Predicting Bike-Sharing Patterns

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Awesome submission! Your code works well! The model has converged to a loss minima! Way to go! 👏

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.

Wonderful job! Your model passes all the tests! :)

The sigmoid activation function is implemented correctly

Good work!
Here are some more activation functions and their characteristics:
https://cs231n.github.io/neural-networks-1/#actfun

Forward Pass

The forward pass is correctly implemented for the network's training.

The implementation of hidden and final layer's parameters are all correct:

Hidden inputs are computed rightly by multiplying inputs with weights_input_to_hidden. ✅
Hidden outputs are computed rightly by passing hidden inputs through sigmoid activation function. ✅
Final inputs are computed correctly by multiplying hidden outputs with weights_hidden_to_output. ✅
Final outputs are computed correctly by directly passing final inputs. It's great that you correctly understood that usage of the non-linear activation function is not required as we are working on a regression problem. ✅

The run method correctly produces the desired regression output for the neural network.

Backward Pass

The network correctly implements the backward pass for each batch, correctly updating the weight change.

A proper implementation, many students get mistakes here due to some calculations error in using numpy,

you have done a great job! Do keep practising numpy functions which will help in learning Deep Learning modules. :)
You can get a good refresher on backprop whenever you have doubts here:
https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

Updates to both the input-to-hidden and hidden-to-output weights are implemented correctly.

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.

A good choice for the number of epochs here! Way to go!
When you find your model loss not improving or your train loss improves but all loss isn't or maybe getting worse, take it as a sign to stop the training - this is called early stopping. This will ensure that your model does not overfit the training data. Also in cases where your model train loss and validation loss isn't improving or improving slowly, it is a sign that the learning rate needs to be tuned, increased as the update steps might be too small to make any meaningful improvement.

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.

Well done!A further recommendation is that it should probably be no more than twice the number of input units, and enough that the network can generalize, so probably at least 8. A good rule of thumb is the halfway between the number of input and output units.
There's a good answer here for how to decide the number of nodes in the hidden layer.
https://www.quora.com/How-do-I-decide-the-number-of-nodes-in-a-hidden-layer-of-a-neural-network

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

So here is my trick(my two cents) I always have a set of learning rates logarithmically decreasing like - [0.7,0.3,0.1,0.05], as the number of epochs gets higher and I see the loss not improving I tend to change my learning rate to a lower number. Try this out when you get time. :)

Additionally here are a few links to understand better:

1. cs231n
2. quora

Kudos!

The number of output nodes is properly selected to solve the desired problem.

The training loss is below 0.09 and the validation loss is below 0.18.

Great Job! You have passed the rubric requirement!

DOWNLOAD PROJECT

RETURN TO PATH