Net Id: N17527962 Name: Devashish Joshi

DSGA-1003 Machine Learning

HW-1

1. **Which function f^* is an obvious Bayes predictor?**
   Risk is the expected loss for f, given by formula:

   $$R(f^*) = \mathbb{E}[\ell(f(x), y)]$$

   Expected loss is the product of probability of the pair (x, y) and the loss for that (x, y)

   It is given that $x \in \mathcal{X} = [0, 1]$ is distributed uniformly ($x \sim \text{Unif}([0, 1])$). This means that the probability of x being equal to any real value $\in [0, 1]$ is constant and is same for all $x \in [0, 1]$. The probability of x being equal to any other value is 0.

   For $x \in [0, 1]$, the probability that y is $a_0 + a_1 * x + a_2 * x^2$, is 1, and the probability that y is any other value is 0.

   The loss for 1 pair of $(x, a_0 + a_1 * x + a_2 * x^2)$ can be calculated as follows:
   (As y is a continuous variable, we will use Mean Squared Loss as this becomes a regression problem)

   $$\begin{aligned}\ell(f(x), y) &= \tfrac{1}{2} * (f(x) - y)^2 \\ &= \tfrac{1}{2} * (f(x) - a_0 + a_1 x + a_2 x^2)^2\end{aligned}$$

   So, if $f(x)$ is $a_0 + a_1 x + a_2 x^2$, then the loss for each pair of (x, y) is 0. Multiplying this with the constant real value of probability and then adding all the resulting terms will still result in 0. The risk cannot be negative, and so 0 is the minimum value of the risk.
   This is why the obvious Bayes predictor is

   $$\boxed{f^* = a_0 + a_1 x + a_2 x^2}$$

2. **Using $\mathcal{H}_2$ as your hypothesis class, which function $f^*_{\mathcal{H}_2}$ is a risk minimizer in $\mathcal{H}_2$? Recall the definition of the approximation error. What is the approximation error achieved by $f^*_{\mathcal{H}_2}$?**
   $\mathcal{H}_2$ is a hypothesis class with set of polynomial functions with degree 2.
   In the previous question, we proved that the optimizer function with minimum (in this case, 0) risk is:

$$f^* = a_0 + a_1 x + a_2 x^2$$

   This is a polynomial function with degree 2, and is therefore included in $\mathcal{H}_2$
   So, $f^*_{\mathcal{H}_2}$ is same as f^*:

$$\boxed{f^*_{\mathcal{H}_2} = a_0 + a_1 x + a_2 x^2}$$

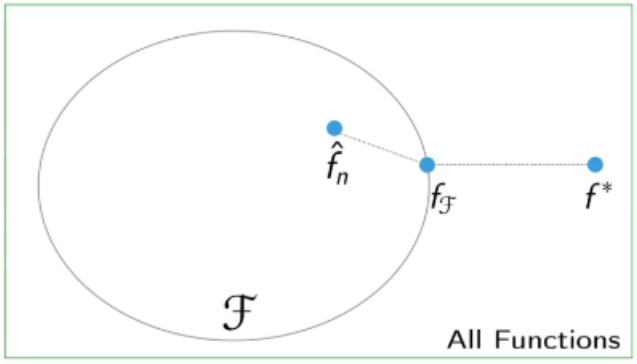
   The approximation error is:

$$\begin{aligned} \text{Approximation Error} &= R(f^*) - R(f^*_{\mathcal{H}_2}) \\ &= 0 - 0 \\ &= 0 \end{aligned}$$

   So, the $\boxed{\text{Approximation Error} = 0}$

3. **Considering now $\mathcal{H}_d$, with $d > 2$. Justify an inequality between $R(f^*_{\mathcal{H}_2})$ and $R(f^*_{\mathcal{H}_d})$. Which function $f^*_{\mathcal{H}_d}$ is a risk minimizer in $\mathcal{H}_d$? What is the approximation error achieved by $f^*_{\mathcal{H}_d}$?**

   For any value of d, such that $d > 2$, $\mathcal{H}_d$ will include set of polynomials with degrees ranging from 0 to d.
   We can interpret this using the figure showed in class.



   As d increases, more polynomials will get added to the set.

   $f^*_{\mathcal{H}_d}$ will move closer to the optimum, i.e. f^* (or if f^* is already included in the set $\mathcal{H}_d$, there will be no change in $f^*_{\mathcal{H}_d}$ as d increases)
   So, we can say that,

$$R(f^*_{\mathcal{H}_d}) >= R(f^*_{\mathcal{H}_2})$$

   Now, we know that $g(x) = a_0 + a_1x + a_2x^2$. The degree of this polynomial function is 2. This means we don't need the higher order terms so the risk minimizer in $f^*_{\mathcal{H}_d}$ is the same as the one showed earlier:

$$\boxed{f^*_{\mathcal{H}_d} = a_0 + a_1x + a_2x^2}$$

   The approximation error will also be same as before.

$$\boxed{\text{Approximation Error} = 0}$$

4. **For this question we assume $a_0 = 0$. Considering $\mathcal{H} = \{f : x \to b_1 x; b_1 \in \mathbb{R}\}$, which function $f^*_{\mathcal{H}}$ is a risk minimizer in $\mathcal{H}$? What is the approximation error achieved by $f^*_{\mathcal{H}}$? In particular what is the approximation error achieved if furthermore $a_2 = 0$ in the definition of true underlying relation $g(x)$ above?**

$$a_0 = 0$$
$$y = a_1 * x + a_2 * x^2$$
$$f(x) = b_1 * x$$

To find the risk minimizer, we need to minimize the risk, which is the expected loss. Loss for a single (x, y) is given by:

$$\ell(f(x), y) = \tfrac{1}{2} * (f(x) - y)^2$$

Substitute values of $f(x)$ and y

$$\ell(f(x), y) = \tfrac{1}{2} * (b_1 * x - a_1 * x + a_2 * x^2)^2$$

Now, risk is given by:

$$R(f^*) = \mathbb{E}[\ell(f^*(x), y)]$$

As x is uniformly distributed, the probability of $(x, a_1 * x + a_2 * x^2)$ is constant for all $x \in [0, 1]$. Let this probability be dx.

So, the risk is:

$$\begin{aligned}
R(f^*) &= \mathbb{E}[\ell(f^*_{\mathcal{H}}(x), y)] \\
&= \int_0^1 \frac{1}{2}(b_1 x - a_1 x + a_2 x^2)^2 \, dx \\
&= \int_0^1 \frac{1}{2}(b_1^2 x^2 + a_1^2 x^2 + a_2^2 x^4 - 2a_1 b_1 x^2 + 2a_2 b_1 x^3 - 2a_2 a_1 x^3) \, dx \\
&= \frac{1}{2}\left[(\frac{1}{3}b_1^2 x^3 + \frac{1}{3}a_1^2 x^3 + \frac{1}{5}a_2^2 x^5 - \frac{2}{3}a_1 b_1 x^3 + \frac{2}{4}a_2 b_1 x^4 - \frac{2}{4}a_2 a_1 x^4)\right]_0^1 \\
&= \frac{1}{2}\left[(\frac{1}{3}b_1^2 + \frac{1}{3}a_1^2 + \frac{1}{5}a_2^2 - \frac{2}{3}a_1 b_1 + \frac{1}{2}a_2 b_1 - \frac{1}{2}a_2 a_1)\right]
\end{aligned}$$

Differentiate the risk ($R(f^*_{\mathcal{H}})$) wrt b_1, and equate the same to 0, to find the value of b_1 which minimizes the risk.

$$0 = \frac{1}{2}\left[(\frac{2}{3}b_1 - \frac{2}{3}a_1 + \frac{1}{2}a_2)\right]$$
$$0 = \frac{2}{3}b_1 - \frac{2}{3}a_1 + \frac{1}{2}a_2$$
$$0 = 2b_1 - 2a_1 + 3a_2$$

$$\boxed{b_1 = a_1 - \frac{3}{2}a_2}$$

Risk minimizer is:

$$\boxed{f^*_{\mathcal{H}} = (a_1 - \frac{3}{2}a_2)x}$$

Now, we will substitute this value in formula for risk ($R(f^*)$)

$$\begin{aligned} R(f^*_{\mathcal{H}}) &= \frac{1}{2}\left[(\frac{1}{3}(a_1 - \frac{3}{2}a_2)^2 + \frac{1}{3}a_1^2 + \frac{1}{5}a_2^2 - \frac{1}{3}a_1(2a_1 - 3a_2) + \frac{1}{2}a_2(a_1 - \frac{3}{2}a_2) - \frac{1}{2}a_2a_1)\right] \\ &= \frac{1}{2}\left[(\frac{1}{3}a_1^2 + \frac{3}{4}a_2^2 - a_1a_2 + \frac{1}{3}a_1^2 + \frac{1}{5}a_2^2 - \frac{2}{3}a_1^2 + a_1a_2 + \frac{1}{2}a_1a_2 - \frac{3}{4}a_2^2 - \frac{1}{2}a_2a_1)\right] \\ &= \frac{1}{2}\left[(\frac{1}{5}a_2^2)\right] \end{aligned}$$

Approximation error is $R(f^*_{\mathcal{H}}) - R(f^*)$

As proved earlier, the term $R(f^*)$ is 0. So, the approximation error is:

$$\boxed{\text{Approximation error} = \frac{a_2^2}{10}}$$

$\boxed{\text{If } a_2 = 0\text{, approximation error} = 0}$

5. **Show that the empirical risk minimizer (ERM) $\hat{b}$ is given by the following minimization $\hat{\boldsymbol{b}} = \arg\min_b \|Xb - \boldsymbol{y}\|_2^2$.**

It is given that functions in $\mathcal{H}_d$ are parametrized by a vector $\boldsymbol{b} = [b_0, b_1, \cdots b_d]^\top$

So, $f_{\boldsymbol{b}}$ is vector of $d+1$ dimensions, formed by multiplying the design matrix X with the weights $\boldsymbol{b}$. This can be interpreted as matrix multiplication, as X has N rows, $d+1$ columns, and $\boldsymbol{b}$ has $d+1$ rows and only 1 column

$$f_{\boldsymbol{b}} = Xb$$

Assuming the output y to be continuous, this becomes a regression problem. The loss is Mean Squared Loss.

$$\ell(f_{\boldsymbol{b}}, y) = \frac{1}{2}(f_{\boldsymbol{b}} - y)^2$$

Generally the ERM is represented by $\hat{f}_{\boldsymbol{b}}$.
But the design matrix X is constant, and so we can consider the minimizer to be just $\hat{\boldsymbol{b}}$

The ERM $\hat{\boldsymbol{b}}$ is given by:

$$\begin{aligned}
\hat{\boldsymbol{b}} &= \arg\min_b \ell(f_{\boldsymbol{b}}, y) \\
&= \arg\min_b \frac{1}{2}(f_{\boldsymbol{b}} - y)^2 \\
&= \arg\min_b (Xb - y)^2 \\
&= \arg\min_b \|Xb - \boldsymbol{y}\|_2^2 \\
\boxed{\hat{\boldsymbol{b}} &= \arg\min_b \|Xb - \boldsymbol{y}\|_2^2}
\end{aligned}$$

6. **If $N > d$ and X is full rank, show that $\hat{b} = (X^\top X)^{-1} X^\top y$. (Hint: you should take the gradients of the loss above with respect to b. Why do we need to use the conditions $N > d$ and X full rank?**

   The loss function is:

$$\ell(f_b, y) = \frac{1}{2}(Xb - y)^2$$

   Expand the square term, (for matrices, $X^2 = X^T X$):

$$\begin{aligned}\ell(f_b, y) &= \frac{1}{2}(Xb - y)^T (Xb - y) \\ &= \frac{1}{2}(b^T X^T - y^T)(Xb - y) \qquad \text{...(Using rule } (Xb)^T = b^T X^T) \\ &= \frac{1}{2}(b^T X^T Xb - y^T Xb - (Xb)^T y + y^T y)\end{aligned}$$

   Let us focus on the 2nd and 3rd term here. Let $f = Xb$

   We need to find value of $y^T f + f^T y$.

   Note that the shape of both the terms is 1x1. The transpose of a 1x1 matrix is the equal to the same matrix. Transpose of the first term is the second term (and vice versa). So, we can conclude that both the terms are equal.

$$\ell(f_b, y) = \frac{1}{2}(b^T X^T Xb - 2(Xb)^T y + y^T y)$$

   Now we will differentiate wrt b, and equate to differentiation of the loss to 0, to find the point at which loss is minimum

   The last term ($y^T y$) is constant wrt b. So, it's differentiation is 0

   In the differentiation of first term, $X^T X$ is constant, so can be treated as a scalar.

   Using the rule $\frac{db^T b}{db} = 2b$,

$$0 = (X^T X)2b - 2X^T y$$

$$2X^T y = (X^T X)2b$$

$$X^T y = (X^T X)b$$

   Before we go to the next step, the matrix $(X^T)X$ should be invertible. A matrix is invertible when it's determinant is non-zero. The determinant of $(X^T)X$ is $\det X^T * \det X$,

which is $(\det X)^2$. So, $\det X$ should be non-zero. This means $\boxed{X \text{ should be full-rank.}}$

Also, the rank of matrix X^TX is same as rank of matrix X. So, for X^TX to be invertible, the number of rows in X should be more than or equal to the number of columns.

$$N > d + 1$$
$$\boxed{N > d}$$

If the above 2 conditions are met, (X^TX) is invertible. So,

$$\boxed{\hat{\boldsymbol{b}} = (X^\top X)^{-1} X^\top \boldsymbol{y}}$$

7. **Write a function called `least_square_estimator` taking as input a design matrix $X \in \mathbb{R}^{N\times(d+1)}$ and the corresponding vector $y \in \mathbb{R}^N$ returning $\hat{b} \in \mathbb{R}^{(d+1)}$. Your function should handle any value of N and d, and in particular return an error if $N \leq d$. (Drawing x at random from the uniform distribution makes it almost certain that any design matrix X with $d \geq 1$ we generate is full rank).**

[]:
```
def least_square_estimator(X, y):
    # Shape of X is N * (d+1)
    N = X.shape[0]
    d = X.shape[1]-1

    if N<=d:
        raise Exception("Sorry, N should be strictly greater then d. \
                        Please check your input and try again")

    b_hat = np.matmul( np.matmul(np.linalg.pinv( np.matmul(
                        np.transpose(X), X) ), np.transpose(X)), y)
    return b_hat
```

8. **Recall the definition of the empical risk $\hat{R}(\hat{f})$ on a sample $\{x_i, y_i\}_{i=1}^N$ for a prediction function $\hat{f}$. Write a function `empirical_risk` to compute the empirical risk of f_b taking as input a design matrix $X \in \mathbb{R}^{N\times(d+1)}$, a vector $y \in \mathbb{R}^N$ and the vector $b \in \mathbb{R}^{(d+1)}$ parametrizing the predictor.**

[]:
```
def empirical_risk(X,y,b):
    f_hat = np.matmul(X,b)            # Find f_hat from weight vector b and
                                      # design matrix X

    loss = 1/2 * ((f_hat-y)**2).sum() # Calculate loss from predicted and
                                      #  true values
    n = X.shape[0]
    return 1/n * loss                 # Return constant probability (1/n)
                                      #  multiplied by loss
                                      # (probability is constant
                                      #  as distribution is uniform)
```

9. **Use your code to estimate $\hat{b}$ from `x_train, y_train` using $d = 5$. Compare $\hat{b}$ and a. Make a single plot (Plot 1) of the plan (x, y) displaying the points in the training set, values of the true underlying function $g(x)$ in $[0, 1]$ and values of the estimated function $f_{\hat{b}}(x)$ in $[0, 1]$. Make sure to include a legend to your plot.**

[4]:
```
deg_true = 2
deg = 5
N = 10
```

[5]:
```
a = get_a(deg_true)
x, y = draw_sample(deg_true,a,N)
```

[6]:
```
x
```

[6]:
```
array([0.03989659, 0.05711273, 0.2266562 , 0.33840313, 0.40000868,
       0.45678672, 0.55294269, 0.58664739, 0.65069397, 0.95293369])
```

[7]:
```
y
```

[7]:
```
array([-6.89885411, -6.69801505, -4.77485621, -3.56158711, -2.91116416,
       -2.32331957, -1.35318084, -1.02068686, -0.39968681,  2.33962894])
```

[8]:
```
X_a = get_design_mat(x, deg_true)
X_b = get_design_mat(x, deg)
```

[9]:
```
b_hat = least_square_estimator(X_b,y)
```

[10]:
```
b_hat
```

[10]:
```
array([-7.36821352e+00,  1.18333108e+01, -1.72728569e+00, -1.62617653e-09,
        3.56840246e-09, -1.31808520e-09])
```

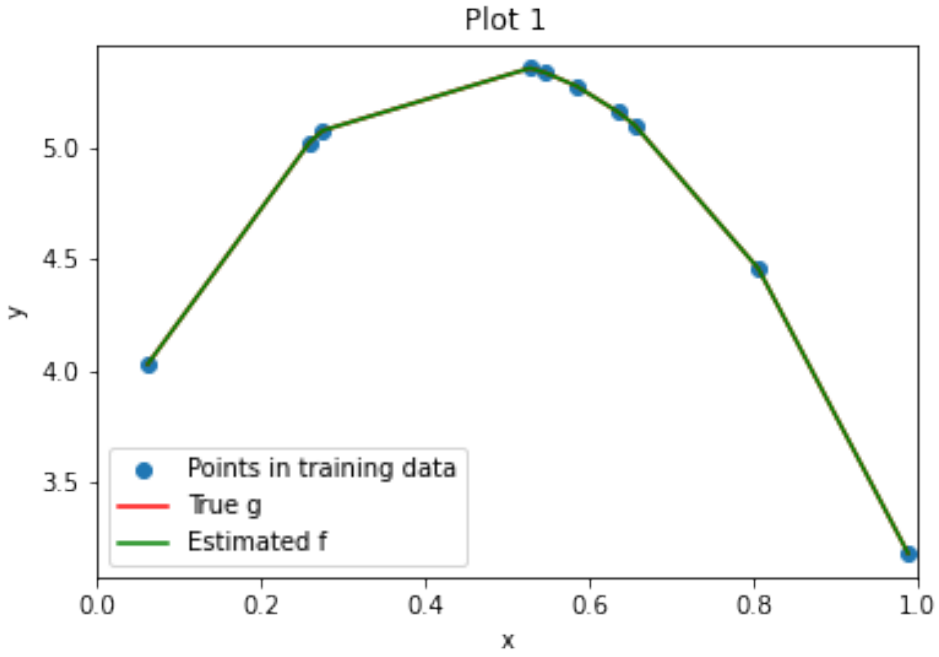
[11]:
```
a
```

[11]:
```
array([-7.36821352, 11.8333108 , -1.72728569])
```

[12]:
```
# The last 3 values of b are of order -8, while the first 3 are of order 0.
# This is consistent with the fact that a has 3 dimensions
```

[13]:
```
f_x = np.matmul(X_b, b_hat)
```

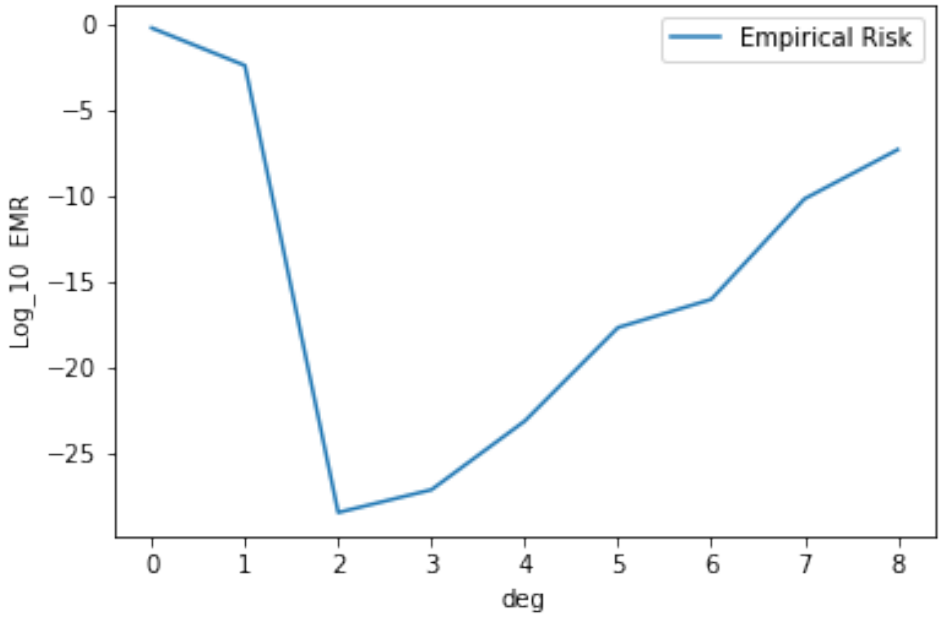
[14]:
```
g_x = y
```



In the figure, the true and estimated functions go through almost the same points, so only green can be seen, no red.

10. **Now you can adjust d. What is the minimum value for which we get a "perfect fit"? How does this result relates with your conclusions on the approximation error above?**

We vary d from 0 to 9, and plot EMR (on Y-axis).
The plot is:



d is the degree of the polynomial. Assuming d cannot be negative, it can be clearly seen from the plot, that the minimum value of d for which we get a minimum empirical risk is

d=2

This is obvious because the true function $g(x)$ has degree 2

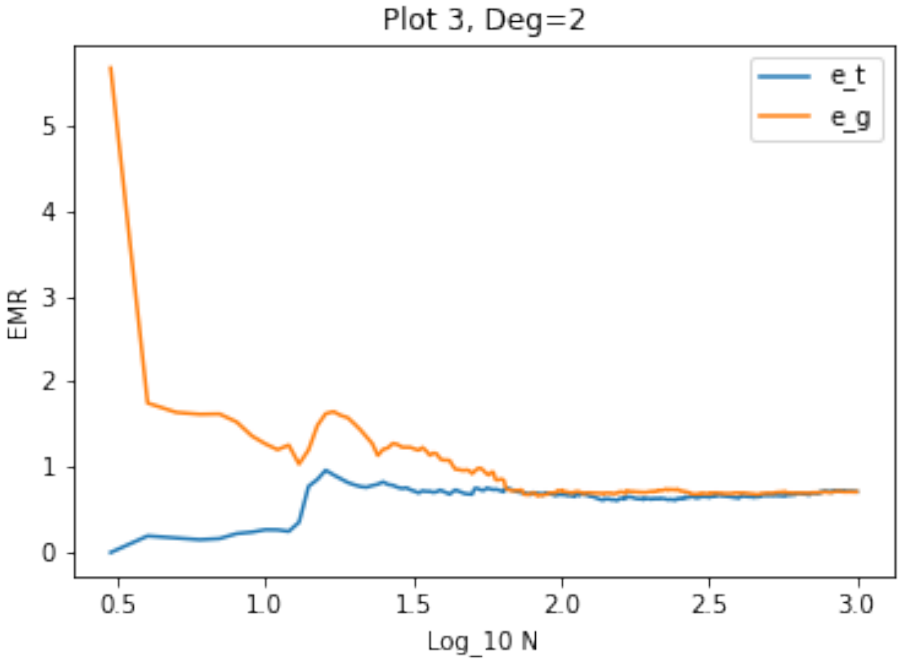
This is **not** consistent with the conclusions on approximation error earlier that approximation error for $d > 2$ is 0. The approximation error increases as d increases.

This is because of the mathematical calculations, which leads to errors, as the decimal precision (number of digits after 0) is limited

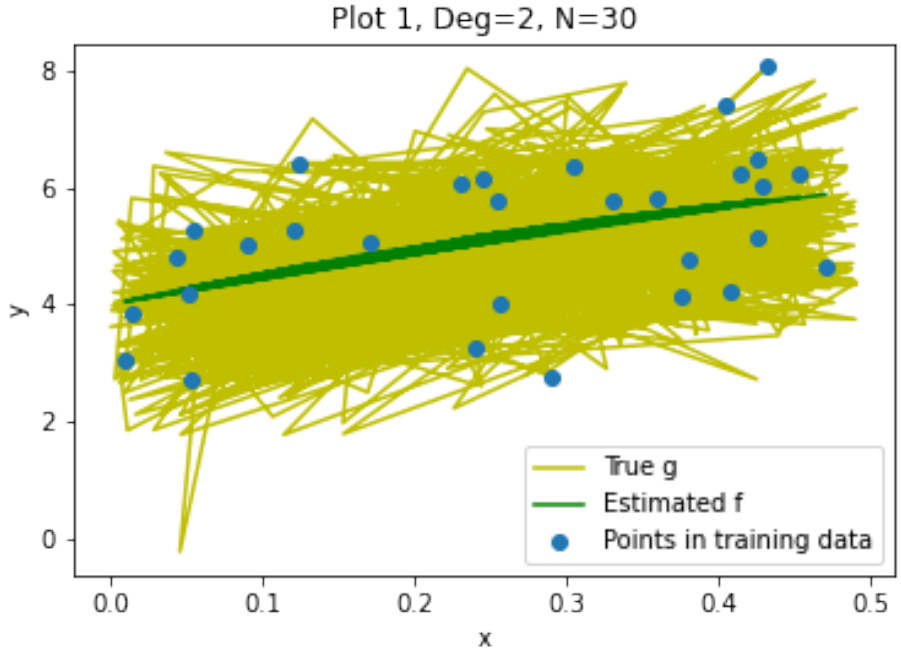
11. **Plot e_t and e_g as a function of N for $d < N < 1000$ for $d = 2$, $d = 5$ and $d = 10$ (Plot 2). You may want to use a logarithmic scale in the plot. Include also plots similar to Plot 1 for 2 or 3 different values of N for each value of d**

Degree=2

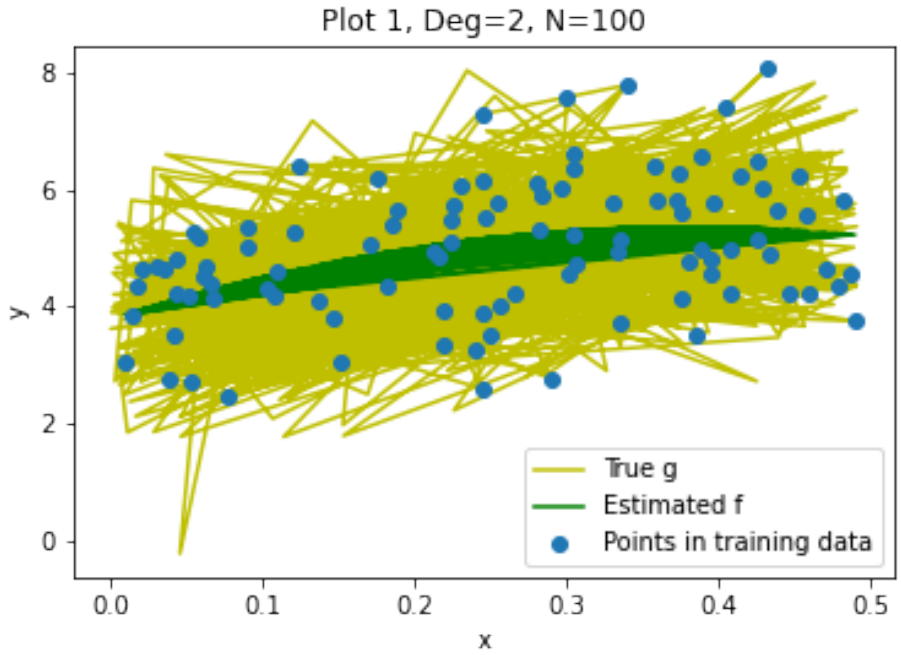
1. Plot of e_t and e_g



2. N=30

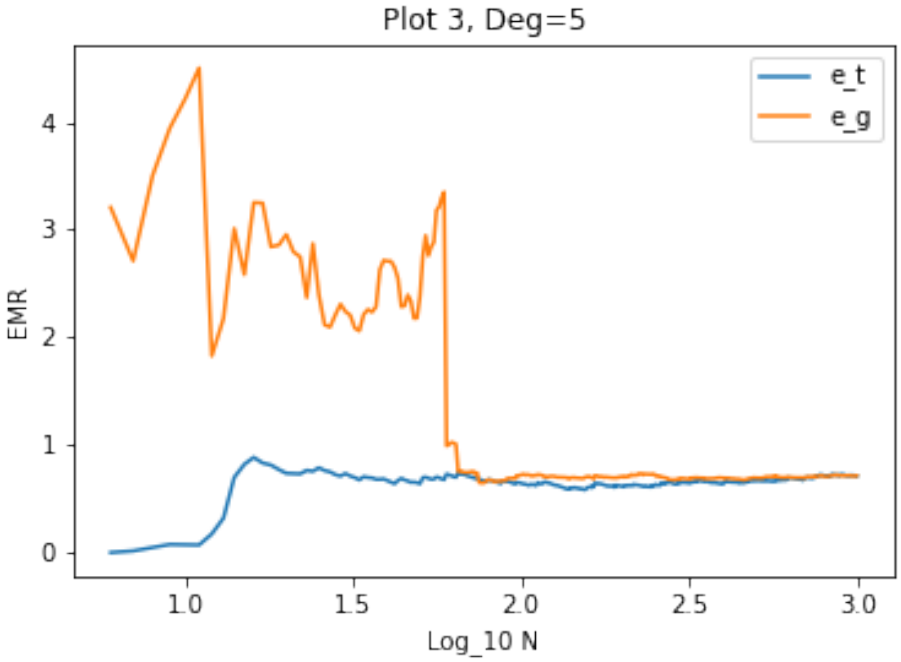

Plot 1, Deg=2, N=30
8
6
4
2
0
y
0.0
0.1
0.2
0.3
0.4
0.5
x
True g
Estimated f
Points in training data

3. N=100

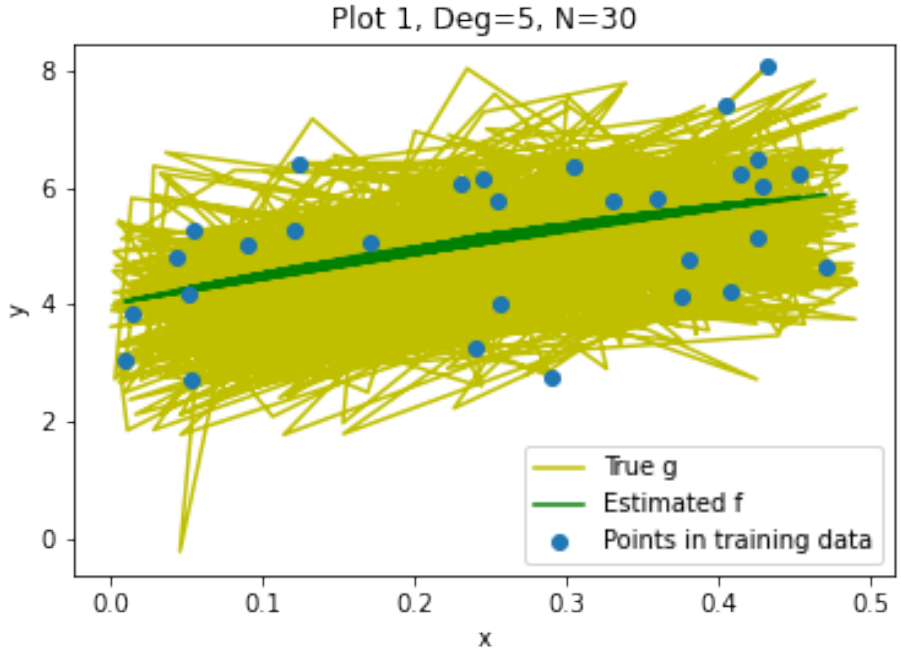

Plot 1, Deg=2, N=100
8
6
4
2
0
y
0.0
0.1
0.2
0.3
0.4
0.5
x
True g
Estimated f
Points in training data

