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

A very impressive and in-depth report for the project! Kudos! 😎

I appreciate your efforts for tuning the hyper parameters here!
I have provided some suggestions on this below. Hope that helps!

Also in machine learning problems there are 2 important components in making a great model :

1. Model Selection and Tuning
2. Feature Engineering and Data Analysis

For the current problem too we have to look at both these aspects. You have done a great job in model selection and tuning. The current model will be enough for predicting the dataset as you can see the model will perform pretty well for the entire data, but if you analyse the data a bit you will find some unique characteristics.
Here is an explanation as to why the december holiday season predictions performing poor is because prior information about such trends was not captured in the data, i.e the dataset that we trained on does not have information on holiday season even for the previous year. So when a new pattern is experienced the model performance is bad. This can be solved by training the model with more data possibly randomised data so that the model captures such patterns and predicts well. Think of it like you are the manager for the bike sharing service, this is your first year there and you know about the trends from January to October, knowing that data you are most likely to anticipate that the trends from your previous experience will hold right? So you would make preparations accordingly, but once december hits the holiday starts and then people stay at their houses maybe and there is a slump, as this is your first time here you only expected as per your experience. This is the same case with the model. You will have this in mind the next year December right? If you notice properly the holiday season extends over a week, but you will find that the days near christmas which are considered holiday season are marked as non-holidays, so a new feature marking the holiday season will be really helpful along with training over additional holiday season data.

Hope this helps.

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

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.

Backward Pass

The network output error is implemented correctly

Updates to both the weights are implemented correctly.

Implementing backprop using numpy is tricky and you have nailed it! 👍

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.

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.

This is the number of batches of samples from the training data we'll use to train the network. 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.

When choosing the number of epochs we should check out the loss plot and make sure that both validation and training loss are decreasing. If you see that the training loss is decreasing while the validation loss is going to increase then that is the sign of overfitting. There is a technique to prevent overfitting called Early-Stopping.

So a quick suggestion here is to gradually note the model training and validation losses over a few thousand epochs with multiple combinations of hyper parameters and choose the combinations where there is clear decrease in training and validation loss and decide whether to continue the iterations further. This is so because when you go for deeper architectures and more data such extensive tests wont be viable and would require a huge amount of compute power.

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.

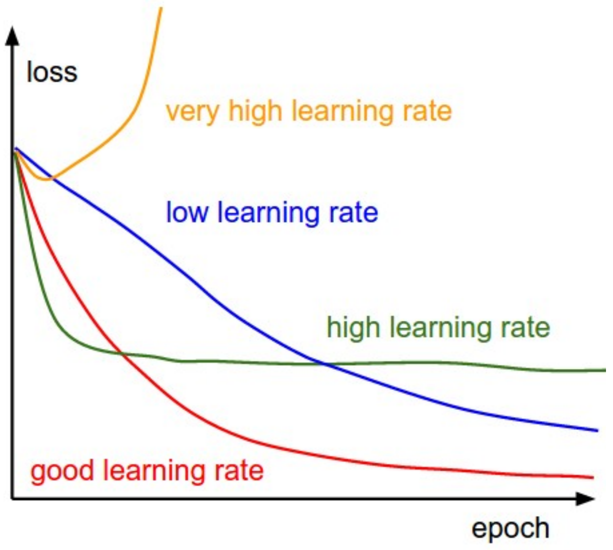
You need to choose a number that would properly capture the variance in the model and be enough to generalise. Having more hidden units will mostly make a model with high variance and hence overfit the training data and fail to generalise properly and similarly having less hidden units will mostly make a model with high bias and hence under fit the training data and fail to fit 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

A good choice here! 😃

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

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.
Please remember there is a trade-off between LR and the Iterations. If you choose smaller LR you will need more iterations to get the model converge and the other way around. Our job is to find the most efficient combination where the training process is time efficient and at the same time is adequately accurate.



DOWNLOAD PROJECT

RETURN TO PATH