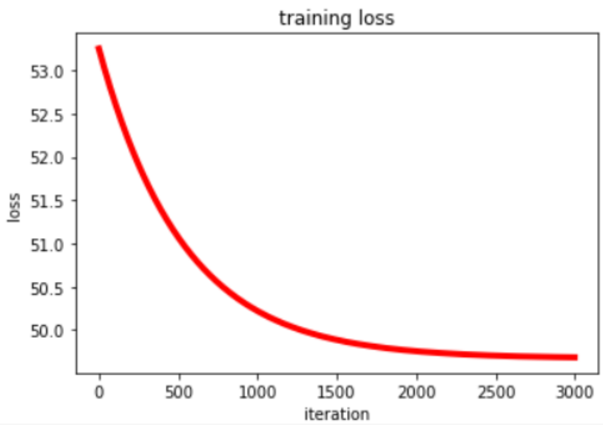
HW1 109550017_黃品云

Part. 1, Coding (60%):

Linear regression model

1.




```
# calculate loss
# loss function: (1/2) * mean ((actual_data - predicted_result)**2)
loss[i] = (1/2) * np.mean((y_train - y_pred)**2)
```

2.

```
MSE: 54.98338097822707
```

```
# Mean Square Error of my prediction and ground truth
mse = (1/2) * np.mean((y_pred - y_test)**2)
print(f'MSE: {mse}')
```

3.

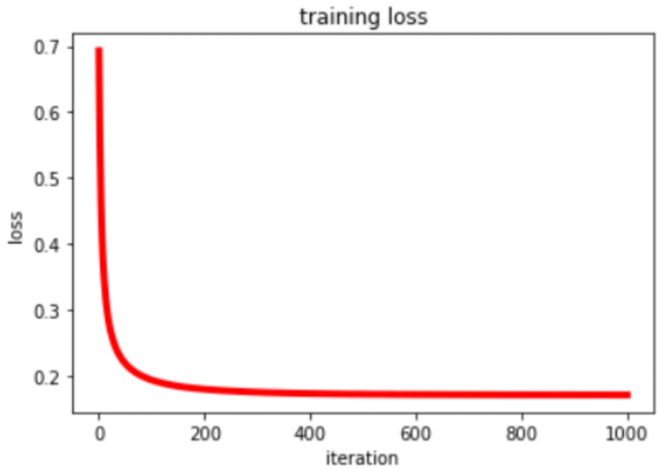
```
weights(m): 52.57670818578662
intercepts(b): -0.3260902413167747
```

```
# gradient of m: mean((m*x + b - y) * x)
tmp = m*x_train + b - y_train
m_gradient = (tmp*x_train).mean()

# gradient of b: mean(m*x + b - y)
b_gradient = tmp.mean()

# update m and b
m += -lr * m_gradient
b += -lr * b_gradient
```

```
# weights and intercepts of my linear model
print(f'weights(m): {m}')
print(f'intercepts(b): {b}')
```

Logistic regression model

1.




```
# calculate loss by Cross Entropy Error
# loss function: (-1/m)*(y_train * log(y_pred) + (1 - y_train)*log(1 - y_pred))
loss[i] = -1*np.mean(y_train * np.log(y_pred) + (1 - y_train) * np.log(1 - y_pred))
```

2.

```
CEE: 0.18690431570340407
```

```
# cross entropy error of my prediction and ground truth
CEE = -1*np.mean(y_test * np.log(y_pred) + (1 - y_test) * np.log(1 - y_pred))
print(f'CEE: {CEE}')
```

3.

```
weights(W): [[4.71934749]]
intercepts(B): 1.6157019268841606
```

```
# update W and B using Gradient Descent
# W is n*1, X is n*m, Y is 1*m
# gradient of W: (1/m)*((y_pred - y_train)·x_train.Transpose)
dW = np.mean(np.dot((y_pred - y_train), x_train.T))
# gradient of B: (1/m)*(sum(y_pred - y_train))
dB = np.mean(np.sum(y_pred - y_train))

# update W and B
W += -lr * dW.T
B += -lr * dB
```

```
# weights and intercepts of my logistic regression model
print(f'weights(W): {W}')
print(f'intercepts(B): {B}')
```

Part 2

1. What's the difference between Gradient Descent, Mini-Batch Gradient Descent, and Stochastic Gradient Descent?

	GD	MBGD	SGD
① The amount of dataset takes into consideration through every iteration	The whole observations	small subset (mini-batch) of observations	one randomly chosen observation
② training time	slow	between GD and SGD	fast
③ noise	noiseless	between GD and SGD	noiser result

(Gradient Descent)

GD :

① Uses the whole observations to calculate in every iteration

② When the amount of observations is large, Gradient Descent takes the longest time to compute compared to the other two approaches

③ Gradient Descent is noiseless and has lower standard error compared the other two approaches

(Mini Batch Gradient Descent)

MBGD :

① Uses a small subset (called mini-batch) and compute the gradient of batch in every iteration

②, ③ : MBGD is considered to the bridge between GD and SGD. Training time and noise are also between GD and SGD

(Stochastic Gradient Descent)

SGD : ① Uses one randomly chosen observation to calculate in every iteration

② SGD uses only one randomly chosen observation from the whole data set, and thus reduces the computation enormously, which is the fastest among three approaches.

