COMP 462 Assignment 5 Report
Regression with Perceptron

Remzi Orak

MEF University
Department of Electrical and Electronics Engineering
orakr@mef.edu.tr
041402011

Abstract. This report based on implementation of a perceptron for regression.

1 Datasets

In this assignment, we used three different dataset, one of which was given, two of which were generated by us. The datasets given in Figure 1. Second dataset generated by adding little noise to $x^3 + 150$ function when x values are between -3 and 10. Third dataset generated by adding little noise to $6 * x + 10$ function when x values are between -3 and 10.

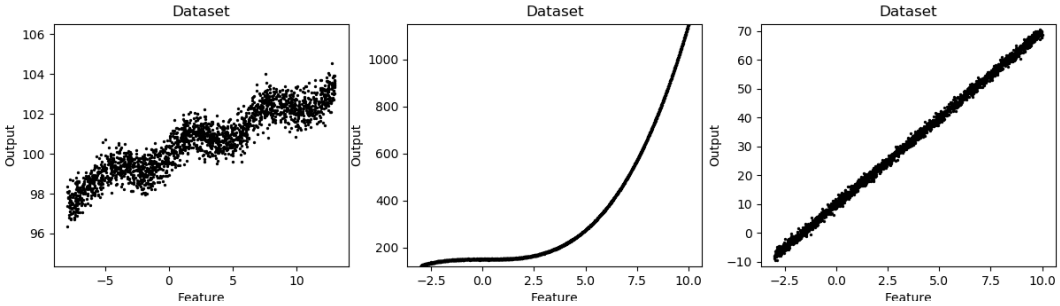


Fig. 1: Three datasets.

2 Stochastic Gradient Descent

In gradient descent, a batch is the total number of examples we use to calculate the gradient in one iteration. We assumed all example in dataset as one batch on standard gradient descent algorithm. When working on dataset contains billions

data example, a large batch may cause long times to compute weights. Therefore, we use a stochastic gradient descent algorithm which update weights for each sample in dataset. The algorithm steps given as Algorithm 1.

Algorithm 1 STOCHASTIC-GRADIENT-DESCENT

Each training example is a pair of the form $\mathbf{x}, t$ where $\mathbf{x}$ is the vector of input values, and t is the target output value. η is the learning rate

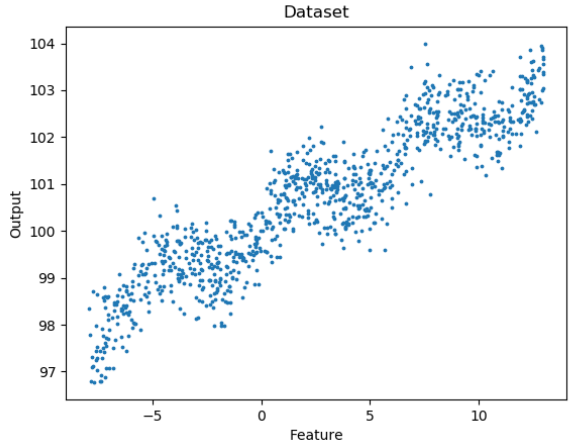
Initialize each w_i to some small random value
Until the termination condition is met, Do
    For each $\mathbf{x}, t$ in the training set, Do
        Input the instance x to the unit and compute the output o
        For each linear unit weight w_i, Do
            $w_i \leftarrow w_i + \eta$(t - o)$\mathrm{x}_i$

3 Regression Results

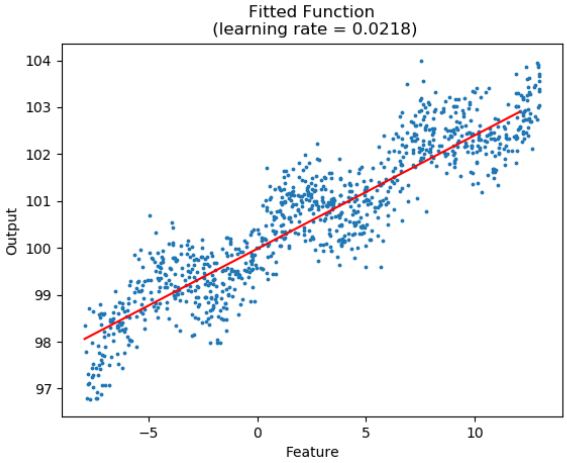
Results for given dataset given in Figure 2. Perceptron easily minimize the loss error and fit the datas. Figure 2.d show that perceptron fit the data better between 0.007 and 0.012 values of learning rate. For greater values of learning rates, error goes to infinity.

Results for second dataset which generated by us given in Figure 3. Second dataset generated by adding little noise to $x^3 + 150$ function when x values are between -3 and 10. Because of our generated data are not linear, the perceptron could not fit the data successfully. Figure 3.d show that loss increase after 0.06 values of leaning rate.

