Model Answer Exercise May 1st

Unless otherwise indicated, we work with the following hyper-parameters:
Learning rate =0.03, Batch Size=10, Noise=0, Training Data=70%.
We examine the behaviour of the networks until about 2500 iterations.

It is very important to note that there will be differences between different runs obtained with the same hyperparameters, because the initial values of the neural network parameters - or weights - are sampled randomly each time (or by ticking to Regenerate button). The comments below characterize the most frequent behaviour we tend to observe.

1. **Start by using a linear activation function and any number of hidden layers and neurons. What do you observe? How do you interpret it?**

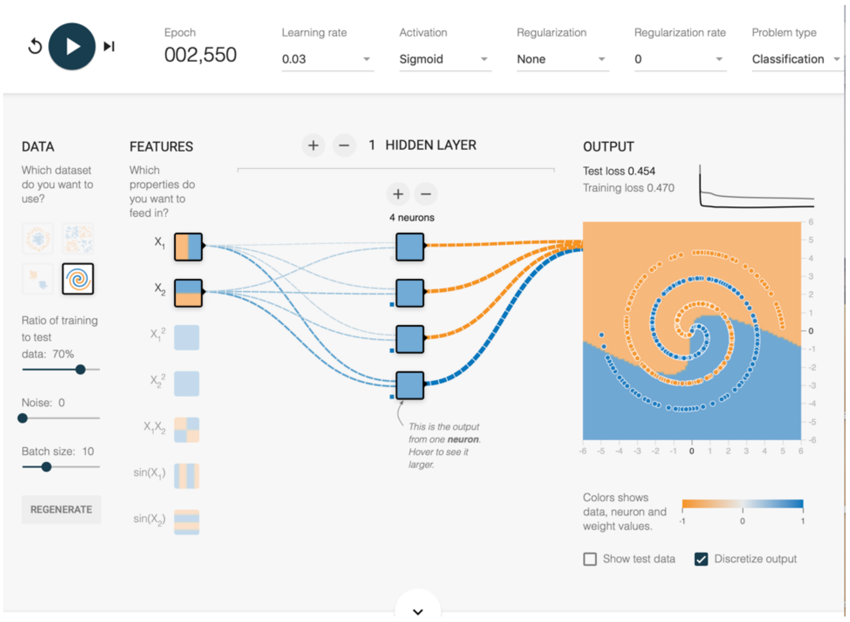
   With a linear activation function, the neural network itself remains a linear function. It cannot do better than build a linear decision boundary.

2. **Now use the sigmoid activation function and no hidden layer. What do you observe? How do you interpret it?**

   With a sigmoid activation function and no hidden layer, we are exactly in the situation of logistic regression. We know that with a sigmoid function the decision boundary is a line.

3. **Now put just one hidden layer, a sigmoid activation function, and make the number of neurons in the hidden layer equal to 4. Then make it equal to the maximum offered by the application, that is 8. What do you observe in both cases?**

   With 4 neurons in the hidden layer, we see that after 2500 epochs a slight non-linearity appears in the Decision Boundary, but on some of the obtained models, there is no progress at all.


Epoch 002,550 · Learning rate 0.03 · Activation Sigmoid · Regularization None · Regularization rate 0 · Problem type Classification

