Deep Learning Lab: Assignment 1

Jury Andrea D'Onofrio

October 2022

1 Polynomial Regression

1.1

Using the given method *plot_polynomial*, I have obtained the plot in Figure 1. To do that, I have set the following parameters:

- *coeffs* $\rightarrow$ a numpy array containing coefficients of a polynomial in the increasing order of degrees.
- *z_range* $\rightarrow$ a list of two elements representing the range of z (min, max).

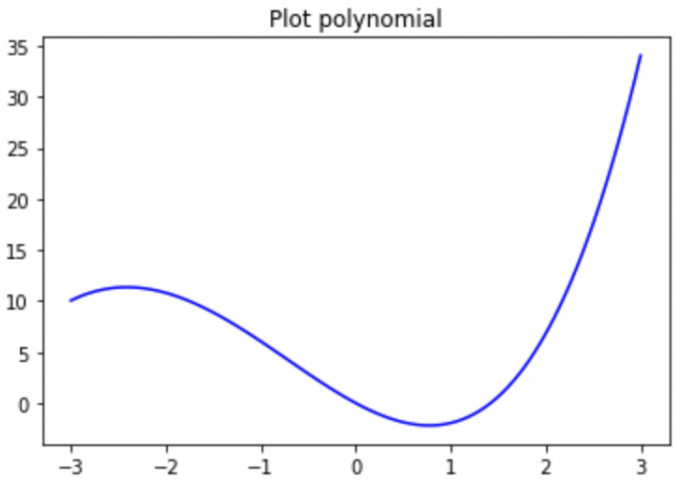

Figure 1: Polynomial using the given method

1.2

I have completed the given method *create_dataset*, in order to return correctly x and y that will become the training and the validation set. x must have as

many rows as sample size, and as many column as the shape of the coefficients vector w. Therefore, the second cycle has to be in the range of w. Each row of the matrix x has to contain the same value of z elevated to an exponent in the range of 0 to 4.

1.3

Using the given parameters, I have created the training and validation sets using the method discussed in section 1.2 whose plots are visible in Figure 2 and Figure 3 in section 1.4.

1.4

The plots obtained using the training and validation test are shown below in Figure 2 and Figure 3. Note that, both of the functions are close to the representation of the polynomial of Figure 1.

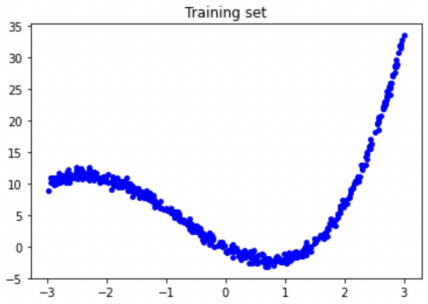

Figure 2: Training set

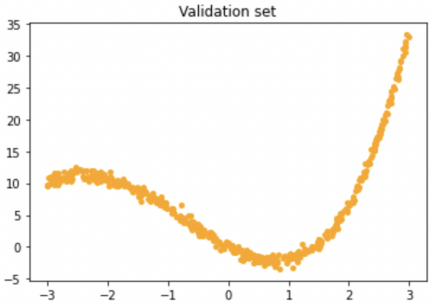

Figure 3: Validation test

1.5

A polynomial $p(z) : \mathbb{R} \to \mathbb{R}$, has the following representation form:

$$p(z) = w_n z^n + w_{n-1} z^{n-1} + ... + w_1 z + w_0 = \sum_{i=0}^{N} w_i z^i$$

where N is the degree of the polynomial.
Since the given polynomial is

$$p(z) = 0.05z^4 + z^3 + 2z^2 - 5z = \sum_{i=0}^{4} w_i z^i$$

it means that w_0 is equals to 0. Hence, the bias of the linear regression model has to be set to false.

1.6

The linear regression code in case 1D, presented in class, has been correctly adapted. The goal is to perform a polynomial linear regression using datasets generated earlier in the section 1.2. The portion of the code, that performs this action, starts from row 67 and I have used the *Linear* class of PyTorch to apply a linear transformation of D'.

1.7

I have done tests and I have found with these hyper parameters a loss function less than 0.6.

- A suitable learning rate is 0.0012.
- A suitable number of epochs is 1550.
- Initial (random) values of $w = [0.25458243, 0.00093084, 0.16313444, -0.21465726, 0.81899738]$.
- Estimate of $w = [0.39906743, -4.12707424, 1.76341951, 0.86563623, 0.07530338]$.

1.8

The plot obtained from the training and the validation loss as a function of the gradient descent iterations is in Figure 4. As visible, the two functions start with an high error and, as iterations pass, they become very close to 0.6.

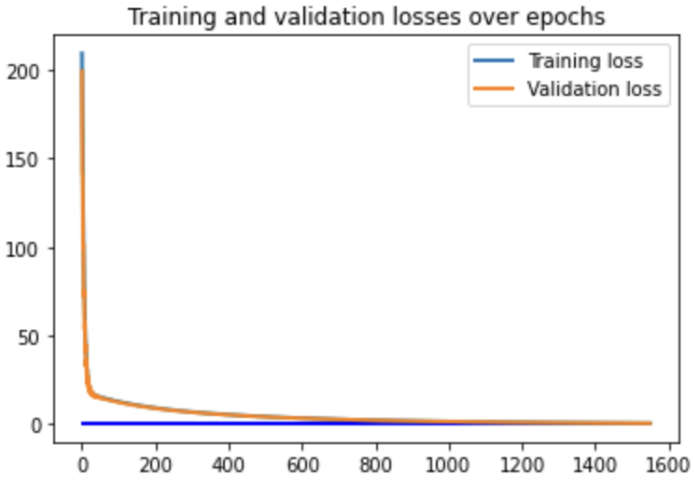

Figure 4: Training and validation loss over epochs

1.9

The polynomial defined by the estimate of w is shown in Figure 5. Note that the predictions made by the model follow the training set.

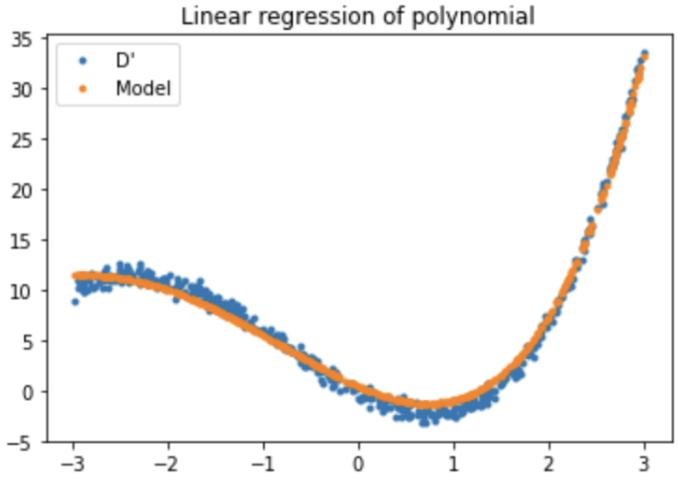

Figure 5: Polynomial defined by the estimate of w

1.10

As Figure 6 shows, the model trains only the points in the training set; in this way, the data structure is not generalized. In this case, the model has too few points with respect to the complexity of the problem. Therefore, the model overfits and will yield wrong predictions for the new data points. The plot in Figure 7 confirms this idea because the validation loss doesn't decrease with the passage of the iterations.

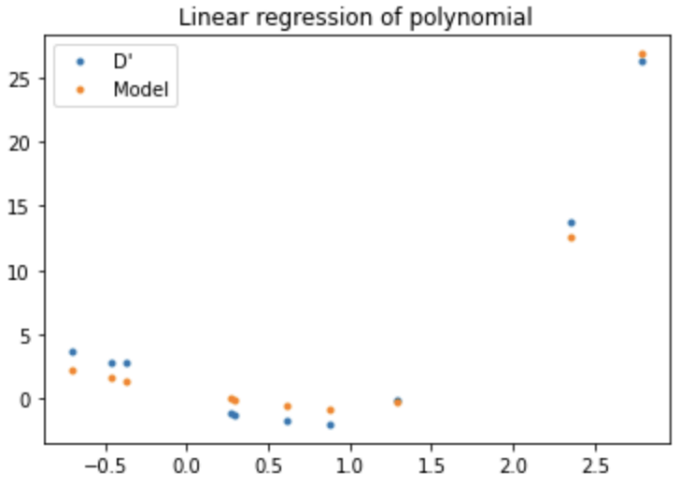

Figure 6: Polynomial defined by the estimate of w

Figure 7: Polynomial defined by the estimate of w

1.11

The evolution of each coefficient of w as a function of the gradient descent iterations is in Figure 8. As visible, the parameters are in the range of [0, 2] except for w_1 that assumes a value near to -4 as iterations pass.

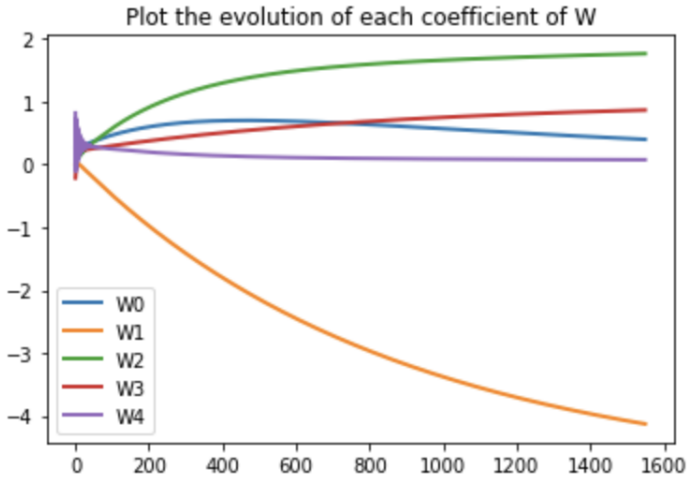

Figure 8: Evolution of each coefficient of w over epochs

2 Questions

2.1

The overfitting occurs when the model doesn't discovery a good structure of the data but it exactly fits the all training set. This means that the model won't predict a reasonable target for a new data point by yielding bad predictions.

2.2

The easiest way to mitigate the overfitting is increase the sample of the training set. For example, it is possible to apply a noise to the sample to create new values for the training set.