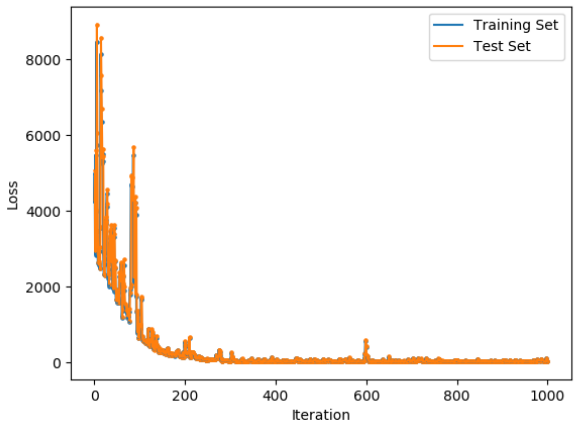
Results for third dataset which generated by us given in Figure 4. Second dataset generated by adding little noise to $6 * x + 10$ function when x values are between -3 and 10. Because of our generated data are crated from linear function, the perceptron fit the data successfully. Figure 4.d show that perceptron fit the data better between 0.001 and 0.04 values of learning rate. For greater values of learning rates perceptron could not fit the data and converges to infinity.



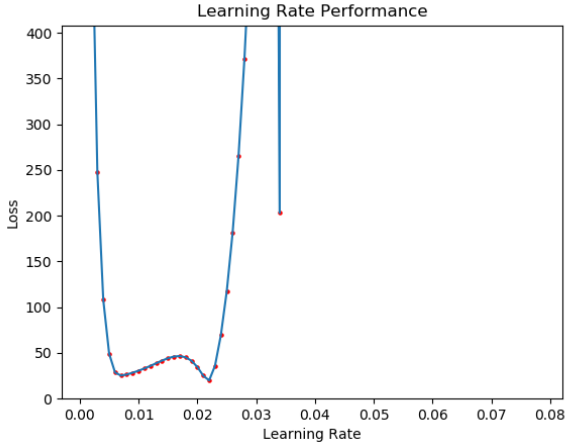
(a) First Dataset



(b) Dataset and a function found by the perceptron.

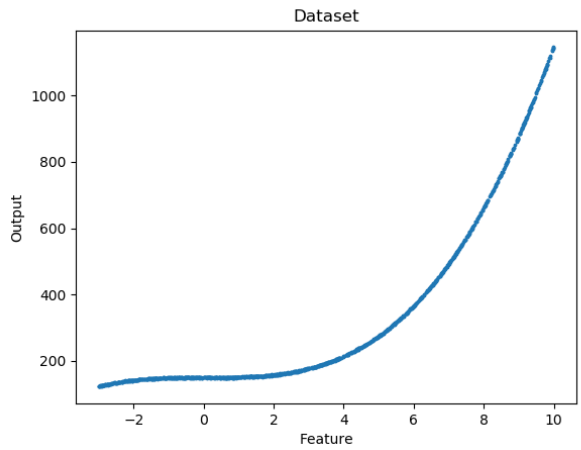


(c) Loss (error) vs iteration

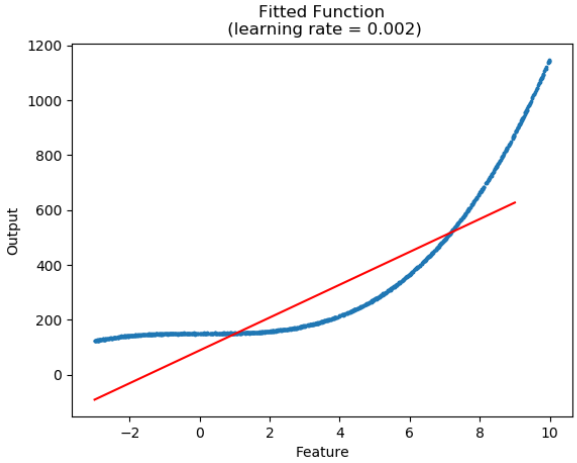


(d) Loss (error) vs learning rate

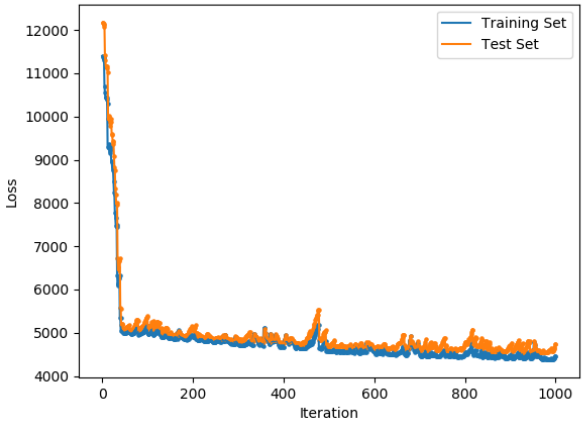
Fig. 2: Results for first dataset



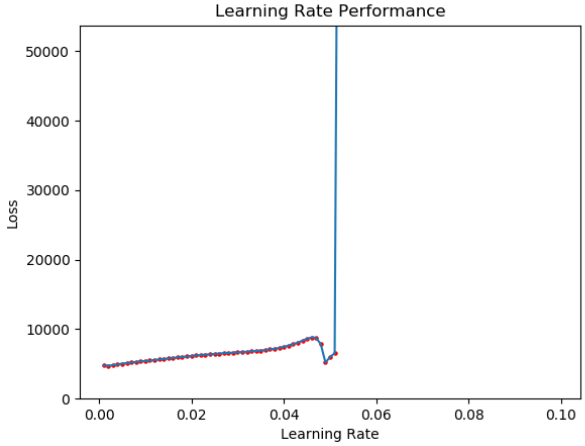
(a) Second Dataset



(b) Dataset and a function found by the perceptron.

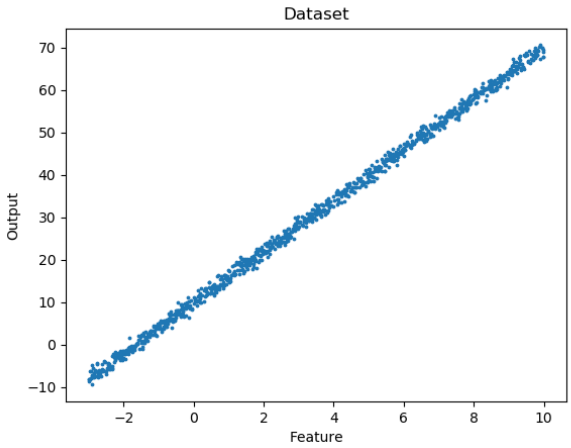


(c) Loss (error) vs iteration

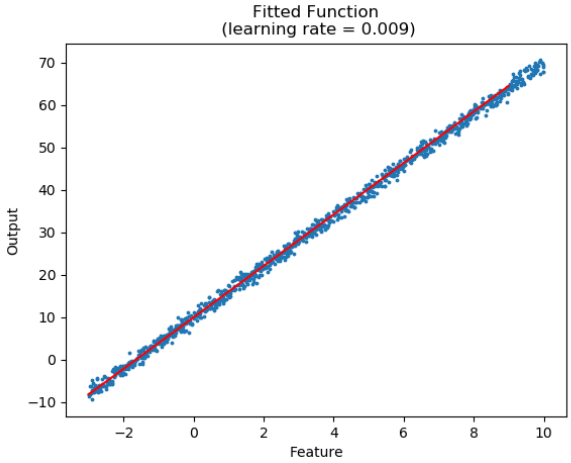


(d) Loss (error) vs learning rate

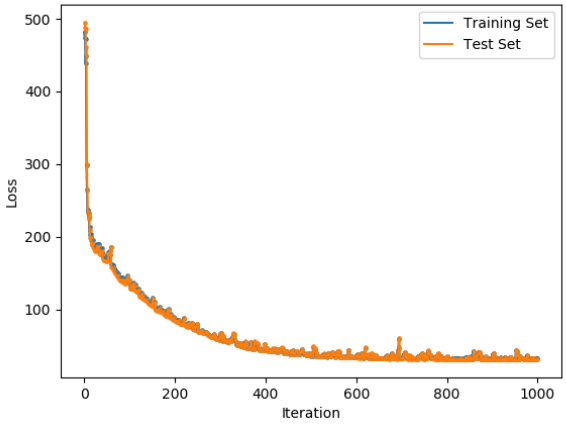
Fig. 3: Results for second dataset



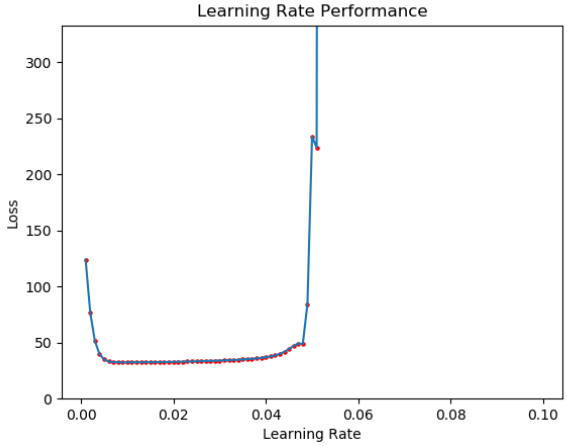
(a) Third Dataset



(b) Dataset and a function found by the perceptron.



(c) Loss (error) vs iteration



(d) Loss (error) vs learning rate

Fig. 4: Results for third dataset