Tensorflow Playgrounds

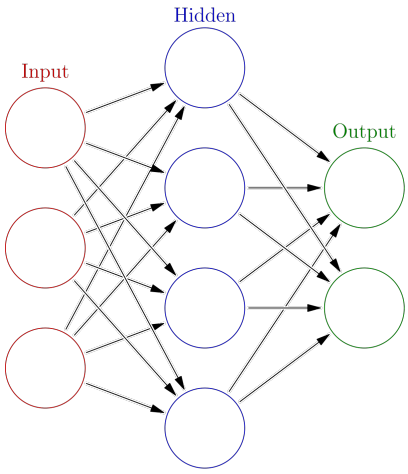
StartOnAI Deep Learning Tutorial 1:

This tutorial helps users to get acquainted with basic deep learning concepts and to understand the process of training and tuning a network

1. What is Deep Learning?

Deep Learning is essentially a branch of machine learning which focuses on learning underlying features in data sets using neural networks (Artificial Neural Networks - ANN). It's shown to have high potential since it is very accurate and efficient as its performance with larger data tends to increase rather than plateau. In simple terms, deep learning can be viewed as "learning from examples". Deep learning teaches a computer how to predict and classify information by filtering inputs through layers, similar to how a brain filters information and is part of the family that learns data representations. Areas such as image recognition, sound recognition, recommender systems have had huge successes with deep learning algorithms and processes and will continue to make new progress.

2. An Investigative Approach to Neural Networks

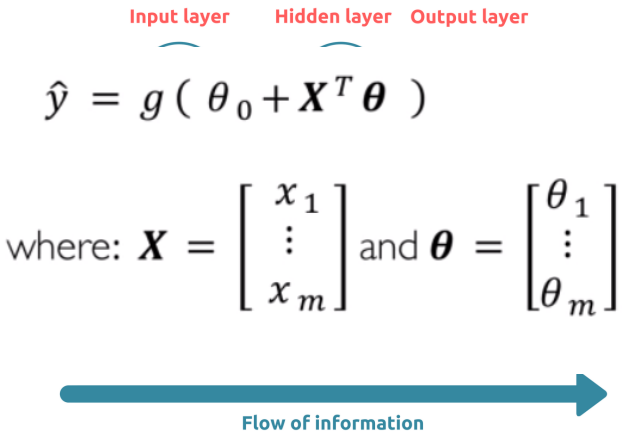


The core idea of deep learning, artificial neural networks, is representative, though not exactly of course, of a neuron inside a human brain. These "neurons" or "Perceptrons" are interconnected within an artificial neural network and are organized into three types of layers - input, hidden, and output layers. You can think of each of the circles as individual neurons that pass on signal from one another. Since the input layer has three neurons (three nodes), it signifies that 3 dimensions are required as an input to

this particular network and since there are two nodes in the output layer, there are two possible outputs for each input. Some of the common layers present in a neural network include:

- Dense Layers
- Convolutional Layers
- Pooling Layers
- Normalization Layers
- Recurrent Layers

These different layers have different roles in the neural network. The stack of hidden layers is referred to as a Multi-Layer Perceptron. The flow of information through a neural network is from left to right as shown in the image above. Each connection also has a weight (numerical value) attached to it and when information or input passes through a node, it will be multiplied by that number. The next node will compute that passed input using an activation function. This will continue until the output layer is reached.



Important Vocab To Know:

- *Activation Function* - This is a function that maps a node's input ot it's particular output

  The Sigmoid, Relu functions, and Hyperbolic are examples of functions built out of certain mathematical techniques

  -The goal of activation functions is to introduce non-linearities and are present in the hidden layers since most data in the world is complex and cannot be classified linearly

3. Training and Tuning Neural Networks

Training and tuning refers to figuring out the accurate weights that can be placed in between the neural network that form the basis for the various connections. Mapping plays a key role in ANN training and will test if the inputs map to the correct output. Network weights need to be optimized during training and an important calculation for loss function must also be done.

Loss Functions - During training, loss will be calculated using the network's output predictions and true labels for the corresponding input. The error is calculated by subtracting your input numerical value from the output numerical value. The *Mean Squared Error* is a procedure which takes the mean of all the sample errors squared. New weights can be computed and updated into the neural network using the following formula:

new weight = old weight- (learning rate * gradient).

*The gradient (slope) of the loss function is calculated with respect to each of the weights within the network

*The learning rate is a small number ranging between 0.0001 - 0.01

4. So… What are Tensorflow Playgrounds?

A TensorFlow PLayground is basically a software application that makes use of Google's TensorFlow machine learning libraries. It's an application that allows individuals to experiment with deep learning and neural networks. The software is built using Typescript using ds3.js. Through this application, individuals are able to create experiments, run small-scale neural networks and obtain results.