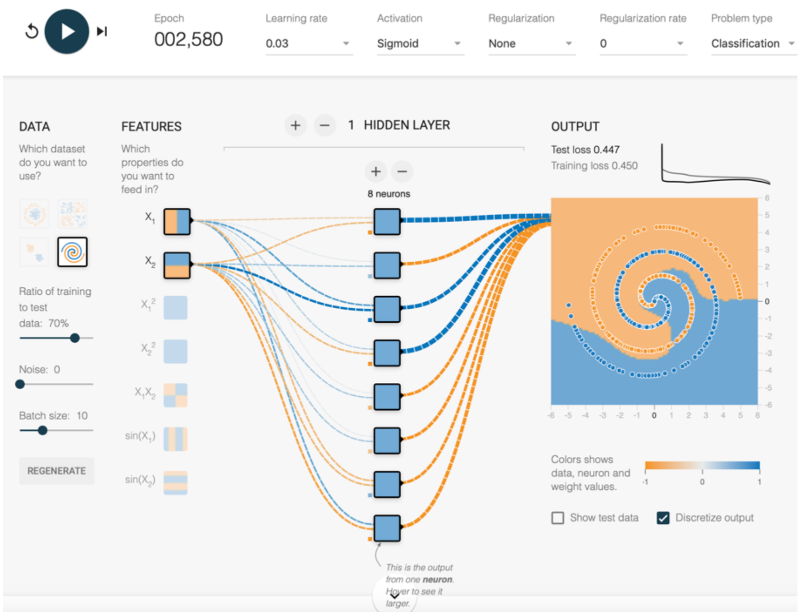
Ratio of training to test data: 70% · Noise: 0 · Batch size: 10

1 HIDDEN LAYER · 4 neurons

Test loss 0.454
Training loss 0.470

DATA
Which dataset do you want to use?
REGENERATE
FEATURES
Which properties do you want to feed in?
OUTPUT
This is the output from one neuron. Hover to see it larger.
Colors shows data, neuron and weight values.
Show test data
Discretize output

- With 8 neurons in the hidden layer, and the sigmoid activation function, we see a slow evolution of the previous non-linearity, and the Decision Boundary remain simplistic. Sigmoid activations are easier to saturate: once a sigmoid reaches either its left or right plateau, it leads to derivatives that are very close to 0.


Epoch 002,580 · Learning rate 0.03 · Activation Sigmoid · Regularization None · Regularization rate 0 · Problem type Classification

Ratio of training to test data: 70% · Noise: 0 · Batch size: 10

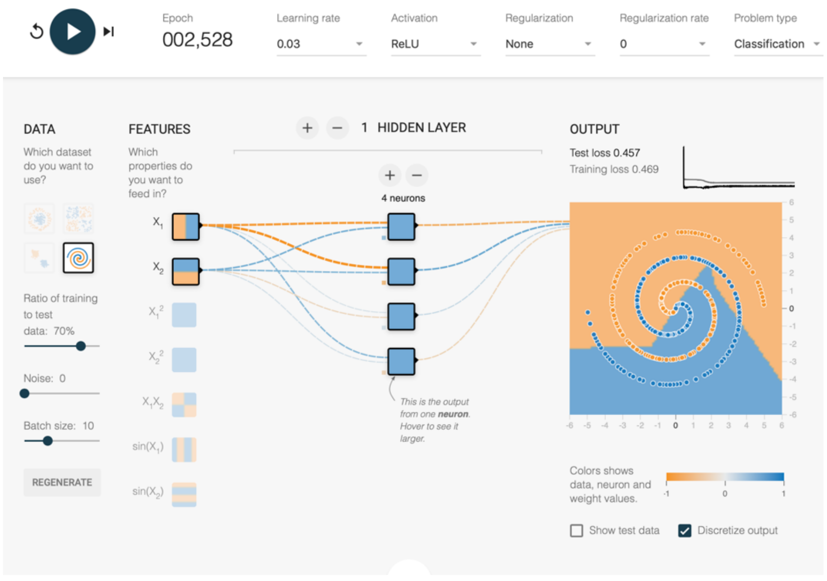
1 HIDDEN LAYER · 8 neurons

Test loss 0.447
Training loss 0.450

DATA
Which dataset do you want to use?
REGENERATE
FEATURES
Which properties do you want to feed in?
OUTPUT
This is the output from one neuron. Hover to see it larger.
Colors shows data, neuron and weight values.
Show test data
Discretize output

4. **Now do the same as in question 3, but using the ReLU activation function. What do you observe?**

   The changes are faster and more apparent with the ReLU, because we do not have the saturation effects of the sigmoid function. But the result is still disappointing after 2500 epochs, both with 4 neurons or 8 neurons in the hidden layer. Clearly, more hidden layers are needed to model such a complex image.


