C1W2_Assignment

May 17, 2021

1 W2 Assignment: Creating a Custom Loss Function

This short exercise will require you to write a simple linear regression neural network that is trained on two arrays: xs (inputs) and ys (labels), where the relationship between each corresponding element is $y = 2x - 1$.

$xs = [-1.0, 0.0, 1.0, 2.0, 3.0, 4.0]$

$ys = [-3.0, -1.0, 1.0, 3.0, 5.0, 7.0]$

You will need to implement a custom loss function that returns the root mean square error (RMSE) of $y_{true} - y_{pred}$. Let's begin!

[1]:
```
import tensorflow as tf
import numpy as np
from tensorflow import keras
from tensorflow.keras import backend as K

import utils
```

[2]:
```
# inputs
xs = np.array([-1.0,  0.0, 1.0, 2.0, 3.0, 4.0], dtype=float)

# labels. relationship with the inputs above is y=2x-1.
ys = np.array([-3.0, -1.0, 1.0, 3.0, 5.0, 7.0], dtype=float)
```

1.0.1 Define the custom loss function (TODO)

Define the custom loss function below called `my_rmse()` that returns the RMSE between the target (`y_true`) and prediction (`y_pred`).

You will return $\sqrt{error}$, where $error = mean((y_{true} - y_{pred})^2)$ - error: the difference between the true label and predicted label. - sqr_error: the square of the error. - mean_sqr_error: the mean of the square of the error - sqrt_mean_sqr_error: the square root of hte mean of the square of the error (the root mean squared error). - Please use `K.mean`, `K.square`, and `K.sqrt` - The steps are broken down into separate lines of code for clarity. Feel free to combine them, and just remember to return the root mean squared error.

[3]:
```
## Please uncomment all lines in this cell and replace those marked with `#␣
↪YOUR CODE HERE`.
## You can select all lines in this code cell with Ctrl+A (Windows/Linux) or␣
↪Cmd+A (Mac), then press Ctrl+/ (Windows/Linux) or Cmd+/ (Mac) to uncomment.



def my_rmse(y_true, y_pred):
    error = (y_true - y_pred)
    sqr_error = K.square(error)
    mean_sqr_error = K.mean(sqr_error)
    sqrt_mean_sqr_error = K.sqrt(mean_sqr_error)
    return sqrt_mean_sqr_error
```

[4]:
```
utils.test_my_rmse(my_rmse)
```

```
All public tests passed
```

1.0.2 Define a model using the custom loss function (TODO)

Similar to the ungraded labs, you will define a simple model and pass the function you just coded as the loss. - When compiling the model, you'll choose the `sgd` optimizer and set the `loss` parameter to the custom loss function that you just defined. - For grading purposes, please leave the other parameter values as is.

[5]:
```
## Please uncomment all lines in this cell and replace those marked with `#␣
↪YOUR CODE HERE`.
## You can select all lines in this code cell with Ctrl+A (Windows/Linux) or␣
↪Cmd+A (Mac), then press Ctrl+/ (Windows/Linux) or Cmd+/ (Mac) to uncomment.



# # define the model architecture
model = tf.keras.Sequential([keras.layers.Dense(units=1, input_shape=[1])])

# use the function you just coded as the loss
model.compile(optimizer='sgd', loss=my_rmse)

# train the model
model.fit(xs, ys, epochs=500,verbose=0)

# test with a sample input
print(model.predict([10.0]))
```

```
[[18.953402]]
```

[6]:
```
utils.test_model_loss(model.loss)
```

```
All public tests passed
```

[ ]: