ML-idea, grades predicted based off of hours studied

Luca Busetti

Introduction to the idea

I aim to predict the grades of various students — an unknown variable — based on the number of hours they studied, which is known. To demonstrate the performance of my model, I will plot the predicted grades as a linear regression line on a graph. At the same time, I will plot the actual grades obtained by the students as scatter points. The objective is to minimize the prediction error, thereby making the model as accurate as possible in estimating students' grades.

About the linear regression

The method of the linear regression is based on the idea of the prediction of an unknown variable using one or more (in this case one) known variables. The core statistical assumptions about the linear regression are the following:

1. linearity (x and y are linearly related)
2. independence of the residuals
3. constant variance of the residuals
4. normal distributions of the residuals

The assumptions about the constant variance of the residuals is important because we can therefore tell that there are not parts of the data which are modelled better than others, because the residuals do not get randomly further/less spread out as the data increases/decreases. Whereas the second assumption about the linear regressions allows us to condlude that the errors are normally distributed and therefore big errors are more unlikely, which makes our model more credible. In our graph the horizontal axis represents the grade gotten whereas the vertical axis represents the hours studied. We are assuming a straight-line relationship between the two variables (I opted for the relationship to be straight-line for the sake of simplicity. Furthermore linearity has an important property: it is easier to interpret and therefore enables us to better comprehend the relationship between hours of study and grades). In order to

ensure that the model predicts the grades as well as possible, we need to ensure that the difference between predicted grades and actual grades is as small as possible. We are therefore aiming at minimizing the error. The slope and the intercept are playing a crucial role in minimizing the error. The intercept predicts the expected grade when the hours of study are zero whereas the slope predicts by how much the grade will increase as the hours of study increase. Basically it tries to model the influence of the hours of study on the grade. We therefore want the model to learn the best fitting slope and vertical intercept so that it can properly predict the grades minimizing the error.

Procedure

1. **Loading and preparing the data**

   The data is loaded from the file `Grades_Data.csv` and is processed using Pandas, which is a Python library that works really well with data in tabular form. Pandas can be used to read the data but also to clean the data, which can be seen in the part where I instructed the program to assign `NaN` to every non-numeric element of the data. Pandas also allows us to select certain parts of the data, which comes in handy when we have to separate the inputs from the outputs.

2. **Preparation of the data for PyTorch**

   We use PyTorch as it is a useful Machine Learning framework which allows us to build neural networks such as our linear regression line for the prediction. However, we have to say that PyTorch can only read data converted to Tensors. Therefore we need to convert all the data accordingly. We extract, using also Pandas, the inputs (x), and the outputs (y). The inputs have to be two-dimensional, whereas the outputs can also be one-dimensional.

3. **Building the model with PyTorch**

   We proceed with the buidling of the linear regressional model. We use the following formula:

   $$\mathrm{y} = w \cdot \mathrm{x} + b$$

   where w stands for weight and b for bias. We are trying to fit a straight line through the data.

4. **Setting up the loss function and the optimizer**

   We need to take into account that the model makes predictions based on it's current parameters which might be completely wrong. Therefore, we need to set up a loss functions which measures the errors and an optimizer which allows us to find a remedy to those errors. We use the mean squared error (MSE) as a loss function, as it computes the average difference between the observed data and the actual data, to the power of two.

The optimizer reduces the error (loss) measured through the loss function. We choose Adam as optimizer, so that it can, at every iteration move the parameters so that the error is smaller at every iteration. In fact we need to iterate this procedure many times so that the optimizer gets as efficient as possible and the loss is as minimized as possible. The more iterarions the better the predictions can be calculated, the more information about possible errors collected and as a consequence the better the optimizer is programmed.

5. **Training of the model**

   We loop 100 times, where in each loop gradients of the loss are computed, with respect to the weight and the bias, using backward, which are then erased at every iteration, as the model needs to find new information. As the model now has the necessary information about how the loss changes with the current weight and bias, it uses the optimizer to minimize the loss. As already said, the more iterations done, the better information can be collected and the better we can use that information.

