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

Congratulations for meeting all specifications for this project! I added some comments and suggestions below, I hope they'll be of use. Keep up the good work!

Code Functionality

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

Suggestion

Before saving your final notebook, it's a good idea to use the "Restart & Run All Cells" command in the Kernel menu to make sure the code will run correctly from start to finish, without relying on any variables or calculations previously performed in the notebook.

The sigmoid activation function is implemented correctly

All unit tests must be passing

Awesome

All code is running and all unit tests are passing. Congrats!

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

Nice job implementing the forward pass of your neural network.

Suggestion

Note how, since the forward pass is performed in both the `run()` and `train()` methods, you had to implement identical lines of code in both of them. One idea to make your code simpler and easier to maintain is to create a new method (something like `forward_pass()`) that both `run()` and `train()` would call in order to perform these calculations. This way you would prevent repeated code, making it easier to make changes if necessary.

Backward Pass

The network output error is implemented correctly

The error propagated back to the hidden layer is implemented correctly

Updates to both the weights are implemented correctly.

Hidden layer gradient(hidden_grad) is calculated correctly.

Awesome

Nice job defining an `output_grad`, even though its value is 1 and it won't modify the calculation results - it helps clarify what's going on when you are reading the code.

Suggestion

It's rare for me to say this, but it may be the case that there's too many comments in your code. :) Comments are important, but they shouldn't get in the way of your code. You can take advantage of Jupyter notebooks to replace much of your comment with actual text explaining what's going on!

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.

Awesome

Excellent job selecting your hyperparameters to both get the loss to converge and avoid overfitting. You are on the limit of what's acceptable here (reviewer guidelines recommend at least 8 hidden units, for instance), but it's working, and it's probably faster than most other implementations I've seen. So, congratulations! :)

Comment

Regarding the written question, take a look at the dates for which the model fails to make good predictions. Is that time of the year special in any way? :)

DOWNLOAD PROJECT

RETURN TO PATH