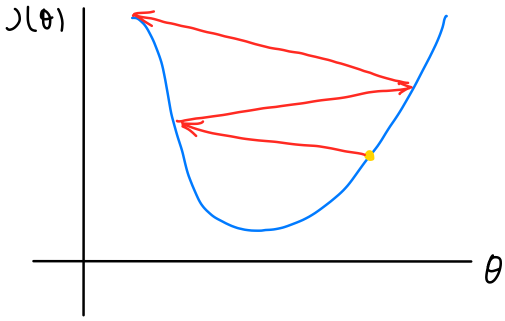
③ SGD has noiser results. Sometimes overfits and results in large variance and small bias.

2. Will different values of learning rate affect the convergence of optimization? Please explain in detail.

Yes. $w^{(\tau+1)} = w^{(\tau)} - \eta \nabla E_D(w)$, where η is the learning rate.

1. If η is small, w will be updated slowly, which requires many updates before reaching the minimum point.

2. If η is large, it might causes drastic updates and makes the result divergent, like the following figure.



3. Show that the logistic sigmoid function (eq. 1) satisfies the property $\sigma(-a) = 1 - \sigma(a)$ and that its inverse is given by $\sigma^{-1}(y) = \ln\{y/(1-y)\}$.

$$\sigma(a) = \frac{1}{1+\exp(-a)} \qquad \text{(eq. 1)}$$

①

$$\sigma(-a) = \frac{1}{1+\exp(a)} = \frac{\frac{1}{\exp(a)}}{\frac{1}{\exp(a)}+1}$$

$$= \frac{\exp(-a)}{1+\exp(-a)} = \frac{1-1+\exp(-a)}{1+\exp(-a)}$$

$$= \frac{1+\exp(-a)}{1+\exp(-a)} - \frac{1}{1+\exp(-a)}$$

$$= 1 - \frac{1}{1+\exp(-a)} = 1-\sigma(a)$$

②

$$y = \frac{1}{1+\exp(-x)}$$

$$\Rightarrow 1+\exp(-x) = \frac{1}{y}$$

$$\Rightarrow \exp(-x) = \frac{1}{y} - 1 = \frac{1-y}{y}$$

$$\Rightarrow -x = \ln\left(\frac{1-y}{y}\right)$$

$$\Rightarrow x = -\ln\left(\frac{1-y}{y}\right) = \ln\left(\frac{y}{1-y}\right)$$

$$\Rightarrow \sigma^{-1}(y) = \ln\left(\frac{y}{1-y}\right)$$

4. Show that the gradients of the cross-entropy error (eq. 2) are given by (eq. 3).

$$E(\mathbf{w}_1, \ldots, \mathbf{w}_K) = -\ln p(\mathbf{T}|\mathbf{w}_1, \ldots, \mathbf{w}_K) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk} \ln y_{nk}$$ (eq. 2)

$$\nabla_{\mathbf{w}_j} E(\mathbf{w}_1, \ldots, \mathbf{w}_K) = \sum_{n=1}^{N} (y_{nj} - t_{nj})\, \boldsymbol{\phi}_n$$ (eq. 3)

Hints:

$$a_k = \mathbf{w}_k^{\mathrm{T}} \boldsymbol{\phi}.$$ (eq. 4)

$$\frac{\partial y_k}{\partial a_j} = y_k(I_{kj} - y_j)$$ (eq. 5)

$$\nabla_{w_j} E(w_1, \cdots, w_K) = \sum_{n=1}^{N} \underbrace{\frac{\partial E}{\partial a_{nj}}}_{①} \underbrace{\nabla_{w_j} a_{nj}}_{②}$$

①: $\frac{\partial E}{\partial a_{nj}} = \sum_{k=1}^{K} \frac{\partial E}{\partial y_{nk}} \frac{\partial y_{nk}}{\partial a_{nj}}$

$\left(\because \ \frac{\partial E}{\partial y_{nk}} = -\frac{t_{nk}}{y_{nk}} \ \text{and} \ \frac{\partial y_{nk}}{\partial a_{nj}} = y_{nk}(I_{kj} - y_{nj}) \right)$

$= -\sum_{k=1}^{K} \frac{t_{nk}}{y_{nk}} \cdot y_{nk}(I_{kj} - y_{nj})$

$= -\sum_{k=1}^{K} t_{nk}(I_{kj} - y_{nj})$

$= -t_{nj} + \sum_{k=1}^{K} t_{nk} y_{nj} = y_{nj} - t_{nj}$

(1-of-k coding scheme $\forall n, \sum_{k} t_{nk} = 1$)

②: $\nabla_{w_j} a_{nj} = \phi_n \ (\because a_{nj} = w_j^T \phi_n)$

$\Rightarrow \nabla_{w_j} E(w_1, \cdots, w_K) = \sum_{n=1}^{N} \frac{\partial E}{\partial a_{nj}} \nabla_{w_j} a_{nj} = \sum_{n=1}^{N} (y_{nj} - t_{nj}) \phi_n$