UDACITY

PROJECT

Your first neural network

A part of the Deep Learning Nanodegree Foundation Program

PROJECT REVIEW

CODE REVIEW

NOTES

SHARE YOUR ACCOMPLISHMENT!

Requires Changes

1 SPECIFICATION REQUIRES CHANGES

Slight variance in validation/training loss curve is accepted. With 1 learning rate on this project variation seen in curev is accepted. There are techniques in which learning rate is reduced as training advances to decrease the step size and also variation in validation loss. You can refer here: http://cs231n.github.io/neural-networks-3/

You did a good job on coding part of project. This is a good first submission. You have implemented the neural network correctly. There is a room for improvement in hyperparameter optimiztaion. Just fiddle a bit with hyperparameters as per guidelines given in respective sections. Also note that apart from validation loss, training time is also an important factor in deciding hyperparameters.

Keep up with the good work!

Code Functionality

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

Code runs correctly. Good job passing the unit tests. Quality work.

The sigmoid activation function is implemented correctly

Correct, Good use of the lambda function.

Forward Pass

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

This is correct. Good use of numpy.dot for the matrix multiplication.

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

Here inputs to hidden layer are being activated with sigmoid layer

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

Nice! dot function is useful.

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

Good! Since this is a regression task, the final output is just the raw input to the output layer.

Backward Pass

The network output error is implemented correctly

Nice! Just a simple subtraction.

Updates to both the weights are implemented correctly.

Awesome! You got this right. You implemented this correctly using `np.dot` and `.T` function. Very few students able to do that in their first submission. You have a very good understanding of the concepts.

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 work! Your plot shows training and validation loss are decreasing. Also as validation loss is not increasing and stabilized, it shows that model is not overfitted.

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.

No of hidden nodes should not be less as it might underfit the network and also should not be very high as this will overfit the network. It should be such that model is able to generalize to unseen data. A good rule of thumb is the half way in between the number of input and output units. See here for more info -

https://www.quora.com/How-do-I-decide-the-number-of-nodes-in-a-hidden-layer-of-a-neural-network

Your choice of number of hidden nodes seems fine.

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

Learning rate should be set such that self.lr / n_records ~ 0.01. As n_records is 128, try setting learning_rate around 1. You may want to retune number of epochs for new learning rate.

For hyperparameter tuning, gridsearch method can be used. For learning about grid-search: https://en.wikipedia.org/wiki/Hyperparameter_optimization

RESUBMIT

DOWNLOAD PROJECT

Best practices for your project resubmission

Ben shares 5 helpful tips to get you through revising and resubmitting your project.

Watch Video (3:01)

RETURN TO PATH

Student FAQ