Assignment 3

Due date: April 19th, Sunday, 11:59 pm
(The late submission policy is the same as that of the previous assignments.)

Backpropagation for Handwritten Digit Recognition

Description

In this assignment, you are to write a program that builds a 3-layer (input layer – hidden layer – output layer) feed-forward network and trains it using the back-propagation algorithm. The problem that the neural network will handle is a 3-class classification problem for recognizing three handwritten digits: 1, 8, and 9. All inputs to the neural network will be numeric.

The neural network has one hidden layer with 10 nodes without counting a bias node. The network is fully connected between consecutive layers. Each node in the hidden layer and the output layer will also have an extra input from a "bias node" that has the constant value +1. In other words, we consider both the input layer and the hidden layer as containing one additional node called a bias node. All nodes in the hidden and output layers (except for the bias nodes) should use the sigmoid activation function. The initial weights of the network will be randomized. You can start with your own weights. You need to determine your own learning rate, too. After each iteration, please check your output's accuracy on the training data. If the accuracy is greater than 90%, you can stop. Otherwise, you need to repeat the process and adjust the weights. Please count the number of epochs until stopping, and explain it in your asn3.pdf.

Your input examples have ***256*** attributes (hence there are 256 input nodes, not counting the bias node) and we will have ***10*** nodes (not counting the bias node) in the hidden layer, and ***3*** nodes in the output layer, then the total number of weights in the network is (256+1)*10 between the input and hidden layers, and (10+1)*3 between the hidden and output layers.

Dataset

The training dataset (name : train.txt) is handwritten digits. It contains binary images of size 16x16 that each data sample means one handwritten digit. Your task is to classify each example as one of the three possible digits: 1, 8, and 9.

You can try any online tool which converts binary digits to pixels, and see the digit corresponding to each data sample. In the data, ‘1’ means black and ‘0’ means white (or the other way around, it does not matter). One data sample has 256 binary numbers which correspond to 16x16 pixel values.

For example, the following three figures are the outputs of the tool https://studio.code.org/s/pixelation/stage/2/puzzle/1 .

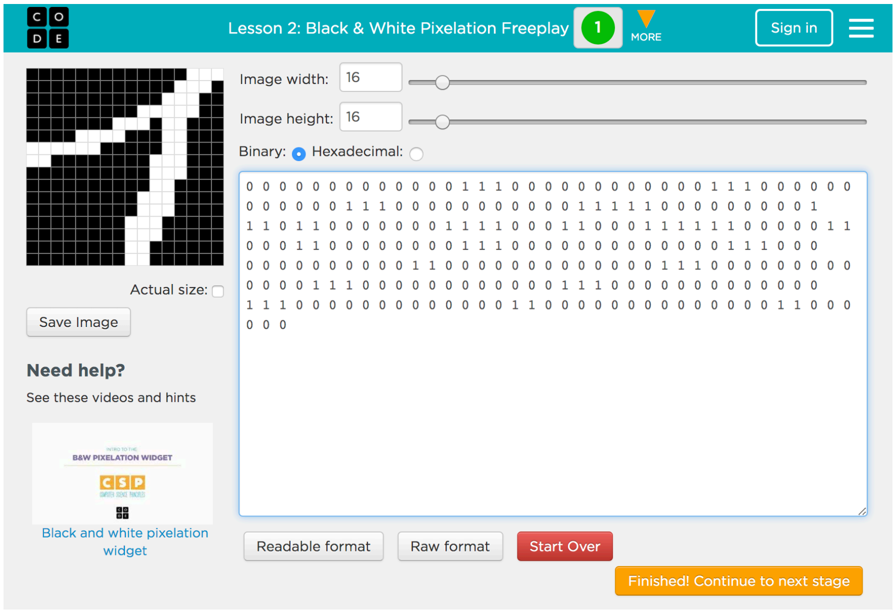
Lesson 2: Black & White Pixelation Freeplay
1
MORE
Sign in
Actual size: □
Save Image
Need help?
See these videos and hints
B&W PIXELATION WIDGET
CSP
Black and white pixelation widget
Readable format
Raw format
Start Over
Finished! Continue to next stage