Degree=5

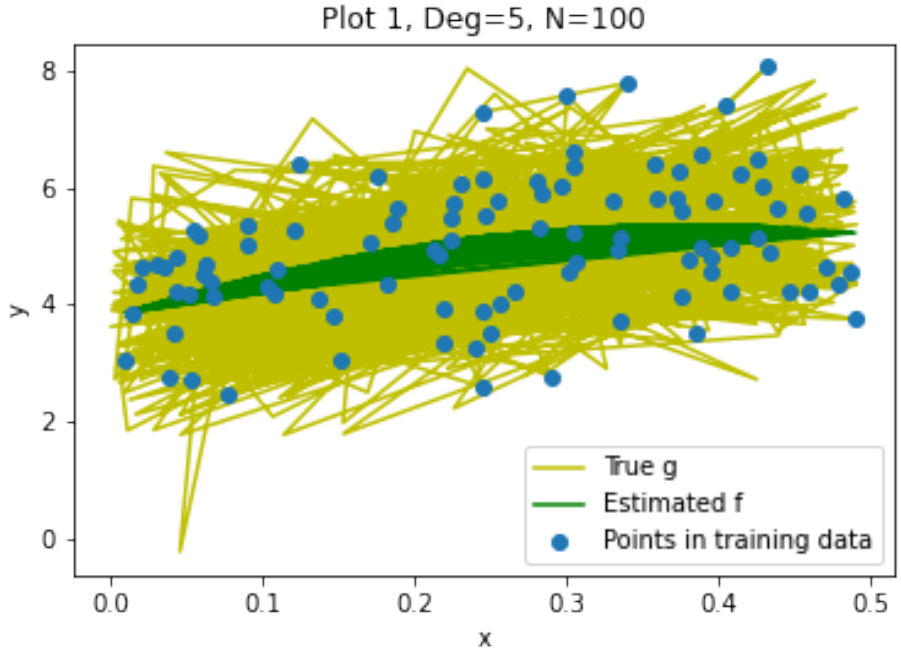
1. Plot of e_t and e_g



2. N=30

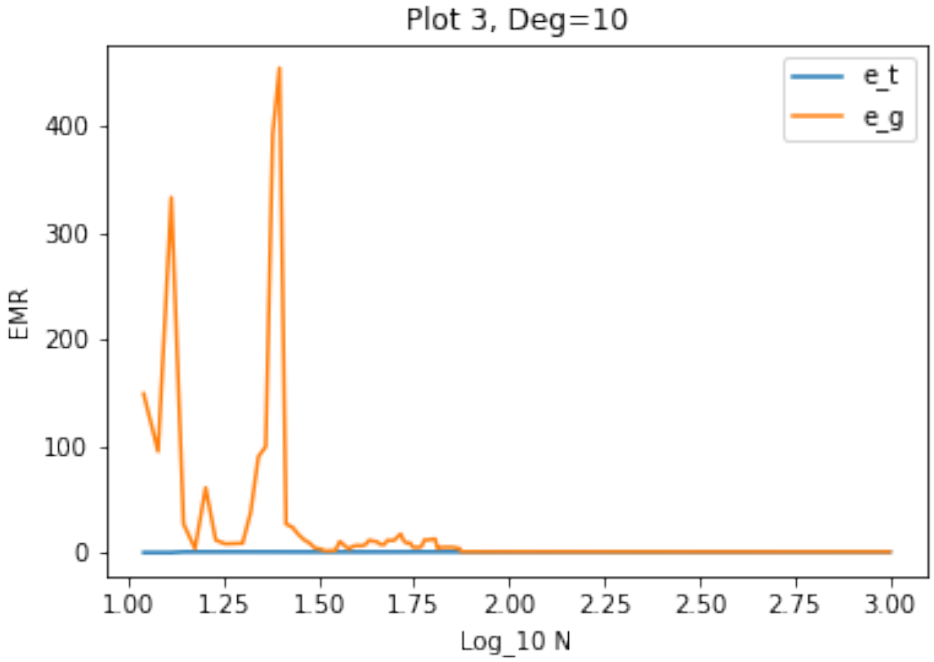

Plot 1, Deg=5, N=30
8
6
4
2
0
y
0.0
0.1
0.2
0.3
0.4
0.5
x
True g
Estimated f
Points in training data

3. N=100

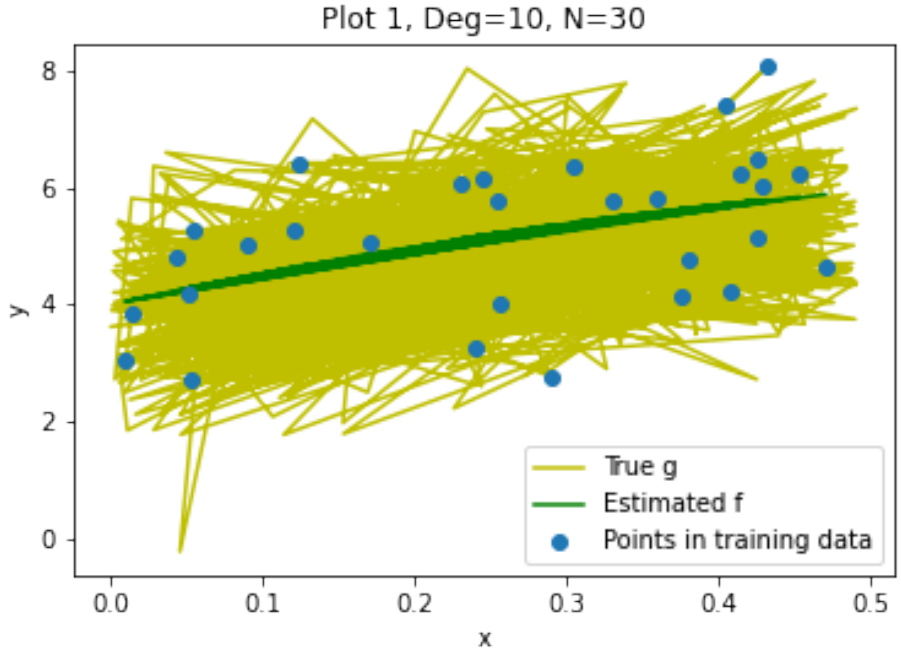

Plot 1, Deg=5, N=100
8
6
4
2
0
y
0.0
0.1
0.2
0.3
0.4
0.5
x
True g
Estimated f
Points in training data

Degree=10

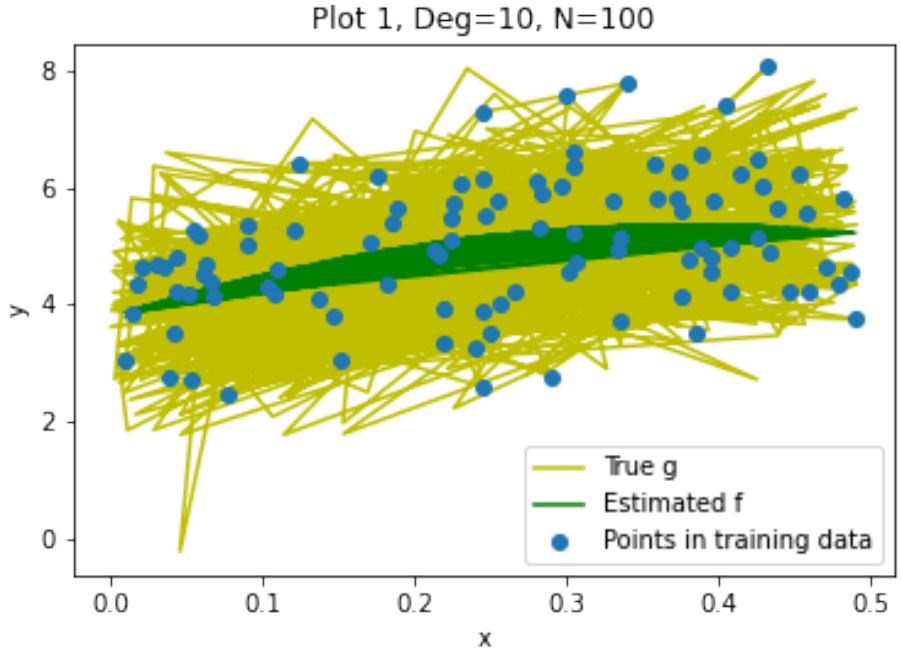
1. Plot of e_t and e_g



2. N=30


Plot 1, Deg=10, N=30
y
x
0
2
4
6
8
0.0
0.1
0.2
0.3
0.4
0.5
True g
Estimated f
Points in training data

3. N=100


Plot 1, Deg=10, N=100
y
x
0
2
4
6
8
0.0
0.1
0.2
0.3
0.4
0.5
True g
Estimated f
Points in training data

12. **Recall the definition of the estimation error. Using the test set, (which we intentionally chose large so as to take advantage of the law of large numbers) give an empirical estimator of the estimation error. For the same values of N and d above plot the estimation error as a function of N (Plot 3).**

Estimation error is given by formula:

$$\text{Estimation error} = R(\hat{f}_n) - R(f^*_{\mathcal{H}})$$

As the value of degree is either of 2, 5, or 10, the degree is always greater than 2. So, the hypothesis space always includes the true function, and so the term $R(f^*_{\mathcal{H}})$ is 0

The first term $R(\hat{f}_n)$ can be calculated as:

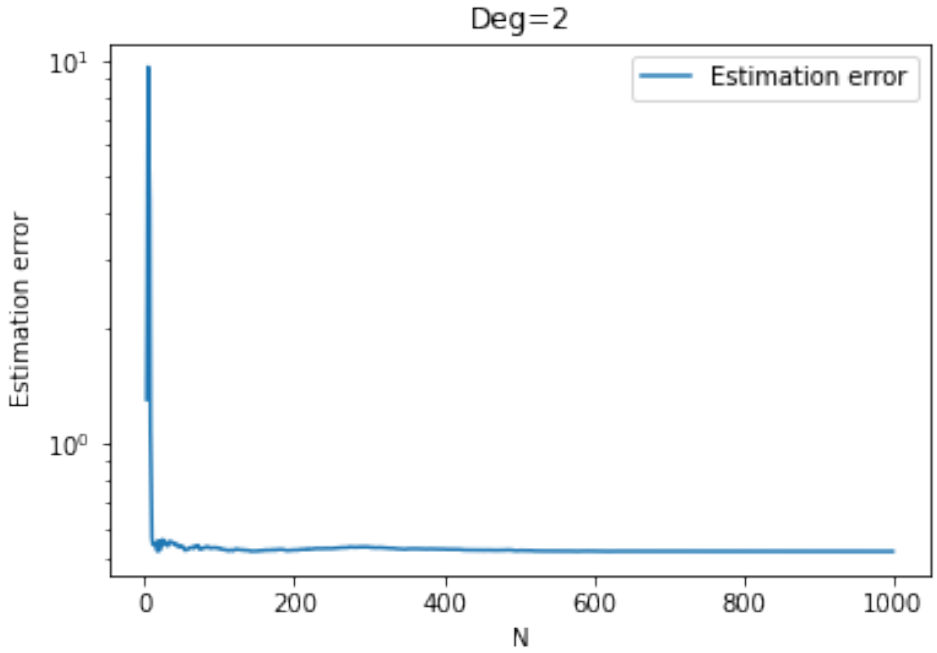
$$R(\hat{f}_n) = \frac{1}{1000}(\frac{1}{2}(X_{test}\hat{b} - y)^2)$$

Here $\hat{b}$ is calculated by training data, using the formula:

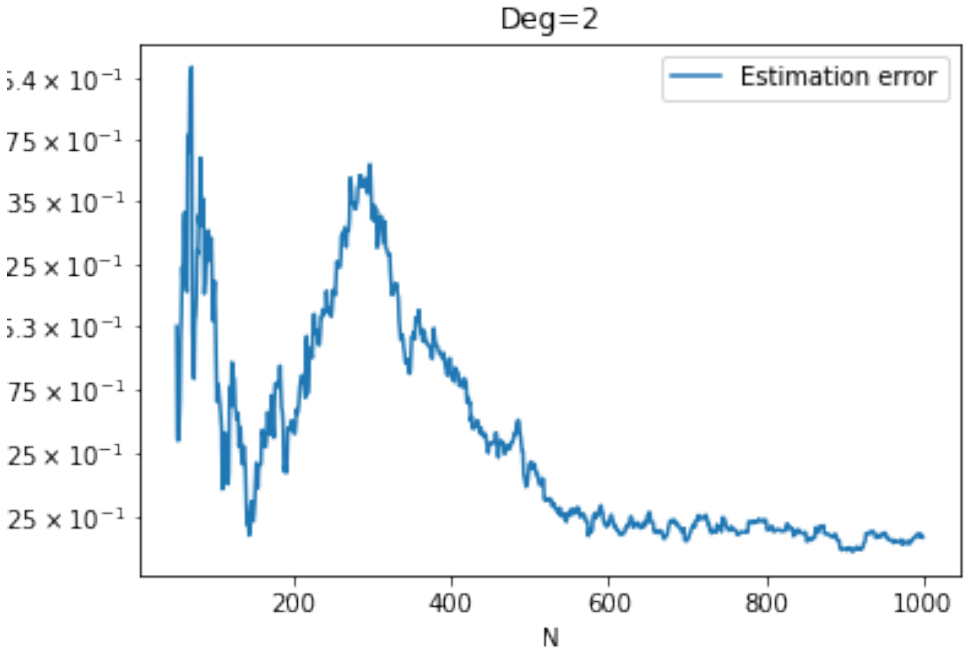
$$\hat{b} = (X^\top_{train}X_{train})^{-1}X^\top_{train}\boldsymbol{y}$$

And the degree d is one of 2, 5, 10. Just as in previous question, this applies to both training and test set design matrices. But the size N only changes for the training set design matrix. For test set, N is fixed to 1000.

