DS-GA 1003 HW1

Buz Galbraith

TOTAL POINTS

33.5 / 37

QUESTION 1

1 Q1 2 / 2

✓ **- 0 pts** *Correct.*

Bayes predictor is right.

- 0.5 pts Unclear or wrong justification.

- 0.5 pts Bayes predictor is not clearly stated.

- 1 pts No justification.

- 1.5 pts Bayes predictor is wrong.

- 2 pts No answer shown.

QUESTION 2

2 Q2 2 / 2

✓ **- 0 pts** *Correct.*

- 0.5 pts Approximation error is wrong, wrong justification, or no justification.

- 0.5 pts Risk minimizer is not clear stated.

- 1 pts Risk minimizer is wrong.

- 2 pts No answer shown.

QUESTION 3

3 Q3 1 / 2

- 0 pts Correct

✓ **- 1 pts** *Wrong/missing inequality: should be R(f_H2) >= R(f_Hd)*

- 0.5 pts Correct inequality but missing/wrong justification: need to reason the inclusion relation between H_2 and H_d

- 0.5 pts Wrong/missing risk minimizer: should be just g(x)

- 0.5 pts Wrong/missing approximation error: should be 0

- 0.5 pts Click here to replace this description.

R(f_H_2) >= R(f_H_d) because H_2 is a subset of H_d. Also please do not write "no approx error", should be approximation error = 0

QUESTION 4

4 Q4 3 / 4

- 0 pts Correct

- 2 pts Wrong/missing risk minimizer, should be f*_H = (a_1 + 0.75*a_2)x, i.e., b_1 = a_1 + 0.75*a_2

✓ **- 1 pts** *Wrong/missing approximation error, should be a_2^2/160 (or a_2^2 / 80 if ignoring the 1/2 in risk)*

- 1 pts Wrong/missing approximation error given a_2 = 0, it's just 0

- 0.5 pts Click here to replace this description.

- 1 pts Missing steps

- 1 pts Wrong derivative

- 2 pts No steps, please include all your work

QUESTION 5

5 Q5 2 / 2

✓ **- 0 pts** *Correct.*

- 0.5 pts No or wrong ERM definition.

- **0.5 pts** No or wrong vector norm justification.
- **1 pts** Incomplete or partially wrong.
- **1.5 pts** Incorrect.
- **2 pts** No answer shown.

QUESTION 6

6 Q6 **3 / 3**

✓ **- 0 pts** *Correct*

- **2 pts** Wrong/missing computations
- **1 pts** Wrong/missing reason for conditions
- **3 pts** empty

QUESTION 7

7 Q7 **2 / 2**

✓ **- 0 pts** *Correct*

- **1 pts** Missing exception when N <= d
- **0.5 pts** Wrong exception condition

Should be `X.shape[0] <= X.shape[1] - 1`

- **0.5 pts** Incorrect function input

QUESTION 8

8 Q8 **1 / 1**

✓ **- 0 pts** *Correct*

- **0.5 pts** Missing 1/2
- **0.5 pts** Missing square
- **0.5 pts** Should be divided by `(2 * len(y))`

QUESTION 9

9 Q9 **3 / 3**

✓ **- 0 pts** *Correct*

- **0.5 pts** Wrong dimension for a, b
- **0.5 pts** Missing a, b comparison
- **0.5 pts** Missing training set points
- **1 pts** Function should be smooth curve (use `np.linspace(0,1,100)` as x values, not x_train)
- **2 pts** Wrong / Missing plot
- **0.5 pts** Click here to replace this description.

QUESTION 10

10 Q10 **0.5 / 1**

- **0 pts** Correct

✓ **- 0.5 pts** *Missing / wrong minimum value of d*

- **0.5 pts** Missing / wrong conclusion on approximation error
- **0.5 pts** Approximation error should be exactly equal to 0

QUESTION 11

11 Q11 **6 / 6**

✓ **- 0 pts** *Correct*

- **1.5 pts** incorrect plot for $$e_t$$
- **1.5 pts** incorrect plot for $$e_g$$
- **1.5 pts** incorrect plots to Plot 1 for 2 or 3 different values of N for each value of d.
- **1 pts** plots for $$e_t$$ and $$e_g$$ should converge to 0.5
- **2 pts** missing the plot for $$e_t$$
- **2 pts** missing the plot for $$e_g$$
- **2 pts** missing the plots similar to Plot 1 for 2 or 3 different values of N for each value of d.

QUESTION 12

12 Q12 **3 / 4**

- **0 pts** Correct
- **0.5 pts** Wrong empirical estimator formula : $$F(f_{n}^*) - F(f_{\mathcal{H}}^*) \approx e_g - \frac{1}{2}$$
- **1 pts** Did not recall the estimation error formula

✓ **- 1 pts** *No empirical estimation error formula*

- 1 pts Wrong plots

- 4 pts No submission

QUESTION 13

13 Q13 **2 / 2**

✓ **- 0 pts** *Correct*

- 1 pts Increasing d tend to increase the generalisation error because of overfitting

- 1 pts Increasing N reduces the effect of the noise

- 2 pts No submission

QUESTION 14

14 Q14 **1 / 1**

✓ **- 0 pts** *Correct*

- 1 pts Optimisation error is 0 because of analytical solution

- 1 pts No submission

QUESTION 15

15 Q15 **2 / 2**

✓ **- 0 pts** *Correct*

- 1 pts wrong/missing plot

- 1 pts no discussion

- 2 pts empty

Homework 1: Error Decomposition & Polynomial Regression

Due: Wednesday, February 1, 2023 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 degree d: $\mathcal{H}_d = \left\{f : x \to b_0 + bx + \cdots b_d x^d; b_k \in \mathbb{R} \forall k \in \{0, \cdots d\}\right\}$. 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^*.

   - risk is defined as $R(f) = E_{x,y\sim P_{X\times Y}}[\ell(f(x), y)]$
   - we know that y is a deterministic function of x, so we can set our optimal function of x as $f^*(x) = y = a_0 + a_1 x + a_2^2 x$
   - if we set $f^*(x) = y(x) = y$ it is clear that our predictor will equal y for any given value of x, and thus as the loss is defined as $\ell(f^*(x)) = \frac{1}{2}(f^*(x) - y) = \frac{1}{2}(y(x) - y(x))^2 = 0$ our loss any given x will be zero. thus setting x as such we face no risk

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 $f^*_{\mathcal{H}_2}$?

   - assuming my argument in the last question is correct, f^* is within $\mathcal{H}_2$ thus $y = y(x) = f^*(x) = f^*_{\mathcal{H}_2} = a_0 + a_1 x + a_2^2 x$
   - since $f^*(x) = f^*_{\mathcal{H}_2}$ the approximation error will be zero, this makes sense as by restraining our selves to polynomials of degree two we do not lose any accuracy.

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 $f^*_{\mathcal{H}_d}$?

   - due to the fact that $f^*(x) = f^*_{\mathcal{H}_2}$ by definition of Bayes risk, it must be the case that $R(f^*_{\mathcal{H}_2}) \leq R(f^*_{\mathcal{H}_d})$

1 Q1 2 / 2

✓ **- 0 pts** *Correct.*

Bayes predictor is right.

- 0.5 pts Unclear or wrong justification.

- 0.5 pts Bayes predictor is not clearly stated.

- 1 pts No justification.

- 1.5 pts Bayes predictor is wrong.

- 2 pts No answer shown.

Homework 1: Error Decomposition & Polynomial Regression

Due: Wednesday, February 1, 2023 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 degree d: $\mathcal{H}_d = \left\{f : x \to b_0 + bx + \cdots b_d x^d; b_k \in \mathbb{R} \forall k \in \{0, \cdots d\}\right\}$. 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^*.

   - risk is defined as $R(f) = E_{x,y\sim P_{X\times Y}}[\ell(f(x), y)]$
   - we know that y is a deterministic function of x, so we can set our optimal function of x as $f^*(x) = y = a_0 + a_1 x + a_2^2 x$
   - if we set $f^*(x) = y(x) = y$ it is clear that our predictor will equal y for any given value of x, and thus as the loss is defined as $\ell(f^*(x)) = \frac{1}{2}(f^*(x) - y) = \frac{1}{2}(y(x) - y(x))^2 = 0$ our loss any given x will be zero. thus setting x as such we face no risk

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 $f^*_{\mathcal{H}_2}$?

   - assuming my argument in the last question is correct, f^* is within $\mathcal{H}_2$ thus $y = y(x) = f^*(x) = f^*_{\mathcal{H}_2} = a_0 + a_1 x + a_2^2 x$
   - since $f^*(x) = f^*_{\mathcal{H}_2}$ the approximation error will be zero, this makes sense as by restraining our selves to polynomials of degree two we do not lose any accuracy.

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 $f^*_{\mathcal{H}_d}$?

   - due to the fact that $f^*(x) = f^*_{\mathcal{H}_2}$ by definition of Bayes risk, it must be the case that $R(f^*_{\mathcal{H}_2}) \leq R(f^*_{\mathcal{H}_d})$

2 Q2 **2 / 2**

✓ **- 0 pts** *Correct.*

- 0.5 pts Approximation error is wrong, wrong justification, or no justification.

- 0.5 pts Risk minimizer is not clear stated.

- 1 pts Risk minimizer is wrong.

- 2 pts No answer shown.

Homework 1: Error Decomposition & Polynomial Regression

Due: Wednesday, February 1, 2023 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 degree d: $\mathcal{H}_d = \left\{f : x \to b_0 + bx + \cdots b_d x^d; b_k \in \mathbb{R} \forall k \in \{0, \cdots d\}\right\}$. 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^*.

   - risk is defined as $R(f) = E_{x,y\sim P_{X\times Y}}[\ell(f(x), y)]$
   - we know that y is a deterministic function of x, so we can set our optimal function of x as $f^*(x) = y = a_0 + a_1 x + a_2^2 x$
   - if we set $f^*(x) = y(x) = y$ it is clear that our predictor will equal y for any given value of x, and thus as the loss is defined as $\ell(f^*(x)) = \frac{1}{2}(f^*(x) - y) = \frac{1}{2}(y(x) - y(x))^2 = 0$ our loss any given x will be zero. thus setting x as such we face no risk

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 $f^*_{\mathcal{H}_2}$?

   - assuming my argument in the last question is correct, f^* is within $\mathcal{H}_2$ thus $y = y(x) = f^*(x) = f^*_{\mathcal{H}_2} = a_0 + a_1 x + a_2^2 x$
   - since $f^*(x) = f^*_{\mathcal{H}_2}$ the approximation error will be zero, this makes sense as by restraining our selves to polynomials of degree two we do not lose any accuracy.

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 $f^*_{\mathcal{H}_d}$?

   - due to the fact that $f^*(x) = f^*_{\mathcal{H}_2}$ by definition of Bayes risk, it must be the case that $R(f^*_{\mathcal{H}_2}) \leq R(f^*_{\mathcal{H}_d})$

- we don't have any restrictions on the values of the constants, so by the same argument as in question one we could pick $f^*_{\mathcal{H}_d} = a_0 + a_1x + a_2x^2 + 0x^3 + 0x^4 ... + 0x^d$ that is we set $b_i = 0 \forall i \in [3, d]$
- further as $f^*_{\mathcal{H}_d} = a_0 + a_1x + a_2x^2 + 0x^3 + 0x^4 ... + 0x^d = a_0 + a_1x + a_2^2x = f^*_{\mathcal{H}_2} = f^*(x) = y(x) = y$ by the same argument as part 2 there is no approximation error

4. (4 Points) For this question we assume $a_0 = 0$. Considering $\mathcal{H} = \{f : x \to b_1x; b_1 \in \mathbb{R}\}$, 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?

   - out goal is to find a single $\alpha \in \mathbb{R}$ such that the risk is minimized.
   - so our problem can be expressed as $R(f^*_H(x)) = E[\ell(f^*_H(x), y)] = E[\frac{1}{2}(\alpha x - a_1x - a_2x^2)] = \frac{1}{2}E[(x(\alpha - a_1) - a_2x^2)^2] = \int_0^1 (x(\alpha - a_1) - a_2x^2)^2 P(x = x)dx = \int_0^1 (x(\alpha - a_1) - a_2x^2)^2 P(x = x)dx$ by the properties of uniform random variables $\int_0^1 (x(\alpha - a_1) - a_2x^2)^2 dx$
   - then we can expand this as $(f^*_H(x)) = \int_0^1 (x(\alpha - a_1) - a_2x^2)^2 dx = \int_0^1 \left(x\left(\alpha - a_1\right)\right)^2 - 2^3a_2 + 2x^3a_1a_2 + a_2^2x^4 dx = \frac{(\alpha - a_1)^2}{3} - \frac{\alpha a_2}{2} + \frac{a_1a_2}{2} + \frac{a_2^2}{5}$
   - our goal is to minimize this function so we can take the derivative $\frac{\partial R}{\partial \alpha} = \frac{2\alpha - 2a_1}{3} - \frac{a_2}{2}$
   - we can take the second derivative $\frac{\partial R^2}{\partial^2 \alpha} = \frac{2}{3} \geq 0$ meaning our risk function is convex and thus any min will be a global minima
   - then setting our first derivative with respect to α equal to zero and solving for α yields $\alpha* = a_1 + \frac{3}{4}a_2$
   - in the case where $a_2 = 0$ and we set $\alpha = \alpha^*$ we can see that our risk will be $R(f^*_H(x)) = E[\ell(f^*_H(x), y)] = E[\frac{1}{2}(\alpha^*x - a_1x - a_2x^2)^2] = E[\frac{1}{2}((a_1 + \frac{3}{4}a_2)x - a_1x - a_2x^2)^2] = \frac{1}{2}E[(a_1x - a_1x)^2 = 0$ meaning that our risk will be zero if $a_2 = 0$

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 $\hat{\boldsymbol{b}} = \arg\min_b \|Xb - \boldsymbol{y}\|_2^2$.

   - for simplicity think of x_i as the ith row of matrix X so x_ib is the vector product of the ith row of x times the matrix b

3 Q3 **1 / 2**

- 0 pts Correct

✓ **- 1 pts** *Wrong/missing inequality: should be R(f_H2) >= R(f_Hd)*

- 0.5 pts Correct inequality but missing/wrong justification: need to reason the inclusion relation between H_2 and H_d

- 0.5 pts Wrong/missing risk minimizer: should be just g(x)

- 0.5 pts Wrong/missing approximation error: should be 0

- 0.5 pts Click here to replace this description.

R(f_H_2) >= R(f_H_d) because H_2 is a subset of H_d. Also please do not write "no approx error", should be approximation error = 0

- we don't have any restrictions on the values of the constants, so by the same argument as in question one we could pick $f^*_{\mathcal{H}_d} = a_0 + a_1x + a_2x^2 + 0x^3 + 0x^4 ... + 0x^d$ that is we set $b_i = 0 \forall i \in [3, d]$
- further as $f^*_{\mathcal{H}_d} = a_0 + a_1x + a_2x^2 + 0x^3 + 0x^4 ... + 0x^d = a_0 + a_1x + a_2^2x = f^*_{\mathcal{H}_2} = f^*(x) = y(x) = y$ by the same argument as part 2 there is no approximation error

4. (4 Points) For this question we assume $a_0 = 0$. Considering $\mathcal{H} = \{f : x \to b_1x; b_1 \in \mathbb{R}\}$, 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?

   - out goal is to find a single $\alpha \in \mathbb{R}$ such that the risk is minimized.
   - so our problem can be expressed as $R(f^*_H(x)) = E[\ell(f^*_H(x), y)] = E[\frac{1}{2}(\alpha x - a_1x - a_2x^2)] = \frac{1}{2}E[(x(\alpha - a_1) - a_2x^2)^2] = \int_0^1 (x(\alpha - a_1) - a_2x^2)^2 P(x = x)dx = \int_0^1 (x(\alpha - a_1) - a_2x^2)^2 P(x = x)dx$ by the properties of uniform random variables $\int_0^1 (x(\alpha - a_1) - a_2x^2)^2 dx$
   - then we can expand this as $(f^*_H(x)) = \int_0^1 (x(\alpha - a_1) - a_2x^2)^2 dx = \int_0^1 \left(x\left(\alpha - a_1\right)\right)^2 - 2^3a_2 + 2x^3a_1a_2 + a_2^2x^4 dx = \frac{(\alpha - a_1)^2}{3} - \frac{\alpha a_2}{2} + \frac{a_1a_2}{2} + \frac{a_2^2}{5}$
   - our goal is to minimize this function so we can take the derivative $\frac{\partial R}{\partial \alpha} = \frac{2\alpha - 2a_1}{3} - \frac{a_2}{2}$
   - we can take the second derivative $\frac{\partial R^2}{\partial^2 \alpha} = \frac{2}{3} \geq 0$ meaning our risk function is convex and thus any min will be a global minima
   - then setting our first derivative with respect to α equal to zero and solving for α yields $\alpha* = a_1 + \frac{3}{4}a_2$
   - in the case where $a_2 = 0$ and we set $\alpha = \alpha^*$ we can see that our risk will be $R(f^*_H(x)) = E[\ell(f^*_H(x), y)] = E[\frac{1}{2}(\alpha^*x - a_1x - a_2x^2)^2] = E[\frac{1}{2}((a_1 + \frac{3}{4}a_2)x - a_1x - a_2x^2)^2] = \frac{1}{2}E[(a_1x - a_1x)^2 = 0$ meaning that our risk will be zero if $a_2 = 0$

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 $\hat{\boldsymbol{b}} = \arg\min_b \|Xb - \boldsymbol{y}\|_2^2$.

   - for simplicity think of x_i as the ith row of matrix X so x_ib is the vector product of the ith row of x times the matrix b

4 Q4 **3 / 4**

- **0 pts** Correct
- **2 pts** Wrong/missing risk minimizer, should be f*_H = (a_1 + 0.75*a_2)x, i.e., b_1 = a_1 + 0.75*a_2

✓ **- 1 pts** *Wrong/missing approximation error, should be a_2^2/160 (or a_2^2 / 80 if ignoring the 1/2 in risk)*

- **1 pts** Wrong/missing approximation error given a_2 = 0, it's just 0
- **0.5 pts** Click here to replace this description.
- **1 pts** Missing steps
- **1 pts** Wrong derivative
- **2 pts** No steps, please include all your work

- we don't have any restrictions on the values of the constants, so by the same argument as in question one we could pick $f^*_{\mathcal{H}_d} = a_0 + a_1x + a_2x^2 + 0x^3 + 0x^4 ... + 0x^d$ that is we set $b_i = 0 \forall i \in [3, d]$
- further as $f^*_{\mathcal{H}_d} = a_0 + a_1x + a_2x^2 + 0x^3 + 0x^4 ... + 0x^d = a_0 + a_1x + a_2^2x = f^*_{\mathcal{H}_2} = f^*(x) = y(x) = y$ by the same argument as part 2 there is no approximation error

4. (4 Points) For this question we assume $a_0 = 0$. Considering $\mathcal{H} = \{f : x \to b_1x; b_1 \in \mathbb{R}\}$, 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?

   - out goal is to find a single $\alpha \in \mathbb{R}$ such that the risk is minimized.
   - so our problem can be expressed as $R(f^*_H(x)) = E[\ell(f^*_H(x), y)] = E[\frac{1}{2}(\alpha x - a_1x - a_2x^2)] = \frac{1}{2}E[(x(\alpha - a_1) - a_2x^2)^2] = \int_0^1 (x(\alpha - a_1) - a_2x^2)^2 P(x = x)dx = \int_0^1 (x(\alpha - a_1) - a_2x^2)^2 P(x = x)dx$ by the properties of uniform random variables $\int_0^1 (x(\alpha - a_1) - a_2x^2)^2 dx$
   - then we can expand this as $(f^*_H(x)) = \int_0^1 (x(\alpha - a_1) - a_2x^2)^2 dx = \int_0^1 \left(x\left(\alpha - a_1\right)\right)^2 - 2^3a_2 + 2x^3a_1a_2 + a_2^2x^4 dx = \frac{(\alpha - a_1)^2}{3} - \frac{\alpha a_2}{2} + \frac{a_1a_2}{2} + \frac{a_2^2}{5}$
   - our goal is to minimize this function so we can take the derivative $\frac{\partial R}{\partial \alpha} = \frac{2\alpha - 2a_1}{3} - \frac{a_2}{2}$
   - we can take the second derivative $\frac{\partial R^2}{\partial^2 \alpha} = \frac{2}{3} \geq 0$ meaning our risk function is convex and thus any min will be a global minima
   - then setting our first derivative with respect to α equal to zero and solving for α yields $\alpha* = a_1 + \frac{3}{4}a_2$
   - in the case where $a_2 = 0$ and we set $\alpha = \alpha^*$ we can see that our risk will be $R(f^*_H(x)) = E[\ell(f^*_H(x), y)] = E[\frac{1}{2}(\alpha^*x - a_1x - a_2x^2)^2] = E[\frac{1}{2}((a_1 + \frac{3}{4}a_2)x - a_1x - a_2x^2)^2] = \frac{1}{2}E[(a_1x - a_1x)^2 = 0$ meaning that our risk will be zero if $a_2 = 0$

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 $\hat{\boldsymbol{b}} = \arg\min_b \|Xb - \boldsymbol{y}\|_2^2$.

   - for simplicity think of x_i as the ith row of matrix X so x_ib is the vector product of the ith row of x times the matrix b

- so it is worth noting for any given data point $f(x_i) = bx_i$
- empirical risk is defined as $\frac{1}{n}\Sigma_{i=1}^{n}\ell(f(x_i), y_i)$ using the loss function above this is more specifically $\frac{1}{n}\Sigma_{i=1}^{n}\ell(f(x_i), y_i) = \frac{1}{2n}\Sigma_{i=1}^{n}(f(x_i) - y_i)^2$
- using the l2 norm we can write $||Xb - y||_2 = \sqrt{\Sigma_{i=1}^{n}(bx_i - y_i)^2}$ squaring this we end up with $||Xb - y||_2^2 = \Sigma_{i=1}^{n}(bx_i - y_i)^2 = \Sigma_{i=1}^{n}(f(x_i) - y_i)^2$
- so note that our empirical risk and$||Xb-y||_2$ only differ by a constant factor $\frac{1}{2n}$ which will not effect optimization thus $\hat{b} = \arg\min_b \|Xb - \boldsymbol{y}\|_2^2 = \arg\min_f \hat{R}_n(f)$ which is the desired result

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 ?

   - we can express our question as $||Xb-y||_2^2 = (Xb-y)^t(Xb-y) = (b^tX^t-y^t)(Xb-y) = b^tX^tXb - b^tX^ty - y^tXb - y^ty$
   - now lets take its gradient in parts
     - first lets consider $\frac{\partial bX^tXb}{\partial b}$ since the product of two matrices are themselves a matrix we can think of this as $\frac{\partial bX^tXb}{\partial b} = \frac{\partial b^tAb}{\partial b}$ for some matrix A which must be symmetric as it is the product of the transpose of a matrix and its self, and we know $\frac{\partial b^tAb}{\partial b} = 2b^t(A) = 2b^t(X^tX)$
     - next we can write $\frac{\partial b^tX^ty}{\partial b} = \frac{\partial b^tv}{\partial b} = v = X^ty$
     - then we have $\frac{\partial y^tXb}{\partial b} = \frac{\partial v^tb}{\partial b} = v^t = y^tX$
     - putting this all together we get $\nabla = 2b^t(X^tX) - X^ty - y^tX = 2b^t(X^tX) - 2 < X, y >$
     - also taking the second derivative we find $\frac{\partial \nabla}{\partial b} = 2X^TX$ which must be a positive definite matrix meaning the problem is convex, and thus min of our gradient will be a global min
   - so now all that remains to do is to solve $\nabla = 2b^t(X^tX) - 2 < X, y >= 0$ this yields $b^t(X^tX) = Y^tX$ we know that X is full rank, and thus invertable. thus we have $b^t = (y^tX)(X^tX)^{-1}$ then we take the transpose to finally get $b = ((y^tX)(X^tX)^{-1})^t = ((X^TX)^{-1})^t(X^ty) = (X^tX)^{-1}(X^ty)$ which is the desired result
   - we need x to be full rank so that it is invertable.
   - we need $N > d$ since if this is not the case we are not guaranteed to have X^TX as positive semi definite, and thus the problem may not be convex meaning our solution may not be a global min.

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

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

5 Q5 2 / 2

✓ **- 0 pts** *Correct.*

- 0.5 pts No or wrong ERM definition.

- 0.5 pts No or wrong vector norm justification.

- 1 pts Incomplete or partially wrong.

- 1.5 pts Incorrect.

- 2 pts No answer shown.

- so it is worth noting for any given data point $f(x_i) = bx_i$
- empirical risk is defined as $\frac{1}{n}\Sigma_{i=1}^{n}\ell(f(x_i), y_i)$ using the loss function above this is more specifically $\frac{1}{n}\Sigma_{i=1}^{n}\ell(f(x_i), y_i) = \frac{1}{2n}\Sigma_{i=1}^{n}(f(x_i) - y_i)^2$
- using the l2 norm we can write $||Xb - y||_2 = \sqrt{\Sigma_{i=1}^{n}(bx_i - y_i)^2}$ squaring this we end up with $||Xb - y||_2^2 = \Sigma_{i=1}^{n}(bx_i - y_i)^2 = \Sigma_{i=1}^{n}(f(x_i) - y_i)^2$
- so note that our empirical risk and$||Xb-y||_2$ only differ by a constant factor $\frac{1}{2n}$ which will not effect optimization thus $\hat{b} = \arg\min_b \|Xb - \boldsymbol{y}\|_2^2 = \arg\min_f \hat{R}_n(f)$ which is the desired result

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 ?

   - we can express our question as $||Xb-y||_2^2 = (Xb-y)^t(Xb-y) = (b^tX^t-y^t)(Xb-y) = b^tX^tXb - b^tX^ty - y^tXb - y^ty$
   - now lets take its gradient in parts
     - first lets consider $\frac{\partial bX^tXb}{\partial b}$ since the product of two matrices are themselves a matrix we can think of this as $\frac{\partial bX^tXb}{\partial b} = \frac{\partial b^tAb}{\partial b}$ for some matrix A which must be symmetric as it is the product of the transpose of a matrix and its self, and we know $\frac{\partial b^tAb}{\partial b} = 2b^t(A) = 2b^t(X^tX)$
     - next we can write $\frac{\partial b^tX^ty}{\partial b} = \frac{\partial b^tv}{\partial b} = v = X^ty$
     - then we have $\frac{\partial y^tXb}{\partial b} = \frac{\partial v^tb}{\partial b} = v^t = y^tX$
     - putting this all together we get $\nabla = 2b^t(X^tX) - X^ty - y^tX = 2b^t(X^tX) - 2 < X, y >$
     - also taking the second derivative we find $\frac{\partial \nabla}{\partial b} = 2X^TX$ which must be a positive definite matrix meaning the problem is convex, and thus min of our gradient will be a global min
   - so now all that remains to do is to solve $\nabla = 2b^t(X^tX) - 2 < X, y >= 0$ this yields $b^t(X^tX) = Y^tX$ we know that X is full rank, and thus invertable. thus we have $b^t = (y^tX)(X^tX)^{-1}$ then we take the transpose to finally get $b = ((y^tX)(X^tX)^{-1})^t = ((X^TX)^{-1})^t(X^ty) = (X^tX)^{-1}(X^ty)$ which is the desired result
   - we need x to be full rank so that it is invertable.
   - we need $N > d$ since if this is not the case we are not guaranteed to have X^TX as positive semi definite, and thus the problem may not be convex meaning our solution may not be a global min.

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

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

6 Q6 **3 / 3**

✓ **- 0 pts** *Correct*

- 2 pts Wrong/missing computations

- 1 pts Wrong/missing reason for conditions

- 3 pts empty

- so it is worth noting for any given data point $f(x_i) = bx_i$
- empirical risk is defined as $\frac{1}{n}\Sigma_{i=1}^{n}\ell(f(x_i), y_i)$ using the loss function above this is more specifically $\frac{1}{n}\Sigma_{i=1}^{n}\ell(f(x_i), y_i) = \frac{1}{2n}\Sigma_{i=1}^{n}(f(x_i) - y_i)^2$
- using the l2 norm we can write $||Xb - y||_2 = \sqrt{\Sigma_{i=1}^{n}(bx_i - y_i)^2}$ squaring this we end up with $||Xb - y||_2^2 = \Sigma_{i=1}^{n}(bx_i - y_i)^2 = \Sigma_{i=1}^{n}(f(x_i) - y_i)^2$
- so note that our empirical risk and$||Xb-y||_2$ only differ by a constant factor $\frac{1}{2n}$ which will not effect optimization thus $\hat{b} = \arg\min_b \|Xb - \boldsymbol{y}\|_2^2 = \arg\min_f \hat{R}_n(f)$ which is the desired result

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 ?

   - we can express our question as $||Xb-y||_2^2 = (Xb-y)^t(Xb-y) = (b^tX^t-y^t)(Xb-y) = b^tX^tXb - b^tX^ty - y^tXb - y^ty$
   - now lets take its gradient in parts
     - first lets consider $\frac{\partial bX^tXb}{\partial b}$ since the product of two matrices are themselves a matrix we can think of this as $\frac{\partial bX^tXb}{\partial b} = \frac{\partial b^tAb}{\partial b}$ for some matrix A which must be symmetric as it is the product of the transpose of a matrix and its self, and we know $\frac{\partial b^tAb}{\partial b} = 2b^t(A) = 2b^t(X^tX)$
     - next we can write $\frac{\partial b^tX^ty}{\partial b} = \frac{\partial b^tv}{\partial b} = v = X^ty$
     - then we have $\frac{\partial y^tXb}{\partial b} = \frac{\partial v^tb}{\partial b} = v^t = y^tX$
     - putting this all together we get $\nabla = 2b^t(X^tX) - X^ty - y^tX = 2b^t(X^tX) - 2 < X, y >$
     - also taking the second derivative we find $\frac{\partial \nabla}{\partial b} = 2X^TX$ which must be a positive definite matrix meaning the problem is convex, and thus min of our gradient will be a global min
   - so now all that remains to do is to solve $\nabla = 2b^t(X^tX) - 2 < X, y >= 0$ this yields $b^t(X^tX) = Y^tX$ we know that X is full rank, and thus invertable. thus we have $b^t = (y^tX)(X^tX)^{-1}$ then we take the transpose to finally get $b = ((y^tX)(X^tX)^{-1})^t = ((X^TX)^{-1})^t(X^ty) = (X^tX)^{-1}(X^ty)$ which is the desired result
   - we need x to be full rank so that it is invertable.
   - we need $N > d$ since if this is not the case we are not guaranteed to have X^TX as positive semi definite, and thus the problem may not be convex meaning our solution may not be a global min.

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

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

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):
    """
    inputs:
    X inputs np.array of xize (Nx(d+1)) where n>d
    y data to fit
    returns:
    b (np .array of size d+1 with least squares estimator of y given x)
    """
    assert len(X)>(len(X[0])-1), "matrix must have N>D"
    return np.linalg.inv(X.T@X)@X.T@y
```

- this is a pretty direct implementation of what was derived in question 6.

8. (1 Points) Recall the definition of the empirical 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 $\boldsymbol{b} \in \mathbb{R}^{(d+1)}$ parameterize the predictor.

```
def empirical_risk(X,y,b):
    """
    inputs:
    X input matrix (NX(d+1))
    y target (N)
    b (d+1) parameters for x
    output:
    int empyrical risk
    """
    return np.mean( ( (X@b) -y ) **2 )/2
```

- here i am assuming that $\ell(f(x), y) = \frac{1}{2}(f(x) - y)^2$ as was defined above question 1, since we never got another loss function.

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 .

7 Q7 2 / 2

✓ **- 0 pts** *Correct*

- 1 pts Missing exception when N <= d

- 0.5 pts Wrong exception condition

Should be `X.shape[0] <= X.shape[1] - 1`

- 0.5 pts Incorrect function input

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):
    """
    inputs:
    X inputs np.array of xize (Nx(d+1)) where n>d
    y data to fit
    returns:
    b (np .array of size d+1 with least squares estimator of y given x)
    """
    assert len(X)>(len(X[0])-1), "matrix must have N>D"
    return np.linalg.inv(X.T@X)@X.T@y
```

- this is a pretty direct implementation of what was derived in question 6.

8. (1 Points) Recall the definition of the empirical 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 $\boldsymbol{b} \in \mathbb{R}^{(d+1)}$ parameterize the predictor.

```
def empirical_risk(X,y,b):
    """
    inputs:
    X input matrix (NX(d+1))
    y target (N)
    b (d+1) parameters for x
    output:
    int empyrical risk
    """
    return np.mean( ( (X@b) -y ) **2 )/2
```

- here i am assuming that $\ell(f(x), y) = \frac{1}{2}(f(x) - y)^2$ as was defined above question 1, since we never got another loss function.

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 .

8 Q8 **1 / 1**

✓ **- 0 pts** *Correct*

- 0.5 pts Missing 1/2

- 0.5 pts Missing square

- 0.5 pts Should be divided by `(2 * len(y))`

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):
    """
    inputs:
    X inputs np.array of xize (Nx(d+1)) where n>d
    y data to fit
    returns:
    b (np .array of size d+1 with least squares estimator of y given x)
    """
    assert len(X)>(len(X[0])-1), "matrix must have N>D"
    return np.linalg.inv(X.T@X)@X.T@y
```

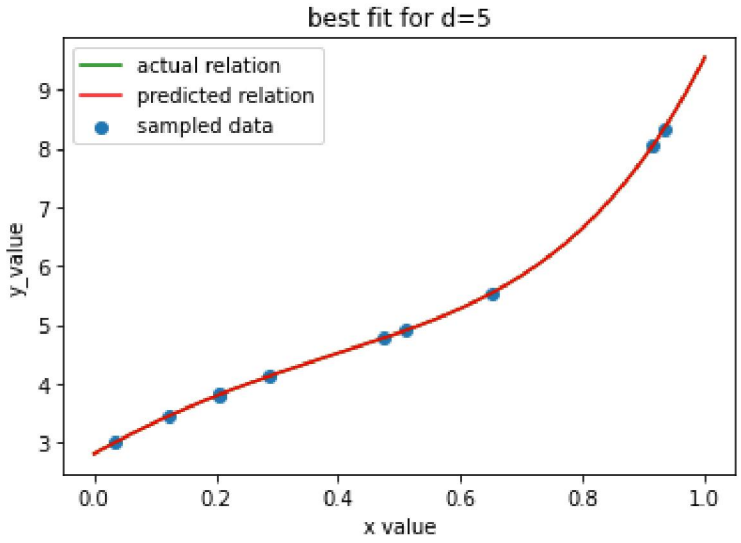
- this is a pretty direct implementation of what was derived in question 6.

8. (1 Points) Recall the definition of the empirical 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 $\boldsymbol{b} \in \mathbb{R}^{(d+1)}$ parameterize the predictor.

```
def empirical_risk(X,y,b):
    """
    inputs:
    X input matrix (NX(d+1))
    y target (N)
    b (d+1) parameters for x
    output:
    int empyrical risk
    """
    return np.mean( ( (X@b) -y ) **2 )/2
```

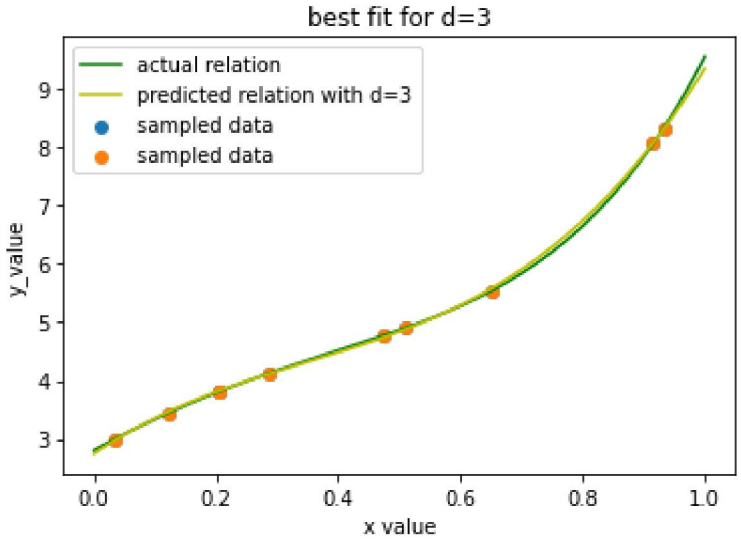
- here i am assuming that $\ell(f(x), y) = \frac{1}{2}(f(x) - y)^2$ as was defined above question 1, since we never got another loss function.

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 .



-
- this plot show that using $d = 5$ we can find a $\hat{b}$ which perfectly estimate $g(X)$

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?



this plot show that we can find very very close to the same relationship using $d = 3$. this contrast with our conclusion above which was that we could model $g(x)$ perfectly with a degree 2 polynomial. The difference however is that in this case we were only allowing our model to learn from 10 data points. presumably if we increased the number of points to learn from we could reduce this number farther

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.

9 Q9 **3 / 3**

✓ **- 0 pts** *Correct*

- 0.5 pts Wrong dimension for a, b

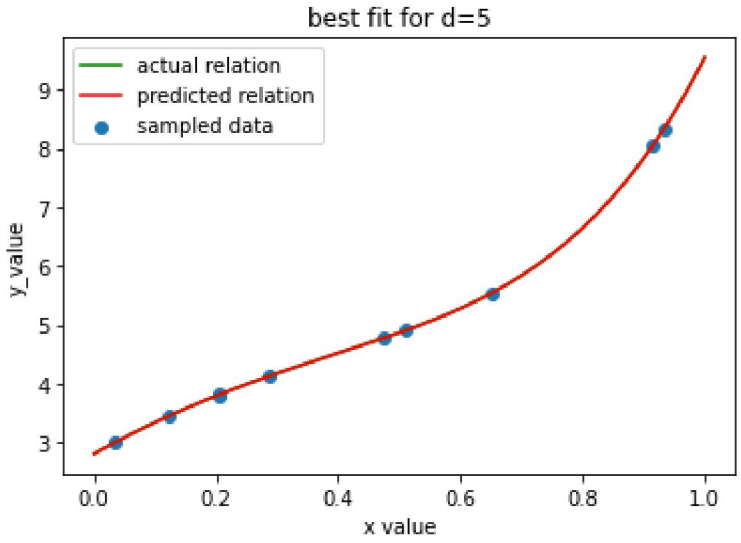
- 0.5 pts Missing a, b comparison

- 0.5 pts Missing training set points

- 1 pts Function should be smooth curve (use `np.linspace(0,1,100)` as x values, not x_train)

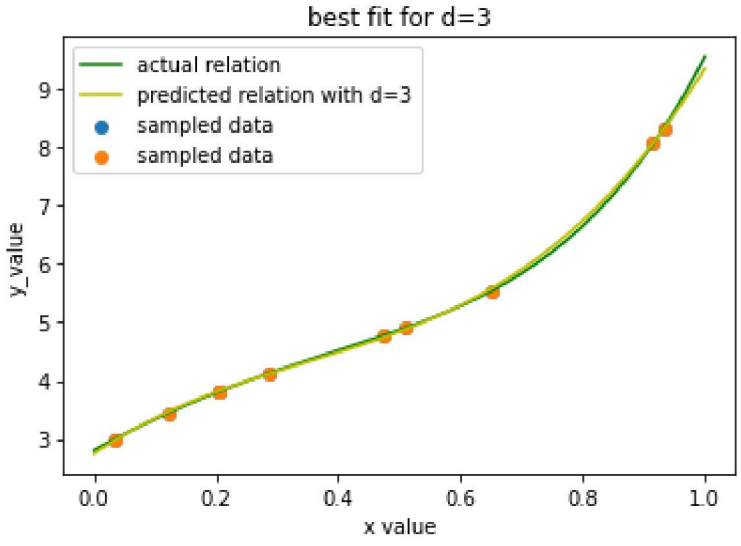
- 2 pts Wrong / Missing plot

- 0.5 pts Click here to replace this description.



-
- this plot show that using $d = 5$ we can find a $\hat{b}$ which perfectly estimate $g(X)$

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?



this plot show that we can find very very close to the same relationship using $d = 3$. this contrast with our conclusion above which was that we could model $g(x)$ perfectly with a degree 2 polynomial. The difference however is that in this case we were only allowing our model to learn from 10 data points. presumably if we increased the number of points to learn from we could reduce this number farther

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.

10 Q10 **0.5 / 1**

- **0 pts** Correct

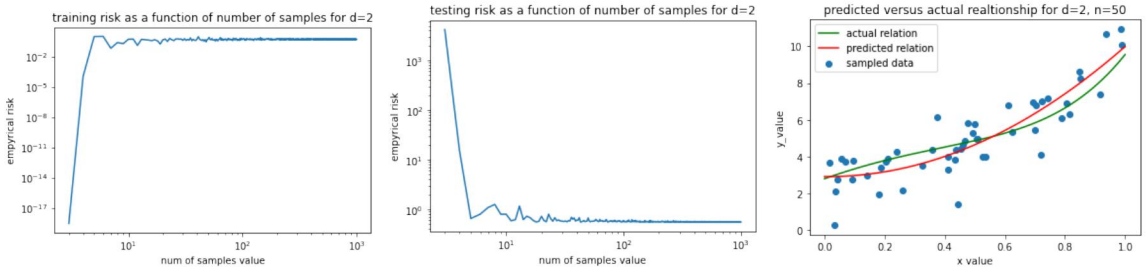
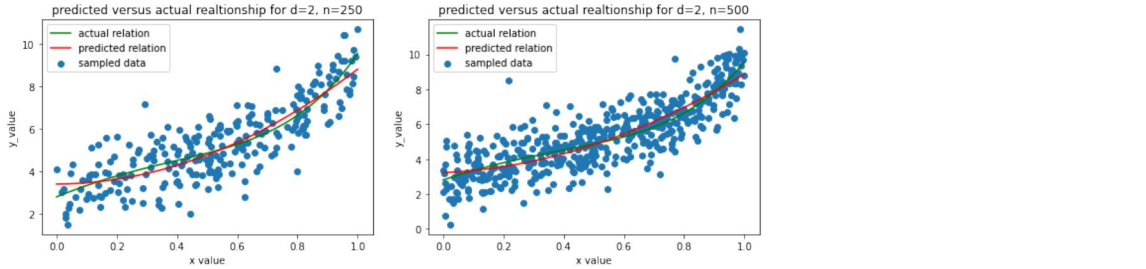
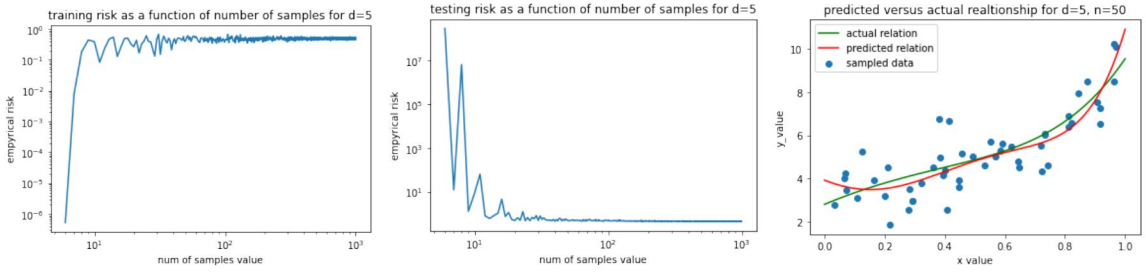
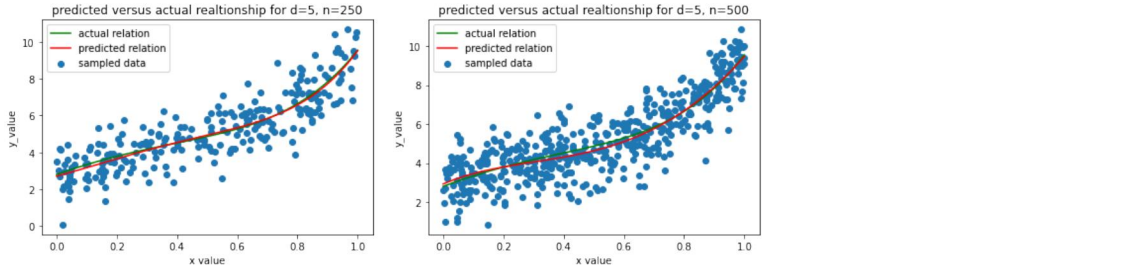
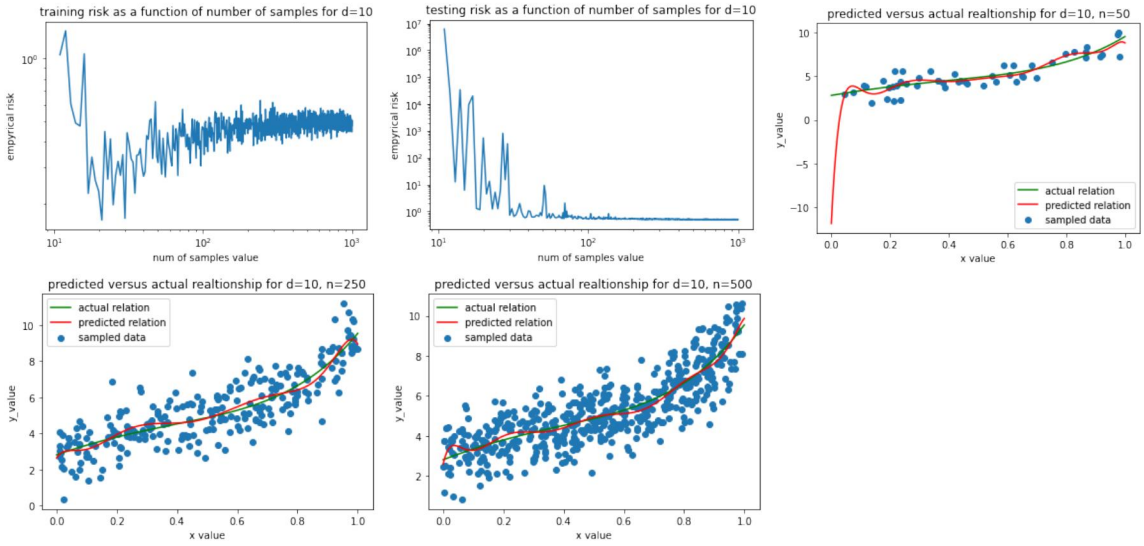
✓ **- 0.5 pts** *Missing / wrong minimum value of d*

- **0.5 pts** Missing / wrong conclusion on approximation error

- **0.5 pts** Approximation error should be exactly equal to 0

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.

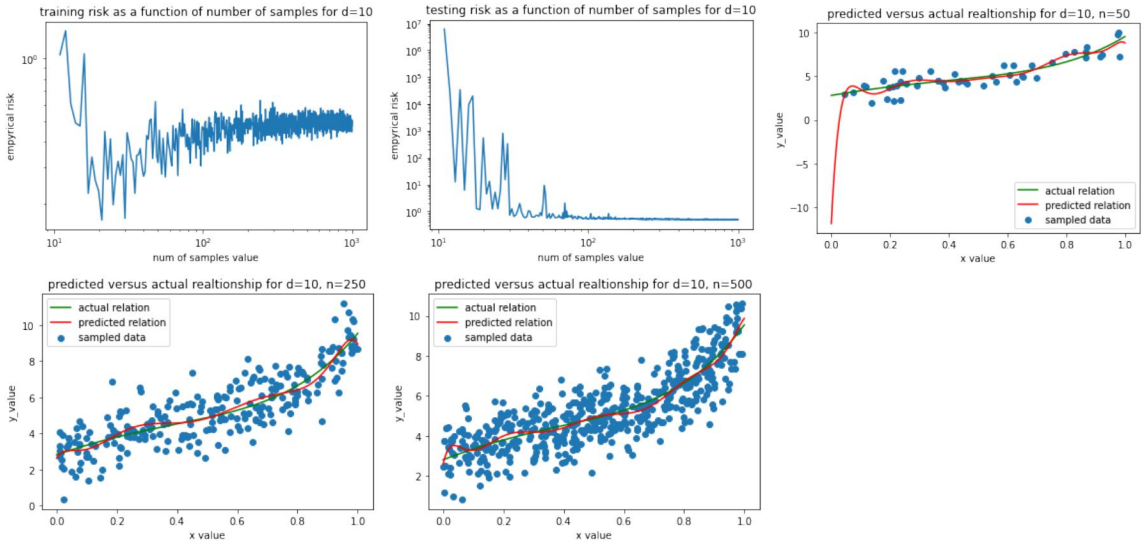
    - the code used to generate the bellow plots is here.
    - ```
      def error_for_n(d):
          epislon_t=[]
          epsilon_d=[]
          X=[]
          Y=[]
          n=[15,20,50,100,250,500]
          b=[]
          x_test_noise,y_test_noise=draw_sample_with_noise(5,a,1000)
          X_mat_test=get_design_mat(x_test_noise,deg=d)
          for i in range(d+1,1001):
              x_train_noise,y_train_noise=draw_sample_with_noise(5,a,i)
              X_mat_train=get_design_mat(x_train_noise,deg=d)
              b_noise=least_square_estimator(X_mat_train,y_train_noise)
              epislon_t.append(empirical_risk(X_mat_train, y_train_noise,
              ↪  b_noise))
              epsilon_d.append(empirical_risk(X_mat_test, y_test_noise,
              ↪  b_noise))
              if(i in n):
                  b.append(b_noise)
                  X.append(x_train_noise)
                  Y.append(y_train_noise)
          plt.plot(range(d+1,1001), epislon_t)
          plt.yscale("log")
          plt.xscale('log')
          plt.xlabel("num of samples value")
          plt.ylabel("empyrical risk")
          plt.title("training risk as a function of number of samples for
          ↪  d={0}".format(d))
          plt.show()
          plt.plot(range(d+1,1001), epsilon_d)
          plt.yscale("log")
          plt.xscale('log')
          plt.xlabel("num of samples value")
          plt.ylabel("empyrical risk")
          plt.title("testing risk as a function of number of samples for
          ↪  d={0}".format(d))
          plt.show()
          x_range = np.linspace(0, 1, 1000)
          for i in range(len(b)):
              plt.plot(x_range,get_design_mat(x_range,5)@a,label="actual
              ↪  relation", color='g')

              ↪  plt.plot(x_range,get_design_mat(x_range,d)@b[i],label="predicted
              ↪  relation", color='r')
      ```
      ```

```
        plt.scatter(X[i],Y[i],label="sampled data")
        plt.xlabel("x value")
        plt.ylabel("y_value")
        plt.title("predicted versus actual realtionship for d={},
        ↪  n={}".format(d,n[i]))
        plt.legend()
        plt.show()

d=[2,5,10]
for i in d:
    error_for_n(i)
```

- 



- 



- 

  - across all d values we can see that training error tends to increase for a period of time in n, this makes sense as a small number of points can be perfectly fit, but as more points are added the effect of noise in our sampling leads to training error. eventually however the increase in training error levels off presumably around the amount caused by noise in the data
  - testing error on the other hand across all values of d has an initial large drop off in testing error as n increases this makes sense as the data used to train the model is becoming more representative of the underlying population, the testing error eventually plateaus again presumably around the level of noise in the data.
  - we can in general the the data does a poor job approximating the actual relationship with low levels of n but does better as n goes up.
  - it seems like d=2 under fits the relationship at low levels of n, and d=10 over fits the data at low levels of n which makes sense since the real data has dimension d=5

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).

    - i defined the empirical estimator of estimation error as
    - ```
      def empyrical_estimation_error(X_test,y_test, b,super_b):
          return empirical_risk(X_test, y_test,b)-
          ↪  empirical_risk(X_test,y_test,super_b)
      ```
    - where b is the minimum mean squared error estimator fit using just the training data, and super b is the minimum mean squared error estimator using the testing data.
    - i think this is a logical approach we are comparing the risk of super b which perfectly fits which is the best fit we could have for our testing error, with the risk of the best function we have found using our training set.
      ```

11 Q11 **6 / 6**

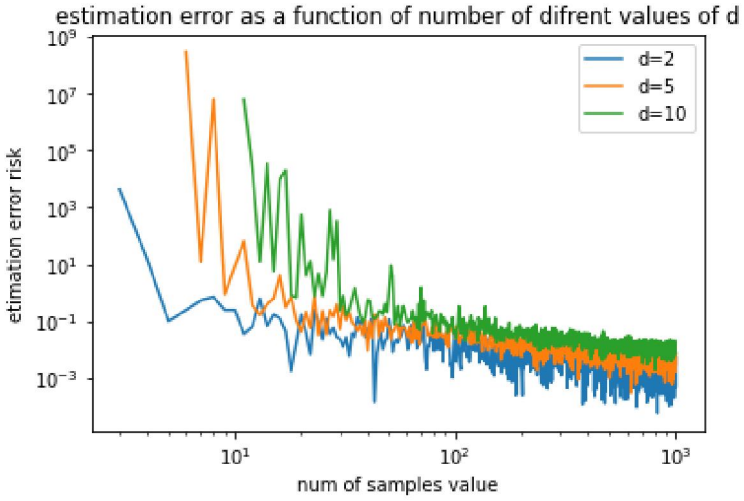
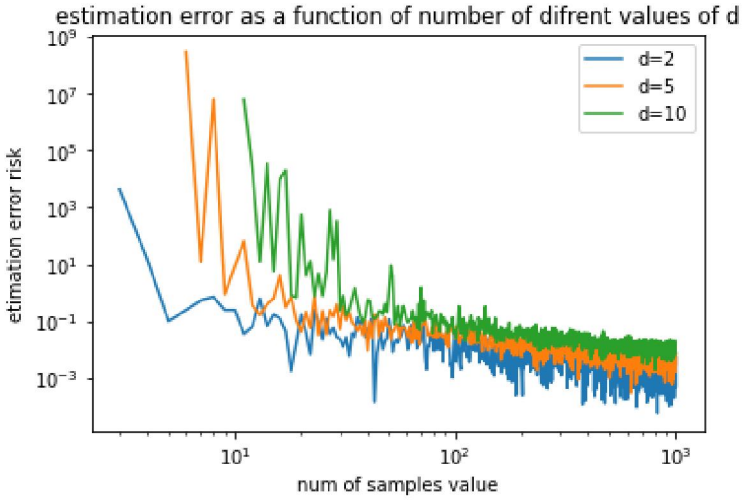
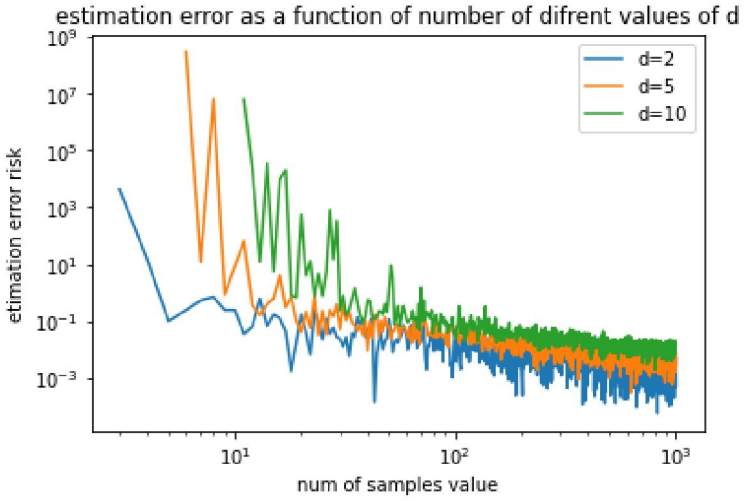
✓ **- 0 pts** *Correct*

- **1.5 pts** incorrect plot for $$e_t$$
- **1.5 pts** incorrect plot for $$e_g$$
- **1.5 pts** incorrect plots to Plot 1 for 2 or 3 different values of N for each value of d.
- **1 pts** plots for $$e_t$$ and $$e_g$$ should converge to 0.5
- **2 pts** missing the plot for $$e_t$$
- **2 pts** missing the plot for $$e_g$$
- **2 pts** missing the plots similar to Plot 1 for 2 or 3 different values of N for each value of d.

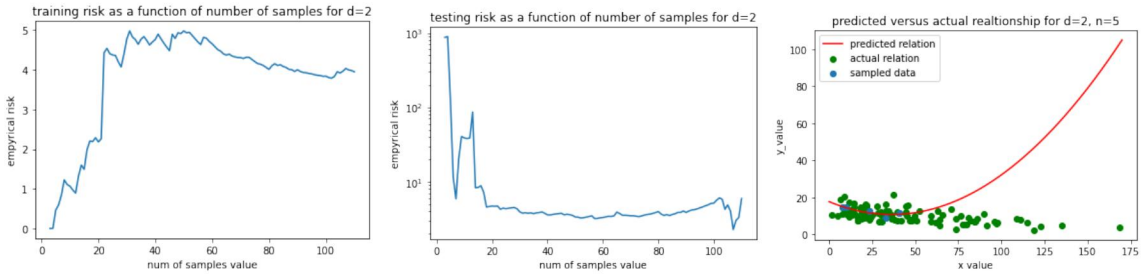
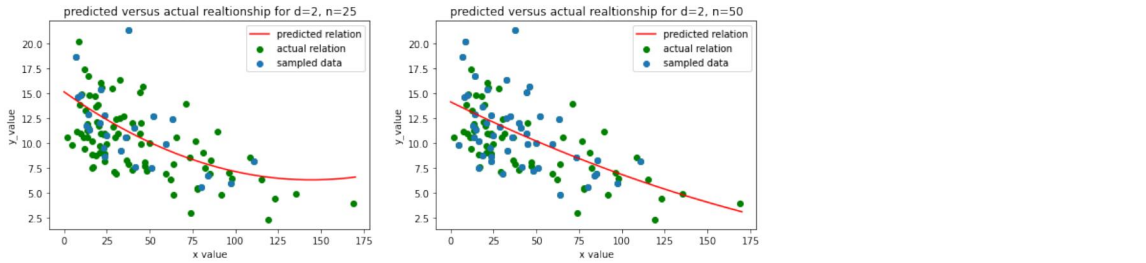
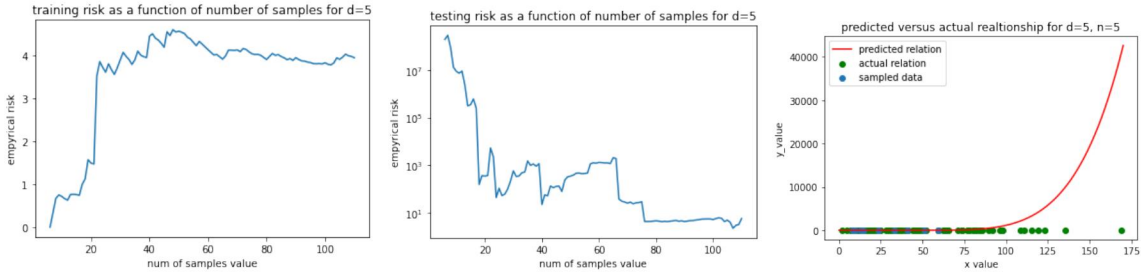
- 

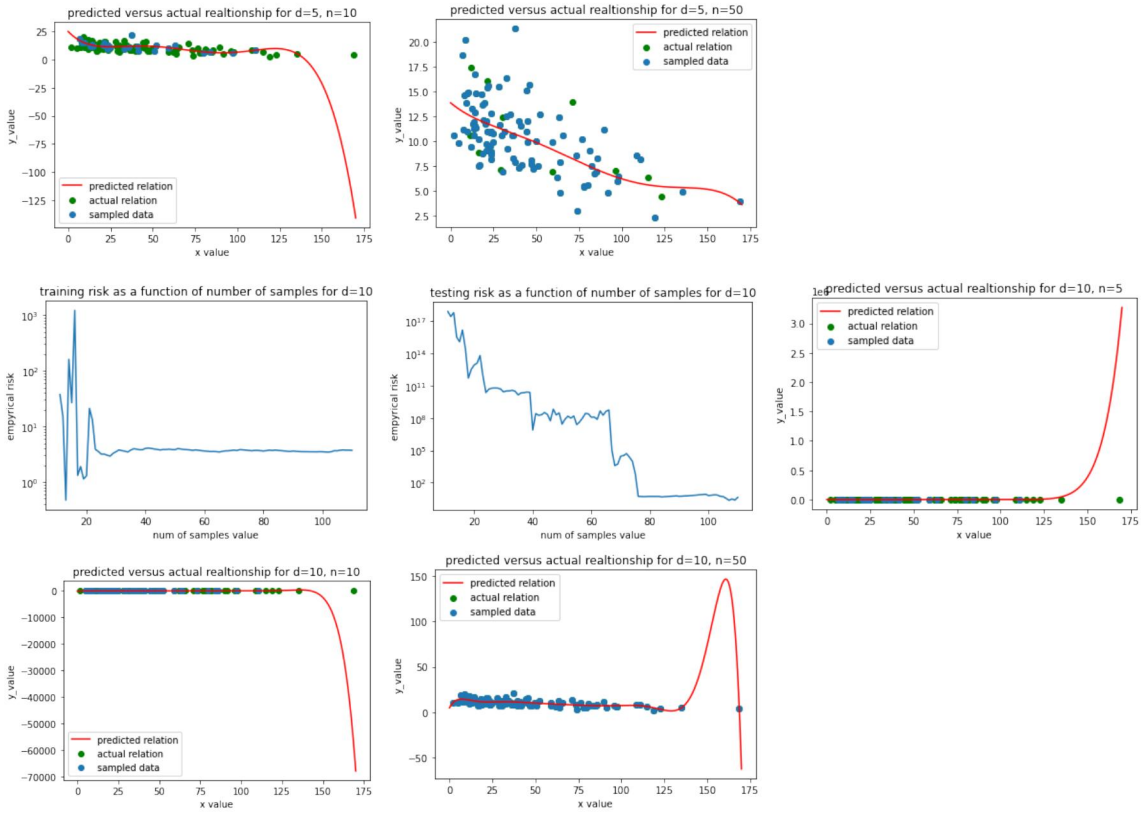
  - across all d values we can see that training error tends to increase for a period of time in n, this makes sense as a small number of points can be perfectly fit, but as more points are added the effect of noise in our sampling leads to training error. eventually however the increase in training error levels off presumably around the amount caused by noise in the data
  - testing error on the other hand across all values of d has an initial large drop off in testing error as n increases this makes sense as the data used to train the model is becoming more representative of the underlying population, the testing error eventually plateaus again presumably around the level of noise in the data.
  - we can in general the the data does a poor job approximating the actual relationship with low levels of n but does better as n goes up.
  - it seems like d=2 under fits the relationship at low levels of n, and d=10 over fits the data at low levels of n which makes sense since the real data has dimension d=5

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).

    - i defined the empirical estimator of estimation error as
    - ```
      def empyrical_estimation_error(X_test,y_test, b,super_b):
          return empirical_risk(X_test, y_test,b)-
          ↪  empirical_risk(X_test,y_test,super_b)
      ```
    - where b is the minimum mean squared error estimator fit using just the training data, and super b is the minimum mean squared error estimator using the testing data.
    - i think this is a logical approach we are comparing the risk of super b which perfectly fits which is the best fit we could have for our testing error, with the risk of the best function we have found using our training set.
      ```



- here we can see that overall empyrical estimation error goes down as n increases, which is logical as the training set is getting larger and generalization is getting better.
- this decrease of course Plautus somewhere around the level of noise in the the testing set
- further it seems that lower values of d have lower testing errors, which makes sense since higher value of d models, require more parameters be fit with the same amount of data.

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?

    - we can see that for all values of d estimation error is decreasing in n.
    - we can see that estimation error eventually plateaus, this makes sense because we are estimating fixed parameters from data with noise so there is some level of ineducable error
    - we can see that higher values of d have higher estimation error and take a larger n to stabilize this makes sense as for higher degree polynomials both noise is propagated and we are using the same amount of data to predict more parameters
    - so a higher n will lead to lower estimation error, and a higher d will lead to higher approximation error.

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?

