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 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^*.

   In this case, a polynomial with degree 2, taking the form $f^* = y = a_0 + a_1 x + a_2 x^2$ would be the obvious choice, as in our problem statement we have defined our function as such. Since we could fit the parameters a_0, a_1, a_2 perfectly to match those in which the function is defined, we would have no loss, as our predictions match the ground truth perfectly. The expected risk would be minimized as

   $$R(f^*) = E(\frac{1}{2}(\hat{y} - y)^2) = 0 \text{ as } \hat{y}_i = y_i$$

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

   Since we are using $\mathcal{H}_2$ as our hypothesis class, this includes all the polynomials of degree two, and thus the Bayes predictor we identified in problem 1. Since the Bayes predictor, a polynomial of degree 2 taking the form $f^* = y = a_0 + a_1 x + a_2 x^2$, is within our hypothesis space, that would be the function we'd use, and thus $f_F = f^*$. Using the definition of Approximation Error:

   $$\textit{Approximation Error} = R(f_F) - R(f^*) = R(f^*) - R(f^*) = 0$$

   Thus, the resulting approximation error would be 0.

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

   Update from campus wire: the question is worded unclearly, and in our new hypothesis space all polynomials of degree 2 are included. In other words, our new hypothesis space, $R(f^*_{\mathcal{H}_d})$, only includes polynomials of degree d or lower. Since our Bayes predictor is still within our hypothesis space, we can still achieve minimal risk minimizer in $R(f^*_{\mathcal{H}_d})$. Therefore, $R(f^*_{\mathcal{H}_2}) = R(f^*_{\mathcal{H}_d})$ in all cases.

   Approximation error would be equal to 0, as our Bayes predictor is within our hypothesis space:

$$Approximation\ Error = R(f_F) - R(f^*) = R(f^*) - R(f^*) = 0$$

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?

   Our prediction $\hat{y}$ now takes the form $\hat{y} = a_1 x$. Applying the loss function to our prediction to calculate risk, we will want to calculate:

$$R(\hat{f}) = E(l(\hat{y}, y)) = E(\frac{1}{2}(b_1 x - a_1 x - a_2 x^2))^2$$

   Since our x is uniform from 0 to 1, we can calculate its expectation by integrating over its domain. Let $h(x)$ be the value of our ground truth function a at point x, and $f(x)$ be the probability density at that point. Since our x is uniform between 0 and 1, $f(x) = 1$ for $0 \leq x \leq 1$ and 0 otherwise.

$$\begin{aligned} E(\frac{1}{2}(b_1 x - a_1 x - a_2 x^2))^2 &= \int_0^1 h(x) f(x) dx \\ &= \int_0^1 \frac{1}{2}(b_1 x - a_1 x - a_2 x^2)^2 dx \\ &= \frac{1}{2}\int_0^1 ((b_1 - a_1)x - a_2 x^2)^2 dx \\ &= (\frac{1}{2}(\frac{1}{3}(b_1 - a_1)^2 x^3) + (\frac{1}{2}(b_1 - a_1)^2 a_2^2 x^4) + \frac{1}{5}a_2^2 x^5)\ |_0^1 \\ &= \frac{1}{6}(b_1 - a_1)^2 + \frac{1}{6}(b_1 - a_1)^2 a_2^2 + \frac{1}{10}a_2^2 \end{aligned} \tag{1}$$

   Taking the derivative with respect to b_1 and setting to 0 to minimize our function:

$$\begin{aligned} \frac{d}{dx}R(f^*_{\mathcal{H}}) &= 0 \\ \frac{1}{3}(b_1 - a_1) - \frac{1}{4}(a_2) &= 0 \\ b_1 &= 3 \times (\frac{1}{4}a_2 + \frac{1}{3}a_1) \\ b_1 &= \frac{3}{4}a_2 + a_1 \end{aligned} \tag{2}$$

We've found our optimal weight for b_1, and now our risk minimizer in $\mathcal{H}$ is equal to $\mathcal{H} = (a_1 + \frac{3}{4}a_2)x$. Now to calculate our expected loss via the risk function:

$$\begin{aligned}\frac{1}{6}(b_1 - a_1)^2 + \frac{1}{6}(b_1 - a_1)^2 a_2^2 + \frac{1}{10}a_2^2 &= \frac{1}{6}((a_1 + \frac{3}{4}a_2) - a_1)^2 + \frac{1}{6}((a_1 + \frac{3}{4}a_2) - a_1)^2 a_2^2 + \frac{1}{10}a_2^2 \\ &= \frac{3}{32}a_2^2 - \frac{3}{16}a_2^2 + \frac{1}{10}a_2^2 = \frac{1}{160}a_2^2\end{aligned} \tag{3}$$

Using the definition of approximation error:

$$\textit{Approximation Error} = R(f^*_{\mathcal{H}}) - R(f^*) = R(f^*_{\mathcal{H}}) - 0 = R(f^*_{\mathcal{H}})$$

We have shown that approximation error will be equal to the following: $R(f^*_{\mathcal{H}}) = \frac{1}{160}a_2^2$. If it were the case that $a_2 = 0$ then approximation error would also be 0 as $\frac{1}{160}0^2 = 0$.

