UDACITY

‹ Back to Deep Learning Nanodegree

Your first neural network

审阅

HISTORY

Meets Specifications

Congratulations on passing the first project of Deep Learning Nanodegree. You have done a small mistake. Only after correcting it add this project to your portfolio.Keep up the great work!
Good Luck for project 2.

Keep Learning! Deep Learning!

Code Functionality

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

All your code is correct but your unit tests are not passing.
I can mark it as requires changes but I liked your approach of hyperparameter tuning 👍. So, I am not gonna mark it as requires changes.
Let me share the reason of unit tests not working correctly. It is due to the weight update step in NeuralNetwork class.
You have used these lines:

```
self.weights_hidden_to_output += learning_rate*np.dot(output_errors,hidden_outputs.T)
self.weights_input_to_hidden += learning_rate*np.dot(hidden_errors *hidden_grad,inputs.T)
```

It should look like these:

```
self.weights_hidden_to_output += self.lr*np.dot(output_errors,hidden_outputs.T)
self.weights_input_to_hidden += self.lr*np.dot(hidden_errors *hidden_grad,inputs.T)
```

For reference check your **init**method.

Correct this logical error and your notebook will run perfectly.

The sigmoid activation function is implemented correctly

Awesome.
The lambda function is a great way to create small anonymous functions. You have correctly implemented the Sigmoid function.

Forward Pass

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

Excellent! The formula for hidden_inputs is correct.

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

The activation function is correctly used to calculate the output of the hidden layer.

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.

This is one of the steps in which many students make a mistake. But you have correctly understood it. Good Job!

Backward Pass

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

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

The formulas of hidden_errors, hidden_grad are precisely coded but the weight update steps should be written like this.

```
`   self.weights_hidden_to_output += self.lr*np.dot(output_errors,hidden_outputs.T)`
`   self.weights_input_to_hidden += self.lr*np.dot(hidden_errors *hidden_
grad,inputs.T)`
```

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 epochs is correctly chosen. You can try some other values too like 1500, 2000, 2500, 3500, 5000. Observe the change in behavior of the model by varying the value of number of 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.

The number of hidden units as 18 is a pretty good choice.
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

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

You have tried to use adaptive learning rate, It's really good.

You can further read about learning rate's influence on models performance by going through this link

下载项目