Epoch 002,528
Learning rate 0.03
Activation ReLU
Regularization None
Regularization rate 0
Problem type Classification

DATA
Which dataset do you want to use?
Ratio of training to test data: 70%
Noise: 0
Batch size: 10

FEATURES
Which properties do you want to feed in?

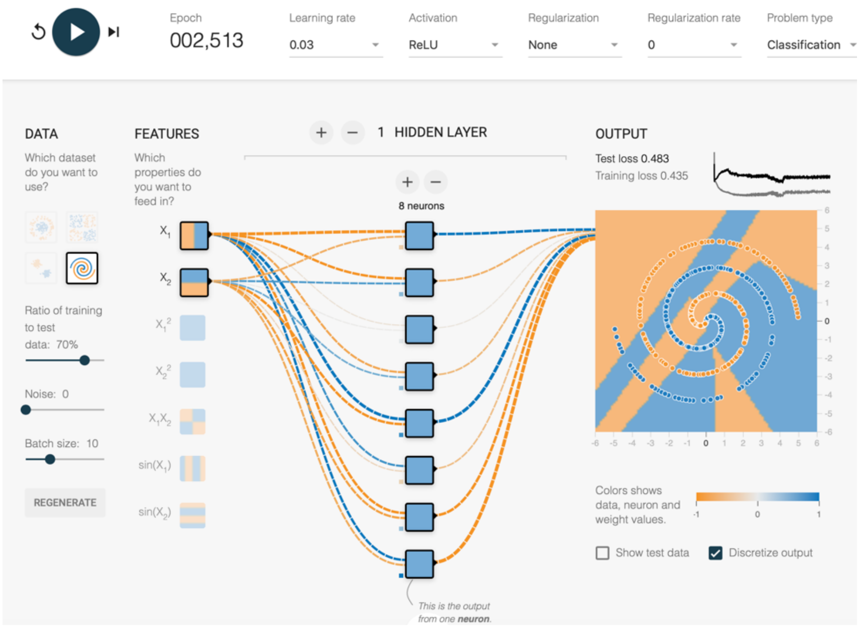
1 HIDDEN LAYER
4 neurons

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

OUTPUT
Test loss 0.457
Training loss 0.469

Colors shows data, neuron and weight values.

Show test data
Discretize output


Epoch 002,513
Learning rate 0.03
Activation ReLU
Regularization None
Regularization rate 0
Problem type Classification

DATA
Which dataset do you want to use?
Ratio of training to test data: 70%
Noise: 0
Batch size: 10

FEATURES
Which properties do you want to feed in?

1 HIDDEN LAYER
8 neurons

This is the output from one **neuron**.

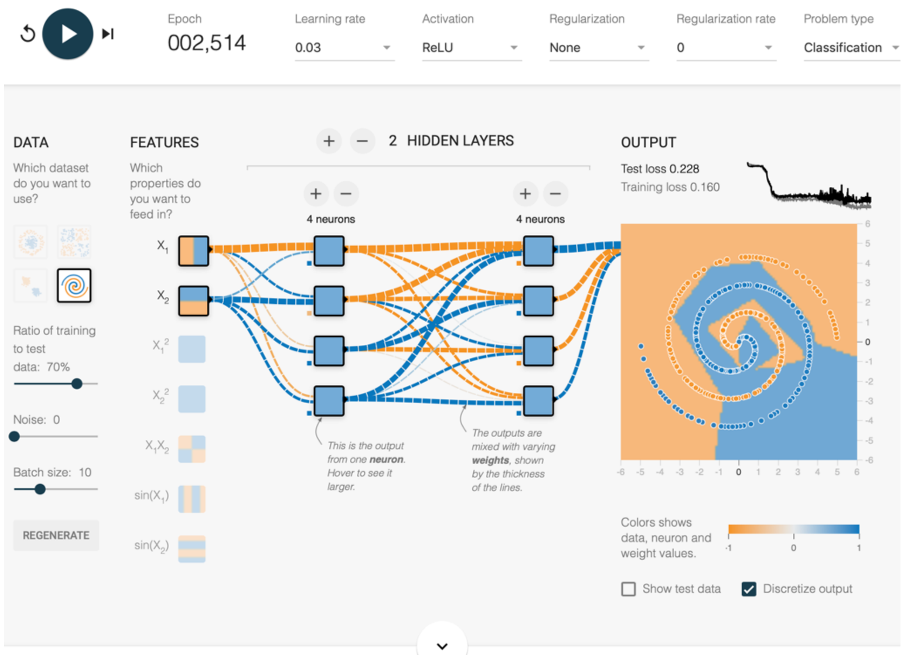
OUTPUT
Test loss 0.483
Training loss 0.435

Colors shows data, neuron and weight values.

Show test data
Discretize output

5. **Now use two hidden layers each with four neurons. What do you observe?**

We observe little improvement with the sigmoid (the result, not shown below, is very similar to that of the one hidden layer case), but there is more change occurring with the ReLU activation function, and now a better-looking shape appears. The Training Loss is much lower than before. Adding a second hidden layer has significantly improved the results.


Epoch 002,514

Learning rate 0.03

Activation ReLU

Regularization None

Regularization rate 0

Problem type Classification

DATA

Which dataset do you want to use?

Ratio of training to test data: 70%

Noise: 0

Batch size: 10

REGENERATE

FEATURES

Which properties do you want to feed in?

2 HIDDEN LAYERS

4 neurons

4 neurons

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.

OUTPUT

Test loss 0.228

Training loss 0.160

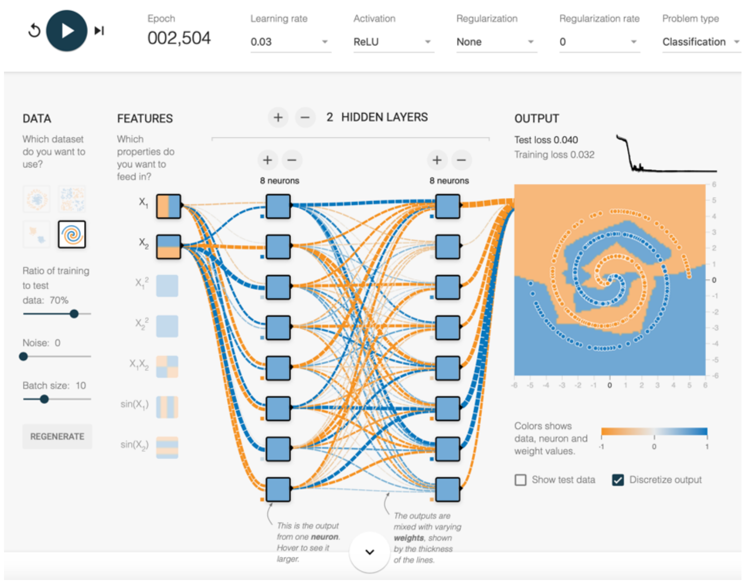
Colors shows data, neuron and weight values.

Show test data

✓ Discretize output

6 Now try two hidden layers with 8 neurons each and a sigmoid activation function. What do you observe ?

The sigmoid provides results which are not much improved compared to those obtained with one hidden layer only. However, ReLU provides a more and more satisfactory result, with a significantly decreased Training Loss and Test Loss.


Epoch 002,504

Learning rate 0.03

Activation ReLU

Regularization None

Regularization rate 0

Problem type Classification

DATA

Which dataset do you want to use?

Ratio of training to test data: 70%

Noise: 0

Batch size: 10

REGENERATE

FEATURES

Which properties do you want to feed in?

2 HIDDEN LAYERS

8 neurons

8 neurons

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.

OUTPUT

Test loss 0.040

Training loss 0.032

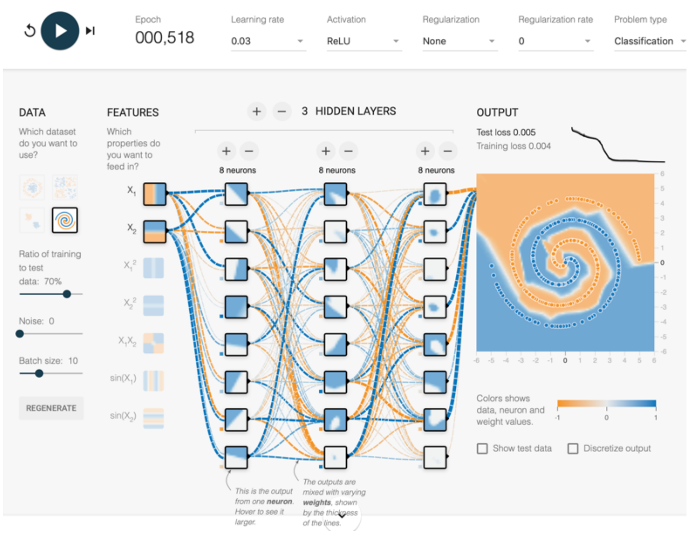
Colors shows data, neuron and weight values.

Show test data

✓ Discretize output

7. Now try three hidden layers with 8 neurons each. What happens if you compare sigmoid and ReLU?

With three hidden layers of 8 neurons each, the ReLU does much better than the sigmoid (which shows little progress) after just about 500 epochs, with both a low Test and Training Loss. It is really interesting (by de-activating the “Discrete Output” visualization) to see the maps corresponding to each neuron of the second and third hidden layers for the ReLU, where some very strongly non-linear shapes appear. The fact that ReLU does consistently better than sigmoid explains why ReLU is almost always preferred, except for the last layer of the network. There does not appear to be over-fitting on this example as Training and Test Loss are not very different. This is a good result.


Epoch 000,518
Learning rate 0.03
Activation ReLU
Regularization None
Regularization rate 0
Problem type Classification

DATA
Which dataset do you want to use?
Ratio of training to test data: 70%
Noise: 0
Batch size: 10

FEATURES
Which properties do you want to feed in?

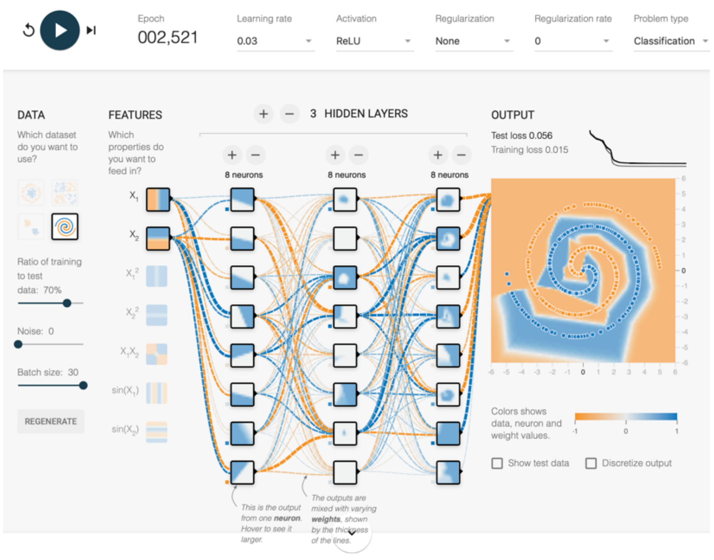
3 HIDDEN LAYERS
8 neurons
8 neurons
8 neurons

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.

