Problem Set 1

February 27, 2023

The main goal of this initial problem set will be to implement ordinary least squares linear regression and a few of its variations which contain key themes related to further machine learning techniques that we will study later in this class.

1. Using the example code we reviewed in class or by creating your own implementation, write a Python function to generate a random dataset in two dimensions, e.g. a collection of points $\{(x_i, y_i)\}$ for $i = 1, \dots, n$. Specifically, use the numpy.random.rand() function to generate random x_i values and corresponding y_i values then use the matplotlib library to plot a scatterplot of your dataset.

2. Next fit the one dimensional linear model

$$f(x) = \beta_0 + \beta_1 x + \epsilon \tag{0.1}$$

   to the data by computing the two estimates for the model parameters

$$\hat{\beta}_1 = \frac{\text{cov}(x, y)}{\text{var}(x)} = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2} \tag{0.2}$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x} \tag{0.3}$$

   and then plot your line $f(x) = \hat{\beta}_0 + \hat{\beta}_1 x$ overlaid upon the previous scatter plot of data.

3. Next, we will compute parameter standard errors and performance metrics of your model. Specifically, first compute the standard errors of both model parameters

$$\text{Std}(\hat{\beta}_1)^2 = \frac{\sigma^2}{\sum_{i=1}^n (x_i - \bar{x})^2}, \quad \text{Std}(\hat{\beta}_0)^2 = \sigma^2 \left[\frac{1}{n} + \frac{\bar{x}^2}{\sum_{i=1}^n (x_i - \bar{x})^2}\right] \tag{0.4}$$

   where here $\sigma^2 = \text{Var}(\epsilon)$ is the residual variance with the data residuals being defined by $\epsilon_i = y_i - f(x_i)$. Then compute the associate t-stat of the slope parameter $t = \hat{\beta}_1 / \text{Std}(\hat{\beta}_1)$. Finally compute the residual standard error and the R-squared of the fit. Specifically, if we let $\hat{y}_i = f(x_i) = \hat{\beta}_0 + \hat{\beta}_1 x_i$, then the residual standard error is

$$\text{RSE} = \sqrt{\frac{1}{n-2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{0.5}$$

and the r-squared is

$$r^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{0.6}$$

4. Next use the ordinary least squares library functions either in the scikit-learn Python package we discussed in class, the statsmodels package, or other related Python libraries (scipy, pandas, etc.), to fit the same dataset and compare your estimates of the model parameters and performance metrics to the output of the library functions.

5. Next, we will implement an optimization based method to identify the best fit model parameters as opposed to directly estimate them from the data. In particular, consider the model error function

$$g(\beta_0, \beta_1) = \sum_{i=1}^{n} l(\epsilon_i) = \sum_{i=1}^{n}(y_i - f(x_i))^2 = \sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 x_i)^2 \tag{0.7}$$

   and use the Python function scipy.optimize.minimize to minimize $g(\beta_0, \beta_1)$ as a function of the two model parameters. How do the optimal points compare to the $\hat{\beta}_0$ and $\hat{\beta}_1$ estimates that you computed above? Next, extend your script to estimate these model parameters for other loss functions. In particular, consider the l_1 loss function that we describe in class $l(x) = |x|$ as well as define your own loss function. Overlay the best fit line with the $|x|$ loss function on the scatter plot of data and the mean squared error minimizing model that you previously displayed. Interpret the differences between these two models.