Alessio Mazzone
CS1571 Artificial Intelligence
Homework 4

1. Use the Gaussian data (the bottom left data set where the orange and blue blobs are nicely separated), the first two features (i.e., just X1 and X2) , 1 hidden layer with 1 neuron. Roughly how many epochs does it take to reach a test loss of 0?

I ran many trials to get an estimate of how many epochs it takes for the test loss to reach 0.000. Upon running many trials with this configuration, it seems like it takes roughly 200 epochs to reach a test loss of 0.000.

2. Change to the exclusive or data (top right). After 2,000 epochs, what is the test loss? Is it correctly classifying most of the data? Does it look like it is getting much better?

After switching datasets, I let the network run for 2,000 epochs. This configuration has two input nodes, with one hidden layer and a single neuron. I ran multiple trials and found that after 2,000 epochs, the test loss was somewhere between 0.370 and 0.425.

During the many trials I ran, I saw that the network is classifying most of the data correctly. By "most", I mean here the majority, or at least 50% of the data. So, by this definition, the network is correctly identifying most, or at least 50%, of the data. I found that throughout multiple trials, the network would correctly classify almost the entirety of one label (almost all positive samples or all negative samples, depending on the trial) but then misclassify about half of the other samples correctly and half incorrectly. This means that that, for example, almost all the positive samples would be correctly classified, but then only half of the negative samples would be correctly classified, and the other half incorrectly classified.

It only takes about 200 epochs for the prediction of the network and the test loss to stabilize, and after that it only improves only slightly as it approaches 2,000 epochs. Thus, the test loss and predictions don't get much better after about only 200 epochs.

3. What is the minimum number of hidden neurons required to quickly (e.g., epochs < 1,000) get nearly all the data right (e.g., test loss < 0.04) at least two-thirds of the time?

I used the exclusive or data for this configuration and a single hidden layer. I found that with three neurons, the test loss after roughly 1,000 epochs was always somewhere between 0.05 and 0.17. The network here with three neurons categorized nearly all the data correctly. But, when I added a fourth neuron, I found that the network much more quickly categorized all the data correctly *and* the test loss was almost always below 0.04. So, I found that the minimum number of neurons required to quickly get nearly all the data categorized correctly most of the time is 4.

4. Switch to the circle data. Does your current network work well on that data? How many epochs does it take to get a test loss that is less than 0.01?

Using the circle data (top left), I used one hidden layer with four neurons. It took roughly some trails as little as 750 epochs for the test loss to fall below 0.01. While other trials took around 1,200 epochs for the test loss to fall below 0.01. There were also some trials that fell in between 750 and 1,200 epochs for the test loss to reach 0.01. The network works well for this data because all of the data looks completely correctly categorized.

5. Switch to the spiral data. Is your network working after 2,000 epochs? Try 8 hidden nodes. What is the test loss after 2,000 epochs? How many weights are in this network?

I used the spiral data, one hidden layer, and four hidden neurons for my first few trials. After a few trials, it became clear that the current network doesn't work well, even after 2,000 epochs. The test loss reaches about 0.420 to 0.450 and then stays there. It's pretty clear this configuration will not produce better results given more epochs. The data does not look correctly classified.

For my second set of trials, I continued to use the spiral data and one hidden layer, but I doubled the hidden neurons from four to eight. I tried many trials with this configuration. The test loss for this configuration was noticeably lower, but now as low as I expected it to be. After 2,000 epochs, the test loss fell to a range of 0.25 to 0.35. In some of my trials with this new configuration, the test loss would not fall below 0.40 after 2,000 epochs.

For this network, there are two inputs. Each input connects to each of the eight hidden nodes. Each hidden node then connects to the single output node. Thus, the total number of weights in this network is (2*8) + 8, or 24 total weights.

6. Try 2 hidden layers of of 4 nodes each. What is the test loss after 3,000 epochs? How many weights are in this network?

