CSE 546 Homework #1 -B

Lu Yu

B.1

a. *[5 points]* Intuitively, how do you expect the bias and variance to behave for small values of m? What about large values of m?
For small values of m, for example ,$m = 1$, the line will be a straight line. The bias will be large and the variance is small.
Reversely, for large values of m, the line will become closer to the true f(x) which means a smaller bias and bigger variance.

b. *[5 points]* show that

$$\frac{1}{n}\sum_{i=1}^{n}(\mathbb{E}[(\hat{f}_m)(x_i)] - f(x_i))^2 = \frac{1}{n}\sum_{j=1}^{n/m}\sum_{i=(j-1)m+1}^{jm}(\bar{f}^{(j)} - f(x_i))^2$$

$$\frac{1}{n}\sum_{i=1}^{n}(\mathbb{E}[\hat{f}_m(x_i)] - f(x_i))^2$$
$$= \frac{1}{n}[\sum_{i=1}^{m}(\mathbb{E}[\hat{f}_m(x_i)] - f(x_i))^2 + \cdots + \sum_{i=n-m+1}^{n}(\mathbb{E}[\hat{f}_m(x_i)] - f(x_i))^2]$$
$$= \frac{1}{n}[\sum_{i=1}^{m}(\mathbb{E}[\sum_{j=1} c_1 \mathbf{1}\{x \in (0, \frac{m}{n})\}] - f(x_i))^2 + \cdots + \sum_{(j-1)m+1}^{jm}(\mathbb{E}[\sum_{j=1} c_{n/m}\mathbf{1}\{x \in (\frac{n-m}{n}, 1)\}] - f(x_i))^2]$$
$$= \frac{1}{n}[\sum_{i=1}^{m}(\mathbb{E}[\frac{1}{m}\sum_{i=1}^{m} y_i \mathbf{1}\{x \in (0, \frac{m}{n})\}] - f(x_i))^2 + \cdots + \sum_{(j-1)m+1}^{jm}(\mathbb{E}[\frac{1}{m}\sum_{(j-1)m+1}^{jm} y_i\mathbf{1}\{x \in (\frac{n-m}{n}, 1)\}] - f(x_i))^2]$$
$$= \frac{1}{n}[\sum_{i=1}^{m}(\frac{1}{m}\sum_{i=1}^{m}\mathbb{E}[y_i]\mathbf{1}\{x \in (0, \frac{m}{n})\} - f(x_i))^2 + \cdots + \sum_{(j-1)m+1}^{jm}(\frac{1}{m}\sum_{(j-1)m+1}^{jm}\mathbb{E}[y_i]\mathbf{1}\{x \in (\frac{n-m}{n}, 1)\} - f(x_i))^2]$$
$$= \frac{1}{n}[\sum_{i=1}^{m}(\frac{1}{m}\sum_{i=1}^{m}\mathbb{E}[f(x_i) + \epsilon_i] - f(x_i))^2 + \cdots + \sum_{(j-1)m+1}^{jm}(\frac{1}{m}\sum_{(j-1)m+1}^{jm}\mathbb{E}[f(x_i) + \epsilon_i] - f(x_i))^2 +]$$
$$= \frac{1}{n}[\sum_{i=1}^{m}(\frac{1}{m}\sum_{i=1}^{m}\mathbb{E}[f(x_i)] - f(x_i))^2 + \cdots + \sum_{(j-1)m+1}^{jm}(\frac{1}{m}\sum_{(j-1)m+1}^{jm}\mathbb{E}[f(x_i)] - f(x_i))^2]$$
$$= \frac{1}{n}[\sum_{i=1}^{m}(\frac{1}{m}\sum_{i=1}^{m}\frac{1}{m}\sum_{i=1}^{m} f(x_i) - f(x_i))^2 + \cdots + \sum_{(j-1)m+1}^{jm}(\frac{1}{m}\sum_{(j-1)m+1}^{jm}\frac{1}{m}\sum_{(j-1)m+1}^{jm} f(x_i) - f(x_i))^2]$$
$$= \frac{1}{n}[\sum_{i=1}^{m}(\frac{1}{m}\sum_{i=1}^{m} f(x_i) - f(x_i))^2 + \cdots + \sum_{i=(j-1)m+1}^{jm}(\frac{1}{m}\sum_{i=(j-1)m+1}^{jm} f(x_i) - f(x_i))^2]$$
$$= \frac{1}{n}[\sum_{i=1}^{m}(\bar{f}^{(1)} - f(x_i))^2 + \cdots + \sum_{i=n-m+1}^{n}(\bar{f}^{(n/m)} - f(x_i))^2]$$
$$= \frac{1}{n}\sum_{j=1}^{n/m}\sum_{i=(j-1)m+1}^{jm}(\bar{f}^{(j)} - f(x_i))^2$$

