APSC 258: Lab 6 Manual

Andre Cox

July 21, 2022

Contents

Chapter 1

Introduction

1.1 Introducing the Dropout Layer

In our last lab you may have noticed that the loss for the training data was very low while the testing loss was quite high. This is because of a phenomenon called overfitting. Overfitting is when the model has learned the detail and noise of the training data but has not learned the general shape of the data. To combat this we will use dropout layers.

1.2 The Problem

You can imagine a neural network as data flowing through a series of functions that are connected to each other. The functions are called neurons and are grouped together to form layers. Multiple layers are connected to each other to form a neural network. When we train the network we adjust these functions to make the network better at predicting the correct output. You can think of this as creating pathways in the network that get activated when a correct input is given. As we continue to train the network these pathways get more entrenched and the network becomes more accurate at predicting the correct output. However you could imagine that if the input into the network is slightly different than the model expects the pathway may not be activated. Now lets think about how we can solve this problem. We could use a huge amount of data fed into the network to create lots of pathways so that our network can predict a wide range of inputs. You can probably imagine that this is may have some downsides. For instance we would need to collect a lot more data, and the network would be more complex.

1.3 Fixing the Problem

A better way to solve this problem is to use dropout layers. These may seem difficult to understand but they are actually very simple. All a dropout layer does is randomly set some of the weights to zero; usually based on a percentage. Why does this work? As mentioned before when we overfit pathways in the network get ingrained to predict specific inputs. If we randomly set some of the weights to zero we are effectively removing some of the pathways in the network. This means that the network will have to adjust itself to more generalize to the shape of the input rather than specific features of the input.

Chapter 2

Implementation

I'll briefly explain how to implement dropout layers in Keras. See the code sample below.

```python
# First we'll import the layers we'll be using
from keras.layers import Dense, Dropout # here's the dropout layer
from keras.models import Sequential # here's the model we'll be using

# Create the model
model = Sequential()

# now lets add some layers to the model
model.add(Dense(units=1, activation='relu')) # first layer
model.add(Dense(units=25, activation='relu')) # second layer
model.add(Dropout(0.2)) # dropout layer with 20% of the units being
    set to zero
model.add(Dense(units=25, activation='relu')) # third layer
model.add(Dense(units=1, activation='sigmoid')) # output layer
```

We can see in the above code sample that our model has four dense layers. In the center of the model we have a dropout layer. This layer randomly sets 20 percent of the weights to zero. This is done to prevent the model from overfitting.

> **Note:** We define percentages as floats between 0 and 1. So 0.2 would be 20 percent. In the dropout layer you can experiment with different percentages to see how the model performs.

Chapter 3

Your Turn

In your last lab you should have implemented Convolutional layers into your model. In this lab you will implement dropout layers into your model. Your goal in this lab is to try to decrease the loss of the validation data. You can go about this in the following steps:

1. Take your model from the previous lab.
2. Record your current loss on the training and validation data.
3. Add a dropout layer to your model.
4. Add a random dropout percentage. Somewhere between 0 and 1.
5. Retrain your model.
6. Record your current loss on the training and validation data.
7. Repeat the process until you can't get the loss to decrease any more.

Hint: A good starting point for picking a random dropout percentage is 0.2 - 0.5. Most convolutional neural networks put the dropout layers in the Dense layers.

Chapter 4

Finishing Up

Once you have implemented dropout layers in your model and are happy with the results you can save your model. Once this is done have a go downloading it and running it on the PiCarV. If you need a reminder on how to do this please refer to lab 4.