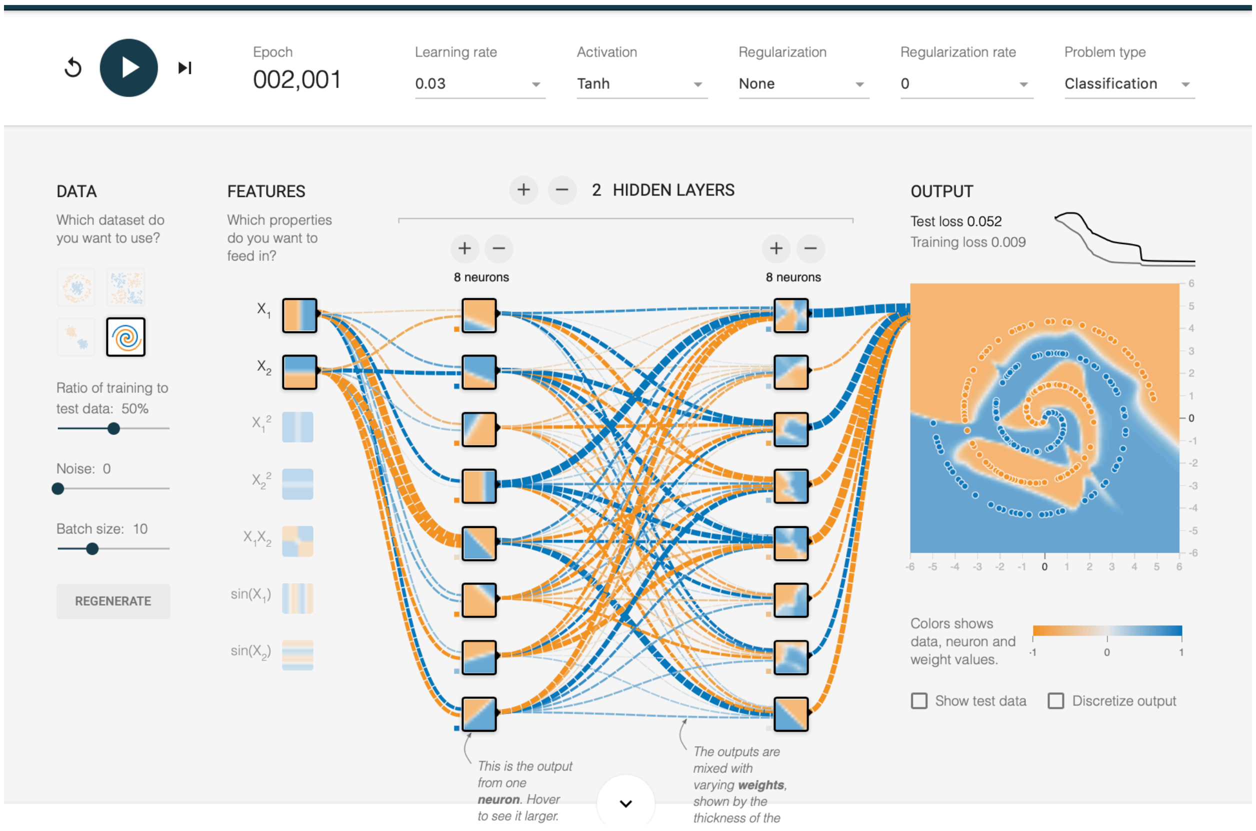
I again used the spiral data for this portion. I used two hidden layers instead of one, but had each hidden layer contain four neurons each. There was still a total of four hidden neurons in total, but now they are split evenly between two hidden layers rather than having a single hidden layer contain them all. After many test runs, the test loss after 3,000 epochs was not as consist not as I expected. Sometimes, it fell to as low as 0.12 to 0.23. Other times, the test loss only reached from 0.35 to 0.45.

This network has two input nodes. Each input node connects to each of the four neurons in the first hidden layer. Each of the four neurons in the first hidden layer connects to each of the four neurons in the second hidden layer. The four neurons in the second hidden. Finally, each of the four neurons in the second hidden layer connects to the single output node. This, the total number of weights in this network is (2*4) + (4*4) + 4, or 28 total weights in the network.

7. Try a full network; let both hidden layers have 8 nodes. What is the test loss after 2,000 epochs? How many weights are in this network? Try this configuration 3 times and make a screenshot of the best one.

For this trial, I used the spiral data still. My configuration had two hidden layers, each containing eight neurons. I ran three different trials. The first trial with this configuration gave me a test loss of 0.419 after 2,000 epochs. The second trial with this configuration gave me a test loss of 0.052 after 2,000 epochs (Screenshot included). The third trial gave me a test loss of 0.114 after 2,000 epochs.

There are two input nodes in this network, each of which connects to the first hidden layer's eight neurons. Each of the eight neurons in the first hidden layer connects to the eight neurons in the second hidden layer. Each neuron in the second hidden layer connects to a single output node. Thus, the total number of weights in this network is (2*8) + (8*8) + 8, or 88 total weights in this network.


Epoch 002,001

Learning rate 0.03

Activation Tanh

Regularization None

Regularization rate 0

Problem type Classification

DATA

Which dataset do you want to use?

Ratio of training to test data: 50%

Noise: 0

Batch size: 10

REGENERATE

FEATURES

Which properties do you want to feed in?

2 HIDDEN LAYERS

8 neurons

8 neurons

OUTPUT

Test loss 0.052

Training loss 0.009

Colors shows data, neuron and weight values.

Show test data

Discretize output

This is the output from one ***neuron***. Hover to see it larger.