OUTPUT
Test loss 0.005
Training loss 0.004
Colors shows data, neuron and weight values.
Show test data
Discretize output

8. **Now keep the three layers with 8 neurons and change the batch size to 30. What happens if you use ReLU?**

On the example below, we observe a slower progression of the ReLU Training Loss, requiring about 2500 epochs to reach a Training Loss still higher than the previous one. There seems to be also some over-fitting as the Test Loss is three times the Training Loss.
Following this morning’s discussion, if we use a batch size of 1, we do stochastic gradient descent. We see (result not shown below) that the Training Loss can evolve in a very chaotic manner: it can significantly increase after reaching a minimum, because of the impact of individual data points on the gradient used and hence on the Training Loss after the associated iteration.


Epoch 002,521

Learning rate 0.03

Activation ReLU

Regularization None

Regularization rate 0

Problem type Classification

DATA

Which dataset do you want to use?

Ratio of training to test data: 70%

Noise: 0

Batch size: 30

FEATURES

Which properties do you want to feed in?

3 HIDDEN LAYERS

8 neurons 8 neurons 8 neurons

OUTPUT

Test loss 0.056

Training loss 0.015

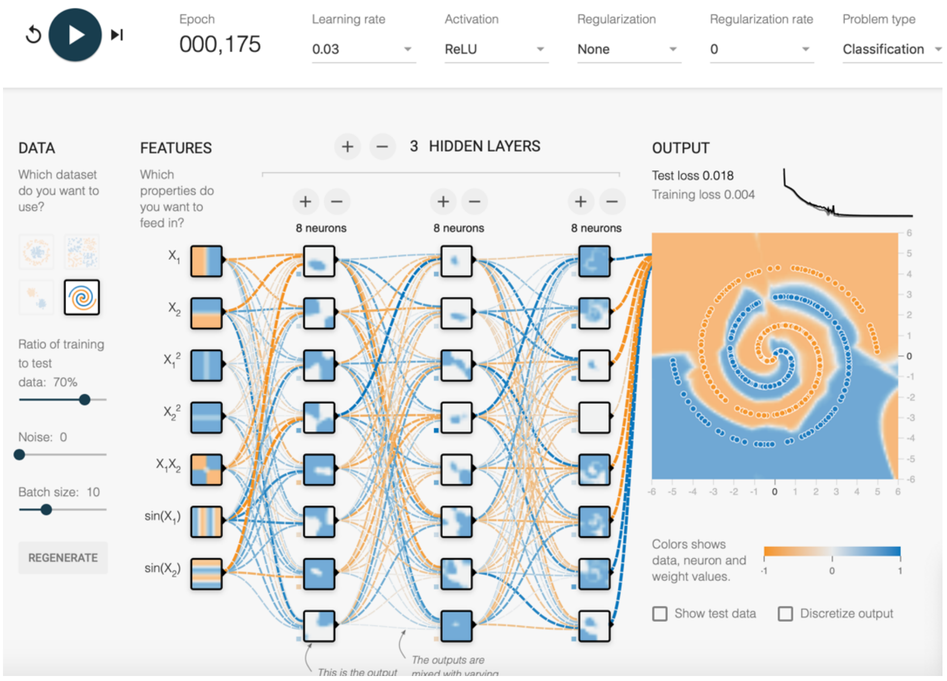
Colors shows data, neuron and weight values.

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.

9. **Now that we have obtained a good model see what happens if you introduce all the 7 input variables : $\boldsymbol{X_1, X_2, X_1^2, X_2^2, X_1X_2, \sin X_1, \sin X_2}$.**

We see that in less than 200 epochs the sigmoid converges to a very good model with a Training Loss identical to that of question 7. But the Test Loss is more than four times the Training Loss. The use of seven instead of two input features drastically increases the number of parameters and hence the risk of over-fitting.


Epoch 000,175

Learning rate 0.03

Activation ReLU

Regularization None

Regularization rate 0

Problem type Classification

DATA

Which dataset do you want to use?

Ratio of training to test data: 70%

Noise: 0

Batch size: 10

FEATURES

Which properties do you want to feed in?

3 HIDDEN LAYERS

8 neurons 8 neurons 8 neurons

OUTPUT

Test loss 0.018

Training loss 0.004

Colors shows data, neuron and weight values.

This is the output

The outputs are mixed with varying