c. *[5 points]* show that

$$\mathbb{E}[\frac{1}{n}\sum_{i=1}^{n}(\hat{f}_m(x_i) - \mathbb{E}[\hat{f}_m(x_i)])^2] = \frac{1}{n}\sum_{j=1}^{n/m} m\mathbb{E}[(c_j - \bar{f}^{(j)})^2] = \frac{\sigma^2}{m}$$

$$
\begin{aligned}
&\mathbb{E}[\frac{1}{n}\sum_{i=1}^{n}(\hat{f}_m(x_i)-\mathbb{E}[\hat{f}_m(x_i)])^2] \\
&=\frac{1}{n}[\sum_{i=1}^{m}\mathbb{E}(\hat{f}_m(x_i)-\mathbb{E}[\hat{f}_m(x_i)])^2+\cdots+\sum_{i=n-m+1}^{n}\mathbb{E}(\hat{f}_m(x_i)-\mathbb{E}[\hat{f}_m(x_i)])^2] \\
&=\frac{1}{n}[\sum_{i=1}^{m}\mathbb{E}(\hat{f}_m(x_i)-\bar{f}^{(1)})^2+\cdots+\sum_{i=n-m+1}^{n}\mathbb{E}(\hat{f}_m(x_i)-\bar{f}^{(n/m)})^2] \\
&=\frac{1}{n}[\sum_{i=1}^{m}\mathbb{E}(c_1\mathbf{1}\{x\in(0,\frac{m}{n})\}-\bar{f}^{(1)})^2+\cdots+\sum_{i=n-m+1}^{n}\mathbb{E}(c_{\frac{n}{m}}\mathbf{1}\{x\in(\frac{n-m}{n},1)\}-\bar{f}^{(n/m)})^2] \\
&=\frac{1}{n}[\sum_{i=1}^{m}\mathbb{E}(c_1-\bar{f}^{(1)})^2+\cdots+\sum_{i=n-m+1}^{n}\mathbb{E}(c_{n/m}-\bar{f}^{(n/m)})^2] \\
&=\frac{1}{n}[mE(c_1-\bar{f}^{(1)})^2+\cdots+mE(c_{n/m}-\bar{f}^{(n/m)})^2] \\
&=\frac{1}{n}\sum_{j=1}^{n/m}m\mathbb{E}[(c_j-\bar{f}^{(j)})^2]
\end{aligned}
$$

$$c_j=\frac{1}{m}\sum_{i=(j-1)m+1}^{jm}y_i=\frac{1}{m}\sum_{i=(j-1)m+1}^{jm}(f(x_i)+\epsilon_i)\qquad,\epsilon_i\sim N(0,\sigma^2)$$

