HW 4

黃熙漢

5. E_{in} of logistic regression given:

$$E_{in} = \frac{1}{N}\sum_{n=1}^{N} \ln(1+\exp(-y_n w^T x_n))$$

$$AE(w) = \nabla^2 E(w)$$

$$g_E(w) = \nabla E(w)$$

$$g_E(w) = \nabla E(w) = -\frac{y_n x_n \exp(-y_n w^T x_n)}{1+\exp(-y_n w^T x_n)} = -y_n x_n \theta(-y_n w^T x_n)$$

$$AE(w) = \frac{\partial g_E(w)}{\partial w} = \frac{1}{N}\sum_{n=1}^{N}\left[-y_n \frac{\partial}{\partial w_k}(x_{n,j}\,\theta(-y_n w^T x_n))\right]$$

$$= \frac{1}{N}\sum_{n=1}^{N}\left[y_n^2 x_{n,j} x_{n,k}\,\theta(-y_n w^T x_n)(1-\theta(-y_n w^T x_n))\right]$$

$y_n^2 = 1$, $y_n \in \{-1, +1\}$:

$$AE(w_t) = \frac{1}{N}\sum_{n=1}^{N}\left[x_n x_n^T\,\theta(-y_n w^T x_n)(1-\theta(-y_n w^T x_n))\right]$$

expressing AE in Matrix form $X^T D X$, with $AE(w_t) = X^T D X$

$$D_{nn} = \theta(-y_n w_t^T x_n)(1-\theta(-y_n w_t^T x_n))$$

$$\theta(-y_n w_t^T x_n) = \begin{cases} 1-h_t(x_n) & \text{if } y_n = +1 \\ h_t(x_n) & \text{if } y_n = -1 \end{cases}$$

$$\theta(-y_n w_t^T x_n) = \frac{1}{1+\exp(y_n w_t^T x_n)}$$

#

$D_{nn} = \theta(-y_n w_t^T x_n)(1-\theta(-y_n w_t^T x_n)) = h_t(x_n)(1-h_t(x_n))$

$D = \text{diagonal}(h_t(x_1)(1-h_t(x_1)), h_t(x_2)(1-h_t(x_2)) \cdots)$

$A \cdot \nabla E(w_t)$ can be expressed as $X^T D X$, where D is an $N \times N$ diagonal matrix define by:

$$D_{nn} = h_t(x_n)(1-h_t(x_n))$$

6.

$$E_{in} = \frac{1}{N}\sum_{n=1}^{N}(-\ln(h(x)))$$

$$= \frac{1}{N}\sum_{n=1}^{N}(-\ln(\frac{\exp(w_y^T x)}{\sum_{k=1}^{k}\exp(w_k^T x)}))$$

$$= \frac{1}{N}\sum_{n=1}^{N}(-w_{y_n}^T x_n + \ln\sum_{k=1}^{k}\exp(w_k^T x_n))$$

gradient (E_{in}):

$$\frac{\partial E_{in}}{\partial w_k} = \frac{1}{N}\sum_{n=1}^{N} -(1_{(y_n=k)})x_n + \frac{\exp(w_k^T x_n)}{\sum_{k=1}^{k}\exp(w_k^T x_n)} x_n$$

$$= -\frac{1}{N}\sum_{n=1}^{N}\left(1_{(y_n=k)} + \frac{\exp(w_k^T x_n)}{\sum_{k=1}^{k}\exp(w_k^T x_n)}\right)x_n$$

$$W^{(t+1)} = W^t + h \cdot V$$

$$V \in \mathbb{R}^K$$

$$1\{y_n = k\} : \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix}, \text{ k dimension}$$

$$\frac{\exp(w_k^T x_n)}{\sum_{k=1}^{K} \exp(w_k^T x_n)} \cdot \begin{bmatrix} p(\text{class } 1) \\ \vdots \\ p(\text{class } k) \end{bmatrix}, \text{ k dimension}$$

$$U = 1\{y_n = k\} + \frac{\exp(w_k^T x_n)}{\sum_{n=1}^{K} \exp(w_n^T x_n)} \#$$

7. $$h_1(x) = \frac{\exp(w_1^{*T} x)}{\exp(w_1^{*T} x) + \exp(w_2^{*T} x)}$$

$$h_2(x) = \frac{\exp(w_2^{*T} x)}{\exp(w_1^{*T} x) + \exp(w_2^{*T} x)}$$

model predict the probability of the positive class as:

$$P(y'=1|x) = \theta(w_{1r}^T x) = \frac{1}{1+\exp(-w_{1r}^T x)}$$

$y_n = 1 \quad y'_n = -1$

$y_n = 2, \quad y'_n = 1$

$h_2(x)$ can be written with:

$$h_2(x) = \frac{\exp(w_2^T x)}{\exp(w_1^T x) + \exp(w_2^T x)} \times \frac{\exp(w_2^T x)}{\exp(w_2^T x)}$$

$$= \frac{1}{1 + \exp(w_1^T x - w_2^T x)}$$

$$\sigma((w_2^* - w_1^*)^T x) = P(y'=1|x) = \sigma(w_{1r}^T x)$$

$$w_{1r}^T x = (w_2^* - w_1^*)^T x$$

$$w_{1r} = w_2^* - w_1^* \quad \#$$

8. given $(x_1, f(x_1))$ $x_2, f(x_2)$

$g = w_0 + w_0 x$

$$\begin{cases} w_0 + w_1 x_1 = f(x_1) = 1 - 2x_1^2 \\ w_0 + w_1 x_2 = f(x_2) = 1 - 2x_2^2 \end{cases}$$

we get:

$$w_1 = \frac{2(x_1^2 - x_2^2)}{x_2 - x_1} = 2(x_1 + x_2)$$

$$w_0 = 1 - 2x_1^2 - w_1 x_1$$

$$= 1 - 4x_1^2 - 2x_1 x_2$$

$$\text{Min}(g) = \frac{1}{2}\left[(g(x_1) - f(x_1))^2 - ((g(x_2) - f(x_2))^2\right]$$

$$= 0$$

$$E_{out}(g) \simeq \int_0^1 (g(x) - f(x))^2 dx$$

1 area of error

$$g(x) - f(x) = [1 - 4x_1^2 - 2x_1x_2 + 2(x_1+x_2)x] - [1 - 2x^2]$$

$$E_{out}(g) = \int_0^1 (g(x) - f(x))^2 dx = \int_0^1 [2x^2 + 2(x_1+x_2)x - 4x_1^2 - 2x_1x_2]^2 dx$$

$$E_D(E_{in}(g) - E_{out}(g)) = E_D(E_{out}(g)), \quad E_{in}(g) = 0$$

Let

$A = -4x_1^2 - 2x_1x_2$

$B = 2(x_1+x_2)x$

$C = 2x^2$

$$(A+B+C)^2 = A^2 + B^2 + C^2 + 2AC + 2AB + 2CB$$

$$E_D(A^2) = 16x_1^4 + 16x_1^3x_2 + 4x_1^2x_2^2$$

$$= 16E[x_1^4] + 16E[x_1^3x_2] + 4E[x_1^2x_2^2]$$

$$E(x_1^4) = \int_0^1 x_1^4 p(x_1)\,dx \quad \text{, continue expect value}$$

$$= \int_0^1 x_1^4 \cdot 1\,dx \quad \text{, uniform}$$

$$= \frac{1}{5}$$

$$E[X_1^3 X_2] = E[X_1^3] \cdot E[X_2] = \frac{1}{4} \cdot \frac{1}{2} = \frac{1}{8}$$

(independent)

$$E[X_1^2 X_2^2] = E[X_1^2] E[X_2^2] = \frac{1}{3} \cdot \frac{1}{3} = \frac{1}{9}$$

$\therefore$ i. $E_D(A^2) = \frac{254}{45}$

ii. $E_D(B^2) = 4(E[X_1^2] + 2E[X_1 X_2] + E[X_2^2]) \cdot x^2$

$= 4(\frac{1}{3} + 2 \cdot \frac{1}{4} + \frac{1}{3}) x^2$

$= \frac{14}{3} x^2$

再對 x integral

$\int_0^1 \frac{14}{3} x^2 dx = \frac{14}{3} \cdot \frac{1}{3} = \frac{14}{9}$

$\therefore$ iii. $E_D(C^2)$

$C = 2X^2$

$C^2 = 4X^4$

$\int_0^1 4X^4 dX = \frac{4}{5}$

IV. $\Xi_D(2AB)$:

$$2AB = -16x_1^3x + (-8x_1^2x_2x)$$

$$\Xi_D(2AB) = -16\frac{1}{4}x - 8\cdot\frac{1}{3}\cdot\frac{1}{2}x$$

$$= -\frac{16}{3}x$$

$$\int_0^1 -\frac{16}{3}x\,dx = \frac{8}{3}$$

V. $\Xi_D(2AC)$:

$$2AC = 2\cdot(-4x_1^2 - 2x_1x_2)\cdot 2x^2 = 16x_1^2x^2 - 8x_1x_2x^2$$

$$\Xi_D(2AC) = -16\cdot\frac{1}{3}x^2 - 8\cdot\frac{1}{4}x^2 = -\frac{16}{3}x^2 - 2x^2 = -\frac{22}{3}x^2$$