Image width: 16

Image height: 16

Binary: ● Hexadecimal: ○

```
0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0
0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1
1 1 0 1 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 1 1 0 0 0 1 1 1 1 1 1 0 0 0 0 0 1 1
0 0 0 1 1 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0
0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0
0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0
0 0 0
```

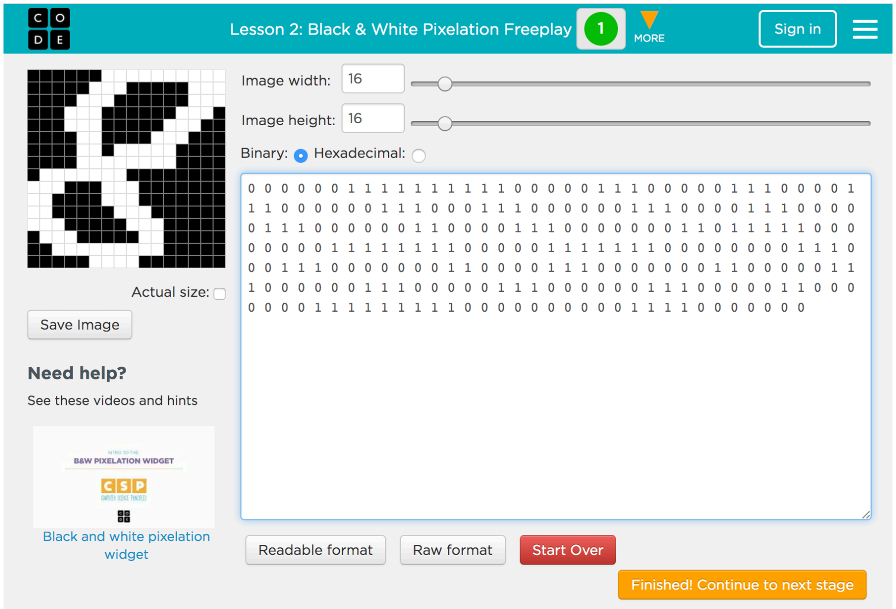
Lesson 2: Black & White Pixelation Freeplay
1
MORE
Sign in
Actual size: □
Save Image
Need help?
See these videos and hints
B&W PIXELATION WIDGET
CSP
Black and white pixelation widget
Readable format
Raw format
Start Over
Finished! Continue to next stage

Image width: 16

Image height: 16

Binary: ● Hexadecimal: ○

```
0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 1
1 1 0 0 0 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 0 0 0 0 1 1 1 0 0 0 0
0 1 1 1 0 0 0 0 0 0 1 1 0 0 0 0 1 1 1 0 0 0 0 0 0 0 1 1 0 1 1 1 1 1 0 0 0
0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 0
0 0 1 1 1 0 0 0 0 0 0 0 1 1 0 0 0 0 1 1 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 1
1 0 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 0 0 0
0 0 0 0 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0
```

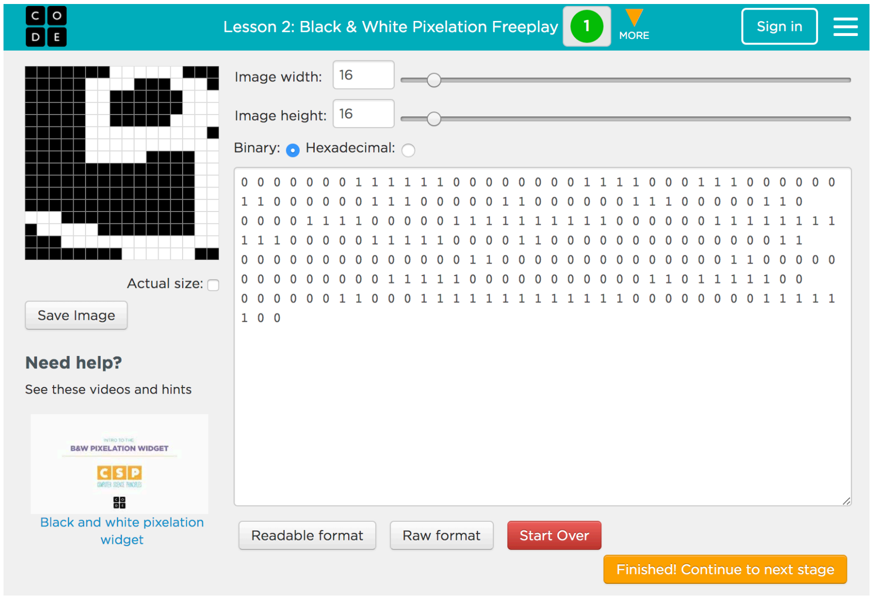
CO
DE
Lesson 2: Black & White Pixelation Freeplay
1
MORE
Sign in
Save Image
Need help?
See these videos and hints
B&W PIXELATION WIDGET
CSP
Black and white pixelation widget
Readable format
Raw format
Start Over
Finished! Continue to next stage

