DeepLearning Lab **2022/2023**

Student: LUCA PERNIGO StudentID:19-993-658

Assignment 1

Due date: 23 October 2022, 10:00 PM

1. Polynominal Regression

1.1.

The model analyzed in this report is a fourth degree polynomial and its plot is reportend in Figure 1. The parameters of weights for the model considered in this assignment are w=[0, 5, 2, 1, 0.05]. To visualize the polynomial the function plot polynomial is called with w as the coefficient input and z as the range of the plot figure.

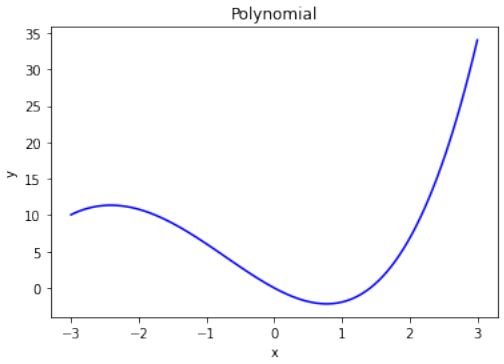

Figure 1: Polynomial visualization

1.2.

In order to create the dataset the create_dataset function has been used. The core part of this function creates the vector $[x^0, x^1, x^2, x^3, x^4]$ for every point in a certain sample and computes the corresponding $y = w^T x$.
See code for its implementation.

1.3.

Calling the create_dataset function, the training and the validation data points have been generated. For both the training and the validation set, 500 observations of x_i and their corresponding y_i have been created; with $y_i = w^T x_i + \text{normal\_random\_noise}$.
Note the seed=0 for the training set and seed=1 for the validation set.

1.4.

Here Figure 2 visualizes the training set while Figure 3 the validation set

Training and Validation sets visualized

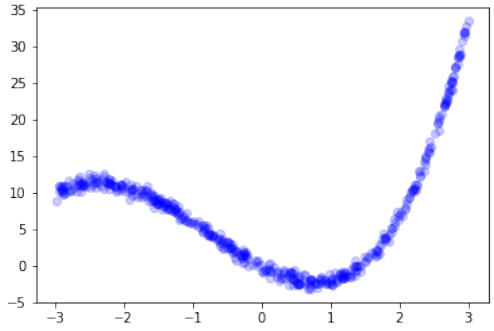

Figure 2: Training set visualization

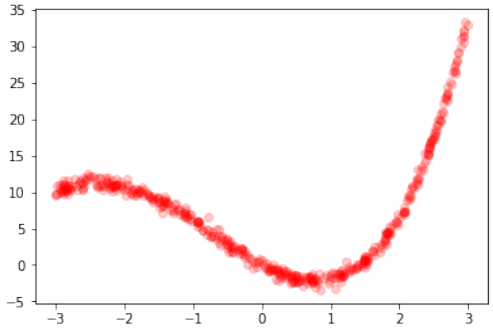

Figure 3: Validation set visualization

1.5.

The function torch.nn.Linear applies an affine transformation to the incoming data. That is it computes $y = xA^T + b$. If not specified in the input arguments, an additive bias b will be added; this is the default behaviour of torch.nn.Linear. Instead, if the bias argument is set to False, no additive bias will be added, hence $y = xA^T + b$ is computed.
In this problem, the goal is to retrieve the coefficients w to learn the model polynomial $p(x) = 0.05x^4 + z^3 + 2x^2 - 5x$, there is no interest in learning the bias. Therefore, the bias flag should be set to False.

1.6.

Next, Polynomial Regression on the training dataset has been performed.
See code for its implementation,

1.7.

After a bit of experimentation the following hyperparameters have been selected:

Learning rate=0.00122

Number of iterations=1600

Table 1: Initial vs Final coefficients

Coefficients	Initial	Final	Real
w_0	-0.2932	0.2576	0
w_1	-0.2378	-4.2487	-5
w_2	-0.2200	1.8475	2
w_3	0.2075	0.8845	1
w_4	-0.3028	0.0663	0.05

1.8.

Hereafter, the evolution of both the training loss(Figure 4) and the validation loss(Figure 5) are visualized singurarly and jointly(Figure 6).

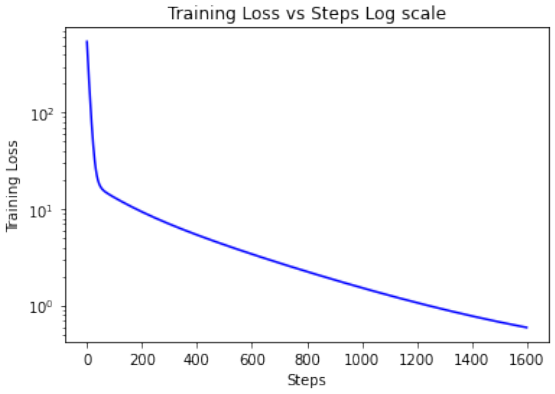

Figure 4: Training Loss over Steps

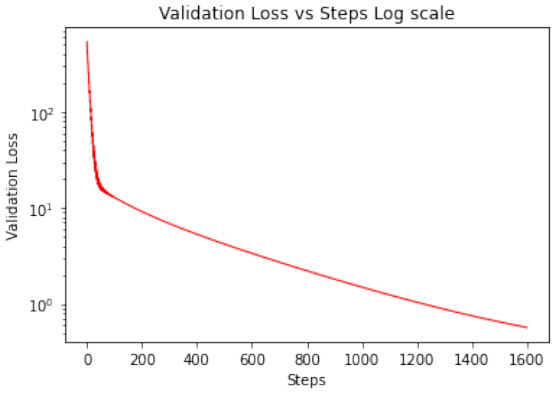

Figure 5: Validation Loss over Steps

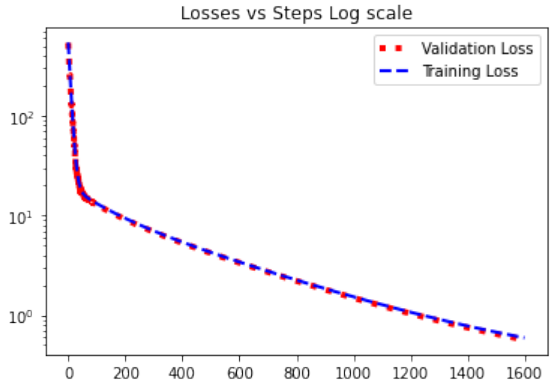

Figure 6: Training and Validation Loss

1.9.

In this subsection, the polynomial defined by the estimate of w is reported (Figure 7).

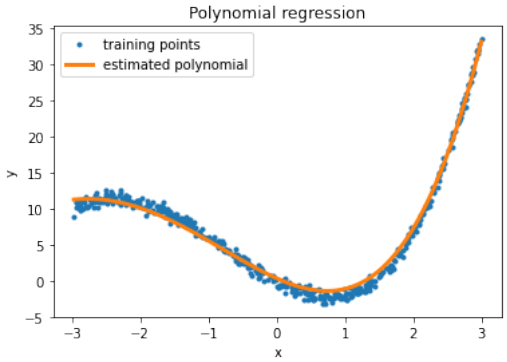

Figure 7: Esimated Polymial

From the picture it is clear that the estimate of w does a good job at approximating the real data generating process.

1.10.

If the training dataset is reduced to 10 observations while keeping the size of the validation dataset unchanged, the training loss decreases over steps while validation loss increases; this is a clear sign of overfitting. To see that, take a look at the following pictures depicting the evolution of the training loss (Figure 8) and the validation loss (Figure 9).

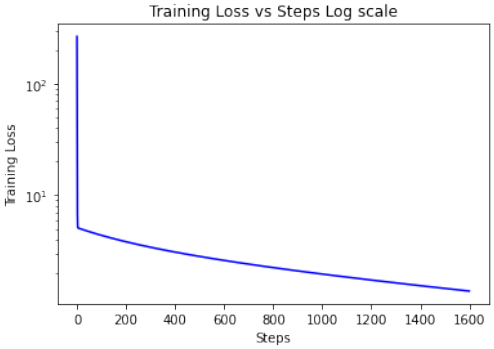

Figure 8: Training Loss over Steps with 10 traning data points

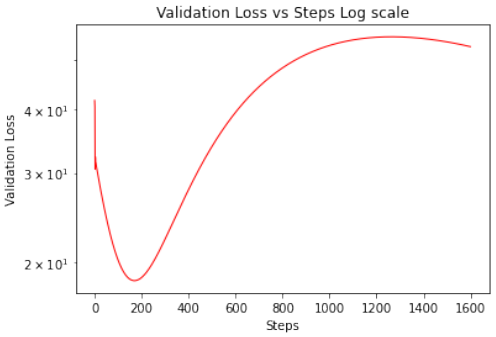

Figure 9: Validation Loss over Steps with 10 traning data points

1.11.

To conclude the analysis of the polynomial regression, the evolution of each w_i coefficient over the gradient descent iterations is reported (Figure 10).

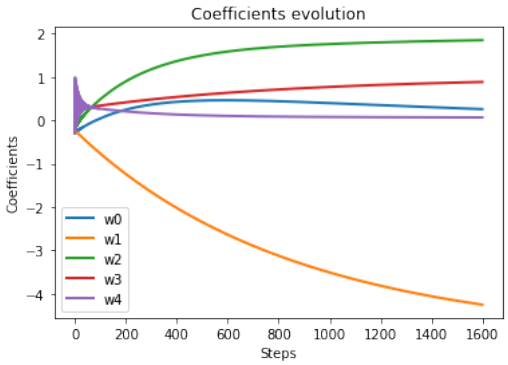

Figure 10: Coefficients evolution

Note the asymptotic behaviour of each coefficient in tending towards its true value.

2. Questions

2.1.

The model is overfitting when empirical risk (training data) is low but true risk (testing data) is high. This follow from the fact that when you are overfitting your model is heavily tuned to the noise of the training data.

2.2.

In order to mitigate overfitting, one possible solution is to include regularization into our model. This procedure involves a penalty factor on the magnitude of the model coefficients; doing so, too complicated models are disregarded and overfitting is avoided.