HW 3

黃熙漢

5. Counter Exp:

$H_1 = \{h(x) = 0\}$, can only assign a label of 0 to any value of input, I guess often use in benchmark for other algorithm

$H_2 = \{h(x) = 1\}$, only assign a label of 1

Following the equation:

$d_{VC}(H_1 \cup H_2) \leq d_{VC}(H_1) + d_{VC}(H_2)$

$d_{VC}(H_1 \cup H_2) = 1$, $\because H_1 = 1$ & $H_2 = 0$, shathered if $N = 1$, all possibilities which include $x_0 = 0$, $x_0 = 1$

$\{0, 1\}$ hypo

$d_{VC}(H_1) = 0$, can't shathered all point

$d_{VC}(H_2) = 0$, same with H_1

$\therefore$, $d_{VC}(H_1 \cup H_2) \leq d_{VC}(H_1) + d_{VC}(H_2)$

$1 \leq 0$ (disprove.)

6.

		predict	
	$h(x)$ / $y(x)$	+1	-1
actual	+1	0	10
	-1	1	0

set: $y=P$ token $= \bar{T}_{i=p}$

$y=N$ token $= \bar{T}_{i=N}$

calculate that expect cost

$$\text{cost}(H) = \bar{T}_{i=p} \; xp(y=-1|x)$$

* which $p(y=-1|x) + p(y=+1|x) = 1$

$$\text{cost}(-1) = \bar{T}_{i=N} \; xp(y=+1|x)$$

compare two expected cost:

$$\text{cost}(+1) \leq \text{cost}(-1)$$

$$\bar{T}_{i=p} \; xp(y=-1|x) \leq \bar{T}_{i=N} \; xp(y=+1|x)$$

which $p(y=-1|x) + p(y=+1|x) = 1$

$\frac{1}{10}$ TIFN = 10 TIFP

$\frac{1}{10}$ TIFN $\times (1 - P(y=+1|x)) \leq$ TIFN $\times P(y=+1|x)$

$P(y=+1|x) \geq \frac{1}{11}$

$\therefore$ Prove that $\alpha = \frac{1}{11}$, idea is sacrifice TIFP to reduce probability of error of TIFN.

7.

Define error for each x

$e(h,x) = P_{y|x}(h(x) \neq y)$

$e(f,x) = P_{y|x}(f(x) \neq y)$

when $h(x) = f(x)$:

$e(h,x) = e(f,x)$

if $h(x) \neq f(x)$:

For given equation we know that, $P(y=f(x)|x) \geq \frac{1}{2}$, so

$P(y=h(x)|x) \leq \frac{1}{2}$)

Therefore $e(h,x) = P_{y|x}(h(x) \neq y) = 1 - P_{y|x}(y=h(x)) \geq \frac{1}{2}$

$$e(f,x) = P_{y|x}(f(x) \neq y) \leq \frac{1}{2}$$

$$e(h,x) = P_{y|x}(y = f(x))$$

$$e(h,x) - e(f,x) = 2P_{y|x}(y = f(x)) - 1 \geq 0$$

Since $P_{y|x}(y = f(x)) \geq \frac{1}{2}$, $e(h,x) - e(f,x) \geq 0$

upper bound:

$$e(h,x) \leq e(f,x) + \begin{cases} 1, & \text{when } h(x) \neq f(x) \\ 0, & \text{else} \end{cases}$$

so,

$$E'_{out}(h) = \mathcal{E}_x[e(h,x)] \leq \mathcal{E}_x[e(f,x) + [h(x) \neq f(x)]] = \mathcal{E}_x[e(f,x)] + \mathcal{E}_x[[h(x) \neq f(x)]]$$

Note that:

$$\mathcal{E}_x[e(f,x)] = E^2_{out}(f)$$

$$\mathcal{E}_x[[h(x) \neq f(x)]] = P_x[h(x) \neq f(x)] = E^1_{out}(h)$$

$\therefore$ prove that $E^{(2)}_{out}(h) \leq E^{(2)}_{out}(f) + E^{(1)}_{out}(h)$,

means for any hypothesis, its error under the noisy distribution $E'_{out}(h)$ is bounded by its error to the target function

$E_{out}(h)$ plus the irreducible error $E_{one}(f)$.

8.

$$X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \end{bmatrix} \quad X' = \begin{bmatrix} 1126 & x_1 \\ 1126 & x_2 \\ 1126 & x_3 \end{bmatrix}$$

relationship between X & X'

$X = X'C$, C is diagonal matrix

Define C : $\begin{bmatrix} 1126 & 0 & \cdots & 0 \\ 0 & 1 & & \\ \vdots & & 1 & \\ 0 & & & 1 \end{bmatrix}$

we know $w_{lin} = (X^T X)^{-1} X^T y$ (unique solution)

$$w_{luck} = (X'^T X')^{-1} X'^T y$$

$$= (SX^T SX)^{-1} SX^T y$$

$$= S^{-1}(X^T X)^{-1} S S^{-1} y$$

$$= S^{-1}(X^T X)^{-1} y$$

$$w_{luck} = S^{-1} w_{Lin}$$

$\therefore$ $w_{lin} = S\, w_{luck}$, which $D = S$

$w_{Lin} = D\, w_{luck}$, I learned how the weight change by scale features.

9. likelihood function: $\prod_{n=1}^{N} (h(x))$

$$\min E_{in}(w) = \frac{1}{N} \sum_{n=1}^{N} -\ln\left(\frac{1}{2}\left(\frac{w^T x}{\sqrt{1+(w^T x)^2}} + 1\right)\right)$$

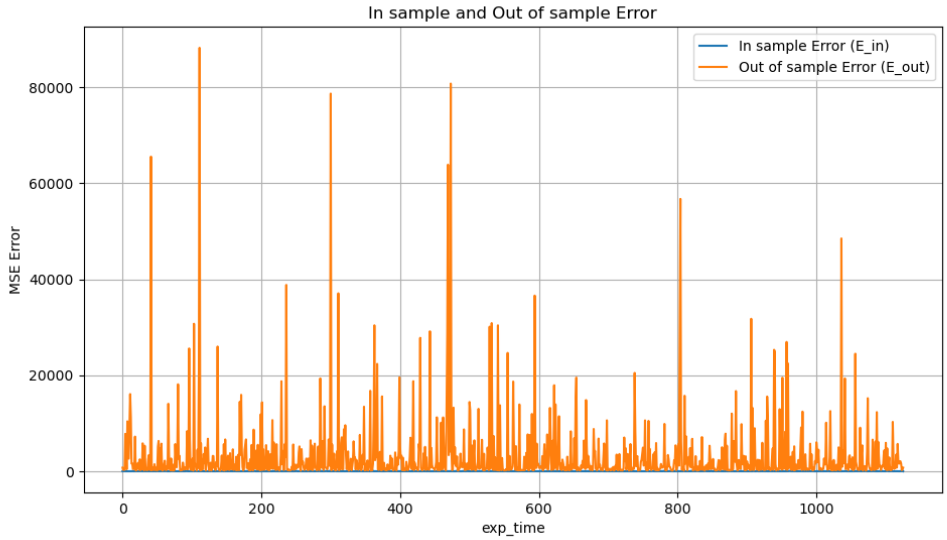
$$\nabla E_{in}(w) = \frac{1}{N} \sum_{n=1}^{N} -\ln\left(\frac{1}{2}\frac{w^T x}{\sqrt{1+(w^T x)^2}} + \frac{1}{2}\right)$$

$$= \frac{1}{N} \sum \left(\frac{1}{\frac{w^T x}{2\sqrt{1+(w^T x)^2}} + \frac{1}{2}}\right) \cdot \frac{1}{2} \cdot \frac{x\sqrt{1+(w^T x)^2} - w^T x \cdot \left((1+(w^T x)^2\right)^{-\frac{1}{2}} \cdot 2w^T x)}{1+(w^T x)^2}$$

#

10. I noticed a significant gap between Eout and Ein. I believe this gap occurs because the training data size is very small, with only 32 samples, which has likely caused the model to overfit. Overfitting happens when the model performs exceptionally well on the training data but struggles to generalize to unseen data. As a result, Ein is much lower than Eout, indicating that the model has learned specific patterns or noise in the training set that don't hold in the test set.

