Lab 5 Exercise – A Little Linear Regression

Justin Ugwudike (jknu1g19@soton.ac.uk)

March 28, 2023

**1.1 Question 1**

The code in Appendix A shows how I implemented the CNN model; the input of the first fully connected layer has dimensions 48 \*40 ^ 2 because the input is the 48 flattened 40x40 feature maps of the convolution layer. Since the stride and padding of the convolution layer is 1, then the dimensions of the feature map will not change.

This model was first trained using a cross-entropy loss function. While training the loss spiralled and became exponentially negative. At the end the loss was less than – 80, 000, 000, 000 and the accuracy was 62%. The magnitude of the loss value led me to experiment with training the model using another loss function.

This time I used the Mean Square Error (MSE) loss function. My reason for this change was because, the cross-entropy loss is not bounded, meaning it can take any value from negative infinity to positive infinity. On top of this, MSE loss is the most common loss function used when performing linear regression.

After this change I was getting a more reasonable value for the loss, around 10.25 (4.s.f) on the validation data. This model took around eight minutes to train on my PC.

Text

Description automatically generated

Figure 1 – Results of the first CNN model trial. These results were hard to capture because the trial would often cause Google Colab to crash.

**1.2 Question 2**

The code in Appendix B shows how I implemented the modified CNN model. The AdaptiveMaxPool2d function has an argument of one to flatten all the feature maps into a vector.

This model performed a little worse but still took eight minutes to train. The new loss value on the validation data was 15.03 (4.s.f). The reason for the increase in the loss could be because the max pooling layer reduces the amount of information outputted by the convolution layer by filtering out the max for every feature map.

Background pattern

Description automatically generated with low confidence

Figure 2 – Results of the second CNN model trial.

**1.3 Question 3**

Adding the addition code to CNN model and training it for third time showed a massive increase in performance, with a loss of 1.145 (4.s.f) around ten times less than the losses obtained with the previous models. This model also took 8 minutes to train.

Text

Description automatically generated

Figure 3 – Results of the third CNN model trial.

The reason for this third change was for the CNN model to have a way to locate each data point. By stacking the indices on every data point before the forward pass, the model can relate the data based on the location and therefore, the importance each data point has to one another, allowing for efficient training.

**2 Appendix**

Appendix A – CNN model implementation

Text

Description automatically generated

Appendix B – Modified CNN model implementation

Text

Description automatically generated