Homework 1: Error Decomposition & Polynomial Regression

Due: Wednesday, February 2, 2022 at 11:59pm

Instructions: Your answers to the questions below, including plots and mathematical work, should be submitted as a single PDF file. It's preferred that you write your answers using software that typesets mathematics (e.g.LaTeX, LyX, or MathJax via iPython), though if you need to you may scan handwritten work. You may find the minted package convenient for including source code in your LaTeX document. If you are using LyX, then the listings package tends to work better. The last application is optional.

General considerations (10 Points)

For the first part of this assignment we will consider a synthetic prediction problem to develop our intuition about the error decomposition. Consider the random variables $x \in \mathcal{X} = [0,1]$ distributed uniformely ($x \sim \text{Unif}([0,1])$) and $y \in \mathcal{Y} = \mathbb{R}$ defined as a polynomial of degree 2 of x: there exists $(a_0, a_1, a_2) \in \mathbb{R}^3$ such that the values of x and y are linked as $y = g(x) = a_0 + a_1 x + a_2 x^2$. Note that this relation fixes the joint distribution $P_{\mathcal{X}\times\mathcal{Y}}$.

From the knowledge of a sample $\{x_i, y_i\}_{i=1}^N$, we would like to predict the relation between x and y, that is find a function f to make predictions $\hat{y} = f(x)$. We note $\mathcal{H}_d$, the set of polynomial functions on $\mathbb{R}$ of at most degree d: $\mathcal{H}_d = \left\{f : x \to b_0 + b_1 x + \cdots b_d x^d; b_k \in \mathbb{R} \forall k \in \{0, \cdots d\}\right\}$. We will consider the hypothesis classes $\mathcal{H}_d$ varying d. We will minimize the squared loss $\ell(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$ to solve the regression problem.

1. (2 Points) Recall the definition of the expected risk $R(f)$ of a predictor f. While this cannot be computed in general note that here we defined $P_{\mathcal{X}\times\mathcal{Y}}$. Which function f^* is an obvious Bayes predictor? Make sure to explain why the risk $R(f^*)$ is minimum at f^*.

   The risk $R(f)$ is the expected loss of f. Since the data is generated by the function $g(x) = a_0 + a_1 x + a_2 x^2$, the function f^* that minimizes the risk is simply $g(x)$. When the function that generates the data and the function that evaluates the data are the same, the expected loss—or the risk—would be zero; so, the risk $R(f^*)$ is minimum at $f^* = g(x) = a_0 + a_1 x + a_2 x^2$.

2. (2 Points) 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 $f^*_{\mathcal{H}_2}$?

   With $\mathcal{H}_2$ as the hypothesis class, the function $g(x) = a_0 + a_1 x + a_2 x^2$ that generates the data is within the set of polynomial functions. Therefore, the function $f^*_{\mathcal{H}_2} = a_0 + a_1 x + a_2 x^2$ is a minimizer in $\mathcal{H}_2$. The approximation achieved by $\mathcal{H}_2$ is zero.

3. (2 Points) 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 $f^*_{\mathcal{H}_d}$?

   Since $d > 2$ and $\mathcal{H}_d$ is the set of polynomial function of $\mathbb{R}$ of at most degree d, the function $g(x) = a_0 + a_1 x + a_2 x^2$ will be within the $\mathcal{H}_d$ no matter what d is. Therefore, the function $f^*_{\mathcal{H}_d} = a_0 + a_1 x + a_2 x^2$ is a risk minimizer in $\mathcal{H}_d$ and the approximation error achieved by $f^*_{\mathcal{H}_d}$ is also zero. Basically, we have $R(f^*_{\mathcal{H}_2}) = R(f^*_{\mathcal{H}_d})$.

4. (4 Points) 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?

   With $a_0 = 0$, our function that generates the data becomes $g(x) = a_1 x + a_2 x^2$ and the prediction function becomes $f(x) = b_1 x$. To find the function $f^*_{\mathcal{H}}$ that minimizes the risk $R(f)$ we first compute the expected loss:

$$\begin{aligned} R(f) = \mathbb{E}(\ell(\hat{y}, y)) &= \int_0^1 \frac{1}{2}(\hat{y} - y)^2 \\ &= \frac{1}{2}\int_0^1 (b_1 x - a_1 x - a_2 x^2)^2 \mathrm{d}x \\ &= \frac{1}{2}\int_0^1 (b_1 - a_1)^2 x^2 - 2(b_1 - a_1)x a_2 x^2 + {a_2}^2 x^4 \mathrm{d}x \\ &= \frac{1}{2}\Big((b_1 - a_1)^2 \frac{x^3}{3} - a_2(b_1 - a_1)\frac{x^4}{2} + {a_2}^2 \frac{x^5}{5}\Big)\Big|_0^1 \\ &= \frac{1}{2}\Big(\frac{{b_1}^2}{3} - \frac{2b_1 a_1}{3} + \frac{{a_1}^2}{3} - \frac{a_2 b_1}{2} + \frac{a_2 a_1}{2} + \frac{{a_2}^2}{5}\Big) \end{aligned}$$

   Then, we take the derivative with respect to b_1 to solve:

$$\begin{aligned} \frac{\mathrm{d}\,\mathbb{E}(\ell(\hat{y}, y))}{\mathrm{d}\ b_1} &= \frac{1}{2}\Big(\frac{2b_1}{3} - \frac{2a_1}{3} - \frac{a_2}{2}\Big) = 0 \\ &\Rightarrow \frac{b_1}{3} - \frac{a_1}{3} - \frac{a_2}{4} = 0 \Rightarrow b_1 = a_1 + \frac{3}{4}a_2 \end{aligned}$$

   Therefore, $f^*_{\mathcal{H}} = b_1 x = (a_1 + \frac{3}{4}a_2)x$ is a risk minimizer in $\mathcal{H}$. Next, we because we know that $R(f^*) = 0$, the approximation error $R(f^*_{\mathcal{H}}) - R(f^*)$ is simply $R(f^*_{\mathcal{H}})$:

$$\begin{aligned} R(f^*_{\mathcal{H}}) = \mathbb{E}\left(\ell(\hat{y}, y)\right) &= \mathbb{E}\left(b_1^* x - a_1 x - a_2 x^2\right)^2 \\ &= \mathbb{E}\left((a_1 + \frac{3}{4}a_2)x - a_1 x - a_2 x^2\right)^2 \\ &= \mathbb{E}\left(\frac{3a_2 x}{4} - a_2 x^2\right)^2 \\ &= \mathbb{E}\left(\frac{9{a_2}^2}{16}x^2 - \frac{3{a_2}^2}{2}x^3 + {a_2}^2 x^4\right) \\ &= \frac{1}{2}\int_0^1 \frac{9{a_2}^2}{16}x^2 - \frac{3{a_2}^2}{2}x^3 + {a_2}^2 x^4 \mathrm{d}x \\ &= \frac{1}{2}\Big(\frac{9{a_2}^2}{16}\frac{x^3}{3} - \frac{3{a_2}^2}{2}\frac{x^4}{4} + {a_2}^2\frac{x^5}{5}\Big)\Big|_0^1 \\ &= \frac{1}{2}\Big(\frac{3{a_2}^2}{16} - \frac{3{a_2}^2}{8} + \frac{{a_2}^2}{5}\Big) \\ &= \frac{{a_2}^2}{160} \end{aligned}$$

   Lastly, by setting $a_2 = 0$, our data-generating function becomes $g(x) = a_1 x$. Since our risk minimizer $f^*_{\mathcal{H}} = b_1 x$, at $a_1 = b_1$ the approximation error achieved would be zero.