The outputs are mixed with varying ***weights***, shown by the thickness of the

8. Try a deeper network. Make a network with 6 hidden layers, with 3 nodes each. What is the test loss after 2,000? How many nodes are in this network?

For this network, I again used the spiral data. I had a total of six hidden layers, each containing three nodes. I ran four separate trials with this network and ended up with the following test losses: 0.483, 0.541, 0.485, and 0.484. I predicted that perhaps having more layers and more nodes would somehow make the network better, but based on my trials, this is not the case. I thought that the deeper the network, the more accurate the categorization would be.

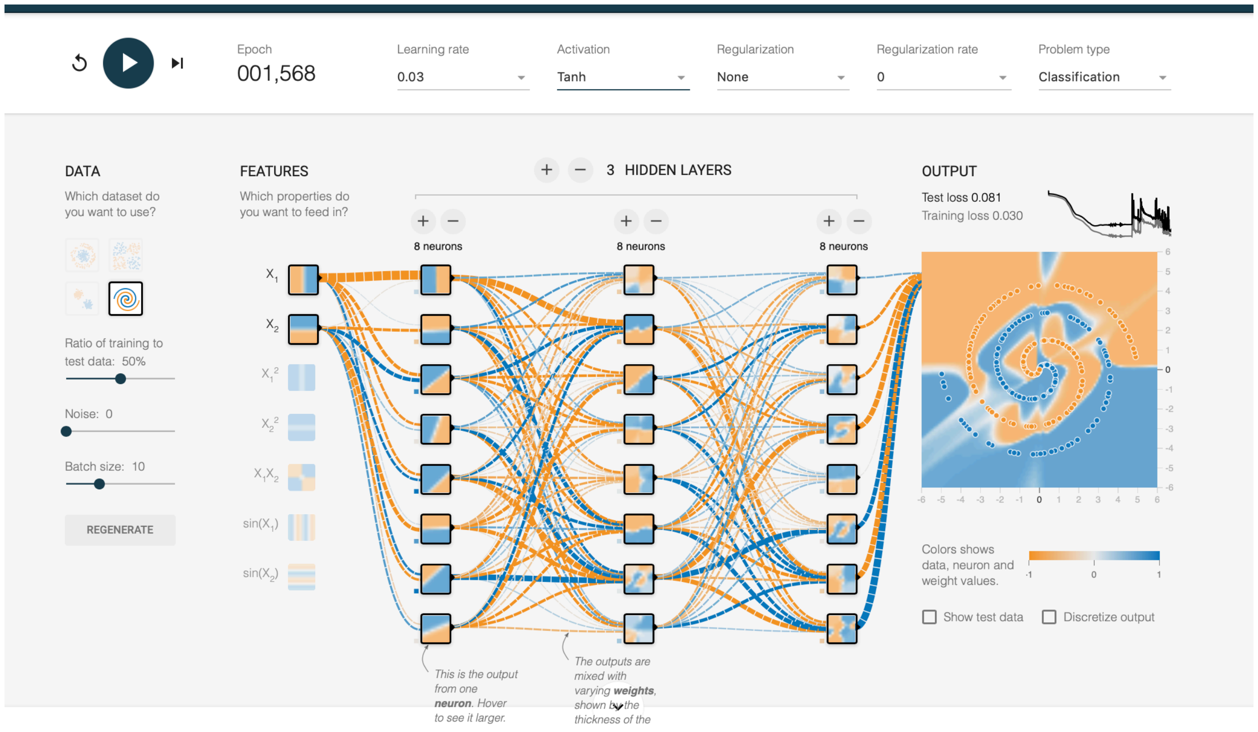
There are a total of 21 nodes in this network: 2 input nodes, 18 hidden nodes (6 layers * 3 nodes each), and one output node.

9. Using the other features, design a network that gets a test loss that is less than 0.1 within 2,000 epochs. Take a screenshot. How many weights does this network have?

For this network, I used the spiral data set. I tried a few different combinations of hidden layers and neurons but with no success. For a few tests I tried two hidden layers, each with five neurons each. I tried increasing the learning rate to something like 0.3 or 1 but the learning rate bottomed off too quickly which tells me the learning rate was too high for my network. I then tried a smaller learning rate, with the same end result.

I then tried a different configuration, still using the spiral data set. I tested with four hidden layers, each with four neurons. I changed my activation function to ReLU as opposed to Tanh and decreased my learning rate 0.01. With this trial, my network came very close, with a test loss of 0.12 after 2,000 epochs. I increased the learning rate to 0.03 in an attempt to quicken the pace of learning, but after 2,000 epochs, the network stabilized at a test loss of about 0.35.

