UDACITY

Return to "Deep Learning" in the classroom

Predicting Bike-Sharing Patterns

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Congratulations!

You have successfully passed the first project. The network is well coded. Hyperparameters are tuned and helping the model to achieve great results. ⭐

Keep Learning! Deep Learning!

Code Functionality

✓

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

All the code cells and unit test functions are running flawlessly. ✌🏾

✓

The sigmoid activation function is implemented correctly

The implementation of the sigmoid activation function is absolutely correct.

Forward Pass

✓

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

Amazing!

The forward propagation is well coded. You have coded the hidden inputs, hidden outputs, final inputs and final outputs rightly. 👌

✓

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.

Good Job!

The backward propagation is precisely coded up. You have implemented the errors, gradient steps and delta weights rightly. This is the trickiest part of the project. 👍

✓

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

The final weight update steps are rightly implemented. This completes the implementation of forward and backward propagation.

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.

Perfect!

The hyperparameter values are well chosen.
The number of iterations as 5000 is a great choice. This provides enough opportunity to the model for learning the hidden patterns. Basically, students should choose the number of epochs such that the loss on the training set is low and the loss on the validation set isn't increasing.

The current training-validation losses graph is not providing much info. I encourage you to try zooming into the graph to observe the response of the curves:

```
plt.plot(losses['train'], label='Training loss')
plt.plot(losses['validation'], label='Validation loss')
plt.legend()
_ = plt.ylim(0, 0.8)
```

✓

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.

The hidden nodes as 15 is a great choice. These many hidden nodes help to build a reasonably complex model. It's learning the patterns and improving.
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.

Learning rate as 0.75 is a great choice. The learning rate helps to update the weights of layers such that they converge to minima.

✓

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!

The network is perfectly trained. It's learning the hidden patterns. The training-validation losses are in acceptable ranges.

DOWNLOAD PROJECT

RETURN TO PATH