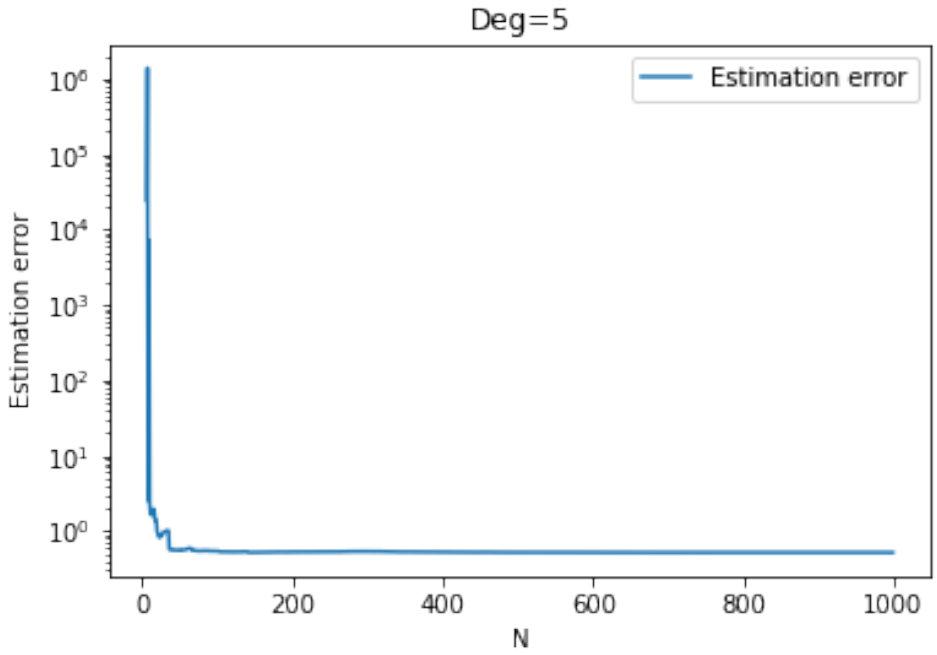
Degree=2

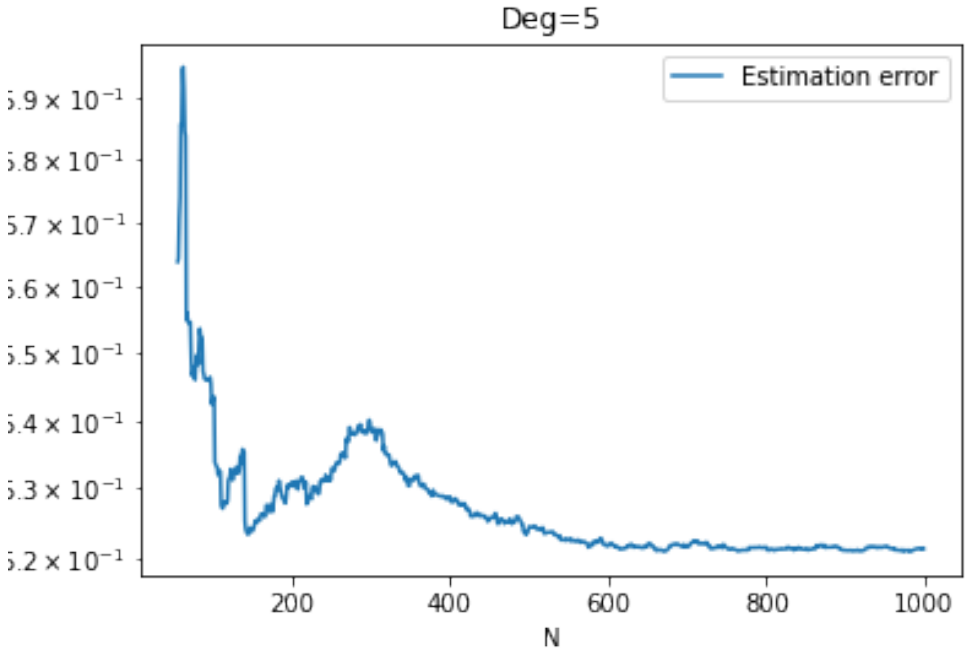


The first plot is a bit difficult to interpret, so we make a similar plot, but start N from 50 instead of $d + 1$. (This will be done for degree 5 and 10 as well)

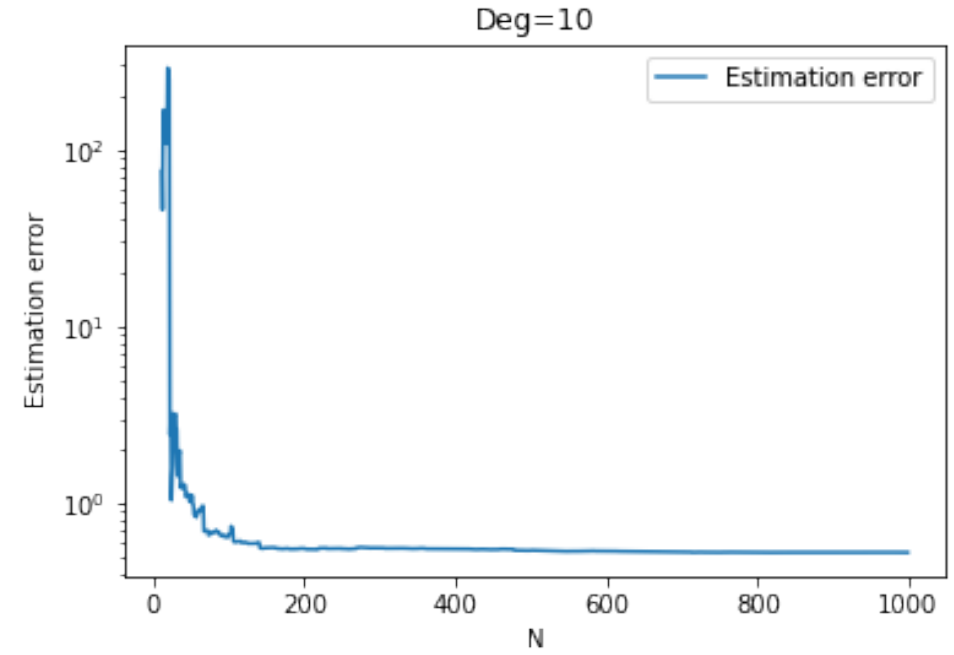


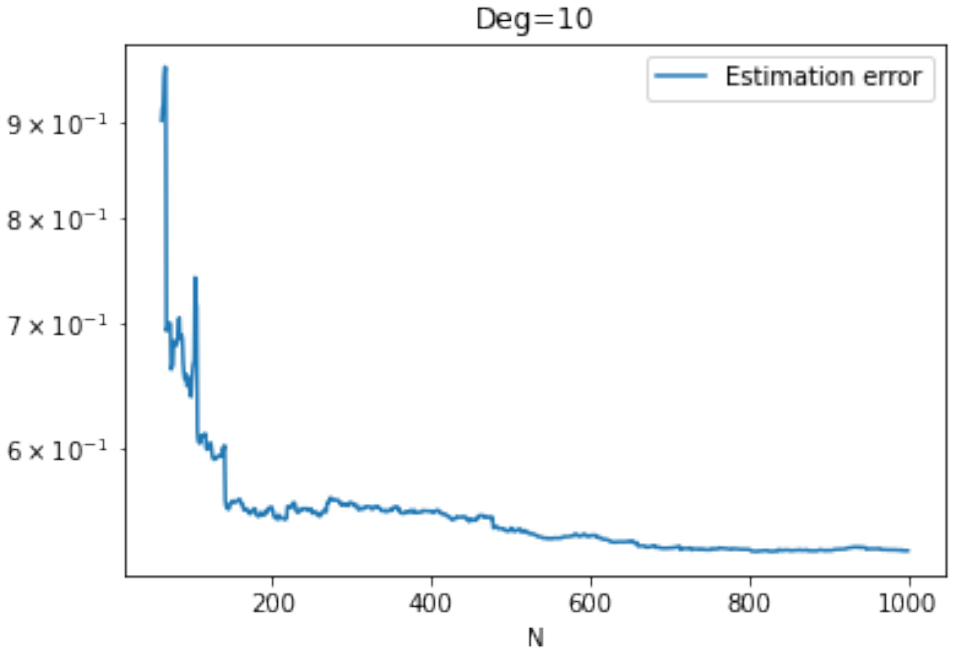
Degree=5


Deg=5
Estimation error
10^6
10^5
10^4
10^3
10^2
10^1
10^0
Estimation error
0
200
400
600
800
1000
N


Deg=5
Estimation error
5.9×10^{-1}
5.8×10^{-1}
5.7×10^{-1}
5.6×10^{-1}
5.5×10^{-1}
5.4×10^{-1}
5.3×10^{-1}
5.2×10^{-1}
200
400
600
800
1000
N

Degree=10


Deg=10
Estimation error
10^2
10^1
10^0
Estimation error
0
200
400
600
800
1000
N


Deg=10
Estimation error
9×10^{-1}
8×10^{-1}
7×10^{-1}
6×10^{-1}
200
400
600
800
1000
N

13. **The generalization error gives in practice an information related to the estimation error. Comment on the results of (Plot 2 and 3). What is the effect of increasing N? What is the effect of increasing d?**

For a fixed d, when N increases, the training error increases, and test error decreases. Overall, the estimation error reduces and approaches a small constant value.

For a fixed N, when d increases, the estimation error increases.

14. **Besides from the approximation and estimation there is a last source of error we have not discussed here. Can you comment on the optimization error of the algorithm we are implementing?**

The optimization error is ideally 0, as we are calculating the optimizer using the formula:

$$\hat{b} = (X^\top X)^{-1} X^\top \boldsymbol{y}$$

But because of mathematical calculations involved, and finite precision when performing algebraic operations, we observe some optimization error in Q7-Q12 (the graphs and comparing $\hat{b}$ with a)