4. Training the Network

Project due Nov 5, 2020 05:29 IST *Completed*

Forward propagation is simply the summation of the previous layer's output multiplied by the weight of each wire, while back-propagation works by computing the partial derivatives of the cost function with respect to **every** weight or bias in the network. In back propagation, the network gets better at minimizing the error and predicting the output of the data being used for training by incrementally updating their weights and biases using stochastic gradient descent.

We are trying to estimate a continuous-valued function, thus we will use squared loss as our cost function and an identity function as the output activation function. $f(x)$ is the activation function that is called on the input to our final layer output node, and $\hat{a}$ is the predicted value, while y is the actual value of the input.

$$C = \frac{1}{2}(y - \hat{a})^2 \tag{6.1}$$

$$f(x) = x \tag{6.2}$$

When you're done implementing the function `train` (below and in your local repository), run the script and see if the errors are decreasing. If your errors are all under 0.15 after the last training iteration then you have implemented the neural network training correctly.

You'll notice that the `train` functin inherits from `NeuralNetworkBase` in the codebox below; this is done for grading purposes. In your local code, you implement the function directly in your Neural Network class all in one file. The rest of the code in `NeuralNetworkBase` is the same as in the original `NeuralNetwork` class you have locally.

In this problem, you will see the network weights are initialized to 1. This is a bad setting in practice, but we do so for simplicity and grading here.

You will be working in the file `part2-nn/neural_nets.py` in this problem

Implementing Train

5.0/5.0 points (graded)

Available Functions: You have access to the NumPy python library as `np`, `rectified_linear_unit`, `output_layer_activation`, `rectified_linear_unit_derivative`, and `output_layer_activation_derivative`

Note: Functions `rectified_linear_unit`, `rectified_linear_unit_derivative`, and `output_layer_activation_derivative` can only handle scalar input. You will need to use `np.vectorize` to use them.

```
class NeuralNetwork(NeuralNetworkBase):

    def train(self, x1, x2, y):

        ### Forward propagation ###
        input_values = np.matrix([[x1],[x2]]) # 2 by 1

        # Calculate the input and activation of the hidden layer
        hidden_layer_weighted_input = self.input_to_hidden_weights.dot(input_values) + self.biases# TODO (
        ReLU_vec = np.vectorize(rectified_linear_unit)
        hidden_layer_activation = ReLU_vec(hidden_layer_weighted_input)# TODO (3 by 1 matrix)

        output = self.hidden_to_output_weights.dot(hidden_layer_activation) # TODO
        activated_output = output_layer_activation(output)# TODO

```

Press ESC then TAB or click outside of the code editor to exit

Correct

Test results

CORRECT

See full output

See full output

Submit

You have used 1 of 50 attempts

Discussion

Hide Discussion

Topic: Unit 3 Neural networks (2.5 weeks):Project 3: Digit recognition (Part 2) / 4. Training the Network

Add a Post

Show all posts

by recent activity

- Another approach - 4. Training The Network — Pinned, Community TA — 41
- STAFF - Grading not working for answer submitted for - Implementing Train — Grading not working for answer submitted for - Implementing Train I submitted my answer well before the due date but grading is n... — 1
- HOW LONG DOES THE GRADER TAKE??? — It's absolutely frustrating that the grader takes forever!!! Implementing a test.py local file is pointless if you can't see the feedback fr... — 13
- 4. Training the Network — I got all the answers correct but the grader market it incorrect. my output Epoch 9 (Input --> Hidden Layer) Weights: [[0.79436072 0.8... — 10
- Code must not contain available functions. — 2
- THERE IS A PROBLEM WITH GRADING — IN SPITE OF CORRECT ANSWER II AM GETTING GRADING IN UNIT TRAINING THE NETWORK . CAN YOU CHECK MANUALY AND DO TH... — 1
- Help to understand the element wise multiplicatoin — Hi everyone! That's very hard project. Wow. Can anyone help understand - based on what knowladge we should use element wise m... — 2
- 4. Training the Network : correct output and my output is exactly same. however, Test cases is falling. can you check manually. — 4. Training the Network : correct output and my output is exactly same. however, Test cases is falling. can you check manually and g... — 1
- Dear Dr. Karene Chu — I was wondering, Are you aware of that is going on here? Have you happened to read some of the comments appeared in this sectio... — 8
- Mathematical Formula for Error — output_layer_error is C right? Am i wrong? Could anyone give clue what is mathematical formulation for Error: 1.output_layer_error_... — 2
- Tips I recommend — Useful tips: 1) As many have said in some posts, delete all those "spaces" around the equal signs such that "x = y" ---> "x=y". This will ... — 2
- [STAFF] Does it really converge at epochs_to_train = 10? — I completed a solution that works locally when I ramp up the epochs_to_train to 1000. It fails at 10 and 100. Thinking that the grader ... — 5
- IMP : Grader [STAFF] — 6

Previous

Next

edX

- About
- Affiliates
- edX for Business
- Open edX
- Careers
- News

Legal

- Terms of Service & Honor Code
- Privacy Policy
- Accessibility Policy
- Trademark Policy
- Sitemap

Connect

- Blog
- Contact Us
- Help Center
- Media Kit
- Donate

GET IT ON Google play

Download on the App Store

深圳市恒宇博科技有限公司 粤ICP备17044299号-2