Lab Course Machine Learning

Exercise Sheet 2

Prof. Dr. Dr. Lars Schmidt-Thieme
Kiran Madhusudhanan

HiWi: Basharat Mubashir Ahmed
Sophia Damilola Lawal
Submission deadline : November 18, 2022

1 Linear Regression on Time Series Data (13 points)

1. **[2 points]** Time Series Exploration : Try to understand the *time series* dataset uploaded along with the exercise. This involves (but is not limited to) plotting the multivariate time series (time on x-axis) clearly labelled and formatted, understanding how the multivariate time series interacts, understanding the correlation between the different variables, plots for variable density functions, identifying inherent seasonality or trend etc.

2. **[1.5 points]** Train/Test split : As a next step, try to split the data into train, validation and test by considering the following rules

   a) train end : 2017-06-26

   b) val end : 2017-10-24

   c) test end : 2018-02-21

   Draw a single plot of the entire time series (for the target variable) and mark clearly within the plot, the train end, the val end and the test end.
   From now on, you are working with the train, test and validation dataset.

3. **[1.5 point]** Scaling data: For each of these datasets perform a standard scaling to scale each variable independently i.e.,

$$z = \frac{(x - \text{mean}(x))}{\text{std}(x)}$$

   Check if the standard scaling worked by computing the mean and standard deviation for a particular variable within the data.

4. **[6 points]** Regression : Next, implement learn a linear regression model using Normal Equations using the algorithm below.

   `learn-linreg-NormEq(Dtrain` $:= \{(x_1, y_1), \ldots, (x_N, y_N)\}$`) :`
   $X := (x_1, x_2, \ldots, x_N)^T$
   $y := (y_1, y_2, \ldots, y_N)^T$
   $A := X^T X$
   $b := X^T y$
   $\hat{\beta} :=$ `solve-SLE`(A, b)
   `return` $\hat{\beta}$

   For solving the Linear Equations (the function *solve-SLE*), use

   a) Gaussian Elimination (implemented in pure python i.e., only numpy)

   b) QR decomposition (implemented in pure python i.e., only numpy)

5. **[2 points]** Loss: Use the parameters ($\hat{\beta}$) learned to calculate the Mean Absolute Error (MAE) on train, validation and test split for the dataset. Additionally, plot the true target and the predicted target for the train, validation and test splits.

2 Gradient Descent and Step Length Controller (7 points)

In this part, you are required to optimize the *booth function* using gradient descent. The Figure below provides a visual representation of the *booth function* in a 3D plot.

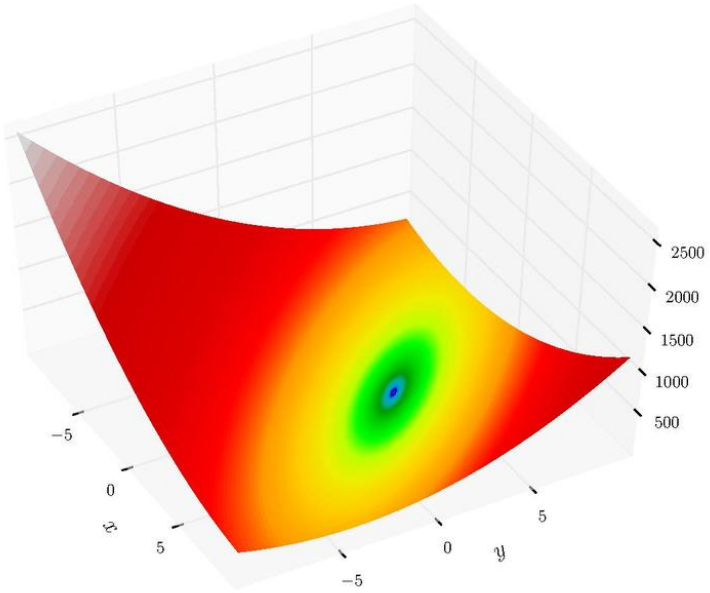

Figure 1: Booth function in a 3D plot

Mathematically, the function can be defined as follows.

$$f(x, y) = (x + 2y)^2 + (2x + y - 5)^2 \tag{1}$$

For this part of the exercise,

1. **[1 point]** Implement a 3D plot to visualize the function (Use Matplotlib's 3D utilities)
2. Derive the partial gradients.
3. **[3 point]** Optimize the function with Gradient Descent. Set the appropriate hyperparameters like initial values of (x,y) and the steplength α through trial and error.
4. **[1 point]** Visualize the trajectory on the same 3D plot. This trajectory should ideally lead to the function minimum. Try to plot the trajectory in a for loop so that the path taken is visible.
5. **[2 point]** Implement the function that controls the step length (*stepsize-backtracking* from the slides).