Open in Colab

T81-558: Applications of Deep Neural Networks

Module 4: Training for Tabular Data

- Instructor: Jeff Heaton, McKelvey School of Engineering, Washington University in St. Louis
- For more information visit the class website.

Module 4 Material

- Part 4.1: Encoding a Feature Vector for Keras Deep Learning [Video] [Notebook]
- Part 4.2: Keras Multiclass Classification for Deep Neural Networks with ROC and AUC [Video] [Notebook]
- Part 4.3: Keras Regression for Deep Neural Networks with RMSE [Video] [Notebook]
- Part 4.4: Backpropagation, Nesterov Momentum, and ADAM Neural Network Training [Video] [Notebook]
- **Part 4.5: Neural Network RMSE and Log Loss Error Calculation from Scratch** [Video] [Notebook]

Google CoLab Instructions

The following code ensures that Google CoLab is running the correct version of TensorFlow.

In [1]:
```
try:
    %tensorflow_version 2.x
    COLAB = True
    print("Note: using Google CoLab")
except:
    print("Note: not using Google CoLab")
    COLAB = False
```

```
Note: not using Google CoLab
```

Part 4.5: Error Calculation from Scratch

We will now look at how to calculate RMSE and logloss by hand. RMSE is typically used for regression. We begin by calculating RMSE with libraries.

In [2]:
```
from sklearn import metrics
import numpy as np

predicted = [1.1,1.9,3.4,4.2,4.3]
expected = [1,2,3,4,5]

score_mse = metrics.mean_squared_error(predicted,expected)
score_rmse = np.sqrt(score_mse)
print("Score (MSE): {}".format(score_mse))
print("Score (RMSE): {}".format(score_rmse))
```

```
Score (MSE): 0.14200000000000007
Score (RMSE): 0.37682887362833556
```

We can also calculate without libraries.

In [3]:
```
score_mse = ((predicted[0]-expected[0])**2 + (predicted[1]-expected[1])**2
+ (predicted[2]-expected[2])**2 + (predicted[3]-expected[3])**2
+ (predicted[4]-expected[4])**2)/len(predicted)
score_rmse = np.sqrt(score_mse)

print("Score (MSE): {}".format(score_mse))
print("Score (RMSE): {}".format(score_rmse))
```

```
Score (MSE): 0.14200000000000007
Score (RMSE): 0.37682887362833556
```

Classification

We will now look at how to calculate a logloss by hand. For this, we look at a binary prediction. The predicted is some number between 0-1 that indicates the probability true (1). The expected is always 0 or 1. Therefore, a prediction of 1.0 is completely correct if the expected is 1 and completely wrong if the expected is 0.

In [4]:
```
from sklearn import metrics

expected = [1,1,0,0,0]
predicted = [0.9,0.99,0.1,0.05,0.06]

print(metrics.log_loss(expected,predicted))
```

```
0.06678801305495843
```

Now we attempt to calculate the same logloss manually.

In [5]:
```
import numpy as np

score_logloss = (np.log(1.0-np.abs(expected[0]-predicted[0]))+\
np.log(1.0-np.abs(expected[1]-predicted[1]))+\
```

```
np.log(1.0-np.abs(expected[2]-predicted[2]))+\
np.log(1.0-np.abs(expected[3]-predicted[3]))+\
np.log(1.0-np.abs(expected[4]-predicted[4])))\
*(-1/len(predicted))

print(f'Score Logloss {score_logloss}')
```

```
Score Logloss 0.06678801305495843
```

In [ ]: