

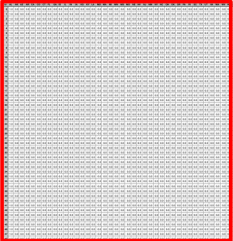
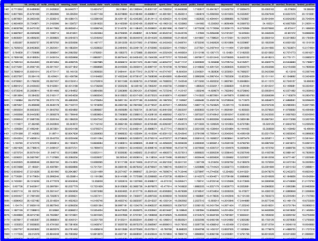
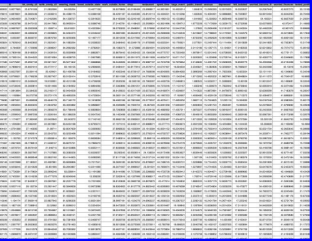
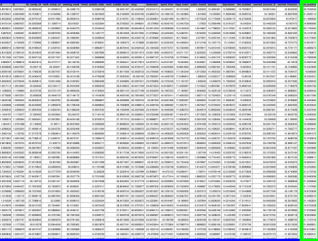

2023年5月2日 星期二

Homework 1

Covid-19 Cases Prediction

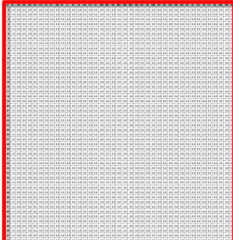
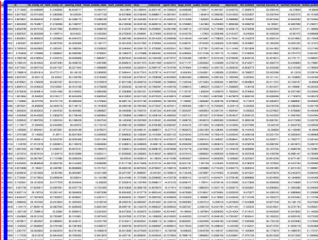
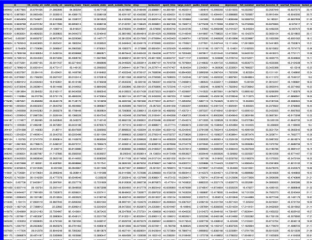
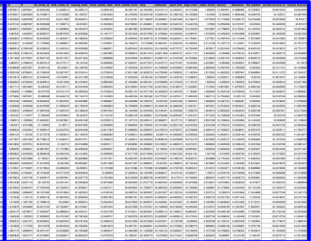
- Data-Training

- covid.train.csv

state one-hot encoding (40)	Day 1 features (18)	Day 2 features (18)	Day 3 features (18)	
				
1 row = 1 sample				tested positive

- Data-Testing

- covid.test.csv

state one-hot encoding (40)	Day 1 features (18)	Day 2 features (18)	Day 3 features (17)
			
1 row = 1 sample			

- Feature Selection

- 透過 sklearn 的 SelectKBest 以 r_regression 進行打分篩選出前25高的分數和對應之 index 。

	FeatureName	Score
75	tested_positive.1	0.991012
57	tested_positive	0.981165
42	hh_cmnty_cli	0.879724
60	hh_cmnty_cli.1	0.879438
78	hh_cmnty_cli.2	0.878218
43	nohh_cmnty_cli	0.869938
61	nohh_cmnty_cli.1	0.869278
79	nohh_cmnty_cli.2	0.867535
40	cli	0.838504
58	cli.1	0.838224
76	cli.2	0.835751
41	ili	0.830527
59	ili.1	0.829200
77	ili.2	0.826075
92	worried_finances.2	0.485843
74	worried_finances.1	0.480958
56	worried_finances	0.475462
91	worried_become_ill.2	0.267610

```
# Read data into numpy arrays
with open(path, 'r') as fp:
    data = list(csv.reader(fp))
    data = np.array(data[1:][:, 1:]).astype(float)

if not target_only:
    feats = list(range(93))
else:
    # TODO: Using 40 states & 2 tested_positive features (indices = 57 & 75)
    feats = [75, 57, 42, 60, 78, 43, 61, 79, 40, 58, 76, 41, 59, 77]

if mode == 'test':
    # Testing data
    # data: 893 x 93 (40 states + day 1 (18) + day 2 (18) + day 3 (17))
    data = data[:, feats]
    self.data = torch.FloatTensor(data)
else:
```

- NeuralNet

- Architecture:

```
def __init__(self, input_dim):
    super(NeuralNet, self).__init__()

    # Define your neural network here
    # TODO: How to modify this model to achieve better performance?
    self.net = nn.Sequential(
        nn.Linear(input_dim, 32),
        nn.BatchNorm1d(32), #使用BN，加速訓練
        nn.Dropout(p=0.2), #使用Dropout，減少過擬和
        nn.LeakyReLU(), #變更激活函數
        nn.Linear(32, 1)
    )

    # Mean squared error loss
    self.criterion = nn.MSELoss(reduction='mean')
```

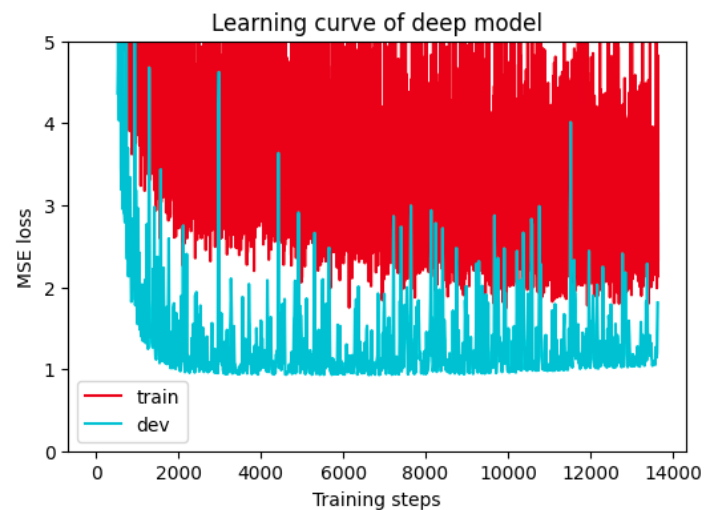
- L2 Regularization:

```
def cal_loss(self, pred, target):
    regularization_loss = 0
    for param in model.parameters():
        # TODO: you may implement L1/L2 regularization here
        # 使用L2 regularization
        regularization_loss += torch.sum(abs(param))
        regularization_loss += torch.sum(param ** 2)
    return self.criterion(pred, target) + 1e-5 * regularization_loss
#return self.criterion(pred, target)
```

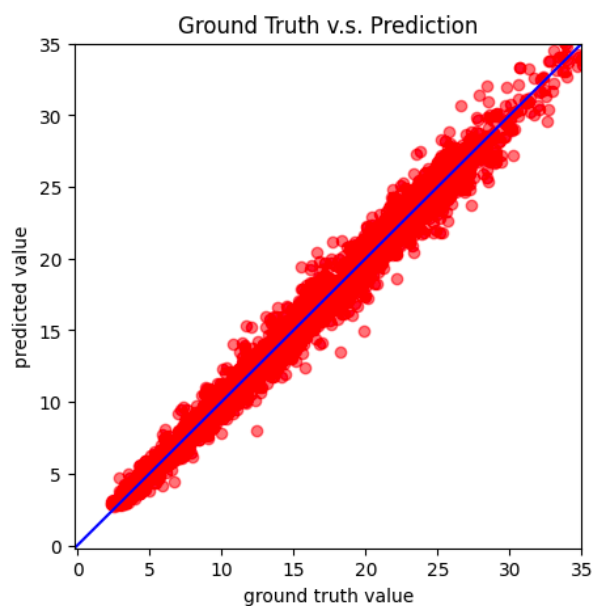
- Hyper-parameter:

```
# TODO: How to tune these hyper-parameters to improve your model's performance?
config = {
    'n_epochs': 5000,                # maximum number of epochs
    'batch_size': 200,               # mini-batch size for dataloader
    'optimizer': 'Adam',             # optimization algorithm (optimizer in torch.optim)
    'optim_hparas': {                # hyper-parameters for the optimizer (depends on which optimizer you are using)
        'lr': 0.003,                 # learning rate of SGD
        'momentum': 0.9              # momentum for SGD
    },
    'l1_lambda': 1e-5 + delta,
    'early_stop': 500,               # early stopping epochs (the number epochs since your model's last improvement)
    'save_path': 'models/model.pth' # your model will be saved here
}
```

- MSE loss:



- Prediction:



- Standard:

#	Team Name	Notebook	Team Members	Score ?
📍	----- strong baseline -----			0.88017
📍	----- medium baseline -----			1.28359
📍	----- simple baseline -----			2.03004

- Result:

- Private Score: 0.89803
- Public Score: 0.88165

- 心得筆記：
 - Epoch：所有的 batch 看過一遍，叫做一個 epoch
 - Batch：
 - 特性：
 1. Batch Size 大的傾向於走到峽谷
 2. Batch Size 小的傾向於走到盆地
 - 結論：
 1. 使用較小的 Batch Size 在更新參數時會有 Noisy (有利於訓練)
 2. 使用較小的 Batch Size 可以避免 Overfitting (有利於測試)
 3. 使用較大的 Batch Size 完成一個 epoch 時間較短
 - Momentum：
 - 定義：

所謂的 momentum 就是 update 的方向不是只考慮現在的 gradient，而是考慮過去所有 gradient 的總合。
 - Tips：
 1. critical points 梯度為 0
 2. saddle point 和 local minima 都屬於 critical point
 - 可由 Hessian matrix 區分
 - local minima 比較少遇到
 - 可藉由沿 Hessian 矩陣的特徵向量方向離開 saddle point
 3. 較小的 batch size 和 momentum 可幫助離開 critical points

- 如何找尋最佳的學習率？
 - 狀況一：loss 不再下降時，未必說明此時到達 critical point，可能只是在山谷的谷壁間來回走
 - 狀況二：使用固定的學習率(Gradient Descent)，即使是在凸面體的優化，都會讓優化的過程非常困難

較大的學習率：loss 在山谷的谷壁間震蕩而不會下降

較小的學習率：梯度較小時幾乎難以移動

- Adaptive Learning Rate
 - 原則：
 1. 平坦 \Rightarrow learning rate 調大一點
 2. 坡度很大 \Rightarrow learning rate 可以設得小一點
 - 類型：
 1. Adagrad：考慮之前所有的梯度大小
 2. RMSProp：添加參數 α 調整當前梯度與過去梯度的重要性 (添越大說明過去的梯度訊息更重要)
 3. Adam = RMSProp + Momentum (最常用的策略)