

說明：請各位使用此template進行Report撰寫，如果想要用其他排版模式也請註明題號以及題目內容（請勿擅自更改題號），最後上傳至cool前，請務必轉成PDF檔，並且命名為report.pdf，否則將不予計分。

備註：

- 所有 advanced 的 gradient descent 技術(如: adam, adagrad 等)都是可以用的
- 第1~2題請都以題目給訂的兩種model來回答

學號：R12945060 系級：生醫電資所碩二 姓名：羅佳蓉

1. (1%) 解釋什麼樣的data preprocessing可以improve你的training/testing accuracy, e.g., 你怎麼挑掉你覺得不適合的data points。請提供數據(例如 kaggle public score RMSE)以佐證你的想法。

- 通過 valid 函數將 $PM2.5 > 20$ 的數據排除，減少了極端異常值對模型的影響，從而提升了訓練和測試集的一致性，模型的學習過程會更加穩定
沒特別排除異常值：

	my_sol.csv Complete · 4d ago	6.99913	<input type="checkbox"/>
---	---------------------------------	---------	--------------------------


排除異常值：

	my_sol.csv Complete · 2d ago	4.78645	<input type="checkbox"/>
---	---------------------------------	---------	--------------------------

- 選擇了與預測 $PM2.5$ 相關的特徵來進行訓練，避免雜訊特徵對模型的干擾
沒篩選相關特徵：

	my_sol.csv Complete · 4d ago	14.87418	<input type="checkbox"/>
---	---------------------------------	----------	--------------------------

篩選相關特徵：

	my_sol.csv Complete · 3d ago	3.53854	<input type="checkbox"/>
---	---------------------------------	---------	--------------------------

2. (1%) 請實作 2nd-order polynomial regression model (不用考慮交互項)。

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 \text{ 其中 } x^2 = [x_1^2, x_2^2, \dots, x_n^2]$$

(a) 貼上 polynomial regression 版本的 Gradient descent code 內容

```

48 def minibatch_2(x, y, config):
49     x_poly = np.hstack((x, x**2))
50
51     index = np.arange(x_poly.shape[0])
52     np.random.shuffle(index)
53     x_poly = x_poly[index]
54     y = y[index]
55
56     batch_size = config.batch_size
57     lr = config.lr
58     lam = config.lam
59     epoch = config.epoch
60     decay_rate = config.decay_rate
61     epsilon = 1e-8
62
63     w = np.full((x_poly.shape[1], 1), 0.1)
64     bias = 0.1
65
66     cache_w = np.zeros_like(w)
67     cache_b = 0.0
68
69     for num in range(epoch):
70         for b in range(int(x_poly.shape[0] / batch_size)):
71             x_batch = x_poly[b * batch_size:(b + 1) * batch_size]
72             y_batch = y[b * batch_size:(b + 1) * batch_size].reshape(-1, 1)
73
74             pred = np.dot(x_batch, w) + bias
75             loss = y_batch - pred
76
77             g_t = np.dot(x_batch.T, loss) * (-2) / batch_size + 2 * lam * w
78             g_t_b = loss.sum(axis=0) * (-2) / batch_size
79
80             cache_w = decay_rate * cache_w + (1 - decay_rate) * g_t**2
81             cache_b = decay_rate * cache_b + (1 - decay_rate) * g_t_b**2
82
83             w -= lr * g_t / (np.sqrt(cache_w) + epsilon)
84             bias -= lr * g_t_b / (np.sqrt(cache_b) + epsilon)
85
86     return w, bias

```

(b) 在只使用 NO 數值作為 feature 的情況下，紀錄該 model 所訓練出的 parameter 數值以及 kaggle public score.

```

Trained model parameters (weights): [[-0.4104965 ]
[ 0.39974413]
[ 0.40903547]
[-0.16608404]
[ 0.01145733]
[ 0.66206191]
[ 0.08918103]
[-0.23014964]
[ 0.04440307]
[-0.03761267]
[-0.02707201]
[ 0.02078605]
[ 0.0067105 ]
[-0.04342381]
[-0.00478789]
[ 0.02885827]]

```

Submission and Description

Public Score ⓘ

Select



my_sol.csv
Complete · now

6.44281



3. (6%) Refer to math problem:

https://drive.google.com/file/d/1c0ath1Un3Gw4RbwGTttI4neX-tcI_qUo/view?usp=drive_link