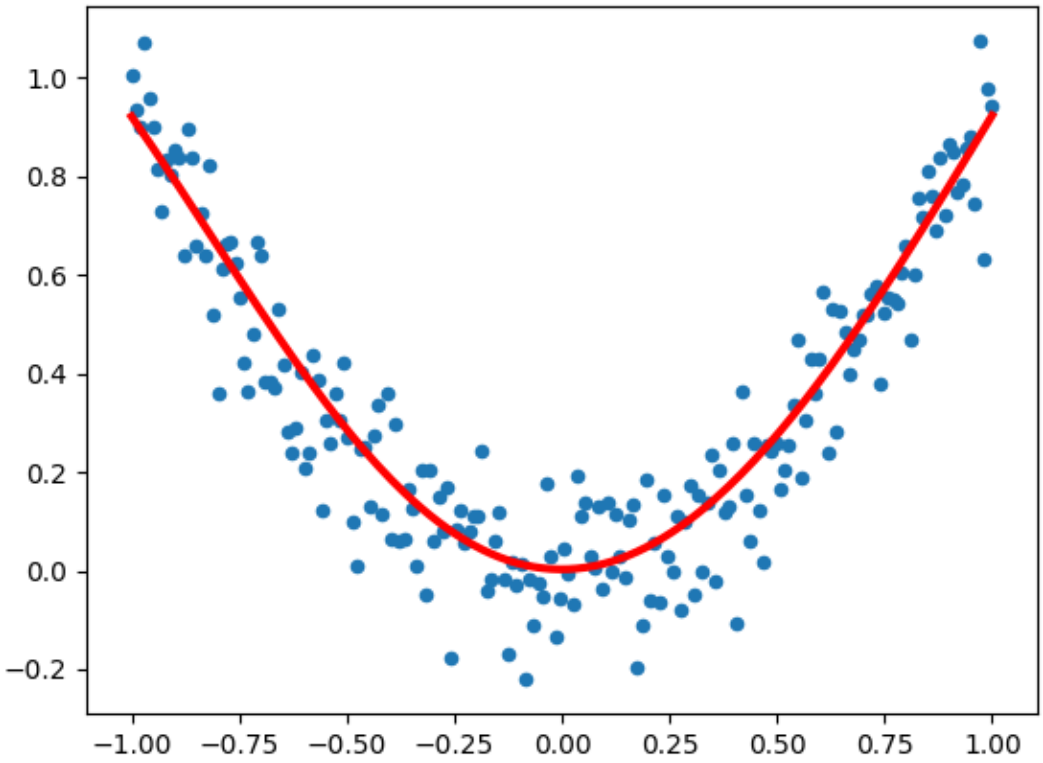
Report

In compute_activation of layers.py file, we compute the input data with weight matrix using dot product and add bias. In compute_gradient, we calculate dw (derivative of weight matrix)by doing dot product of input_data and output_error_gradient, calculate db (derivative of bias matrix) by applying dot product of output_error_gradient and matrix of 1 with row equals column of output_error_gradient and column equals 1, and lastly calculate input_error_gradient by using dot product of output_error_gradient and weight matrix. In update_weight, we update weight and bias by taking original matrix of weight and bias minus its derivative multiply by learning_rate respectively.

In neural_network.py file, to compute output we first compute_activation value of x for the first layer and use the result of it for the next layer recursively. Loss is genereated by using compute_acivation with output got from above and given target as 2 inputs. For compute_gradient, first we have to compute_gradient of loss and use that result to set output error gradient for each layer in reversed order. Computing gradient in that layer and pass it to previous layer recursively until all layers are finished. In update_weight, we update_weight for each layer using given learning_rate as well as update_weight for loss.

As we have trained the model toy_example_regressor.py, we obtain validation loss equals to 0.0112853… After we trained the model prime_classifier.py, we obtain validation loss of 0.02454448 and validation accuracy percentage of 97.12.


1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00