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^*.

   A Bayes predictor is a function that achieves the minimum risk among all possible functions.

   $\hat{y} = f^*(x) = b_0 + b_1 x + b_2 x^2$ where b = a

   Risk $\to E[l(\hat{y}, y)] = E[\frac{1}{2}(\hat{y} - y)^2] = \frac{1}{2}E[(b_0 + b_1 x + b_2 x^2 - (a_0 + a_1 x + a_2 x^2))^2] = 0$

   The risk of our function is 0 (lowest possible value), which follows the definition of a Bayes predictor.

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

   Given that $\mathcal{H}_2$ is our Hypothesis space, the Bayes predictor used in the previous question is included in it and therefore it is also a risk minimizer in $\mathcal{H}_2$.

   Approximation error consists on the risk difference between the Bayes predictor and the risk minimizer of the Hypothesis space. As mentioned previously, our Bayes predictor is in our Hypothesis space, thus the approximation error is 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 $f^*_{\mathcal{H}_d}$?

It is clear that $R(f^*_{\mathcal{H}_2}) \leq R(f^*_{\mathcal{H}_d})$ because $R(f^*_{\mathcal{H}_2})$ is already at a minimum (it is the Bayes predictor), and expanding our Hypothesis space to $\mathcal{H}_d$ will simply add additional complexity to the model. The equality would only happen when all coefficients $d > 2$ of the predictor in $\mathcal{H}_d$ are 0.

A function $f^*_{\mathcal{H}_d}$ in $\mathcal{H}_d$ would be a polynomial $f^*_{\mathcal{H}_d} = b_0 + b_1 x + \ldots + b_d x^d$ where b_i for $i > 2$ go to 0. Additionally, in Campuswire, there was a change in this question, precisely on the definition of the Hypotyhesis space $\mathcal{H}_d$. The comment mentioned that it should have been defined as "of degree at most d". If that is the case, our Bayes predictor $b_0 + b_1 x + b_2 x^2$ is still included in this Hypothesis space and therefore it would be a risk minimizer in $\mathcal{H}_d$ and its approximation error would also be 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?

$$R(f^*_{\mathcal{H}}) = E[l(\hat{y}, y)] = \int_{x=0}^{x=1} \frac{1}{2}(b_1 x - a_1 x - a_2 x^2)^2 dx = \frac{1}{6}(b_1 - a_1)^2 - \frac{1}{4}(b_1 - a_1)a_2 + \frac{1}{10}a_2^2$$

Now let's take the derivative with respect to b and set it equal to 0.

$$\frac{\partial}{\partial b} R(f^*_{\mathcal{H}}) = \frac{1}{3}(b_1 - a_1) - \frac{1}{4}a_2 = 0 \to b_1 = 3 * (\frac{1}{4}a_2 + \frac{1}{3}a_1) = \frac{3}{4}a_2 + a_1$$

The risk minimizer function $f^*_{\mathcal{H}}$ is therefore $f^*_{\mathcal{H}} = (\frac{3}{4}a_2 + a_1)x$

Approximation error is defined as the difference between the risk of the Bayes predictor and the risk of the risk minimizer function contained in our Hypothesis space. The risk of our Bayes predictor was calculated in Question 1, it is 0. Therefore the approximation error is the risk of our new risk minimizer $R(f^*_{\mathcal{H}})$.

$$\begin{aligned} R(f^*_{\mathcal{H}}) = & \\ & = E[l(\hat{y}, y)] = \frac{1}{2}E[(\hat{y} - y)^2] = \frac{1}{2}E[((\frac{3}{4}a_2 + a_1)x - (a_0 + a_1 x + a_2 x^2))^2] \\ & = \frac{3}{32}a_2^2 - \frac{2}{16}a_2^2 + \frac{1}{10}a_2^2 = \frac{1}{160}a_2^2 \end{aligned}$$

Furthermore, if $a_2 = 0$, then the risk is 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 $\hat{\boldsymbol{b}} = \arg\min_b \|Xb - \boldsymbol{y}\|_2^2$.

   Let's first substitute our definition for loss: $l(f^*(x_i), y_i) = \frac{1}{2}(Xb - y)^2$. Now, we need to find a b that minimizes the empirical risk of our function.

   $$\hat{b} = argmin \frac{1}{N} \sum_{i=1}^{N} \frac{1}{2}(Xb - y)^2$$

   Following the definition of the L_2 Norm, $\sqrt{\sum_{i=1}^{N} \frac{1}{2}(Xb - y)^2} = ||Xb - y||_2$

   Therefore,

   $$\hat{b} = argmin \frac{1}{2N} ||Xb - y||_2$$

   The constant $\frac{1}{2N}$ does not change the argmin, therefore:

   $$\hat{b} = argmin ||Xb - y||_2$$

   Geometrically, the empirical risk minimizer aims to minimize the distance between Xb and the solution vector y. The vector y is rarely in the span of X, so it tries to minimize the orthogonsl distance from the hyperplane spanned by Xb and the vector y, which is expressed as $Xb - y$. Additionally, taking the 2 norm of a vector represents its distance (represented in the graph as e).

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 ?

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

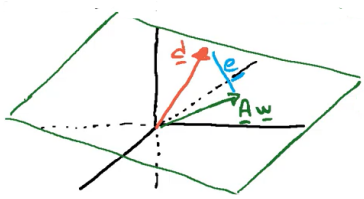

Figure 1: Linear Least Squares

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

$$\frac{\partial}{\partial b} = 2X^T Xb - 2X^T y = 0 \rightarrow X^T Xb = X^T y \rightarrow b = (X^T X)^{-1} X^T y$$

The matrix X has dimensions Nx(d+1), so it is necessary that $N > d$. If that is not the case, then the matrix would have more columns than rows, which means that it could have infinitely many solutions. Additionally, it is important that the matrix is full rank because if X is full rank, then $X^T X$ is also full rank and therefore it is invertible. If $X^T X$ was not full rank, we would not be able to solve the equation $\hat{b} = (X^T X)^{-1} X^T y$.

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.