$$\int_0^1 -\frac{22}{3}x^2dx = -\frac{22}{9}$$

VI $\Xi_D(2BC)$:

$$2BC = 2\cdot 2(x_1+x_2)x\cdot 2x^2$$

$$= 8(x_1+x_2)x^3$$

$$\Xi_D(2BC) = 8\cdot\left(\frac{1}{2}+\frac{1}{2}\right)x^3 = 8\cdot 1\cdot x^3 = 8x^3$$

$$\int_0^1 8x^3dx = \frac{1}{4}\cdot 8 = 2$$

$$\therefore E_D(E_{out}(g)) = \int_0^1 E_D[Ax^2+Bx+C]^2 dx$$

$$= \frac{254}{45} + \frac{14}{9} + \frac{4}{5} - \frac{8}{3} - \frac{22}{9} + 2$$

$$= \frac{254}{45} + \frac{70}{45} + \frac{36}{45} - \frac{120}{45} - \frac{110}{45} + \frac{90}{45} = \frac{26}{9} \#$$

9. $X_h^T X_h$

$$X_h = [\vec{x}]$$

$X_h^T X_h$

$$X_h^T X_h = x^T x + x^T \tilde{x} + \tilde{x}^T x + \tilde{x}^T \tilde{x}$$

$\therefore E(x^T \tilde{x}) = x^T x$ (non random)

$\therefore E(\tilde{x}^T \tilde{x})$

$\tilde{X} = X + E$

$E(X^T \tilde{X}) = E(X^T(X+E)) = X^T X + X^T E(E)$

$= X^T X$ ↓ zero mean

iii.

$E(\tilde{X}^T X) = X^T X$

iv. $E(\tilde{X}^T \tilde{X})$

$\tilde{X}^T \tilde{X} = (X+E)^T (X+E)$

$= X^T X + X^T E + E^T X + E^T E$

ions 1 | 0 | 0

$E(E^T E)$

$E^T E = \sum_{n}^{N} \epsilon_n^T \epsilon_n$

$E(\epsilon_n^T \epsilon_n) = E(\epsilon_n^T \epsilon_n) = \sigma^2 I_{d+1}$

$E(E^T E) = N\sigma^2 I_{d+1}$

$$\mathbb{E}(X_h^T X_h) = X^T X + X^T X + X^T X + (X^T X + N\sigma^2 I_{d+1})$$
$$= 4X^T X + N\sigma^2 I_{d+1}$$

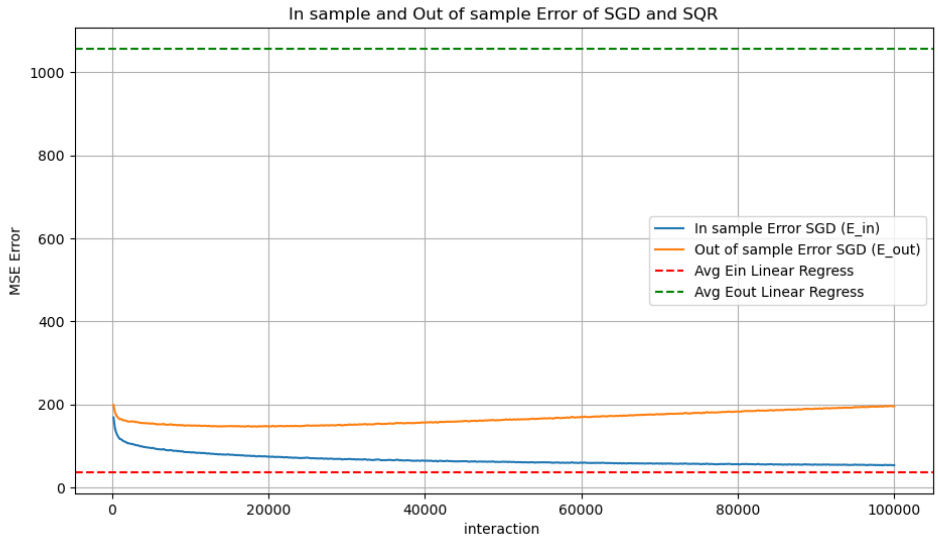
for the given:

$$\mathbb{E}(X_h^T X_h) = \alpha X^T X + \beta\sigma^2 I_{d+1}$$

$$\alpha = 4$$
$$\beta = N \quad \#$$

10.Description: The results show that as the number of iterations increases, the in-sample error Ein(SGD) converges to the in-sample error Ein(Lin), which aligns with theoretical expectations. However, a curious observation is that the out-of-sample error Eout(SGD) begins to diverge as the number of iterations increases. I suspect this occurs because the model is trained on a very small subset of only 64 examples, and with 100,000 iterations, it ends up overfitting to the training data.

