UDACITY

PROJECT

Your first neural network

A part of the Deep Learning Nanodegree Foundation Program

PROJECT REVIEW

CODE REVIEW

NOTES

SHARE YOUR ACCOMPLISHMENT!

Requires Changes

1 SPECIFICATION REQUIRES CHANGES

Overall this is a very good submission, and you only have minor adjustments to make in order to meet all the specs.

The reason your network performs differently each time you run it is that we are using random batches of 128 records when we train. To make your results reproducible you can always set a random seed... `np.random.seed(42)`

You're very close to completion, so keep at it! 😃

Code Functionality

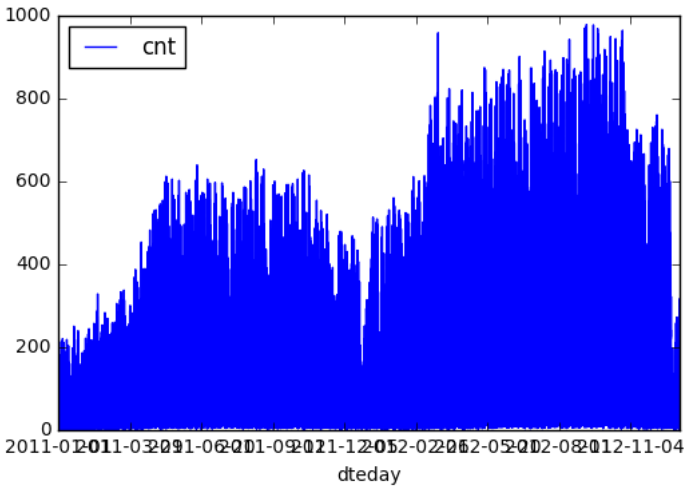
All the code in the notebook runs in Python 3 without failing, and all unit tests pass.

All the code runs well. 😎

Also, nice comments about the model and where it struggles with predictions during the holiday season.

As you mention, a likely cause for the test error is because there isn't enough data for the network to learn about the lower ridership during the holiday season. (we only have 2 years of data, and the network only saw this last holiday period once)

```
rides.plot(x='dteday', y='cnt');
```



(it looks like Christmas week in 2012 has a bigger drop in rides than in 2011)

The sigmoid activation function is implemented correctly

Good work with the lambda function. We could also implement with a function like this...

```
def sigmoid(x):
    return 1/(1+np.exp(-x))
```

Forward Pass

The input to the hidden layer is implemented correctly in both the train and run methods.

Good implementation of the matrix multiplication of the inputs and hidden layer weights to calculate `hidden_inputs`.

If you wanted, you could also include a bias term in the layer.

The output of the hidden layer is implemented correctly in both the `train` and `run` methods.

Nice job applying the activation function to the hidden layer inputs.

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.

Great work generating the final output of the network — our network is being used for regression, so the final output is simply the raw input to the final output unit, `f(x) = x`.

Backward Pass

The network output error is implemented correctly

Updates to both the weights are implemented correctly.

Great job implementing the backprop correctly, this is definitely the part of the code where many students run into trouble. Kudos! 😎

While deep learning frameworks such as keras and tensorflow can abstract away this process, it's still good to understand just how neural nets actually learn...

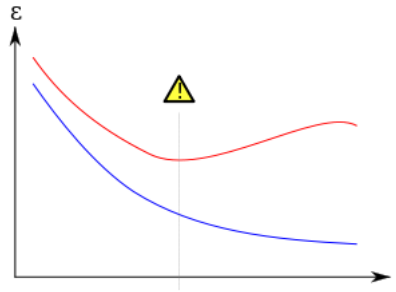
https://medium.com/@karpathy/yes-you-should-understand-backprop-e2f06eab496b

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.

Try to train the model a bit more here, and increase the value of `epochs`.

Looking at the training plot, the Validation loss appears to be still decreasing, which is a good indication the model can be trained further and is not overfitting.



(wikipedia)

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.

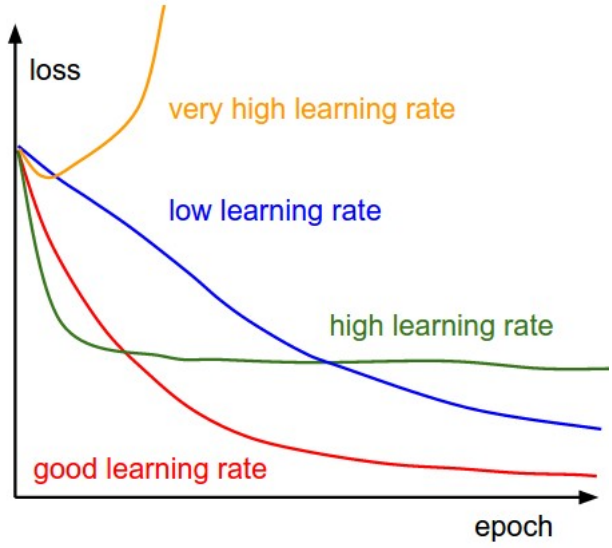
Choosing the number of nodes is somewhat subjective, but try to increase `hidden_nodes` while keeping it less than 2X the number of input units. (our number of input features is `N_i = 56`)

A good rule of thumb is the halfway point between the number of input & output units. (anywhere from 8 to 30 nodes works well)

See below for further guidance:
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.

The choice of learning rate is also somewhat subjective. Just make sure you choose a rate that's appropriate for your number of `hidden_nodes`, and that you train with enough `epochs` to help the model avoid any local minima.



RESUBMIT

DOWNLOAD PROJECT

Best practices for your project resubmission

Ben shares 5 helpful tips to get you through revising and resubmitting your project.

Watch Video (3:01)

Have a question about your review? Email us at review-support@udacity.com and include the link to this review.

RETURN TO PATH

Student FAQ