PROJECT

Your first neural network

A part of the Deep Learning Nanodegree Foundation Program

PROJECT REVIEW | CODE REVIEW | NOTES

Requires Changes

SHARE YOUR ACCOMPLISHMENT

3 SPECIFICATIONS REQUIRE CHANGES

A wonderful submission with proper code implementation, just a bit of tuning and you will get an awesome model which almost perfectly predicts for most of the year.... Spoiler: You will find that the model(after tuning) will perform well for most of the days except for a certain few days.... Your task will then be to look up the optional question and try to understand why from the data. All the best! 😃

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**

Properly implemented! 👍

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.**

Forward pass implemented properly! 😃

Backward Pass

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

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

Awesome work! This part is tricky - implementing backprop using numpy! 😃

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 epochs is a bit low here, usually the model converges by ~4000-5000 epochs. You need to choose the number of epochs such that the loss stagnates or stops decreasing, this would mean either the model has either converged or other hyper parameters needs to be tuned. The more iterations you use, the better the model will fit the data. However, if you use too many iterations, then the model with not generalize well to other data, this is called overfitting. You want to find a number here where the network has a low training loss, and the validation loss is at a minimum. As you start overfitting, you'll see the training loss continue to decrease while the validation loss starts to increase. You need to get training loss around ~0.07 and validation loss ~0.17 for the model to converge.

🔁 **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.**

You need to choose a number that would properly capture the variance in the model and be enough to generalise. Having 2 hidden units will mostly make a model with high bias and hence underfit the data and fail to generalise properly. The general rule of thumb here is to choose a number that is about half way between the number of inputs and the number of outputs, use at least 8 hidden units.

Please do read this thread in quora

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

Your learning rate seems to be a bit low to start with, try choosing a learning rate that gets the ratio of learning rate/the number of records(128) to be around 0.01, try out values such as [0.5,0.8,0.9]. A good learning rate is key to making the gradient descent converge in reasonable amount of time. Choosing a low learning rate will mean that the weight updates will be slower and hence the model will take longer to converge, choosing a high learning rate will over shoot the gradient as it takes bigger steps and probably never converge to minima. So it is crucial to tune the learning rate.

Additionally here are a few links to understand better:

1. cs231n
2. quora

RESUBMIT PROJECT

DOWNLOAD PROJECT

Learn the best practices for revising and resubmitting your project.

RETURN TO PATH