Result:



Code :

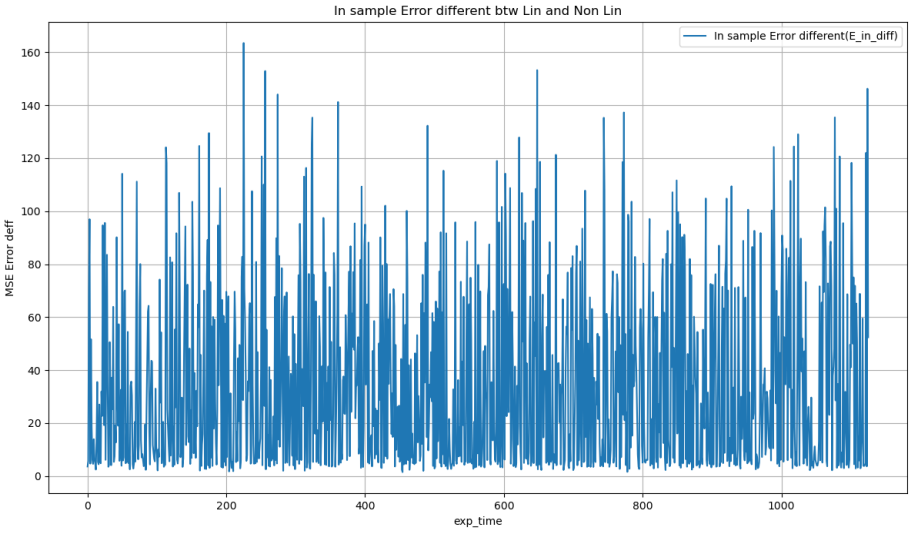
```
#SGD time
for t in range(1,w_interact+1):
    #sgD : w_t+1 = w_t + a * Del[x]

    random_n = np.random.randint(0,N)
    x_i = xtrain_set[random_n]
    y_i = ytrain_set[random_n]
    y_hat_i = np.dot(w_init, x_i)
    gradient = -2 * x_i * (y_i - y_hat_i)
    w_init = w_init - alpha * gradient


    if t % 200 == 0 : #這樣可以保留 t = 200 , 400 , 600 。而且t 會增長
      y_hat_train = xtrain_set @  w_init
      y_hat_test = xtest_set @  w_init
      E_in = np.mean((ytrain_set - y_hat_train)**2)
      E_out = np.mean((ytest_set - y_hat_test)**2)
      E_in_t.append(E_in)
      E_out_t.append(E_out)
      t_val += 1
E_all.append({'Ein': E_in_t,'Eout' : E_out_t})
E_out_LR.append(per_E_out) , E_in_LR.append(per_E_in)
```

```
E_in_LR_fin = np.mean(E_in_LR)
E_out_LR_fin = np.mean(E_out_LR)

E_in_avg  = []
E_out_avg = []
#mean allvalue exp of 200 , then 400..
for i in range(len(E_all[0]['Ein'])) :
    E_in_avg.append(np.mean([E_all[exp]['Ein'][i] for exp in range(experiment_time)])) #n
    E_out_avg.append(np.mean([E_all[exp]['Eout'][i] for exp in range(experiment_time)]))


plt.figure(figsize=(11,6))
plt.plot(t_values,E_in_avg , label = "In sample Error SGD (E_in)")
plt.plot(t_values,E_out_avg, label = "Out of sample Error SGD (E_out)")
plt.axhline(E_in_LR_fin, ls = "--",color = 'r', label = "Avg Ein Linear Regress")
plt.axhline(E_out_LR_fin, ls = "--", color = 'g' , label = 'Avg Eout Linear Regress')
```

11.Description: The results show that Ein(Lin) is larger than Eout(Non_Lin) , which aligns with the theory. Non-linear models have a higher degree of freedom than linear models (and also a higher VC dimension), allowing them to fit the data more closely.

