Project X Experimentation with neural network mathematics

The aim of this report is, to understand the math behind neural networks a little bit better. After reading the slides about the basics and the process of one epoch again, the most interesting part for me is the influence on the learning rate, since I realized in the practical exercises, that it makes a difference, how big the LR-Value is.

LR and Epochs

In the original notebook, the LR-Rate was **0.025**. I changed the LR to the value I used in one of the practical task, which is **0.001**. When I looked at the loss per epoch, the losses where significantly smaller. At 0.025, the loss at Epoch 1 was ~84 and ~34 in epoch 2. For the last 1000 epochs, the loss stayed the same. When I used a LR at 0.001, the loss started 39 at Epoch 1, ~4 at epoch 2 and even goes down to ~2.083 at Epoch 1000. So should the value as small as possible? I tried out the values 0.0001 and 0.00001. For LR = 0.0001 the value started at 170 and goes down quite quickly. At epoch 12, the loss value was already below 4 and from there it did not change much. At Epoch 1000, it was ~3.482. When I defined the LR as 0.00001, the loss value also at 170 but it decreases a lot slower. At Epoch 25, it was still 70. At epoch 1000, the value was 3.405, so even though it learns slower, the last loss value is about the same. After this, I was wondering, when the LR gets so low, that the loss at the last epoch is a lot worse. I defined the LR to 0.000001 and now, it starts at 173 and at Epoch 1000 the loss is still 167. This experiment makes me realize, that a higher LR has the biggest loss, but if the LR is too small, the neural network learns too slow. This makes me think, that if I use a very low LR, but increase the epochs, a very small LR should not be a problem, since it has more possibilities to improve. I used 0.0000001as a LR again and changed the epochs from 1000 to 1 000 000. This approach took a few minutes, but I could already see at Epoch 100 000 that the loss stayed at ~3.384. and did not get better. So, adding epochs only makes sense to a certain extent because there is a point, where the model does not learn more. I still wanted to find out, if there is a value for the learning rate, where I could get a loss that is even less than with the learning rate of 0.001. With a LR of 0.0004, I got a loss at Epoch 1000 of ~1.95. A LR of 0.00039 would raise the loss value again. It is important to note that the effect of the LR depends on the data but since I got the best results with a LR of ~0.0001 in my practical exercises as well, I might start with a LR of 0.0001-0.0005 in my future projects and see how this plays out.

Weights and Biases

The next topic, I was interested in was the weights and biases. In the provided notebook and presentation, it was mentioned that the values of weights and biases are usually generated randomly. Now my question would be: How big is the impact of these values on the performance of the model? If the impact is quite high, the quality of the model would be different for every randomization. To test this, I would just define different values for the bias first. Since the values in the notebook are between -1 and 1, in this case bias1 = 1, bias2 = -0.35 and bias3 = 0.5, I assume the values would be generated in a similar category. I am still interested, how the model would react when I put bias values that are much bigger than the current ones. For bias1, bias2 and bias3, I defined 100 as a new value. When I ran the code, the rise raised from 1.94 to 33.4. This shows that there is a clear impact, if your values of biases are too high. If we start to look at the new weights and biases, the new values differ a lot from the new values from the starting values before. As an example, if I use bias1, bias2, bias3 = 100, the highest weight is 295 while the smallest is -582. When I used 1, -0.35 and 0.5 as biases, weights the lowest is -0.23 and the highest 2.12. The difference between new biases is not as high, new biases before had a value between -3.3 and 1.67 while the new biases choosing bias1, bias2, bias3 = 100 are between 2.8 and 12.36. It makes sense, that the weights differ a lot more than the biases because the weights have to scale the bias values, so they are similar in the end. I think if the value of the weights are no direct indicator on how effective the model is, since it really depends on the bias value. If the bias value is too large, it makes sense that weights are higher as well to bring them down to the correct size.

The next issue I was interested in was, if the model gets better if the model gets better if I increase the weights as well. It turns out: Not at all. The new weights and biases are larger than before. The loss however stayed at ~33. What I also had a look at was the prediction at the end of the notebook. When I used the given starting values with the best LR I found before, the prediction was ~9.54. When I changed all the weights and biases, the prediction was ~12, which is a significant change since the random value, that is added to the equation is between 0 and 5.

After realizing, that biases and weights should have values at around 0, at least in this example, I am still wondering if you could just use the same value for each weight or bias. Since the numbers are randomized in practical examples, it should not make a huge difference, if for example I put every weight at 0.5 and every bias as 1. The result was that the loss increased from 1.94 to 4.11 which is not huge. The prediction is ~9.59, which is basically the same as before. Next, I tried if the loss or the model gets better when I use negative and positive values. For w2, w4, w6 and bias2 I changed the value to a negative value (from 0.5 to -0.5). The numbers did barely change prediction and loss were basically the same. I thought that the weights and biases being different is slightly better for the model.

From there, I asked myself the question what would happen, if every weight and bias is 0. I thought that the model might not be able to learn at all. I tried it out and the loss at epoch was ~33.42, so way higher. The prediction was also at 12. What was surprising was, that all weights stayed at 0 but only bias3 was 12.36. I thought about this result, but I cannot explain it.

Using random data

The last experiment I wanted to test is, how the model reacts, if the values are random and do not have a trend or pattern. My assumption is that the loss is not reducing in every epoch but just changing randomly. After changing this line “n3 = n1 ** 2 + n2 + np.random.randint(0, 5)

“ to “n3 = np.random.radint(0, 100)”, my assumption was not true. The model still learns but the loss is very high (after epoch 1000, it was still over 800). The prediction at the end was ~51. This makes a lot of sense: since the model tries to minimize the loss function, it will predict the average value since the loss is the lowest on average, meaning a prediction of ~51 makes sense.