5. counter-example

Collaborators:

B11901016 張均豪

$$\begin{cases} \mathcal{H}_1 = \{ h_1(\underline{x}) = 1 \} \\ \mathcal{H}_2 = \{ h_2(\underline{x}) = -1 \} \end{cases}$$

$$d_{vc}(\mathcal{H}_1) = d_{vc}(\mathcal{H}_2) = 0$$

$$d_{vc}(\mathcal{H}_1 \sqcup \mathcal{H}_2) = 1$$

6. no collaborators

		g: +1	g: −1
f	+1	no error	false negative
	−1	false positive	no error

$\Rightarrow$

		g: +1	g: −1
f	+1	0	10
	−1	1	0

$$\Rightarrow \alpha = \frac{10}{1+10} = \frac{10}{11}$$

7.

Collaborators:
B11901073 林禹融

Let the probability of noise is p

That is, $E_{out}^{(2)}(f) = \mathbb{E}_{\underline{x} \sim P(\underline{x}),\, y \sim P(y|\underline{x})} [\![f(\underline{x}) \neq y]\!] = p$

$$E_{out}^{(2)}(h) = \frac{1}{N} \left(p \sum_n [\![h(\underline{x}) = f(\underline{x})]\!] + (1-p) \sum_n [\![h(\underline{x}) \neq f(\underline{x})]\!] \right)$$

$$= E_{out}^{(2)}(f) \frac{1}{N} \left(N - \sum_n [\![h(\underline{x}) \neq f(\underline{x})]\!] \right) + \left(1 - E_{out}^{(2)}(f) \right) \frac{1}{N} \sum_n [\![h(\underline{x}) \neq f(\underline{x})]\!]$$

$$= E_{out}^{(2)}(f) \left(1 - E_{out}^{(1)}(h) \right) + \left(1 - E_{out}^{(2)}(f) \right) E_{out}^{(1)}(h)$$

$$= E_{out}^{(2)}(f) + E_{out}^{(2)}(h) - 2 E_{out}^{(2)}(f) E_{out}^{(1)}(h)$$

$$\leq E_{out}^{(2)}(h) + E_{out}^{(2)}(f)$$

Q.E.D.

8. From page 11 in lecture 09, no collaborators

$$\underline{w_{LIN}} = X^{\dagger} y$$
$$= (X^T X)^{-1} X^T y \text{ , where}$$

$$X = \underbrace{\begin{bmatrix} 1 & — & x_1^T & — \\ 1 & — & x_2^T & — \\ & & \vdots & \\ 1 & — & x_N^T & — \end{bmatrix}}_{N\times(d+1)}, \quad y = \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}}_{N\times 1}$$

Now, $x_0 = 1 \longrightarrow x_0 = 1126$

$$X' = \underbrace{\begin{bmatrix} 1126 & — & x_1^T & — \\ 1126 & — & x_2^T & — \\ & & \vdots & \\ 1126 & — & x_N^T & — \end{bmatrix}}_{N\times(d+1)} = XM \text{, where } M = \begin{bmatrix} 1126 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}$$

and $M^T = M$

$$\therefore \underline{w_{LUCKY}} = ((X')^T X')^{-1} (X')^T y$$
$$= ((XM)^T (XM))^{-1} (XM)^T y$$
$$= (M^T (X^T X) M)^{-1} (M^T X^T) y$$
$$= M^{-1} (X^T X)^{-1} M^{-1} (M X^T) y$$
$$= M^{-1} (X^T X)^{-1} X^T y$$
$$= M^{-1} \underline{w_{LIN}}$$

$$\Rightarrow \underline{w_{LIN}} = M \underline{w_{LUCKY}}$$

$$\therefore D = M = \begin{bmatrix} 1126 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}$$

9. $h(\underline{x}) = \dfrac{1}{1+e^{-\underline{w}^T\underline{x}}}$ no collaborators

$$\tilde{h}(\underline{x}) = \frac{1}{2}\left(\frac{\underline{w}^T\underline{x}}{\sqrt{1+(\underline{w}^T\underline{x})^2}}+1\right) \qquad \theta(s) = \frac{1}{2}\left(\frac{1}{\sqrt{1+s^{-2}}}+1\right)$$

$$= \frac{1}{2}\frac{1}{\sqrt{1+(\underline{w}^T\underline{x})^{-2}}}+\frac{1}{2}$$

$$\max_{\tilde{h}} \text{likelihood}(\text{logistic } \tilde{h}) \propto \prod^{N} \tilde{h}(y_n \underline{x_n})$$

$$\Rightarrow \max_{\underline{w}} \text{likelihood}(\underline{w}) \propto \prod_{n=1}^{N} \theta(y_n \underline{w}^T \underline{x_n})$$

$$\Rightarrow \max_{\underline{w}} \ln \prod_{n=1}^{N} \theta(y_n \underline{w}^T \underline{x_n})$$

$$\Rightarrow \min_{\underline{w}} \frac{1}{N}\sum_{n=1}^{N} -\ln\theta(y_n \underline{w}^T \underline{x_n})$$

$$\Rightarrow \min_{\underline{w}} \frac{1}{N}\sum_{n=1}^{N} -\ln\left(\frac{1}{2}\left(\frac{1}{\sqrt{1+(y_n\underline{w}^T\underline{x_n})^{-2}}}+1\right)\right)$$

$$\Rightarrow \min_{\underline{w}} \frac{1}{N}\sum_{n=1}^{N}\left[\ln 2+\ln\sqrt{1+(y_n\underline{w}^T\underline{x_n})^{-2}} - \ln\left(1+\sqrt{1+(y_n\underline{w}^T\underline{x_n})^{-2}}\right)\right]$$

$$\Rightarrow \min_{\underline{w}} \underbrace{\frac{1}{N}\sum_{n=1}^{N} err(\underline{w}, \underline{x_n}, y_n)}_{\tilde{E}_{in}(\underline{w})}$$

$$err(\underline{w}, \underline{x}, y) = \ln 2+\ln\sqrt{1+(y\,\underline{w}^T\underline{x})^{-2}} - \ln\left(1+\sqrt{1+(y\,\underline{w}^T\underline{x})^{-2}}\right)$$

$$\tilde{E}_{in}(\underline{w}) = \frac{1}{N}\sum_{n=1}^{N}\left[\ln 2+\ln\sqrt{1+(y_n\underline{w}^T\underline{x_n})^{-2}} - \ln\left(1+\sqrt{1+(y_n\underline{w}^T\underline{x_n})^{-2}}\right)\right]$$

$$\Rightarrow \frac{\partial \tilde{E}_{in}(\underline{w})}{\partial w_i} = \frac{1}{N}\sum_{n=1}^{N}\Bigg[\left(\frac{1}{\sqrt{1+(y_n\underline{w}^T\underline{x_n})^{-2}}}\right)\left(\frac{-1}{(y_n\underline{w}^T\underline{x_n})^3\sqrt{1+(y_n\underline{w}^T\underline{x_n})^{-2}}}\right)(-y_n x_{n,i})$$
$$-\left(\frac{1}{1+\sqrt{1+(y_n\underline{w}^T\underline{x_n})^{-2}}}\right)\left(\frac{-1}{(y_n\underline{w}^T\underline{x_n})^3\sqrt{1+(y_n\underline{w}^T\underline{x_n})^{-2}}}\right)(-y_n x_{n,i})\Bigg]$$

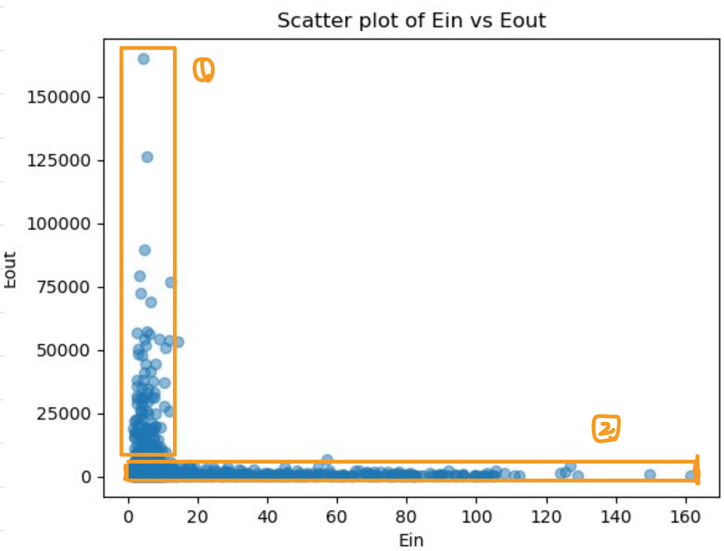
$$= \frac{1}{N}\sum_{n=1}^{N}\left(\frac{1}{\sqrt{1+(y_n\underline{w}^T\underline{x_n})^{-2}}} - \frac{1}{1+\sqrt{1+(y_n\underline{w}^T\underline{x_n})^{-2}}}\right)\left(\frac{y_n x_{n,i}}{(y_n\underline{w}^T\underline{x_n})^3\sqrt{1+(y_n\underline{w}^T\underline{x_n})^{-2}}}\right)$$

$$= \frac{1}{N}\sum_{n=1}^{N}\left(\frac{y_n x_{n,i}}{(1+(y_n\underline{w}^T\underline{x_n})^{-2})(1+\sqrt{1+(y_n\underline{w}^T\underline{x_n})^{-2}})(y_n\underline{w}^T\underline{x_n})^3}\right)$$

$$\therefore \nabla\tilde{E}_{in}(\underline{w}) = \frac{1}{N}\sum_{n=1}^{N}\left(\frac{y_n \underline{x_n}}{(1+(y_n\underline{w}^T\underline{x_n})^{-2})(1+\sqrt{1+(y_n\underline{w}^T\underline{x_n})^{-2}})(y_n\underline{w}^T\underline{x_n})^3}\right)$$

10.

no collaborators



① $E_{out}(w_{lin})$ is far from $E_{in}(w_{lin})$ and $E_{in}(w_{lin}) \approx 0$

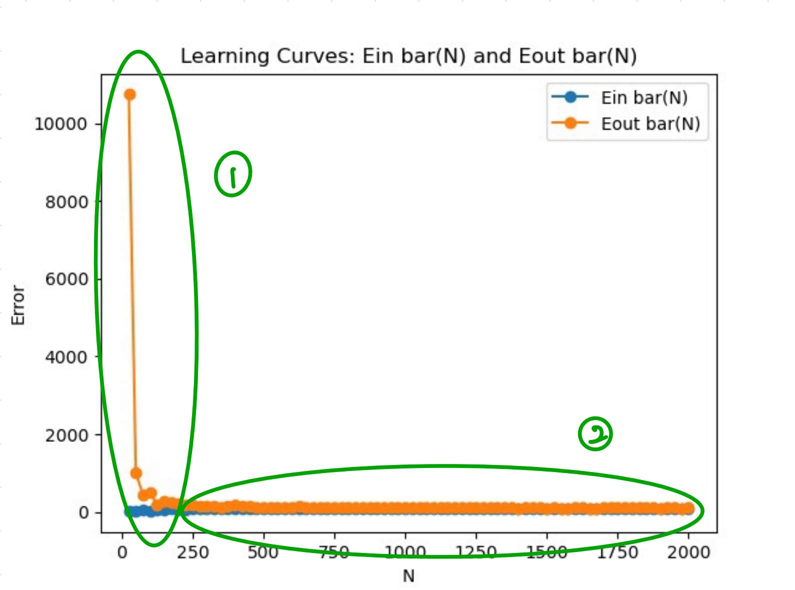
② $E_{out}(w_{lin}) \approx E_{in}(w_{lin})$ but $E_{in}(w_{lin})$ varies from 0 to 160

Increase N to let $E_{out}(w_{lin}) \approx E_{in}(w_{lin})$ but $E_{in}(w_{lin})$ may increase.

hw3-10.py > ...

```python
import numpy as np…

def download_and_parse_data():…

# Linear regression
def linear_regression(X, y):
    X_dagger = np.linalg.pinv(X)  # pseudo-inverse of X
    wlin = X_dagger @ y  # wlin
    return wlin

# Squared error
def squared_error(y_true, y_pred):
    return np.mean((y_true - y_pred) ** 2)

# Error calculation for in-sample and out-of-sample
def calculate_errors(X_in, y_in, X_out, y_out, wlin):
    y_in_pred = X_in @ wlin
    y_out_pred = X_out @ wlin
    Ein = squared_error(y_in, y_in_pred)
    Eout = squared_error(y_out, y_out_pred)
    return Ein, Eout

# Plot the results
def plot_results(Ein_array, Eout_array):…

num_examples = 8192
N = 32
num_experiments = 1126
Ein_array = []
Eout_array = []

def main():
  X, y = download_and_parse_data()

  for _ in range(num_experiments):
    indices = np.random.choice(num_examples, N, replace=False) # Randomly sample N examples for training
    X_in = X[indices]
    y_in = y[indices]
    mask = np.ones(num_examples, dtype=bool)
    mask[indices] = False
    X_out = X[mask]
    y_out = y[mask]

    wlin = linear_regression(X_in, y_in)
    Ein, Eout = calculate_errors(X_in, y_in, X_out, y_out, wlin)
    Ein_array.append(Ein)
    Eout_array.append(Eout)

  plot_results(Ein_array, Eout_array)

if __name__ == '__main__':
    main()
```

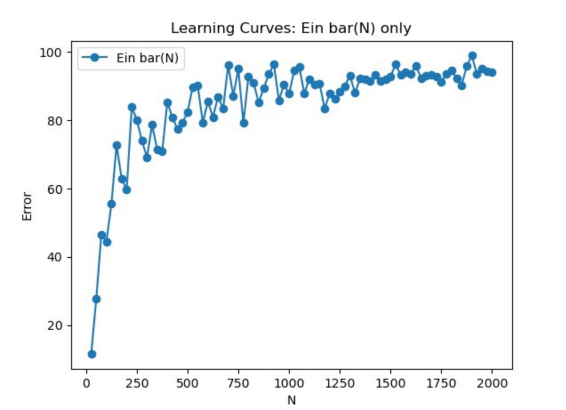
11. no collaborators



① $\bar{E}_{out}(N)$ are much larger than $\bar{E}_{in}(N)$ when N is small

② $\bar{E}_{out}(N) \approx \bar{E}_{in}(N)$ and $\bar{E}_{in}(N)$ are lower than 100

auxiliary:

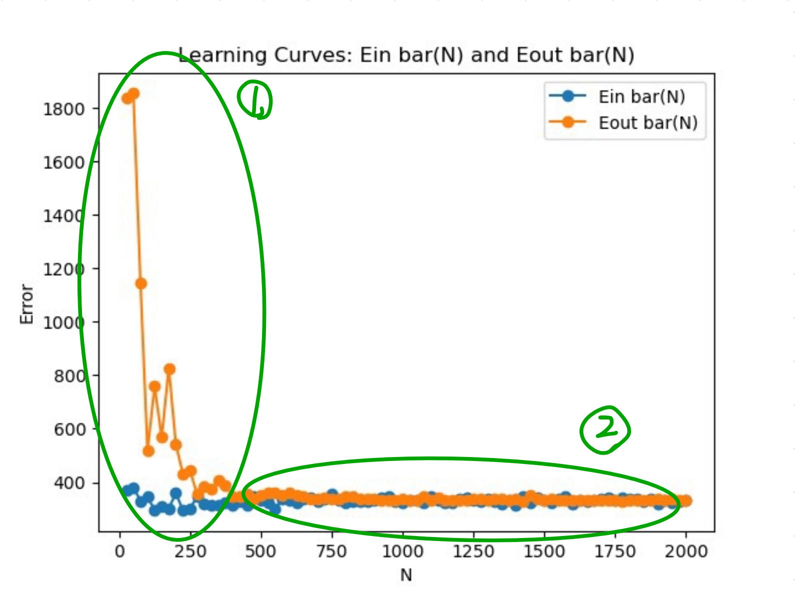


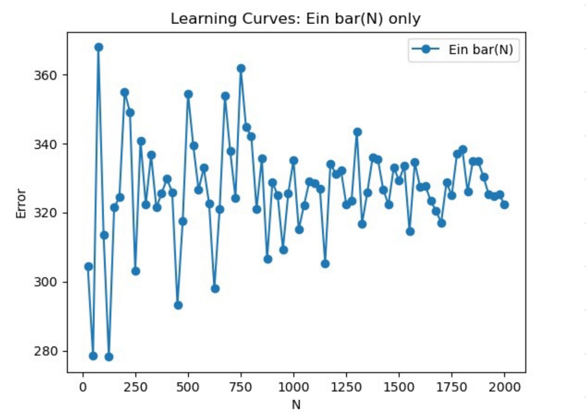

```python
hw3-11.py > ...
> import numpy as np⋯

> def download_and_parse_data():⋯

# Linear regression
> def linear_regression(X, y):⋯

# Squared error
> def squared_error(y_true, y_pred):⋯

# Error calculation for in-sample and out-of-sample
> def calculate_errors(X_in, y_in, X_out, y_out, wlin):⋯
  Ctrl+L to chat, Ctrl+K to generate
# Plot the learning curves
> def plot_learning_curves(Ein_avg, Eout_avg):⋯

num_examples = 8192
N_array = range(25, 2001, 25)  # N values from 25 to 2000
num_experiments = 16

def main():
  X, y = download_and_parse_data()
  Ein_avg = []
  Eout_avg = []
  for N in N_array:
    Ein_array = []
    Eout_array = []
    for _ in range(num_experiments):
      indices = np.random.choice(num_examples, N, replace=False)  # Randomly sample N examples for trainin
      X_in = X[indices]
      y_in = y[indices]
      mask = np.ones(num_examples, dtype=bool)
      mask[indices] = False
      X_out = X[mask]
      y_out = y[mask]

      wlin = linear_regression(X_in, y_in)
      Ein, Eout = calculate_errors(X_in, y_in, X_out, y_out, wlin)

      Ein_array.append(Ein)
      Eout_array.append(Eout)

    # Calculate average Ein and Eout for the current N
    Ein_avg.append(np.mean(Ein_array))
    Eout_avg.append(np.mean(Eout_array))

  # Plot the learning curves
  plot_learning_curves(Ein_avg, Eout_avg)

if __name__ == '__main__':
  main()
```

12. no collaborators

① $\bar{E}_{out}(N)$ are larger than $\bar{E}_{in}(N)$ when N is small, and both $\bar{E}_{out}(N)$ and $\bar{E}_{in}(N)$ oscillate with N

② $\bar{E}_{out}(N) \approx \bar{E}_{in}(N)$ and $\bar{E}_{in}(N)$ are lower than 400 but higher than 250





hw3-12.py > download_and_parse_data

```python
import numpy as np…

def download_and_parse_data():
    # Downloading the dataset
    url = 'https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/regression/cpusmall_scale'
    response = requests.get(url)
    data = response.text

    # Preprocess the data
    lines = data.strip().split('\n')
    labels = []
    features = []
    for line in lines:
        tokens = line.split()
        labels.append(int(tokens[0]))  # The first value is the target (y)
        feature = [1]  # x0 = 1
        for token in tokens[1:3]: # Use only the first 2 features
            _, value = token.split(":")
            feature.append(float(value))
        features.append(feature)

    X = np.array(features)
    y = np.array(labels)

    return X, y

# Linear regression
def linear_regression(X, y):…

# Squared error
def squared_error(y_true, y_pred):…

# Error calculation for in-sample and out-of-sample
def calculate_errors(X_in, y_in, X_out, y_out, wlin):…

# Plot the learning curves
def plot_learning_curves(N_array, Ein_avg, Eout_avg):…

# Experiment parameters
num_examples = 8192
N_array = range(25, 2001, 25)  # N values from 25 to 2000
num_experiments = 16

def main():…

if __name__ == '__main__':
    main()
```