Average In-sample Error Difference: 34.518965

Result:



Code:

```
z_nl = np.zeros((data_num, feat*R + 1))
for i  in range(data_num) :
    x_l[i,0] = 1 #set x0
    for j in range(1,feat + 1) :
        x_l[i,j] = data[i][j]
for i in range(data_num):

    z_nl[i,0] = 1
    i_idx = 1
    for l in range (1,R + 1) :

        for j in range (1, feat + 1) :
            z_nl[i,i_idx] = data[i][j]**l
            i_idx += 1


for t in range(experiment_time) :
```

```
ztrain_set  = z_nl[train_set]
ztest_set   = z_nl[test_set]

ytrain_set  = arr_label[train_set]
ytest_set   = arr_label[test_set]

#W_Lin = x^pseudoinv @ y
x_pseudo_inv = np.linalg.pinv(xtrain_set)
w_Lin        = x_pseudo_inv@ytrain_set

# y = wx (outsample)
y_ht_train = xtrain_set @ w_Lin
y_ht_test = xtest_set @ w_Lin

#E_in = (1/N)(y - y^)^2 , y^ = wLin@X ,
per_E_in     = np.mean((ytrain_set - y_ht_train)**(2))
per_E_out    = np.mean((ytest_set - y_ht_test)**(2))

#polynomial
z_p_i = np.linalg.pinv(ztrain_set)
w_nonlin = z_p_i @ ytrain_set

y_hat_non_train = ztrain_set @ w_nonlin
```

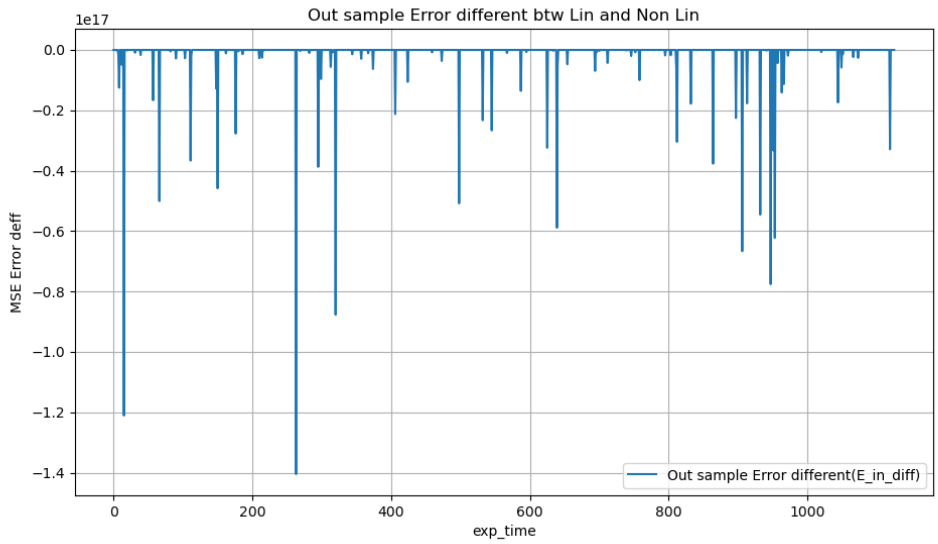
```
    #Error
    E_in_non_each  = np.mean((y_hat_non_train - ytrain_set)**2)

    E_in_nonl.append(E_in_non_each)
    E_out_lin.append(per_E_out) , E_in_lin.append(per_E_in)

E_in_diff = []
```

S OUTPUT TERMINAL PORTS

12. Description: As a result, we observe that the out-of-sample error Eout(Lin) for the linear regression model is smaller than Eout(Non_Linear) for the nonlinear regression model. This outcome aligns with theoretical expectations. Nonlinear models possess a higher degree of freedom and a larger VC dimension compared to linear models. These characteristics make nonlinear models more flexible and capable of capturing complex patterns in the data. However, this increased flexibility also makes them more susceptible to overfitting, especially when the model is not regularized. Additionally, the training set size is relatively small (only 64 examples), which exacerbates the overfitting problem. With limited data, the model has fewer opportunities to generalize its learning, making it easier to memorize noise rather than identifying the underlying true patterns.

Average Out-sample Error Difference: -1288079828051806.500000

Result:



Code:

```
    #polynomial
    z_p_i = np.linalg.pinv(ztrain_set)
    w_nonlin = z_p_i @ ytrain_set

    y_hat_non_test  = ztest_set @ w_nonlin

    #Error
    E_out_non_each       = np.mean((y_hat_non_test - ytest_set)**2)

    E_out_nonl.append(E_out_non_each)
    E_out_lin.append(per_E_out) , E_in_lin.append(per_E_in)

E_in_diff = []
E_out_diff = []


E_out_diff = np.array(E_out_lin) - np.array(E_out_nonl)
E_out_avg = np.mean(E_out_diff)
print(f"Average Out-sample Error Difference: {E_out_avg:.6f}")
```