Polynomial regression as linear least squares (5 Points)

In practice, $P_{\mathcal{X}\times\mathcal{Y}}$ is usually unknown and we use the empirical risk minimizer (ERM). We will reformulate the problem as a d-dimensional linear regression problem. First note that functions in $\mathcal{H}_d$ are parametrized by a vector $\boldsymbol{b} = [b_0, b_1, \cdots b_d]^\top$, we will use the notation $f_{\boldsymbol{b}}$. Similarly we will note $\boldsymbol{a} \in \mathbb{R}^3$ the vector parametrizing $g(x) = f_{\boldsymbol{a}}(x)$. We will also gather data points from the training sample in the following matrix and vector:

$$X = \begin{bmatrix} 1 & x_1 & \cdots & x_1^d \\ 1 & x_2 & \cdots & x_2^d \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_N & \cdots & x_N \end{bmatrix}, \quad \boldsymbol{y} = [y_0, y_1, \cdots y_N]^\top. \tag{1}$$

These notations allow us to take advantage of the very effective linear algebra formalism. X is called the design matrix.

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

   By definition, the empirical risk minimizer $\hat{\boldsymbol{b}}$ is the following:

$$\hat{\boldsymbol{b}} = \arg\min \frac{1}{n}\sum_{i=1}^{n} \ell(f(x_i), y_i)$$

   In the general considerations section, we're given that:

$$\hat{y} = f(x) = b_0 + b_1 x + \cdots b_d x^d$$

   and the squared loss function:

$$\ell(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$$
$$\Rightarrow \ell(f(x), y) = \frac{1}{2}(f(x) - y)^2$$

   Plugging this into the ERM equation, we have:

$$\begin{aligned} \hat{\boldsymbol{b}} &= \arg\min \frac{1}{2N}\sum_{i=1}^{N}(f(x_i) - y_i)^2 \\ &= \arg\min \frac{1}{2N}\sum_{i=1}^{N}\left(\sum_{j=0}^{d} b_j X_{i,j} - y_i\right)^2 \\ &= \arg\min \frac{1}{2N}\left\|\sum_{i=1}^{N}\sum_{j=0}^{d} b_j X_{i,j} - y_i\right\|_2^2 \\ &= \arg\min \frac{1}{2N}\|Xb - \boldsymbol{y}\|_2^2 \end{aligned}$$

   Lastly, because the $\frac{1}{2N}$ constant wouldn't affect the output of the $\arg\min$, we have shown that the empirical risk minimizer (ERM) $\hat{\boldsymbol{b}}$ is:

$$\hat{\boldsymbol{b}} = \arg\min \frac{1}{N}\sum_{i=1}^{N} \ell(f(x_i), y_i) = \arg\min_b \|Xb - \boldsymbol{y}\|_2^2$$

6. (3 Points) If $N > d$ and X is full rank, show that $\hat{\boldsymbol{b}} = (X^\top X)^{-1}X^\top \boldsymbol{y}$. (Hint: you should take the gradients of the loss above with respect to $\boldsymbol{b}$ [1]). Why do we need to use the conditions $N > d$ and X full rank ?

   From the previous question, we showed that our loss function can be written in the form $\|Xb - \boldsymbol{y}\|_2^2$. If we expand the terms:

$$\begin{aligned}
\|Xb - \boldsymbol{y}\|_2^2 &= \|Xb\|^2 + \|\boldsymbol{y}\|^2 - 2\langle Xb, \boldsymbol{y}\rangle \\
&= (Xb)^\top Xb + \boldsymbol{y}^\top \boldsymbol{y} - 2(b^\top X^\top \boldsymbol{y}) \\
&= b^\top X^\top Xb + \boldsymbol{y}^\top \boldsymbol{y} - 2\big((\boldsymbol{y}^\top Xb)\big) \\
&= b^\top (X^\top X)b + \boldsymbol{y}^\top \boldsymbol{y} - 2\big((X^\top \boldsymbol{y})^\top b\big)
\end{aligned}$$

   and then take the gradient:

$$\begin{aligned}
\nabla\|Xb - \boldsymbol{y}\|_2^2 &= \big(X^\top X + (X^\top X)^\top\big)b + 0 - 2(X^\top \boldsymbol{y}) \\
&= 2(X^\top X)b - 2(X^\top \boldsymbol{y}) \\
&= 2(X^\top Xb - X^\top \boldsymbol{y})
\end{aligned}$$

   Next, we set the gradient to zero and solve for b:

$$\begin{aligned}
2(X^\top Xb - X^\top \boldsymbol{y}) &= 0 \\
X^\top Xb - X^\top \boldsymbol{y} &= 0 \\
X^\top Xb &= X^\top \boldsymbol{y} \\
b &= (X^\top X)^{-1}X^\top \boldsymbol{y}
\end{aligned}$$

   Since this is the global minimum of the function, it shows that b is the minimum of $\hat{\boldsymbol{b}}$; hence, we have shown that $\hat{\boldsymbol{b}} = (X^\top X)^{-1}X^\top \boldsymbol{y}$. We need to use the conditions $N > d$ and X full rank because that ensures that $X^\top X$ is also full rank and is therefore invertible.

Hands on (7 Points)

Open the source code file *hw1_code_source.py* from the *.zip* folder. Using the function `get_a` get a value for $\boldsymbol{a}$, and draw a sample `x_train, y_train` of size $N = 10$ and a sample `x_test, y_test` of size $N_{\text{test}} = 1000$ using the function `draw_sample`.

7. (2 Points) 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 $\boldsymbol{y} \in \mathbb{R}^N$ returning $\hat{\boldsymbol{b}} \in \mathbb{R}^{(d+1)}$. 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):
    N = X.size
    d = X.ndim - 1
    if N <= d:
        print("Error!")
    else:
        b_hat = np.linalg.inv(X.T @ X) @ X.T @ y
    return b_hat
```

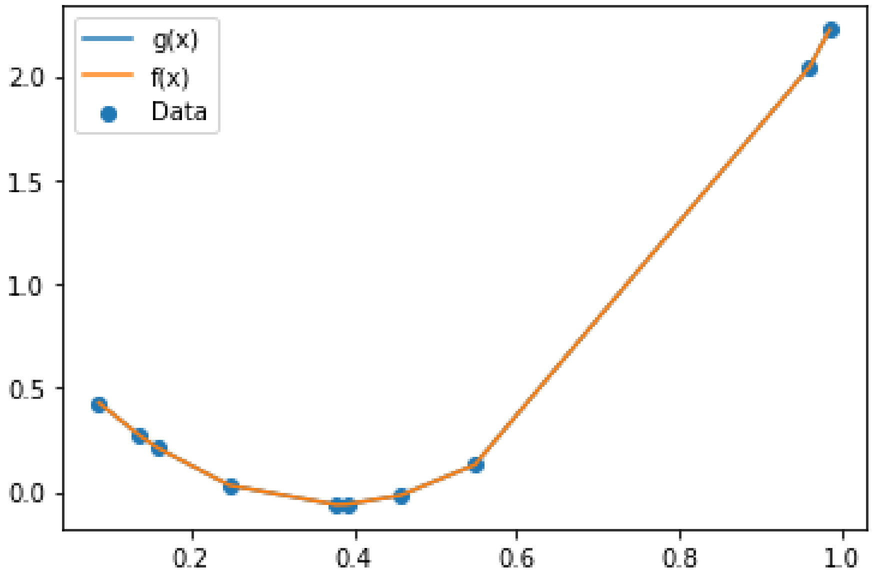
[1] You can check the linear algebra review here if needed `http://cs229.stanford.edu/section/cs229-linalg.pdf`

8. (1 Points) 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 $f_{\boldsymbol{b}}$ taking as input a design matrix $X \in \mathbb{R}^{N\times(d+1)}$, a vector $\boldsymbol{y} \in \mathbb{R}^N$ and the vector $\boldsymbol{b} \in \mathbb{R}^{(d+1)}$ parametrizing the predictor.

```
def empirical_risk(X, y, b):
    N = X.size
    matrix = X @ b - y

    risk = matrix.T @ matrix / (2 * N)

    return risk
```

9. (3 Points) Use your code to estimate $\hat{\boldsymbol{b}}$ from `x_train, y_train` using $d = 5$. Compare $\hat{\boldsymbol{b}}$ and $\boldsymbol{a}$. 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 $f_{\hat{\boldsymbol{b}}}(x)$ in $[0, 1]$. Make sure to include a legend to your plot .

```
print(b_hat)
[ 7.71568786e-01 -4.48900685e+00  6.04984835e+00
-1.42517820e-09 1.78420123e-09 -3.70320663e-10]

print(a)
[ 0.77156879 -4.48900684  6.04984835]
```



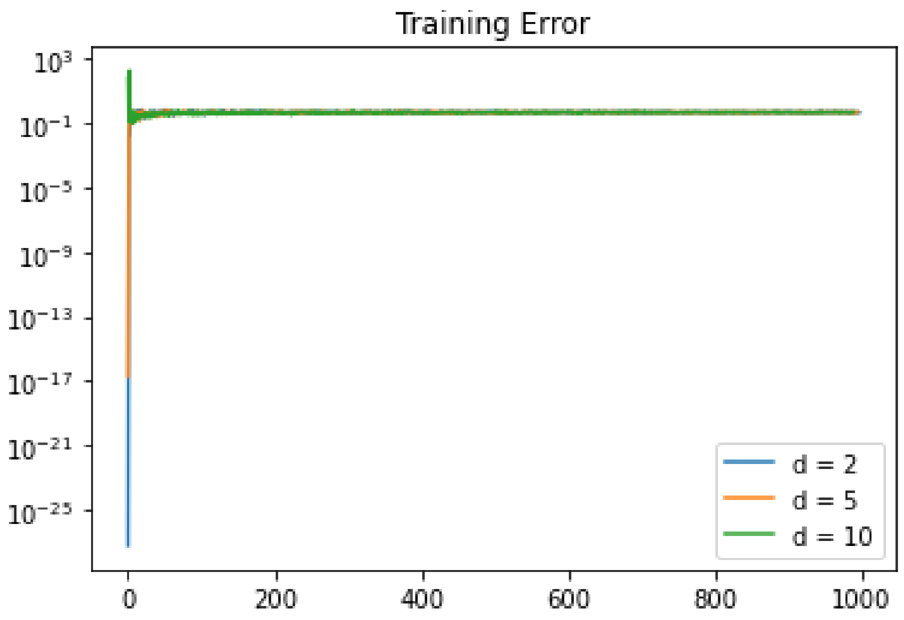
10. (1 Points) Now you can adjust d. What is the minimum value for which we get a “perfect fit”? How does this result relates with your conclusions on the approximation error above?

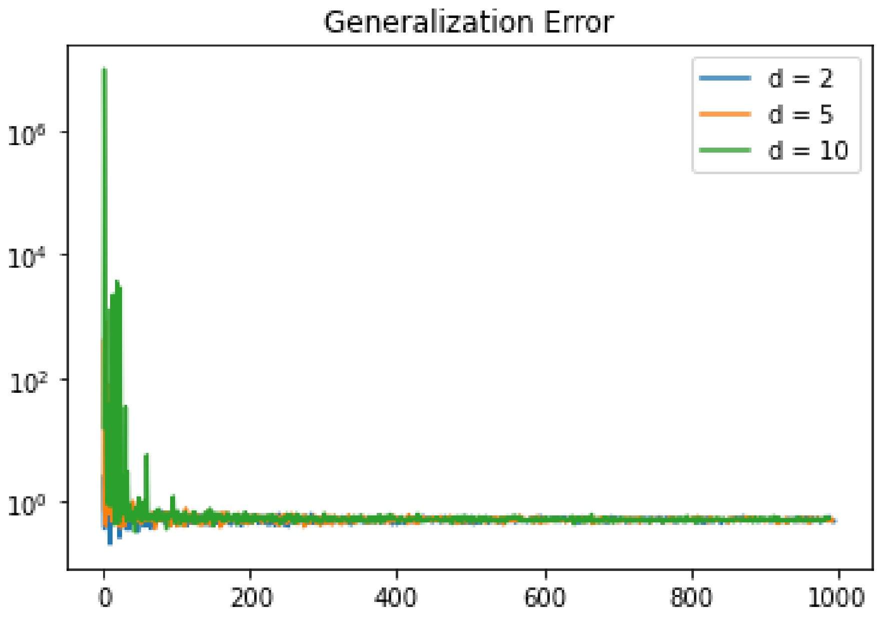
The lowest risk is achieved at $d = 2$. This makes sense because it is the same degree as when we generated the data.

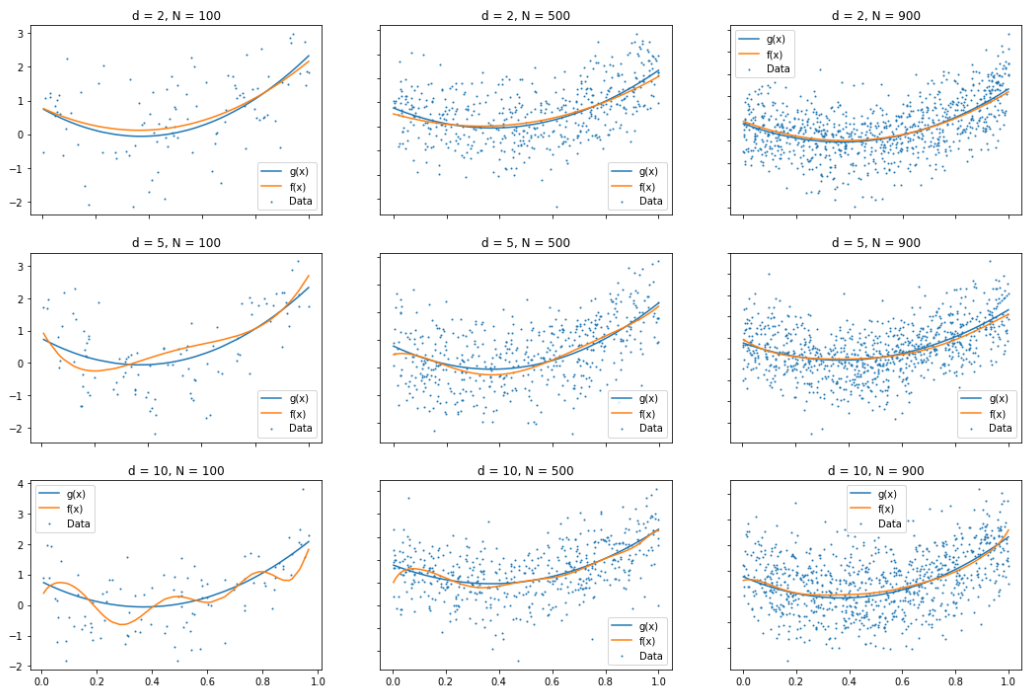
In presence of noise (13 Points)

