UDACITY Logout

PROJECT

Your first neural network

A part of the Deep Learning Nanodegree Foundation Program

PROJECT REVIEW | CODE REVIEW | NOTES

Meets Specifications

SHARE YOUR ACCOMPLISHMENT

Code is functional, clean, and easy to read.Overall a very good neural network project. I would suggest trying to manipulate the three factors that affect output of the network (`epochs`, `learning_rate` and `hidden_nodes`) in order to better understand the role they play in generalization vs. over-fitting.

Code Functionality

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

✓ The sigmoid activation function is implemented correctly

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.

Rate this review

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

Backward Pass

✓ **The network output error is implemented correctly**

✓ **Updates to both the weights are implemented correctly.**

Updates to both the weights are implemented correctly. Excellent job, this part is very tricky making sure the data structure has the correct number of dimensions as well as the transpose on inputs.T

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.**

Plot of the training and validation loss looks good. A small suggestion for future visualizations is to always include descriptive axis labels. This functionality is available from within matplotlib. You can add text, y axis labels, x axis labels, titles, subtitles and annotations. For example to add a title as well as x axis and y axis labels to the plot:

```
fooplot.set_title('axes title')
fooplot.set_xlabel('xlabel')
fooplot.set_ylabel('ylabel')
```

✓ **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.**

There are no set criteria for determining the number of hidden units. It is a good rule of thumb to be no more than twice the number of inputs and enough to generalize the problem space well. A good starting point is half way in between the number of inputs and output units.

To minimize the error and have a trained network that generalizes well, you need to pick an optimal number of hidden layers, as well as nodes in each hidden layer.

Too few nodes will lead to high error for your system as the predictive factors might be too complex for a small number of nodes to capture.

Too many nodes will over-fit to your training data and not generalize well.

It is important to strike a balance when trying to determine the number of nodes and since there are no concrete rules it can sometimes come down to trial and error. Having a solid data pipeline is key to being able to quickly experiment and optimize outputs.

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