Execution

First try

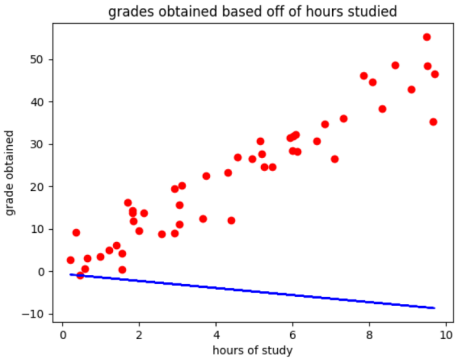

Figure 1: Try n1

As can be seen from the figure, I was not able to construct a model which properly predicts the grades. In fact, the weight and the bias have been computed completely wrong. I came to the conclusion that this was due to the fact that the model was learning a negative weight. I attributed this to the fact that the model was trained on raw values and not normalized. This leads the model to struggle if x and y have very different ranges, which in turn induces the model to choose a shortcut which might mathematically speaking reduce the error but is complete nonsense intuitively and conceptually speaking.

Second try

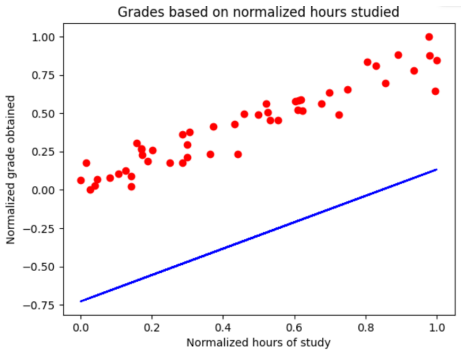

Figure 2: Try n2

In this case I was able to correct the slope of the linear regression line, such that the line is now positively sloped. However, the bias is still incorrect. I concluded that this was to be attributed to a bad normalization procedure. In fact I normalized the data when it came down to train the model, through min/max normalization procedure, but I did not "denormalize" the data when it came down to predict the values. This confuses the model as the predictions (plotted with the regression line) are now sigificantly lower with respect to the actually obtained values (the scattered points).

Try n3

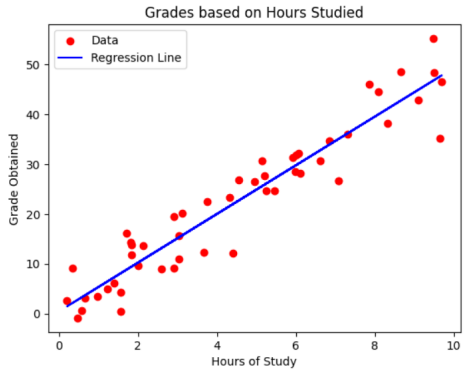

Figure 3: Try n3

I was finally able to adjust the previously discussed effect of bad normalization. When it came down to plot the predictions I used predictions in original size. For the future I know now that PyTorch works better with normalized data but in order to properly use MathPlotLib we have to revert the data back to original size. Furthermore for the sake of perfectionism I added a label to the regression trendline and also to the scattered points.

Evaluation

In order to evaluate the performance of the model and how exactly it predicts the grades based off of the hours studied. As a metric I chose the determination score as it is the tool I am most familiar with, given my statistics knowledge. The determination score is computed with the following formula:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

The determination score of this model is 0.9044698476791382, which highligts its high trustwortiness and efficiency.

Conclusions

I was able to construct a model which succesfully predicts in an efficient way the correlation between hours studied and grades obtained. I was able to recognize the importance of normalized data, as otherwise the model might wrongfully detect and process them which could be seen from the negatively sloped linear regression line. To adress this issue I scaled the data down to the [0,1] range so that PyTorch and the optimizer Adam could work better with them. To also

plot them realistically, I had to scale the results back to original. In the end I was able to construct a model which, as can be concluded intuitively, depicts that a higher study time, typically, leads to higher grades. As a whole it can be concluded that the project shows how well machine learning framweworks like PyTorch can be used for predictive analysis when combined with visualization tools such as Matplotlib.