Phase 2

Kel Gruber

November 26, 2023

1 Phase Goal

In this phase, I intentionall built over-fit neural models using the "Bank Marketing Data Set" from Kaggle Data Science that I cleaned in Phase 1. The goal of this phase it to learn the ways that an overfit model can be built and to understand and recognize when a model has become overfit.

2 What is Overfitting a Model?

Overfitting occurs when the neural network model becomes too closely fit to the training dataset. The model essentially memorizes the training set data and then when used to make predictions has poor results because it cannot make generalizations. This can happen in a few different ways. The first is that the training data size is too small and does not contain enough instances to accurately represent all possible input data values. The second is when the training data contains large amounts of irrelevant features. The third is when the model trains for too long on a single sample set of data. The last is when the model complexity is too high so the model learns the noise within the training data.

In this phase, I used the entire dataset to build and train a model. I did not split our data into training and test data because we want to intentionally create an overfit model. As mentioned above in the third way a model can become overfit, by using all of the data and training it for too long we can create an overfit model.

3 Building Overfit Models

I struggled with this phase of the project. My initial models started off with low accuracy and continued to stay at the same low accuracy levels no matter how much training or adjustments to the learning rate, batch size, number of epochs, and features I included. My initial models from this phase had an accuracy of 25%, 18%, and 12%. Eventually, I moved on to Phase 3 and in this phase, I decided to use a different data normalization process that I saw in one of the instruction videos for Phase 3. These models behaved better and with this experience from Phase 3, I decided to return to Phase 2 and redo the data normalization process with the new method.

This normalization method worked way better and these models immediately had better accuracy results. I then built the following sized neural networks in the table below and trained them all for a minimum of 300 epochs. I broke the training into sessions of 150 epochs each and would examine the history after each session. If the accuracy was still increasing at the end of the session I would complete another session or two so most models completed about 450 to 600 epochs before I would decide to increase the size of the neural network. With the last model that had 497 neurons, I decided that perhaps I was having difficulty reaching 100% accuracy because I had still not trained my model for enough epochs so I trained it for a few more sessions more to see if I could bring accuracy up even higher than the initial 93% I had gotten with the basic training that all models had received. When I stopped due to time constraints this large model had an accuracy of 96%.

Model	Total Neurons	Layers	Final Accuracy
Single Neuron	1	1	73%
2-1 Model	3	2	73%
4-2-1 Model	7	3	74%
8-4-2-1 Model	15	4	75%
16-8-4-2-1 Model	31	5	77%
32-16-8-4-2-1 Model	63	6	79%
64-32-16-8-4-1 Model	125	6	88%
128-64-32-16-8-1 Model	249	6	91%
256-128-64-32-16-1 Model	497	6	96%

Table 1: Model Sizes and Results. Please see the Phase 2 Jupyter Notebook for more details on the Phase 2 execution and results.

4 Results

While I did not reach the full 100% accuracy for any of my models I did come close and I do believe that a few of these models are overfit and are far too large for this problem. In my largest model, the 256-128-64-32-16-1 Model with 497 neurons I was able to train it to have an accuracy of 96% on the entire dataset. I do believe that this model's architecture was far too large for this problem though and if given time it would have memorized the entire dataset and had a 100% accuracy but I had to end this phase because of time constraints.

5 Challenges

One of the biggest challenges in this project for me was knowing when to decide it was time to stop training the model because it was no longer improving its accuracy. I had the early stopping flag turned on so that if no changes were made within 20 epochs it would stop. I know this can be used to avoid overfitting but I was hoping that I could also use this to determine if my model was too small by using it to determine if a model was no longer increasing accuracy at a reasonable pace. The few times the early stopping flag was triggered I examined the training history and determined if the model was still improving enough to try to continue training it some more or if it was time to move on to a larger architecture.

References

[AWS, 2023a] (2023a). Model fit: Underfitting vs. overfitting - amazon machine learning.

[AWS, 2023b] (2023b). What is overfitting? - overfitting in machine learning explained - aws.

[Dolphin, 2022] Dolphin, R. (2022). Overfitting in ml: Avoiding the pitfalls.

[Soni, 2023] Soni, B. (2023). Topic 1: Standardization and normalization.