Image width: 16

Image height: 16

Binary: ● Hexadecimal: ○

Actual size: □

```
0 0 0 0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 1 1 1 0 0 0 0 0 0
1 1 0 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 0
0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 1 1 1 1 1 1 1 1
1 1 1 0 0 0 0 0 1 1 1 1 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1 1 0 0
0 0 0 0 0 0 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1
1 0 0
```

The figures show the digits of '1', 8', and '9' respectively.

You can assume that the number of classes will always be 3 for this homework. The first output node should produce the largest value when the instance is determined to be in class 1 (digit 1). The second output node should produce the largest value when the instance is in class 2 (digit 8), and the third output node should produce the largest value when the instance is in class 3 (digit 9). In the dataset, each line has the values of 256 attributes followed by the values for the 3 classes.

The following is an example line:

```
0.0000 0.0000 0.0000 0.0000 0.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 1.0000 1.0000 1.0000 1.0000 1.0000 0.0000 0.0000 0.0000
1.0000 1.0000 1.0000 0.0000 0.0000 0.0000 0.0000 1.0000 1.0000 1.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 1.0000 1.0000 0.0000 0.0000 0.0000 1.0000 1.0000 1.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
1.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 1.0000  1.0000
1.0000 1.0000 1.0000 1.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
1.0000 1.0000 1.0000 1.0000 0.0000 1.0000 1.0000 1.0000 0.0000 0.0000 0.0000 0.0000 0.0000
1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 0.0000 0.0000 0.0000 1.0000 1.0000 1.0000 1.0000
1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 1.0000 1.0000 1.0000 1.0000 1.0000 0.0000 0.0000 1.0000 1.0000 1.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 1.0000 1.0000 1.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
1.0000 1.0000 1.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 1.0000 1.0000 1.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 1.0000 1.0000 1.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 1.0000 1.0000 1.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 1.0000 1.0000 1.0000 0.0000
```

```
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
1.0000 1.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0 0 1
```

If the last 3 values for an instance are “1 0 0 ”, then the instance is the digit 1. Similarly, “0 1 0” means digit 8, and “0 0 1” means digit 9.

Classification

You will classify an instance as the digit of the output node with the maximum value. For example, if one instance has outputs [0.1, 0.5, 0.2], this instance will be classified as digit 8. If there is a tie between multiple values, the instance will be classified as the dominant digit in the training set. For example, if outputs are [0.1, 0.4, 0.4], it should be classed as 9, because the training data has more cases of digit ‘9’ than digit ‘8’.

Training result

I will execute your program through the following command

```
python asn3.py train.txt train_output.txt
```

“train_output.txt” should show your predicted digit and target (correct value) as following:

```
my_predicted_digit    target(correct_digit)
8                           1
1                           1
8                           8
9                           1
9                           8
…
Accuracy: 398/429 = 92.77%
```

Test result

I have provided test set ‘test.txt’ without the target information. Then, I will execute your program through the following command

```
python asn3.py test.txt test_output.txt
```

Then your program will run the constructed neural network and choose one class which produces the largest value.

The output should be written in test_output.txt. Following is the example of the test_output.txt

```
8
8
1
8
1
1
9
…
```

I have provided ‘correct_answer.txt’. Please compare your test output with correct_answer.txt, and analyze your test accuracy overall and accuracy for each digit class. Then explain your analysis in asn3.pdf.

Deliverables

Submit

(1) asn3.py,
(2) train_output.txt,
(3) test_output.txt,
(4) asn3.pdf which explains your initial weights, learning rate, the number of epochs, and your test accuracy. Please also analyze your test accuracy for each class of 1, 8, and 9.