Step 1: Run below command in the root directory. ‘pip install udacity-pa’

Step 2: Download the jwt file by clicking here, <https://project-assistant.udacity.com/auth_tokens/new>

Copy the contents to the Jwt file in the project folder.

Step 3: Run the submit command from the first-neural-network folder

cd first-neural-network

/home/workspace/first-neural-network# **udacity submit first-neural-network**

This will create the zip file

Step 4: Upload the file under the dashboard.

**Links:**

[GitHub - udacity/deep-learning-v2-pytorch: Projects and exercises for the latest Deep Learning ND program https://www.udacity.com/course/deep-learning-nanodegree--nd101](https://github.com/udacity/deep-learning-v2-pytorch)

<https://www.manning.com/cart>

[CS231n Convolutional Neural Networks for Visual Recognition](https://cs231n.github.io/neural-networks-1/#actfun)

classroom.udacity.com/me

[(29) How do I decide the number of nodes in a hidden layer of a neural network? I will be using a three layer model. - Quora](https://www.quora.com/How-do-I-decide-the-number-of-nodes-in-a-hidden-layer-of-a-neural-network-I-will-be-using-a-three-layer-model)

[Gradient (video) | Khan Academy](https://www.khanacademy.org/math/multivariable-calculus/multivariable-derivatives/gradient-and-directional-derivatives/v/gradient)

[Vanishing gradient problem - Wikipedia](https://en.wikipedia.org/wiki/Vanishing_gradient_problem)

[Geometric Sequences and Exponential Functions - Algebra | Socratic](https://socratic.org/algebra/exponents-and-exponential-functions/geometric-sequences-and-exponential-functions)

[sketch-rnn (tensorflow.org)](https://magenta.tensorflow.org/assets/sketch_rnn_demo/index.html)

[yunjey/pytorch-tutorial: PyTorch Tutorial for Deep Learning Researchers (github.com)](https://github.com/yunjey/pytorch-tutorial)

Evolution of RNNs:

[Time delay neural network - Wikipedia](https://en.wikipedia.org/wiki/Time_delay_neural_network)

[Recurrent neural network - Wikipedia](https://en.wikipedia.org/wiki/Recurrent_neural_network#Elman_networks_and_Jordan_networks)

[nc.dvi (jku.at)](http://www.bioinf.jku.at/publications/older/2604.pdf)

[Sepp Hochreiter - Wikipedia](https://en.wikipedia.org/wiki/Sepp_Hochreiter)

[Juergen Schmidhuber's home page - Universal Artificial Intelligence - New AI - Deep Learning - Recurrent Neural Networks - Computer Vision - Object Detection - Image segmentation - Goedel Machine - Theory of everything - Algorithmic theory of everything - Computable universe - Zuse's thesis - Universal learning algorithms - Universal search - Kolmogorov Complexity - Algorithmic information - Super Omega - Speed Prior - Independent component analysis - ICA - Financial forecasting - Evolution - Reinforcement learning - POMDPs - Reinforcement learning economy - Hierarchical learning - Metalearning - Learning to learn - Self-Improvement - Genetic programming - Attentive vision - Active exploration - Theory of beauty - Theory of creativity - Theory of Humor - Facial Attractiveness - Low-complexity Art - Lego Art (idsia.ch)](https://people.idsia.ch/~juergen/)

[deeplearning4j.org](https://deeplearning4j.org/lstm.html)

**Project 1 Review notes:**

**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.**

Great work.  
Summary:

* Forward propagation sequentially calculates and stores intermediate variables within the computational graph defined by the neural network. It proceeds from the input to the output layer.
* Backpropagation sequentially calculates and stores the gradients of intermediate variables and parameters within the neural network in the reversed order.
* When training deep learning models, forward propagation and back propagation are interdependent. Training requires significantly more memory than prediction.
* For more basic understanding, check <http://jalammar.github.io/visual-interactive-guide-basics-neural-networks/>
* This comprehensive explanation will help you get a better intuition into forward propagation <https://towardsdatascience.com/forward-propagation-in-neural-networks-simplified-math-and-code-version-bbcfef6f9250>

**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.**

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, and increased as the update steps might be too small to make any meaningful improvement.  
As you can see the model loss stagnates after ~4000 epochs, so you can stop training by around 4000 epochs.

**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.**

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](http://cs231n.github.io/neural-networks-3/#loss)
2. [quora](https://www.quora.com/In-an-artificial-neural-network-algorithm-what-happens-if-my-learning-rate-is-wrong-too-high-or-too-low)

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!

But as you can see, due to the high training of your model for higher epochs the validation loss has slightly increased beyond our goal.