    - this is a linear regression problem with a known closed form, so i don't think there is any numeric approximation going on (outside of maybe while inverting matrices) thus optimization error should be frailly minimal

Application to Ozone data (optional) (2 Points)

You can now use the code we developed on the synthetic example on a real world dataset.

12 Q12 **3 / 4**

- 0 pts Correct

- 0.5 pts Wrong empirical estimator formula : $$F(f_{n}^*) - F(f_{\mathcal{H}}^*) \approx e_g - \frac{1}{2}$$

- 1 pts Did not recall the estimation error formula

✓ **- 1 pts** *No empirical estimation error formula*

- 1 pts Wrong plots

- 4 pts No submission



- here we can see that overall empyrical estimation error goes down as n increases, which is logical as the training set is getting larger and generalization is getting better.
- this decrease of course Plautus somewhere around the level of noise in the the testing set
- further it seems that lower values of d have lower testing errors, which makes sense since higher value of d models, require more parameters be fit with the same amount of data.

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?

    - we can see that for all values of d estimation error is decreasing in n.
    - we can see that estimation error eventually plateaus, this makes sense because we are estimating fixed parameters from data with noise so there is some level of ineducable error
    - we can see that higher values of d have higher estimation error and take a larger n to stabilize this makes sense as for higher degree polynomials both noise is propagated and we are using the same amount of data to predict more parameters
    - so a higher n will lead to lower estimation error, and a higher d will lead to higher approximation error.

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?

    - this is a linear regression problem with a known closed form, so i don't think there is any numeric approximation going on (outside of maybe while inverting matrices) thus optimization error should be frailly minimal

Application to Ozone data (optional) (2 Points)

You can now use the code we developed on the synthetic example on a real world dataset.

13 Q13 **2 / 2**

✓ **- 0 pts** *Correct*

- 1 pts Increasing d tend to increase the generalisation error because of overfitting

- 1 pts Increasing N reduces the effect of the noise

- 2 pts No submission



- here we can see that overall empyrical estimation error goes down as n increases, which is logical as the training set is getting larger and generalization is getting better.
- this decrease of course Plautus somewhere around the level of noise in the the testing set
- further it seems that lower values of d have lower testing errors, which makes sense since higher value of d models, require more parameters be fit with the same amount of data.

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?

    - we can see that for all values of d estimation error is decreasing in n.
    - we can see that estimation error eventually plateaus, this makes sense because we are estimating fixed parameters from data with noise so there is some level of ineducable error
    - we can see that higher values of d have higher estimation error and take a larger n to stabilize this makes sense as for higher degree polynomials both noise is propagated and we are using the same amount of data to predict more parameters
    - so a higher n will lead to lower estimation error, and a higher d will lead to higher approximation error.

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?

    - this is a linear regression problem with a known closed form, so i don't think there is any numeric approximation going on (outside of maybe while inverting matrices) thus optimization error should be frailly minimal

Application to Ozone data (optional) (2 Points)

You can now use the code we developed on the synthetic example on a real world dataset.

14 Q14 **1 / 1**

✓ **- 0 pts** *Correct*

- 1 pts Optimisation error is 0 because of analytical solution

- 1 pts No submission

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.

    - here is my train test split code
    - ```
      ozone_data=np.loadtxt("ozone_wind.data")
      def train_test_split(n):
          training_indecies=np.random.choice(range(len(ozone_data)), n,
          ↪  replace=False)
          testing_indecies=[i for i in range(len(ozone_data)) if i not in
          ↪  training_indecies]

          ↪  X_train,y_train=ozone_data[training_indecies,0],ozone_data[training_indecies,1]

          ↪  X_test,y_test=ozone_data[testing_indecies,0],ozone_data[testing_indecies,1]
          return X_train, y_train, X_test, y_test
      ```
    - 

    - 

    - 

      ```

- 


- we can see that for all values of d as n increases training error initially spikes, and then stabilizes around a certain value. this makes sense as it easy to achieve low training error on a few points regardless of noise, but as the number of points in the training set go up there needs the noise is observed in the training error
- testing error on the other hand initially decreases in n and then stabilizes. This makes sense, as when $b$ is trained on only a few points, it will generalize well but as n goes up the model is more able to generalize, until it plateaus around the level of risk that comes from noise
- looking at d it seems that d=2 under fits the training data, and d=10 over fits the training data, so perhaps the degree of the polynomial is close to 5.
- this is all consistent with what we observed above

*15* Q15 **2 / 2**

✓ **- 0 pts** *Correct*

**- 1 pts** wrong/missing plot

**- 1 pts** no discussion

**- 2 pts** empty