$$\bar{f}^{(j)}=\frac{1}{m}\sum_{i=(j-1)m+1}^{jm}f(x_i)$$

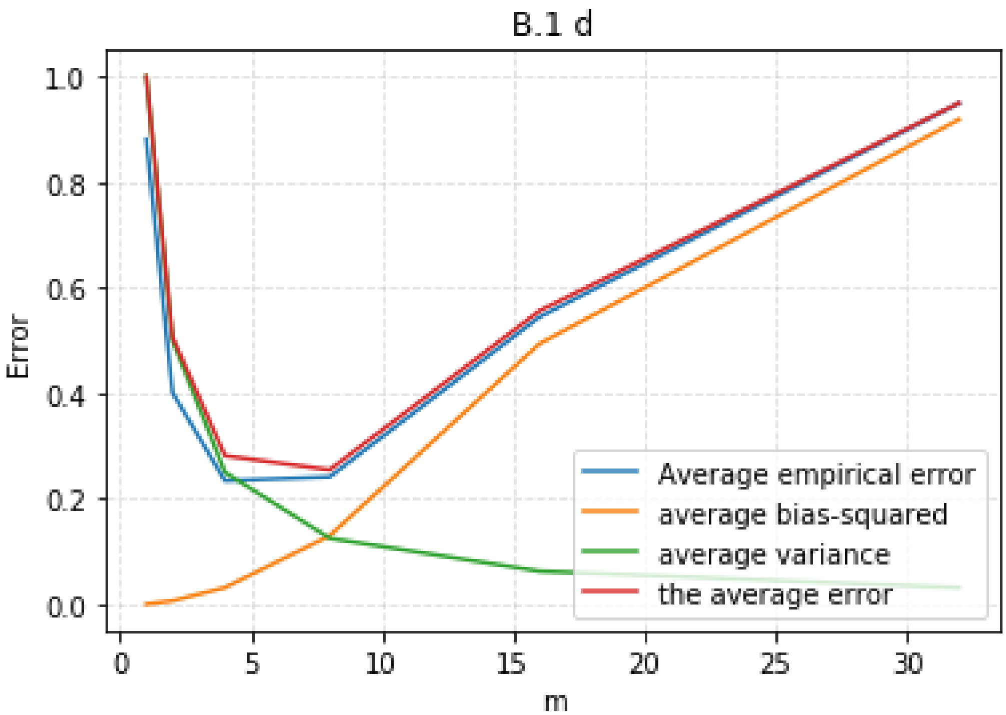
$$\mathbb{E}[(c_j-\bar{f}^{(j)})^2]=\frac{\sigma^2}{m}$$

$$\mathbb{E}[\frac{1}{n}\sum_{i=1}^{n}((\hat{f})_m(x_i)-\mathbb{E}[(\hat{f}_m)(x_i)])^2]=\frac{1}{n}\sum_{j=1}^{n/m}m\mathbb{E}[(c_j-\bar{f}^{(j)})^2]=\frac{1}{n}\frac{n}{m}m\frac{\sigma^2}{m}=\frac{\sigma^2}{m}$$

d. *[15 points]*

[4]:
```
"""
Created on Tue Apr 14 22:28:26 2020
CSE 546 HW 1 B1
@author: Leah
"""

import numpy as np
#import math
import matplotlib.pyplot as plt
%matplotlib inline
n = 256
sigma = 1

df = np.ones((n,15))
df2 = np.ones((6,5))
```

```
##x_i
df[:,0] = np.arange(1,n+1)/float(n)
##f(x_i)
df[:,1] = 4* np.sin(np.pi * df[:,0]) * np.cos(6 * np.pi * df[:,0]**2)
##y_i
df[:,2] = 4* np.sin(np.pi * df[:,0]) * np.cos(6 * np.pi * df[:,0]**2) + np.
 ↪random.randn(n)
for k in range(0,6):
    m = 2**k
    df2[k,0] = m
    for j in range(1,int(n/m)+1):
        ##hat{f_m(x^i)}
        df[(j-1)*m:j*m,k+3] = np.mean(df[(j-1)*m:j*m,2])
        df[(j-1)*m:j*m,k+9] = np.mean(df[(j-1)*m:j*m,1])
    df2[k,1] = ((df[:,k+3]-df[:,1])**2).sum()/n
    df2[k,2] = ((df[:,k+9]-df[:,1])**2).sum()/n
    df2[k,3] = sigma**2/m
    df2[k,4] = df2[k,2] + df2[k,3]


# Create the vectors X and Y
x_axix = df2[:,0]
#y1= df2[:,1]
# Create the plot
plt.plot(x_axix,df2[:,1],label='Average empirical error') # Add a title
plt.plot(x_axix,df2[:,2],label='average bias-squared')
plt.plot(x_axix,df2[:,3],label='average variance')
plt.plot(x_axix,df2[:,4],label='the average error')
plt.title('B.1 d')
# Add X and y Label
plt.xlabel('m')
plt.ylabel('Error')
# Add a grid
plt.grid(alpha=.4,linestyle='--')
# Add a Legend
plt.legend()
# Show the plot
plt.show()
```



e. *[5 points]* By the Mean-Value theorem...
From B.1b, $\bar{f}^{(j)} = \mathbb{E}[\hat{f}_m(j)] = \sum_{j=1}^{n/m} c_j \mathbf{1}\{j \in I_j\},\ I_j = (\frac{(j-1)m}{n}, \frac{jm}{n}],\ N_j = \{i : x_i \in I_j\}$

$$
\begin{aligned}
Average\ bias - squared &= ||\bar{f} - f||^2 \\
&= \int_0^1 (\bar{f}^{(j)} - f(x_i))^2 dx \\
&= \sum_{j=1}^{n/m} \int_{I_j} (\bar{f}^{(j)} - f(x_i))^2 dx \\
&= \sum_{j=1}^{n/m} \int_{I_j} (c_j - f(x_i))^2 dx \\
&= \sum_{j=1}^{n/m} \int_{I_j} (\frac{1}{m}(\sum_{i \in N_j} f(\frac{i}{n})) - f(x_i))^2 dx \\
&= \sum_{j=1}^{n/m} \int_{I_j} (\frac{1}{m} \sum_{i \in N_j} (f(\frac{i}{n}) - f(x_i)))^2 dx \\
&\leq \sum_{j=1}^{n/m} \int_{I_j} (\frac{1}{m} \sum_{i \in N_j} |f(\frac{i}{n}) - f(x_i)|)^2 dx \\
&\leq \sum_{j=1}^{n/m} \int_{I_j} (\frac{1}{m} \sum_{i \in N_j} \frac{Lm}{n})^2 dx \\
&= \sum_{j=1}^{n/m} \int_{I_j} (\frac{Lm}{n})^2 dx \\
&= \sum_{j=1}^{n/m} \frac{m}{n} (\frac{Lm}{n})^2 dx \\
&= (\frac{Lm}{n})^2
\end{aligned}
$$

$$
Let \quad total\ error\ E = (\frac{Lm}{n})^2 + \frac{\sigma^2}{m}
$$

$$
E' = \frac{2L^2 m}{n^2} - \frac{\sigma^2}{m^2} = 0
$$

$$
m^\star = (\frac{n^2 \sigma^2}{2L^2})^{\frac{1}{3}}
$$

Intuitivelly, with the increasing of L and the decreasing of n, the total error will raise up. It is easy to see that there are more data points, the total error will decrease. Regard to L, according to L-Lipschitz, if the tolerance of changing speed of function is high, there will be more error.

