PROJECT

Your first neural network

A part of the Deep Learning Nanodegree Foundation Program

PROJECT REVIEW | CODE REVIEW | NOTES

Meets Specifications

SHARE YOUR ACCOMPLISHMENT

Well done! Your submission meets all the required specifications. Please keep going on this track! Good luck! 😄

Code Functionality

All the code in the notebook runs in Python 3 without failing, and all unit tests pass.
Awesome job 👍 all the code is running without errors!

The sigmoid activation function is implemented correctly
Nice job implementing sigmoid function! To learn more about sigmoid or other activation functions like RelU and Tanh check out this link.

All unit tests must be passing

All good here.

Forward Pass

The input to the hidden layer is implemented correctly in both the train and run methods.

Well done 👏
All parameters for Forward Pass are implemented correctly!

The output of the hidden layer is implemented correctly in both the `train` and `run` methods.

The input to the output layer is implemented correctly in both the train and run methods.

Flawless!

The output of the network is implemented correctly in both the train and run methods.

Nicely done! Here the network is being used for regression, so the output of the network is the raw input to the output unit. That is, there is not activation function for the output unit, f(x) = x. So then, `final_outputs = final_inputs`.

Backward Pass

The network output error is implemented correctly

Nice job implementing `output_errors = targets - final_outputs`.

The error propagated back to the hidden layer is implemented correctly

Awesome!

Updates to both the weights are implemented correctly.

Awesome job! In case you wanna read more about backpropagation here is a nice article. However, you have already demonstrated your skills 😉

Hidden layer gradient(hidden_grad) is calculated correctly.

Well done! In case you wanna learn more about Sigmoid function and its derivative, here is a nice article.

Hyperparameters

The number of epochs is chosen such the network is trained well enough to accurately make predictions but is not overfitting to the training data.

Good choice of epochs. Since both validation and training loss has the same behavior and the validation loss is not going to increase it's good to continue the training, however when you see that the validation is going to increase then you have to stop the training to prevent overfitting!

The number of hidden units is chosen such that the network is able to accurately predict the number of bike riders, is able to generalize, and is not overfitting.

Amazing job tweaking the hyperparameters! 👏🏽

The learning rate is chosen such that the network successfully converges, but is still time efficient.

All good here.