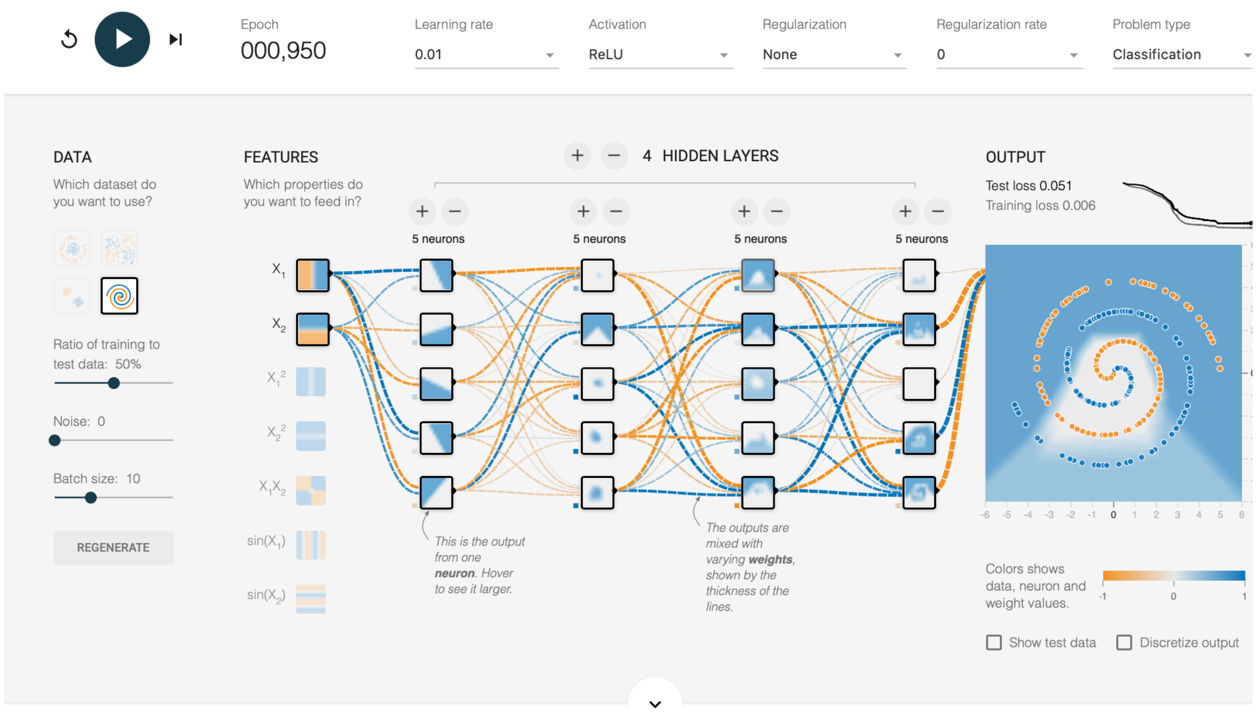
I managed to get a test loss of less than 0.1 within 2,000 on two different occasions, both of which I've included screen shots. My first case is shown below:


DATA
Which dataset do you want to use?
FEATURES
Which properties do you want to feed in?
3 HIDDEN LAYERS
8 neurons
8 neurons
8 neurons
OUTPUT
REGENERATE
Colors shows data, neuron and weight values.
Show test data
Discretize output
This is the output from one neuron. Hover to see it larger.
The outputs are mixed with varying weights, shown by the thickness of the

Epoch 001,568
Learning rate 0.03
Activation Tanh
Regularization None
Regularization rate 0
Problem type Classification
Ratio of training to test data: 50%
Noise: 0
Batch size: 10
Test loss 0.081
Training loss 0.030

In this case, my configuration was very similar to question number seven. My network has two inputs. It also has three hidden layers, each with eight neurons. I used a learning rate of 0.03 and Tanh as the activation function. It took longer than I expected to reach a test loss below 0.1, given that I added an entire extra hidden layer. But, the categorization looks correct and the criteria for the question were met.

There was a second configuration that also worked fairly quickly, pictured below:



Epoch 000,950

Learning rate 0.01

Activation ReLU

Regularization None

Regularization rate 0

Problem type Classification

Which dataset do you want to use?

Ratio of training to test data: 50%

Noise: 0

Batch size: 10

Which properties do you want to feed in?

This is the output from one *neuron*. Hover to see it larger.

The outputs are mixed with varying *weights*, shown by the thickness of the lines.

Test loss 0.051

Training loss 0.006

Colors shows data, neuron and weight values.

This configuration has two input nodes. It also has four hidden layers, each with five neurons. Also, I decreased the learning rate from 0.3 to 0.1, and changed the activation function from Tanh to ReLU. This configuration fell below a test loss of 0.1 very quickly, much faster than I expected. But, looking at the categorization, it doesn't look accurate at all, which is very strange. It looks as though almost all the orange samples were incorrectly classified. Yet, this network's test loss fell well below 0.1 in less time than my first network that worked.