UDACITY

PROJECT

Your first neural network

A part of the Deep Learning Nanodegree Foundation Program

PROJECT REVIEW

CODE REVIEW

NOTES

SHARE YOUR ACCOMPLISHMENT!

Meets Specifications

Awesome!

Congratulations on passing all the requirements for this project. I think you have a solid grasp of building a simple neural network. I took a look at your commented matrix shapes checks. I think it's great that you spent time on this. As for leaving them there, maybe it's not the best option. Just remember to remove these unused comments from the notebook in your GitHub repo when you get the chance.

I also saw outputs for various hyperparameters that you tried. This gave me the idea that you could use Ensemble Averaging to 'average out' the performance for each set of hyperparams. This could be useful if you're interested in improving this project at some point. It would really show anyone looking at the project that you made the extra effort--which is always a good trait to have.

Code Functionality

✓

All the code in the notebook runs in Python 3 without failing, and all unit tests pass.

Awesome

All the code in the notebook runs without failing

✓

The sigmoid activation function is implemented correctly

Good job. You could also use the lambda function to implement this.

```
lambda x: 1/(1+np.exp(-x))
```

✓

All unit tests must be passing

Forward Pass

✓

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

✓

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.

✓

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

Awesome.

Good job at implementing the output of the network. The forward pass implementation looks great.

Backward Pass

✓

The network output error is implemented correctly

I guess this works. It's not a easy to read as opposed to `output_errors = targets - final_outputs` though

✓

The error propagated back to the hidden layer is implemented correctly

Good job at implementing this.

✓

Updates to both the weights are implemented correctly.

✓

Hidden layer gradient(hidden_grad) is calculated correctly.

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.

✓

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.

✓

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

Free Form Question

✓

PLEASE NOTE:
This section is optional and does not affect the pass/fail criteria for the student.

Insightful comments made. It's interesting how you put things into perspective by using the error in terms of bike rentals per hour. It made me think that maybe the perceived low error rates of the neural network isn't really that low! I also liked your take on the holidays. It would be really interesting if we could get more data for these periods and then train the neural network to see how it performs then. But anyhow, good work on this.

DOWNLOAD PROJECT

RETURN TO PATH

Student FAQ