Homework 4 - MLP for Regression

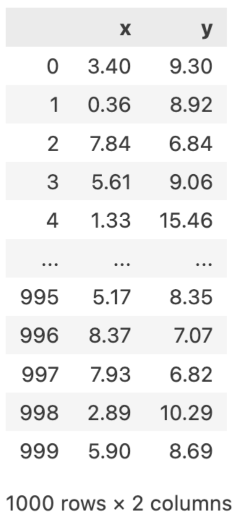
Robert Mateescu

1 Introduction

In this assignment, you will implement a Multi-Layer Perceptron (MLP) for Regression. In class we went over an MLP for XOR data, which was a classification task. You will modify that code to make it work for non-linear regression.

2 Data

The data is given in the file 'data.csv', and it looks like in Figure 1. There are 1,000 data points, where column x is the independent (or predictor) variable, and column y is the dependent (or response) variable. We want to learn a function that maps x to y, based on these 1,000 training examples.



	x	y
0	3.40	9.30
1	0.36	8.92
2	7.84	6.84
3	5.61	9.06
4	1.33	15.46
...	...	...
995	5.17	8.35
996	8.37	7.07
997	7.93	6.82
998	2.89	10.29
999	5.90	8.69

1000 rows × 2 columns

Figure 1: Data

If we used a linear regression on this data, we would get the line shown in Figure 2. When you use an MLP with hidden layers and non-linear activations, you will get a non-linear fit to the data.

3 MLP Class

For regression, you do not need the Softmax and Argmax in your MLP, but will rather collect a single value (meant to be the continuous output of the non-linear function you are modeling). When you write the MLP class, use three **torch.nn.Linear** layers. The first two are hidden layers and the third one is the output

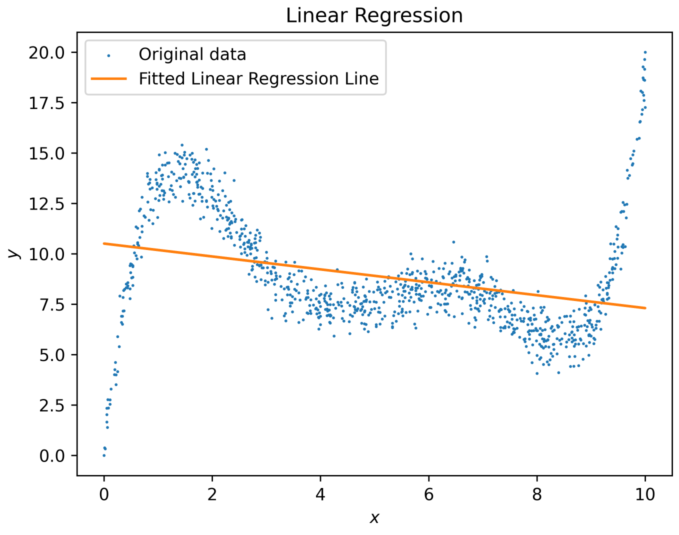


Figure 2: Linear Regression

layer. You can start with a small number of neurons in the hidden layers (e.g., 50, 25.) The input to the first layer should have the dimension equal to the number of features (which is 1 in our case, corresponding to the dimension of x), and the output of the last layer should be 1-dimensional (corresponding to the dimension of y).
After you load the data into PyTorch tensors, you may end up with X_train having a shape:

```
X_train.shape
torch.Size([1000])
```

You may find it useful to reshape the tensor with something like:

```
X_train = X_train.view(-1,1)
```

so that it plays well with some of the torch.nn layers.

4 Data Normalization

In the XOR example we did not normalize the data, because it was already centered at zero, and had relatively small values. We could have used normalization for XOR too.

In the case of regression, you are asked to normalize the data first, by doing z-score standardization. Find x_mean, x_std, y_mean, y_std for the data and then normalize the data. You will use x_mean and x_std to make predictions at the end, when you plot the regression curve.

5 Loss Function

We do not use the cross-entropy loss for this regression task. The most common used loss function for regression is the Mean Squared Error. In the definition below, $y^{[i]}$ is the true output for the i-th data point, while $\hat{y}^{[i]}$ is the output of the MLP model (the equation uses actual target minus predicted target).

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y^{[i]} - \hat{y}^{[i]})^2$$

Make sure to change the loss function to F.mse_loss() in the training loop.

6 Dataset and Dataloader

The Dataset class is almost the same. Use tensors for X and y. You will only use a train_loader, no need to split into validation and test sets.

7 Evaluate Model and Plot Regression Curve

After you train your model, plot the regression curve. Generate points as x values, for example:

X_range = torch.arange(X_train.min(), X_train.max(), 0.01).view(-1, 1)

Normalize this evaluation data using the original mean and std from the training data. Make predictions using the MLP. Then un-normalize the target values that you obtained, using the original y_std and y_mean. You are now ready to plot the regression curve.

8 Training Loss Plot

Show a plot of the training loss (MSE) as a function of epoch.

9 Hyper-Parameter Search

You are encouraged to explore various combinations of hyper-parameters, including:

- batch size (e.g., 5, 10, 20, 50, 100, 200 etc.)
- number of epochs (e.g., 10, .., 50, ..., 100, ..., 1000 etc.)
- learning rate (e.g., 0.1, 0.01, 0.001 etc.)
- activation functions (e.g., ReLU, sigmoid etc.)
- optimizer (e.g., SGD, Adam etc.)
- number of layers
- number of neurons per layer

Pick the best combinations that you could find and include the corresponding plots in your report. Include plots for a second combination of hyper-parameters and explain the differences.

10 Grading Rubric

These instructions are intended to help you organize your work. The grading is going to be done according to the Rubric on Canvas. Please check the Rubric for the detailed allocation of the points.