MSE Result :

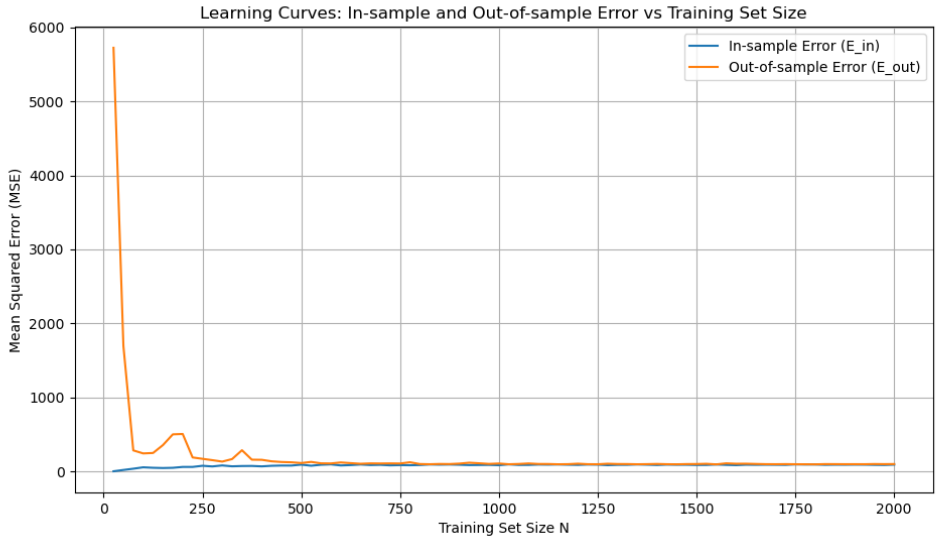


Code :

```
for i  in range(data_num) :
    x[i,0] = 1 #設置x0
    for j in range(1,feat + 1) :
        x[i,j] = data[i][j] #data[第幾個字典][第幾個鍵值]

for t in range(experiment_time) :
    splice_data = np.random.permutation(data_num)
    train_set   = splice_data[:N] #N_exp 個 traindata
    test_set    = splice_data[N:]

    xtrain_set  = x[train_set]  # train set 是一組數字 index, x【train_set] 就會對著他要的行來取。
    xtest_set   = x[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 #don't forget mtrx rule
    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))
```

11. Description: As shown in the graph, we can observe that as we use more training data, both Ein and Eout begin to converge. Initially, there is a noticeable gap between E(in and out) , which indicates overfitting when the model is trained on smaller datasets. However, as the amount of training data increases, the gap narrows, and the errors converge, indicating that the model is generalizing better and overfitting is reduced.

Result:



Code:

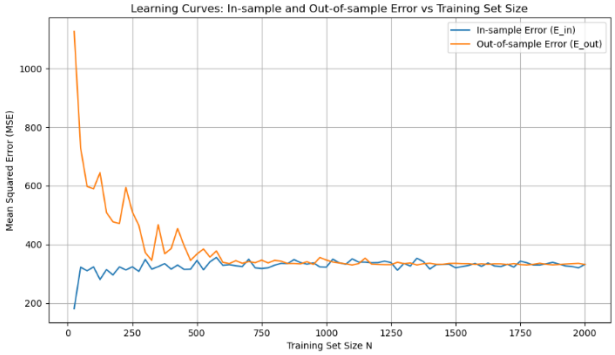
```
label, data = svm_read_problem('ml_hw_2_feat')

arr_label   = np.array(label)

N_Range  = list(range(25,2001,25))
data_num = 8192
experiment_time = 16
feat  = 12

E_in  = []
E_out = []

np.random.seed(1)

x = np.zeros((data_num, feat + 1))

for i  in range(data_num) :
    x[i,0] = 1 #設置x0
    for j in range(1,feat + 1) :
        x[i,j] = data[i][j] #data[第幾個字典][第幾個鍵值]

for N in N_Range :
    E_in_avg = []
    E_out_avg = []
    for t in range(experiment_time) :
        splice_data = np.random.permutation(data_num)
        train_set   = splice_data[:N] #N_exp 個 traindata
```

```python
E_out_avg = []
for t in range(experiment_time) :
    splice_data = np.random.permutation(data_num)
    train_set   = splice_data[:N] #N_exp 個 traindata
    test_set    = splice_data[N:]

    xtrain_set  = x[train_set]  # train set 是一組數字 index, x
    xtest_set   = x[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 #don't forget mtrx rule
    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))

    E_out_avg.append(per_E_out) , E_in_avg.append(per_E_in)
E_out.append(np.mean(E_out_avg))
E_in .append(np.mean(E_in_avg))
```

12.

Description: We could see that final Eout values of the two models are very close, which indicates that the noise level has a greater impact on the model's performance, and increasing the number of features did not significantly reduce the generalization error. I guess it could be because most of the useful information in the data has already been captured by the first two features, and adding more features did not provide much additional valuable information.

Result:



Code:

```
feat  = 2
```