而下降，其中kaggle score 最高的是lamda = 0.0001的情況。

learning rate	0.005	0.001	0.0001	0.00001	0.000001
validation	-6733	781	-371	554	5291

4. Discussion on learning rate.
固定regulariztion的lamda與iterations後，learning rate以0.0001的誤差最小。

5. Further discussion:

在做這次作業時先後做過幾種不同的作法

(a). 取feature:

最一開始使用1x18作為一筆data，後來改採用了9x18，因為覺得比較貼近test data的條件。切成9x18之後，原本沒有管跨月問題，結果跑出了自己最高的kaggle score，後來處理跨月問題後反倒沒有變高。

(b). Iterations:

一開始以為iterations越多就會越準確，殊不知我的Gradient Descent 跑的不夠快，居然跑50000 iterations跑了十幾小時，雖然有過public baseline但是翻去private就彈出界外了QQ。於是我改採stochastic， 變快許多，我便嘗試了一筆500萬iterations，結果kaggle score竟然降低，我想可能是overfitting，總之讓我理解到了iterations絕對不是越多越好！

(c). 永遠要備份code：

記得開始寫的第二天過了baseline之後就很開心地慢慢優化，結果沒想到一去不復返，不管是怎麼調learning rate跟iterations，就是回不去原本的分數，只能盡量貼近而已XD