UDACITY

‹ Return to "Deep Learning" in the classroom

Predicting Bike-Sharing Patterns

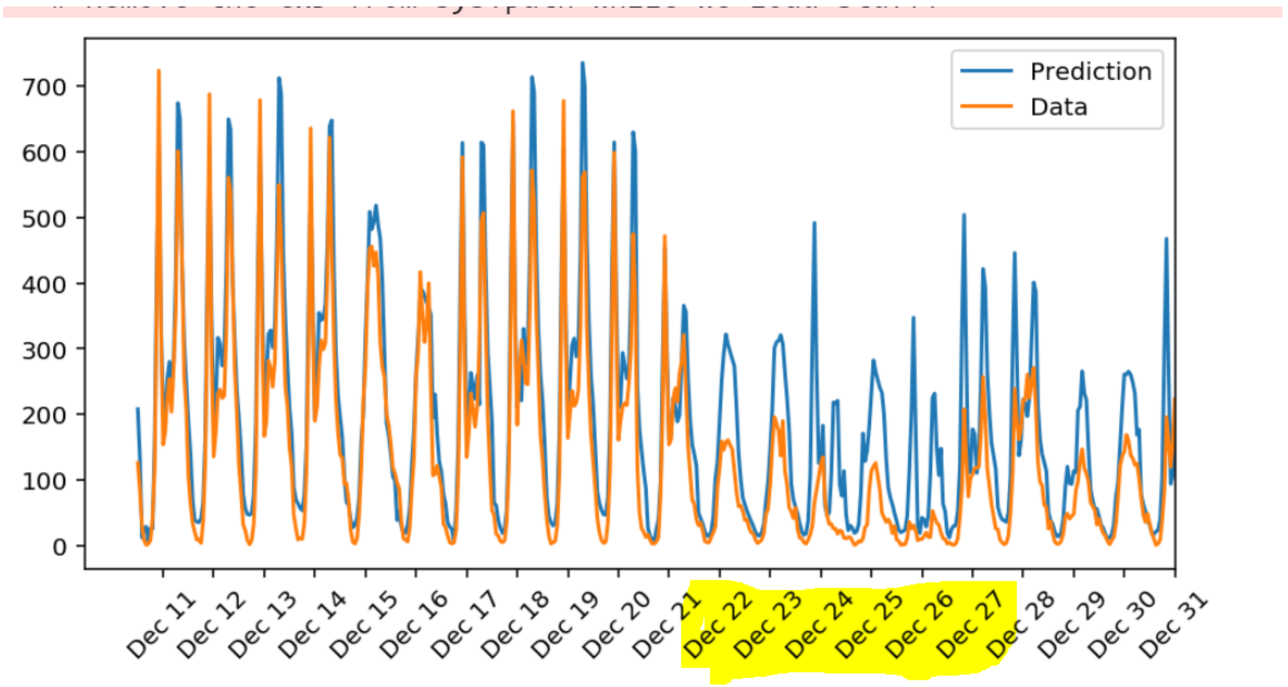
REVIEW

CODE REVIEW

HISTORY

Meets Specifications

The model actually fails to perform after 21st December because of the Christmas Holidays in which bicycle usage dropped due to the festive. Neural net failed to capture this pattern during the Christmas festival because this dataset only has two years of data. You need to keep in mind that the neural networks need a lot of data to train on so that it can make accurate predictions.



Please note that this review is independent of the previous reviews.

Code Functionality

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

All the unit tests are passing given in the notebook and also the tests by udacity-pa are passing. That's great! 👍

Unit testing is one of the steps to write industrial bug free code. You will come across a lot in this nanodegree. I'd suggest you to learn about unit testing. Here's a introductory tutorial on unit testing
https://www.youtube.com/watch?v=6tNS--WetLI&t=1816s

The sigmoid activation function is implemented correctly

A sigmoid function maps any value to a value between 0 and 1. We use it to convert numbers to cumulative probabilities.

Check out this lesson video to know about other kinds of non-linear activation units
https://www.youtube.com/watch?v=kA-1vUt6cvQ

Forward Pass

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

Correct implementation!
In the project, the network is being used for regression. So the output of the network is the raw input to the output unit. That is, there is linear activation function for the output unit, `f(x) = x`

`final_outputs = self.activation_function(final_inputs)` would have been a correct implementation if it was the classification problem that is when the output value is discrete. However final output of cnt is the continuous valued so it is a regression problem and we cannot use activation function to get the output unit.

Here is a link for reference
http://stats.stackexchange.com/questions/218542/which-activation-function-for-output-layer

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

The difference between the `run` method and `forward_pass_train` method is that the run method is used to give out the prediction output values and there's no weight updates in the `run` method. On the other hand, `forward_pass_train` method is used to calculate the weight values by using gradient descent and minimize the prediction error by the back-propagation algorithm.

Backward Pass

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

Correct Implementation! The weights are updated using the gradient descent algorithm. The gradient descent has been extensively covered in the lessons.

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

Correct Implementation! The output error is propagated back to the hidden layer with the hidden-to-output weights, and the hidden gradient should also be calculated. After this, the weights are updated.

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.

The number of iterations of 3000 seems to be more than enough. Because you need to have larger training iterations in order for the model to learn and so the training loss continues to go down and eventually converges to the lower level of the training loss value.

This video is about the number of epochs https://www.youtube.com/watch?time_continue=11&v=TTdHpSb4DV8

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.

For this project, the number of nodes is fairly open. I recommend that number of hidden nodes should probably be no more than twice the number of input units, and enough that the network can generalize. A good rule of thumb is the half way in 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

This lesson video is about tuning the number of hidden nodes https://www.youtube.com/watch?time_continue=2&v=IkGAIQH5wH8

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

The learning rate is often depended on the batch size/number of training examples in each iterations. Check out this lesson video to see the relationship between the batch size and the learning rate https://www.youtube.com/watch?v=hMLUgM6kTp8&list=PLAwxTw4SYaPn_OWPFT9ulXLuQrImzHfOV&index=20

This lesson video is about tuning the learning rate value; https://www.youtube.com/watch?time_continue=6&v=HLMjeDez7ps

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

Correct!

Check out this link for the neural net to be used as regression https://deeplearning4j.org/logistic-regression

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

Your project meets the loss requirements!

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review