Neural and Adaptive Systems

ANN Playground

# Project Overview:

For this activity we’ll be using online resources that require no special setup.  
This exercise is designed to provide you with a chance to see Neural Network based learning models in action and feed your imagination for future study, research, and application in Ai.

**(Activity)**

Open the page and follow the instructions below: [https://playground.tensorflow.org](https://playground.tensorflow.org/)

# Objectives:

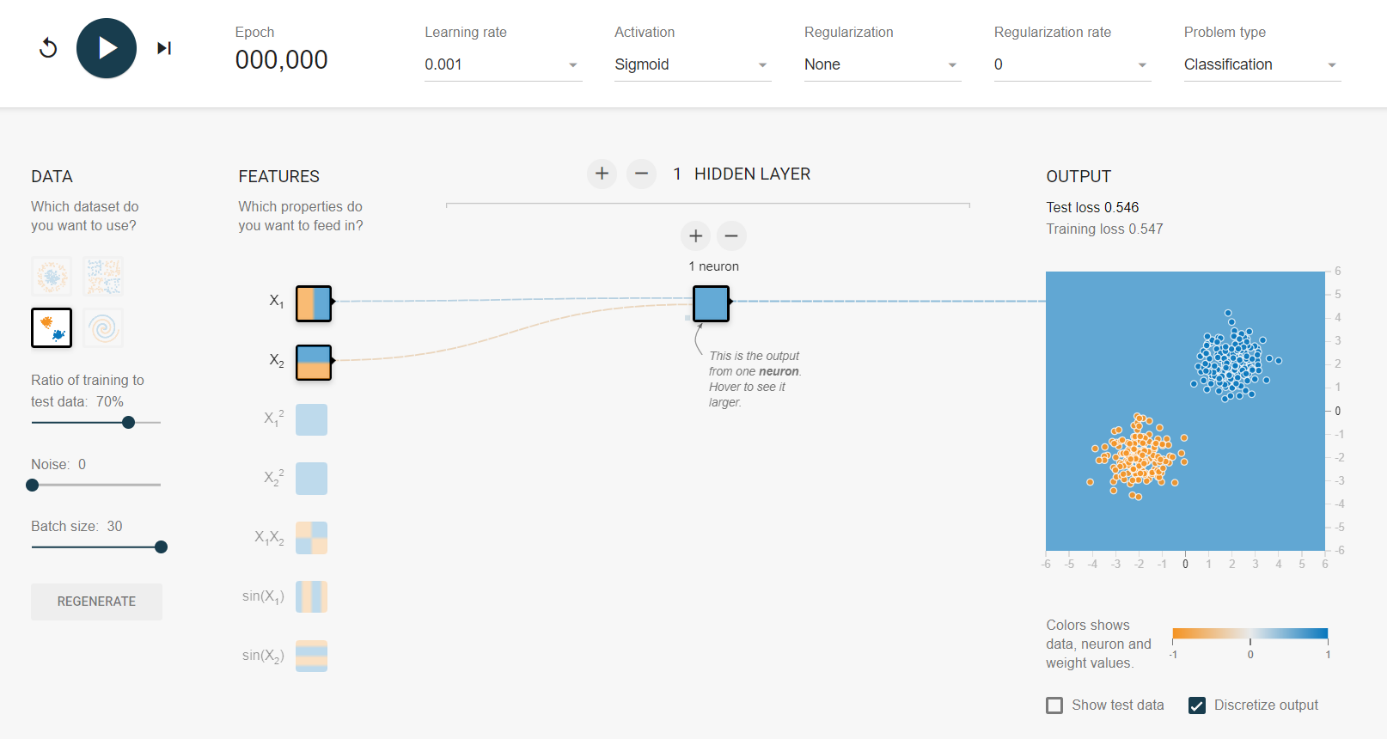
* Visualize and connect concepts related to classification and regression tasks for Perceptron and Multi-Layer Perceptron networks.
* Select Training Parameters and Network structure that fits selected data
* Analyze and Explain training events as they relate to error, neuron activations and optimizations (learning rates and adaptations).

# Requirements/Task(s):

Part 1: Step by Step Training

Open a browser window and navigate to the web address provided in the overview.

On this page you set and see all the parameters we’ll be working with:



Please select the parameters for the first set of training runs as shown:

‘TRAINING’ (top menu):   
Learning Rate – 0.001

Activation – Sigmoid

Regularization – None

Regularization Rate – 0

Problem Type – Classification

DATA (left menu):

Gaussian (Bottom Left)

Ratio – 70% (remainder will be used for testing – no Validation is used by this training)

Noise – 0

\*Batch Size – 30

FEATURES:

Select X1 and X2 raw data as the features.

HIDDEN LAYERS and NEURONS:

**1** Hidden Layer and **1** Neuron

OUTPUT:

There are two boxes at the bottom here – You can **Select** Discretize Output to better visualize the training process.

Graphical user interface, application

Description automatically generated*\*Quick Note: A ‘****batch’*** *is a small random group of training examples that are fed in together as a mini training set. The network runs one forward propagation, calculates a back propagation and updates the weights using the output. In this way, weights are updated more often then they would be using the full training set in hopes of faster overall learning (convergence to a stable predictive network).*

Startup:

Step by Step training -

Using the controls in the upper left -- step forward (shown highlighted in red above) and see how many steps it takes for the error of the resulting model to reach a value **lower** than **0.010** on the **testing** data [Test Loss above the graph of the right].

Observe the output as the samples are classified by the decision surface between the highlighted regions.  
**Record** the number of Steps (epocs) needed and **RESET** (shown highlighted in blue above) the model. Adjust the learning rate to the next value (0.003) and repeat the process. Do this for all values up to a learning rate of 1.

Try another **activation** function, selecting different **features**, adding **noise**, and adding **nodes** and **layers**. Select another **dataset** and try again with the same values. When you feel ***comfortable***, move on to the tasks below.

For the remainder of the exercise you’ll need to **experiment** a bit to select the best parameters and the features to give each network. If you find it difficult to train all networks to perform perfectly, when adding noise for instance, be sure to start with lower noise values, and add nodes or layers as you increase the complexity of separating the data.

Sometimes you will get better performance from increasing or changing the features in the same network – try adding and subtracting features while using a higher learning rate to do a quick test.

General Guideline: Add layers only when adding neurons isn’t generating any improvement. Remove some neurons when adding a layer to simplify the features you are sending to the second layer – too many features can generate problems making quality gradient estimates.

**Be Aware: Epochs can be higher than 10,000 in some cases and training will take several minutes!**

Try Adjusting

* **Activation Function**
* **Learning Rate – You can do this while training! (annealing?)**
* **Features**
* **Neurons and Layers  
  Experiment a bit!**

**Each of the following tasks are suggestions – Make up your own values if you like, this is just to encourage exploration. You might not know what you’re adjusting, but you’ll get a “feel” for it if you pay attention.**

**Task 1**:

Noise and Stability:   
Using the **Circle** data for this task, and adjust the training division to 60%

Noise increase of **5** per configuration.  
Modify the network as necessary so that your training results in a **Testing loss < 0.17**Try about **5** configurations -- if your models don’t converge (achieve stable performance), just indicate what you think you might change next.

**Task 2**:

Complexity:

Using the **Gaussian** data set, adjust the noise value to 35, and adjust the training division to 80%

Generate a network with at least a **testing loss < 0.15**.   
Try increasing your **learning rate** and letting your network run for a few hundred epochs to see if the performance is improving.   
(Maybe change learning rate while it’s running to see what happens!)

Try about **5** configurations -- if your models don’t converge (achieve stable performance), just indicate what you think you might change next.

**Task 3:**

Using the **Spiral** dataset, with a noise factor of 35 or greater and adjust the training division to 50%

Build a network that generates at a **testing loss (approx.) < 0.10**.   
\*Note the steps you took, and the parameters you used.  
Start with smaller batch sizes (around 10) to see how the network is learning early on. Also -- sometimes testing loss will drop even after going up for a while (don’t give up too early).  
Be sure to try different activation functions, training ratios as well.

Try about **10** configurations -- if your models don’t converge (achieve stable performance), just indicate what you think you might change next.

Watch the simulation of the surface for a spiral like the one you’re working with at the link below.

<https://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html>

# Use the File (Task Record) to make notes on the parameters and results of your training. You can print them and fill them in by hand to make it easier.

# Deliverable:

Record your \*observations for the exercise **(~ 2 paragraphs).**

Include your response to the following questions:

1. Did adjusting the learning rate make a difference?
2. Can you tell the difference between activation functions?
3. Did adding layers have the impact you expected?
4. At high learing rates, what did you notice about the output? Was the output stable?

Submit via Canvas