https://review.udacity.com/#!/reviews/457935/shared

Meets Specifications

An awesome submission this one. Well done, I sense a machine learning ninja in the making here 😎

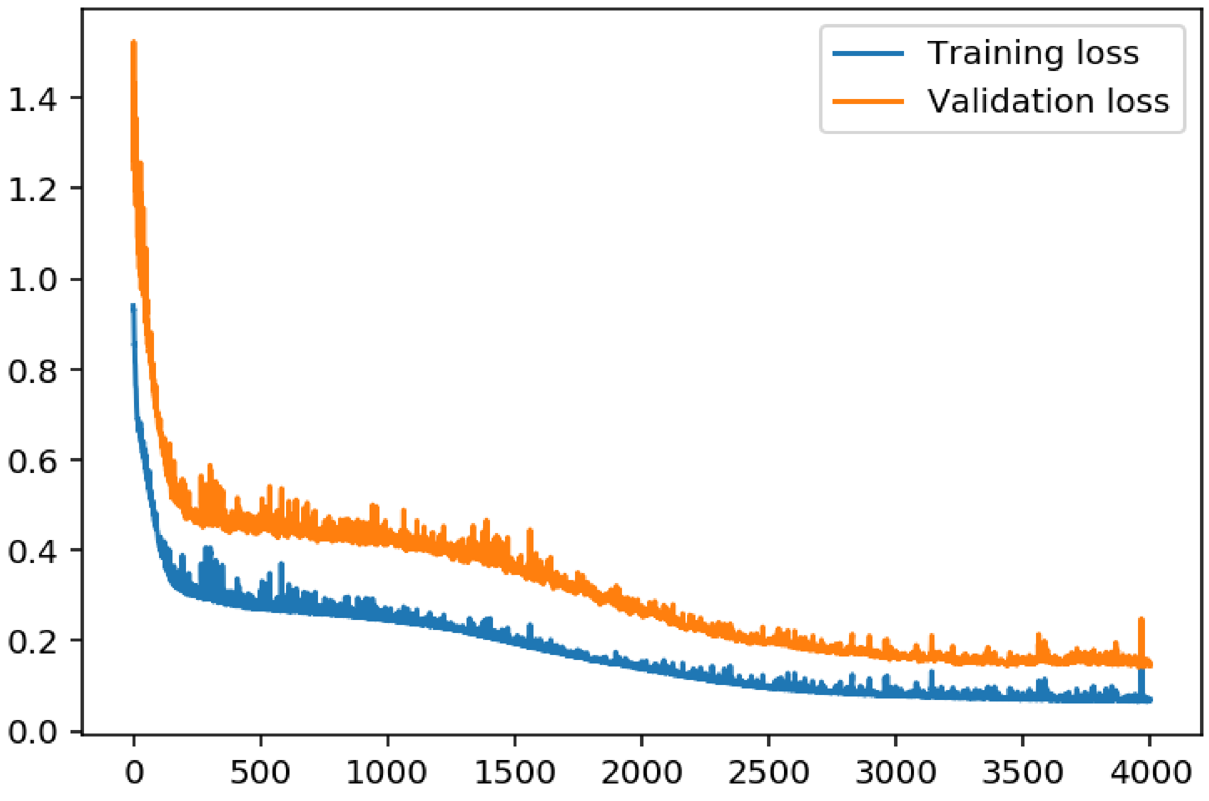
A very good and neatly presented answer for the optional questions. You have a good insight for the data which is crucial for machine learning. Yes, a new feature/flag for marking holiday season would be great, as you will find that usually holiday season is taken as an entire week holiday and you will find that 24th is not considered so in the data, this along with a few more historical data would make the model awesome.

It is a good practise to randomly choose and try out hyper parameters and that is what we use in the industry, additionally we use gridsearch with a few hand picked hyper parameters which we think will be a good selection for the problem. How do we know its a good pick? with practise you tend to get some insight on the data, we start with a random value for example the learning rate 0.5 for instance and we visualise the loss over a certain number of epochs and see how well the learning rate does. Then it is usually a good practise to take well spaced out selections so that we know the sweet spot range, for example lr ~ [0.5,0.3,0.1,0.05,0.03,0.01] and perform gridsearch. So if the model performs we in the range between 0.5 and 0.3 we tend to start with 0.5 and slowly decrease the learning rate with the number of iterations as the gradient moves to minima a smaller learning rate will have a greater chance of convergence. Hope this helps.

Additionally I ran your model with a range of learning rates and your model does converge a bit earlier than ~7000 iterations, try out with a few more learning rate combinations to achieve similar results. 😉

Here is the result I got:

All the best! 👋



Code Functionality

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

Perfect! Just perfect first submission. Your code passes all the tests! Perfect implementation of both the forward and backward passes. Well done! 👏

The sigmoid activation function is implemented correctly

A good use of python lambda functions here! 👍

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.

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 number of epochs chosen. The model converges really well! A great combination of hyper parameters here.

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.

> A nice choice for the number of hidden units. Well done! 👌

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

> A perfectly tuned learning rate that fits well for the problem. Well done! 😃