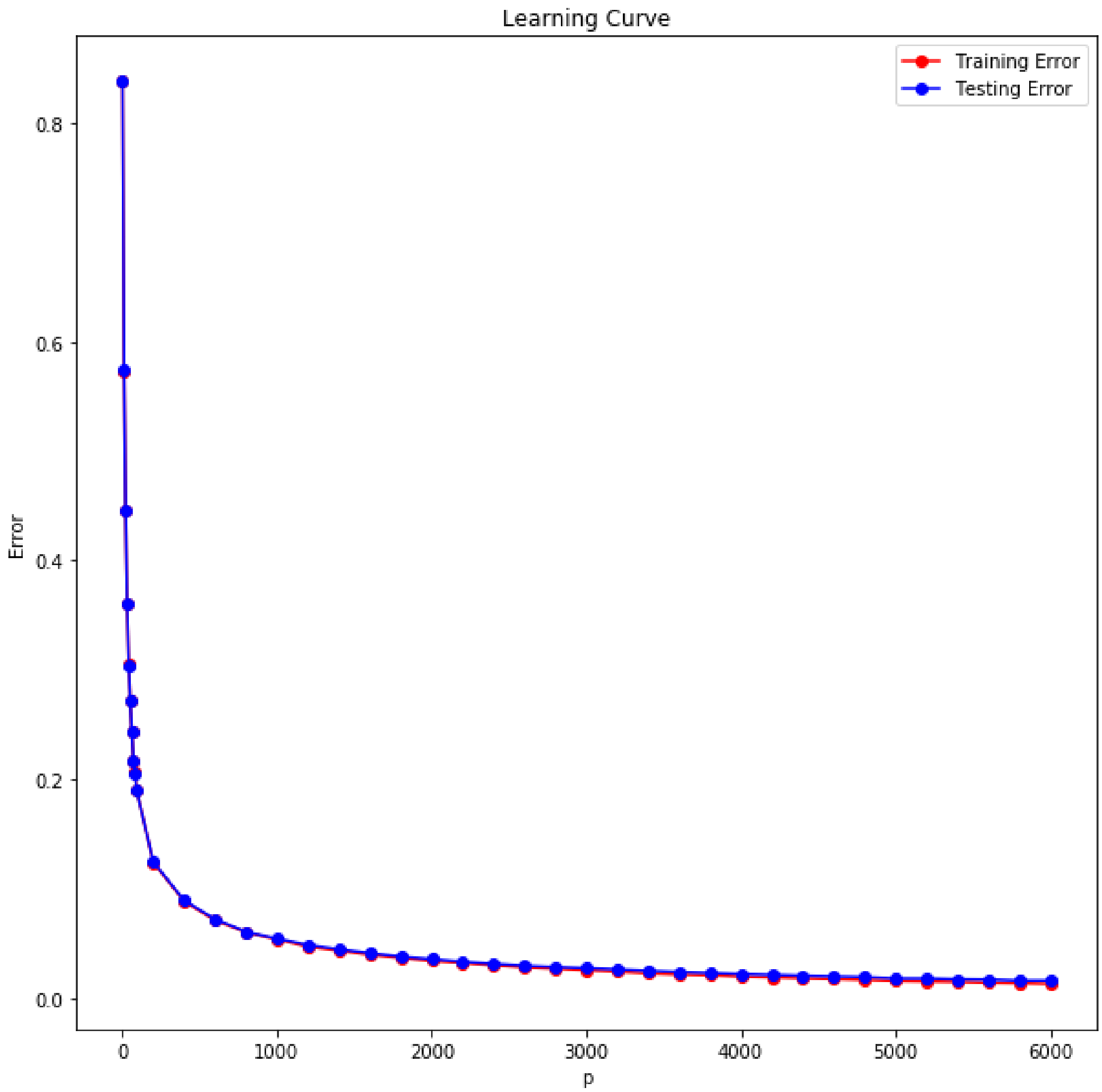
B.2

[5]:
```python
"""
Created on Thu Apr 18 13:10:52 2020
CSE 546 HW 1 b2
@author: Leah
"""
import numpy as np
from mnist import MNIST
import matplotlib.pyplot as plt
from sklearn import model_selection
import scipy

img_h = img_w = 28                                          #␣
↪MNIST images are 28x28
img_size_flat = img_h * img_w                               #␣
↪28x28=784, the total number of pixels
n_classes = 10                                              #␣
↪Number of classes, one class per digit

def load_dataset():
    mndata = MNIST('./data/')
    X_train, labels_train = map(np.array, mndata.load_training())
    X_test, labels_test = map(np.array, mndata.load_testing())
    X_train = X_train/255.0
    X_test = X_test/255.0

    return X_train, labels_train, X_test, labels_test

def trans_coef(p, d):
    mu, sigma = 0, 0.1 # mean and standard deviation
    G = np.random.normal(mu, sigma, (p,d))
    b = np.random.uniform(0,2 * np.pi,p)
    return G, b

def transform(G,b,X,p=1000):
    n, d = X.shape
    h = np.cos(G.dot(X.T) + np.repeat(b,n).reshape((p,n))).T
    return h

def train(h,Y,lamda,p):
    W_hat = scipy.linalg.solve(
            a = h.T.dot(h)+ lamda * np.eye(p),
            b = h.T.dot(Y)
            )
    return W_hat
```

```
def predict(W, h):
    X_ = h.dot(W)
    pre_ = X_.argmax(axis=1)
    return pre_



#-----------------------------------------------------------------
#  Main
#-----------------------------------------------------------------
def main():
    n = 10
    gap = 10
    error_table = np.zeros((n-1,3))                    #record the two errors

    XX_train, labelss_train, XX_test, labelss_test = load_dataset()           ␣
 ↪      #LOAD data

    #Redistribution data 80/20
    X_ = np.r_[XX_train,XX_test]
    labels_ = np.r_[labelss_train,labelss_test]
    rand_schedule = np.random.permutation(range(len(X_))).tolist()
    slice_list = [0, int(len(X_)/5), int(len(X_)/5*2), int(len(X_)/5*3),␣
 ↪int(len(X_)/5*4), len(X_)]
    sliced_schedule = [rand_schedule[slice_list[i]: slice_list[i + 1]] for i in␣
 ↪range(len(slice_list) - 1)]

    loo = model_selection.LeaveOneOut()


    for p in range(1,n*gap,gap):
        itrial = 0
        errorTrains = np.zeros((5, 1))
        errorVal = np.zeros((5, 1))
        print(p)
        for train_index, test_index in loo.split(sliced_schedule):
            #print (train_index, test_index)
            X_train = np.vstack((X_[sliced_schedule[i]] for i in␣
 ↪range(0,len(train_index))))
            X_test = X_[sliced_schedule[test_index[0]]]

            labels_train = np.vstack((labels_[sliced_schedule[i]] for i in␣
 ↪range(0,len(train_index)))).reshape(int(len(X_)*0.8),)
            labels_test = labels_[sliced_schedule[test_index[0]]]

            #X_train, X_test, labels_train, labels_test = train_test_split(X_,␣
 ↪labels_, test_size=0.2)
```

```
            ## transform labels_train(n-by-1) into n-by-k
            Y_train = np.zeros((len(labels_train),n_classes))

            for i in range(0,len(Y_train)):
                Y_train[i,labels_train[i]] = 1

            G, b =  trans_coef(p, X_train.shape[1])
            #get w_hat
            X_train = transform(G,b,X_train, p=p)
            X_test = transform(G,b,X_test,p=p)

            W_hat = train(X_train,Y_train,lamda = 0.0001,p=p)
            #print(W_hat.shape[0],W_hat.shape[1])
            #predict
            Train_pre = predict(W_hat, X_train)
            Test_pre = predict(W_hat, X_test)

            errorTrains[itrial, :] = sum(Train_pre != labels_train)/
 ↪len(labels_train)
            errorVal[itrial, :] = sum(Test_pre != labels_test)/len(labels_test)
            itrial = itrial + 1

        ##training, test error
        error_table[int(p/gap-1),0] = p
        error_table[int(p/gap-1),1] = errorTrains.mean()
        error_table[int(p/gap-1),2] = errorVal.mean()

    import pandas as pd
    error_table = pd.read_excel("1.xlsx")
    error_table = error_table[error_table["a"]>0]

    plt.figure(figsize=(10,10))
    plt.plot(error_table[:,0], error_table[:,1], 'r-o')
    plt.plot(error_table[:,0], error_table[:,2],'b-o')
    plt.legend(['Training Error', 'Testing Error'], loc='best')
    plt.title('Learning Curve')
    plt.xlabel('p')
    plt.ylabel('Error')

    return error_table

def find_p_hat():
    error_table = main()
    p_hat = int(error_table[error_table[:,2].argmin(axis = 0),0])
    XX_train, labelss_train, XX_test, labelss_test = load_dataset()
    Y_train = np.zeros((len(labelss_train),n_classes))
```

```
    for i in range(0,len(Y_train)):
        Y_train[i,labelss_train[i]] = 1

    G, b =  trans_coef(p_hat, XX_train.shape[1])

    #get w_hat
    X_train = transform(G,b,XX_train, p=p_hat)
    X_test = transform(G,b,XX_test,p=p_hat)

    W_hat = train(X_train,Y_train,lamda = 0.0001,p=p_hat)

    #predict
    Test_pre = predict(W_hat, X_test)

    test_error= sum(Test_pre != labelss_test)/len(labelss_test)

    # apply Hoeffding's inequality
    delta = 0.05
    m = len(XX_test)
    ci = np.sqrt(np.log10(2/delta) / (2*m))

    low_bound = test_error - ci
    up_bound = test_error + ci

    # result
    print("p_hat: " + str(p_hat))
    print()
    print("Test Error: " + str(test_error))
    print()
    print("Confidence Interval: " + str(low_bound) + "<= mean error <= " +
 ↪str(up_bound))
    print()

find_p_hat()
```




```
p_hat: 5801

Test Error: 0.0266

Confidence Interval: 0.017649972091306188<= mean error <= 0.03555002790869381
```

So the CI is [0.017649972091306188, 0.03555002790869381]