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:

$$X = \begin{bmatrix} 1 & x_1 & \cdots & x_1^d \\ 1 & x_2 & \cdots & x_2^d \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_N & \cdots & x_N \end{bmatrix}, \quad \boldsymbol{y} = [y_0, y_1, \cdots y_N]^\top. \tag{4}$$

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

   The empirical risk minimizer being $\hat{\boldsymbol{b}} = \arg\min_b \|Xb - \boldsymbol{y}\|_2^2$ follows directly from the definition of our loss function, as it is merely restated in the language of linear algebra.

   Let us first remember an important Lin Alg. concept: the Euclidean Norm. The Euclidean norm is defined as:

   $$||x||_2 = \sqrt{\langle x, x\rangle} = \sqrt{\sum_{i=1}^{n} x_i^2} \text{ where } x \in \mathbb{R}^n$$

   Squaring the norm simply removes the square root term outside of the summation. Using the definition of our empirical risk minimizer and its loss function:

   $$\begin{aligned} R(\hat{f}) &= E(l(\hat{y}, y)) \\ &= E((\frac{1}{2}(\hat{y} - y)^2) \\ &= \frac{1}{2}E((\hat{y} - y)^2) \\ &= \frac{1}{2 * N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 \end{aligned} \tag{5}$$

The data set of our prediction, $\hat{y}$, can be expressed as Xb, where $X \in \mathbb{R}^{N\times(d+1)}$ is our design matrix of our $d+1$ polynomial terms, $N \in \mathbb{R}$ is the number of observations, and $b \in \mathbb{R}^{d+1}$ is the vector that parameterizes our design matrix. We can use this representation in our formula above.

$$\begin{aligned}\frac{1}{2*N}\sum_{i=1}^{N}(\hat{y}-y_i)^2 &= \frac{1}{2*N}\sum_{i=1}^{N}(Xb_i-y_i)^2 \\ &= \frac{1}{2*N}\langle Xb-y, Xb-y\rangle \\ &= \frac{1}{2N}||Xb-y||_2^2\end{aligned} \tag{6}$$

Taking the argmin with respect to b yields the optimal coefficient weights for our polynomial terms, and thus the vector that parameterizes our polynomials to minimize loss, $\hat{b}$. We can drop the fraction in front of our equation as it will not affect the answer of $\hat{b}$. This yields the answer to our problem:

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

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 ?

   Firstly, we must use the condition that $N > d$ as when we generate a polynomial, it has $d+1$ terms. For example, when we consider a polynomial of degree 2, it has 3 terms: $\{a_0, a_1, a_2\}$. Due to Rank-Nullity, the Rank of a matrix (the dimension of its image) is defined as: $Rank(A) \leq min(m,n)$ where $A \in \mathbb{R}^{n\times m}$. Since our design matrix will have $d+1$ columns (n in this case), at minimum, for our our matrix to be full rank it must have at least $d+1$ rows, (m in this case) justifying the strict inequality $N > d$.

   Also, credit to Rank-Nullity, we also know that $Rank(A^T) = Rank(A)$, and that $Rank(AB) \leq min(m,n,r,k)$ where $A \in \mathbb{R}^{n\times m}$ $B \in \mathbb{R}^{r\times k}$. We also know that Therefore, for A^TA to be an invertible matrix, A must be full rank, as it therefore will not have a null space. We need the matrix to be full rank to be able to compute $(X^TX)^{-1}$.

   To arrive at our closed form solution for $\hat{b}$ we consider the following:

$$\begin{aligned}\hat{\boldsymbol{b}} &= \arg\min_b \|Xb-\boldsymbol{y}\|_2^2 \\ &= \arg\min_b \ \langle Xb-y, Xb-y\rangle \\ &= \arg\min_b \ ||Xb||^2 + ||y||^2 - 2\langle Xb, y\rangle \\ &= \arg\min_b \ b^TX^TXb + y^Ty - b^TX^Ty\end{aligned} \tag{7}$$

   Since X is full rank, the expression we arrive to is strongly, (and thus strictly), convex. To

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

compute the minimum, we can differentiation both sides with respect to $\hat{b}$ and set to 0.

$$\begin{aligned} 2X^TXb - 2Xy^T &= 0 \\ X^TXb &= X^Ty \\ b &= (X^TX)^{-1}X^Ty \\ \hat{\boldsymbol{b}} &= (X^\top X)^{-1}X^\top \boldsymbol{y} \ \square \end{aligned} \tag{8}$$

We have shown that the b that minimizes $||Xb - y||^2$ is $\hat{b} = (X^TX)^{-1}X^Ty$, assuming X when we assume X is full rank.

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).

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.

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 .

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?

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?

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?

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.