Now we will modify the true underlying $P_{\mathcal{X}\times\mathcal{Y}}$, adding some noise in $y = g(x)+\epsilon$, with $\epsilon \sim \mathcal{N}(0,1)$ a standard normal random variable independent from x. We will call training error e_t the empirical risk on the train set and generalization error e_g the empirical risk on the test set.

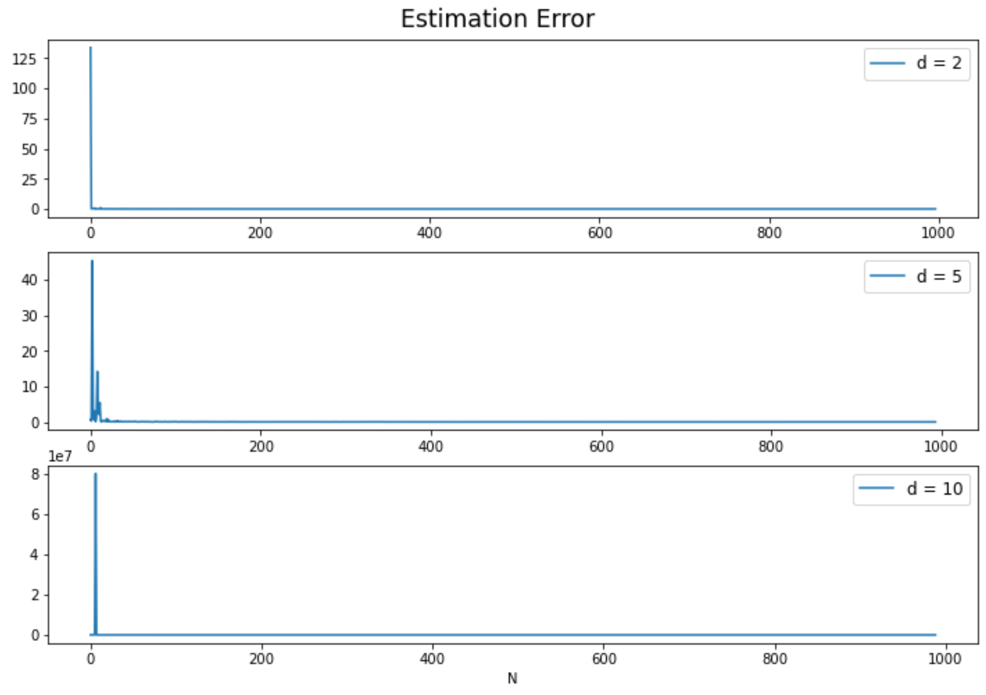
11. (6 Points) Plot e_t and e_g as a function of N for $d < N < 1000$ for $d = 2$, $d = 5$ and $d = 10$ (Plot 2). You may want to use a logarithmic scale in the plot. Include also plots similar to Plot 1 for 2 or 3 different values of N for each value of d.







12. (4 Points) 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).



13. (2 Points) 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?

    Based on the results of Plot 2 and 3, we can see that as N increases, the error converges to around zero for the training error, the generalization error, and the estimation error. However, when d increases, different effects (or lack of) are observed: the training error converges much faster as d increases; the generalization error converges much slower as d increases; and the convergence of the estimation error is unaffected when d changes (i.e. there is no clear relationship). This makes sense because the estimation error is calculated as a subtraction of two empirical risks that are generated from design matrices in the same dimension; therefore, the two components of the subtraction would remain relatively similar even when the degree is altered.

14. (1 Points) 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 exists when one attempts to approximate the "perfect fit" function because arriving at the actual "perfect fit" function is not achievable due to computational cost or time constraint. However, in the algorithm we are implementing, the "perfect fit" function is known; therefore, we are able to reduce to optimization error to zero.

Find codes for questions 7 $\sim$ 12 at: `https://github.com/anchengyoung/MS_Data_Science/blob/main/machine_learning/hw1/homework_1.py`

Application to Ozone data (optional) (2 Points)

You can now use the code we developed on the synthetic example on a real world dataset. Using the command `np.loadtxt('ozone_wind.data')` load the data in the *.zip*. The first column corresponds to ozone measurements and the second to wind measurements. You can try polynomial fits of the ozone values as a function of the wind values.

15. (2 Points) Reporting plots, discuss the again in this context the results when varying N (subsampling the training data) and d.