

EdX and its Members use cookies and other tracking technologies for performance, analytics, and marketing purposes. By using this website, you accept this use. Learn more about these technologies in the [Privacy Policy](#).



MITx: 6.86x

Machine Learning with Python-From Linear Models to Deep Learning

[Help](#)[sandipan\\_dey](#)

[Course](#) > [Unit 3 Neural networks \(2.5 weeks\)](#) > [Project 3: Digit recognition \(Part 2\)](#) > 5. Predicting the Test Data

## 5. Predicting the Test Data

Now fill in the code for the function predict, which will use your trained neural network in order to label new data.

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

### Implementing Predict

5/5 points (graded)

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

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

```
1 class NeuralNetwork(NeuralNetworkBase):
2
3     def predict(self, x1, x2):
4
5         input_values = np.matrix([[x1],[x2]])
6
7         # Compute output for a single input(should be same as the forward propagation in training)
8         hidden_layer_weighted_input = self.input_to_hidden_weights @ input_values + self.biases # TODO
9         hidden_layer_activation = hidden_layer_weighted_input.copy()
10        for i in range(len(hidden_layer_activation)):
11            hidden_layer_activation[i] = rectified_linear_unit(hidden_layer_weighted_input[i].item()) # TODO (3 by 1 matrix)
12        output = self.hidden_to_output_weights @ hidden_layer_activation # TODO
13        activated_output = output_layer_activation(output) # TODO
14
15        return activated_output.item()
16
```

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

Correct

## Test results

[Hide output](#)**CORRECT**

Test: local testpoints tuned params

Testing datapoints from local tests

**Output:**

```
(Input --> Hidden Layer) Weights:  [[1.04754206 1.06143111]
 [1.04754206 1.06143111]
 [1.04754206 1.06143111]]
(Hidden --> Output Layer) Weights:  [[1.10707298 1.10707298 1.10707298]]
Biases:  [[0.01026478]
 [0.01026478]
 [0.01026478]]
Test Passed
Test Passed
Test Passed
Test Passed
Test Passed
Test completed
```

Test: more testpoints tuned params

Testing datapoints from local tests

**Output:**

```
(Input --> Hidden Layer) Weights:  [[1.04754206 1.06143111]
 [1.04754206 1.06143111]
 [1.04754206 1.06143111]]
(Hidden --> Output Layer) Weights:  [[1.10707298 1.10707298 1.10707298]]
Biases:  [[0.01026478]
 [0.01026478]
 [0.01026478]]
Test Passed
Test Passed
Test Passed
Test completed
```

Test: predict above line default params

Testing random datapoint predictions near a random line

**Output:**

```
(Input --> Hidden Layer) Weights:  [[1 1]
 [1 1]
 [1 1]]
(Hidden --> Output Layer) Weights:  [[1 1 1]]
Biases:  [[0]
 [0]]
Point, (40, 1150) Prediction, 3570.000000
Point, (32, 925) Prediction, 2871.000000
Point, (-4, -125) Prediction, 0.000000
Point, (77, 2226) Prediction, 6909.000000
Point, (16, 463) Prediction, 1437.000000
Point, (-99, -2882) Prediction, 0.000000
Point, (12, 342) Prediction, 1062.000000
Point, (-31, -904) Prediction, 0.000000
Point, (10, 280) Prediction, 870.000000
Point, (-73, -2123) Prediction, 0.000000
Test completed
```

Test: predict above line random params

Testing random datapoint predictions near a random line, with random network parameters

**Output:**

```
(Input --> Hidden Layer) Weights:  [[0.272614  0.073678]
[0.00451  0.261516]
[0.47835  0.932721]]
(Hidden --> Output Layer) Weights:  [0.277808  0.092931  0.266236]
Biases:  [[0.457848]
[0.659228]
[0.532457]]
Point, (-38.39833909389459, -207.39003456336755) Prediction, 0.000000
Point, (18.708050958155418, 132.24830574893252) Prediction, 42.898786
Point, (99.07906880368296, 615.4744128220977) Prediction, 200.886135
Point, (30.609323400775573, 205.65594040465345) Prediction, 66.836208
Point, (83.65475878460317, 525.928552707619) Prediction, 171.501712
Point, (46.69261859111744, 302.15571154670465) Prediction, 98.392897
Point, (-37.660397665056124, -200.96238599033674) Prediction, 0.000000
Point, (-50.62384599812762, -278.7430759887657) Prediction, 0.000000
Point, (58.996773552749474, 380.9806413164969) Prediction, 124.000092
Point, (-31.801057784781896, -162.80634670869136) Prediction, 0.000000
Test completed
```

[Hide output](#)

Submit

You have used 1 of 20 attempts

✓ Correct (5/5 points)

When you're done, run the script and make sure that all of your predictions pass the test cases.


Discussion

Hide Discussion


Topic: Unit 3 Neural networks (2.5 weeks):Project 3: Digit recognition (Part 2) / 5. Predicting the Test Data

Add a Post


Show all posts ▾by recent activity ▾

 [Bias term](#)

Hi, I see that there is a Bias to be used in the hidden layer activation but I was under the assumption that for the second step we should also have a bias (when consolidating t... 3 ▾

 [Inconsistency in Predict](#)

Grader accepts two tests and reject the third one. The traceback is Traceback (most recent call last): File "submission.py", line 40, in predict W1 = self.input to hidden weights... 2 ▾

 [Community TA](#)

Learn About Verified Certificates