Jared Kelnhofer
Machine Learning Project 5
April 8, 2020

1)
  a)
    i) The output would be 0
    ii) The output would be 17
  b) Because 0 is not a negative number, the output can never be negative.
  c) If the inputs are vector A, and the weights are Vector B, then the sum of all inputs multiplied by their respective weights could be represented as the dot product of A and B. We can perform ReLU by simply using the function: f(x) = {x=x, x>=0
     x=0, x<0

2)
  a) There are two classes, "positive" and "negative."
  b) The default learning rate is 0.03, and it seems to work pretty well with most configurations of the network.
  c) There are 2 default input features, and it looks like the other features are simply the results of mathematical operations done on the two default features. This seems almost like creating brand-new features out of thin air, similar to when we chose "rooms per household" as a more useful feature than "rooms per sample area" during our time with the California housing dataset.
  d) I think that the "discretize output" button is toggling whether or not we want to see what a given instance **will** be predicted as, instead of how certain the network is of that prediction. It's sort of showing us the areas where it thinks all the features live with hard-drawn boundaries.
  e) The test data is emphasized with a more saturated color, and with a black outline. There is also (usually) a lot less of testing data than training data.
  f) Because it is a binary classifier, the untrained network should have around a 50% chance of guessing correctly each time. Because of this, we can expect 50% accuracy and loss at the beginning.

g) I went with “test loss” for this question, and the average seems to be around 50 epochs.

h) Increasing the number of neurons helped in certain cases, as well as choosing which features to use/discard. The learning rate can greatly reduce the amount of epochs needed to create a clean model, if it is high enough.

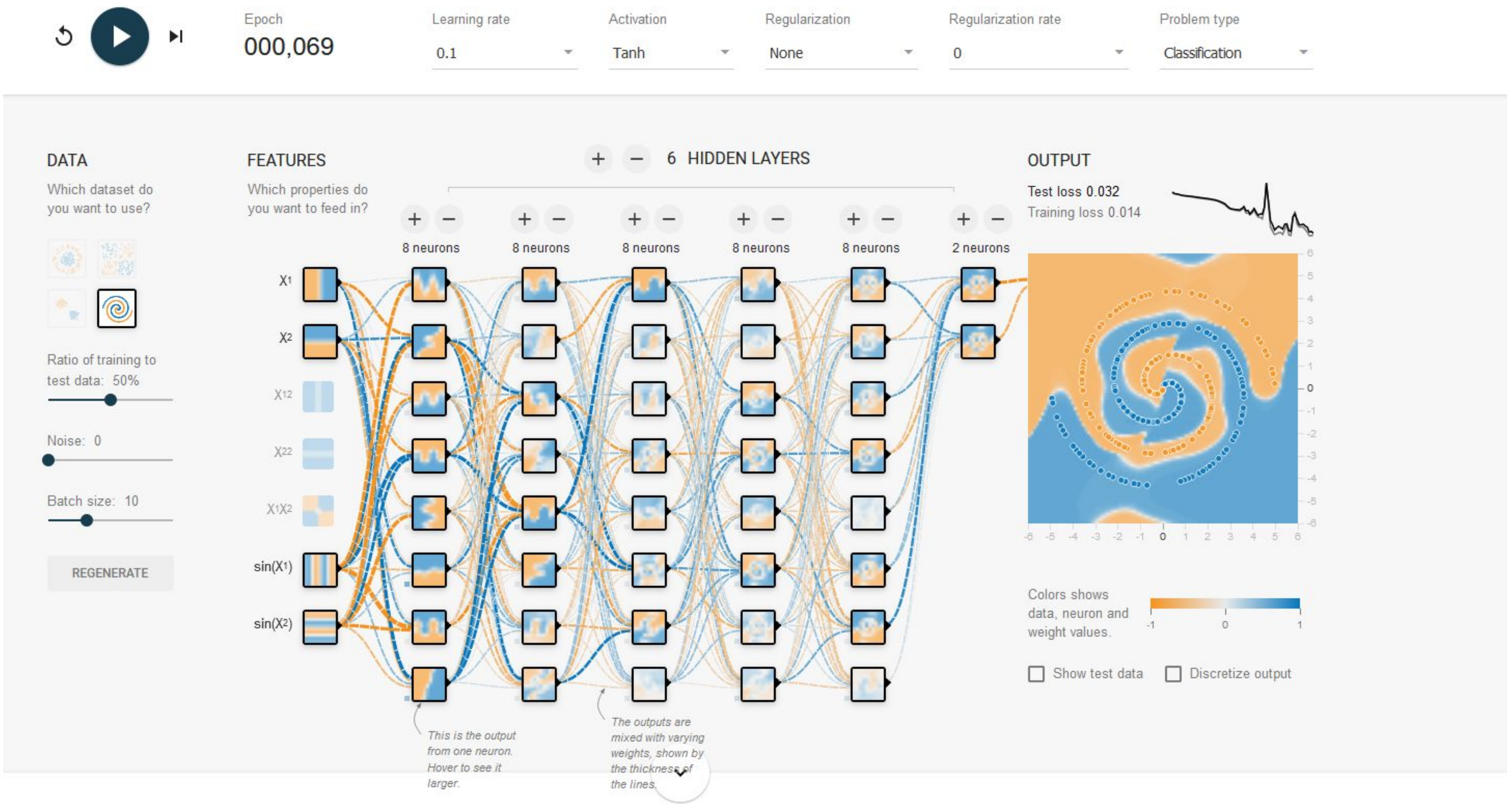
i) The hardest one for me was the dataset that looked like a jelly roll. That dataset needed a huge degree of flexibility, and when the learning rate was too high, the model would go all over the place.

j) They show the areas in X1/X2 space that would register as either the negative of positive class if that particular input was an output neuron. It’s to give the viewer an idea of what that node is doing in the background. We don’t need all of them to be 2D, because sometimes the node will output the negative or positive class regardless of what is fed into it. (For example, X1^2 will always output a positive number. (with the exception of a 0 X1))

k) The images on the hidden layer nodes seem to show the feature combinations that would result in a positive/negative output if that node were an output node. Unlike the input nodes, however, these nodes can gain a high level of complexity, depending on how the weights and nodes preceding them have been trained.

l) I ended up using the Tanh activation function, because I was able to achieve the best results all around with that. I also used

m) This is the best network I could come up with. I judged network quality based mostly on how long the network took to train. Some simpler networks could be made that achieve the same accuracy levels, but those generally took a lot longer to train. This whole process was sort of like stumbling around in the dark, but after a while I figured out which features seemed to be more important than others, and which learning rates were helpful.



Epoch 000,069

Learning rate 0.1

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?

X_1

X_2

X_1^2

X_2^2

X_1X_2

$\sin(X_1)$

$\sin(X_2)$

6 HIDDEN LAYERS

8 neurons

8 neurons

8 neurons

8 neurons

8 neurons

2 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 0.032

Training loss 0.014

Colors shows data, neuron and weight values.

Show test data

Discretize output