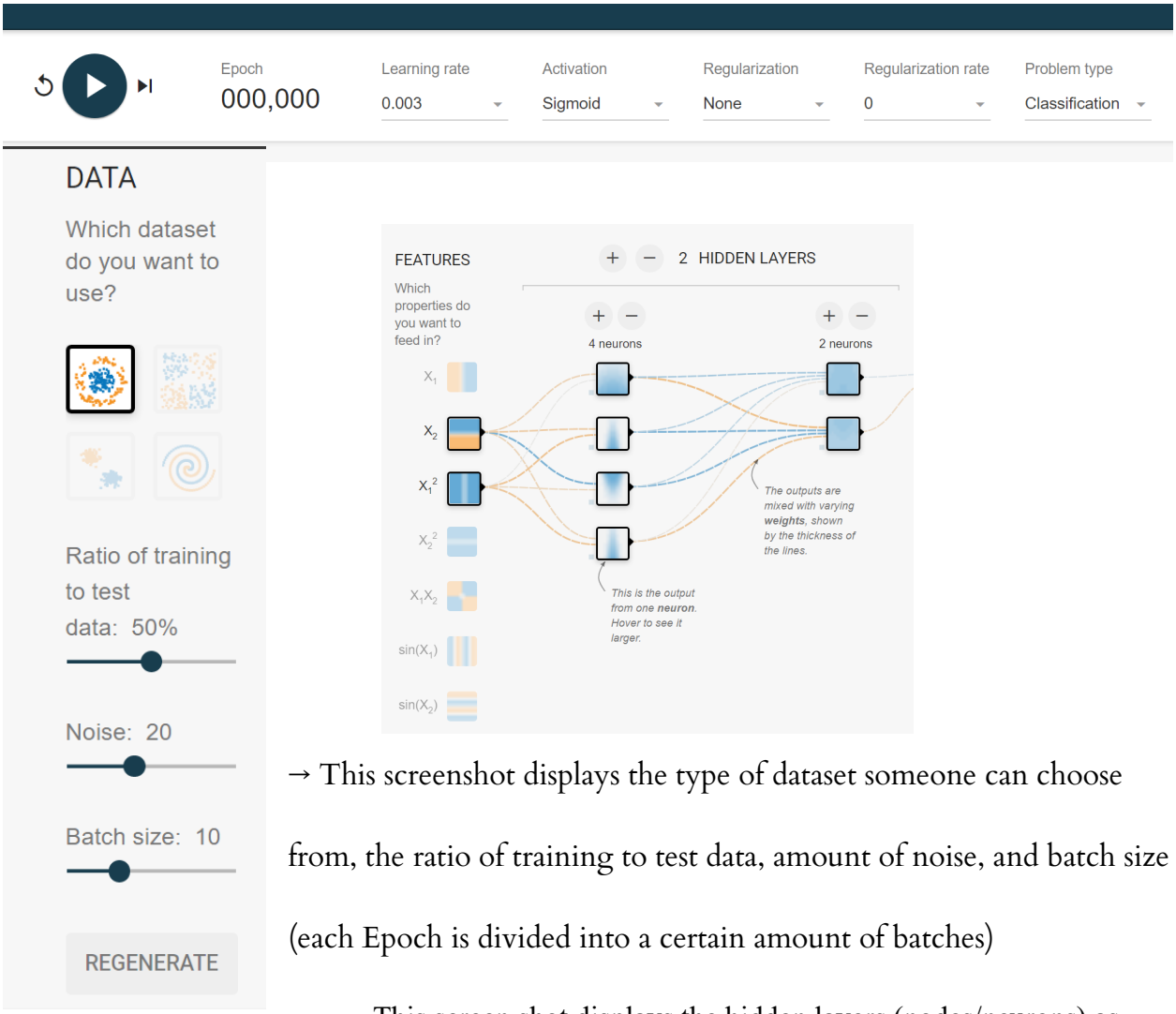
5. Sample Demonstration of a TensorFlow Playground

- Link to a Sample TensorFlow Playground: Tensorflow Playground Link

The following screenshot displays the various features with which an individual can play with

- A person may choose a specific learning rate, activation function, problem type, as well

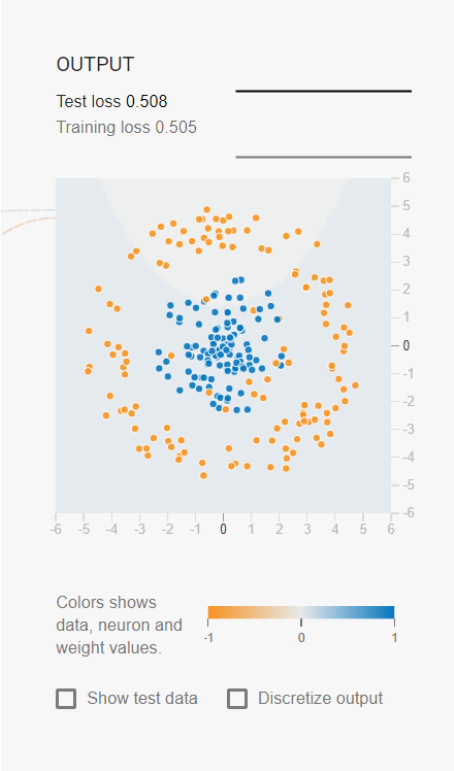
as if they wish to have regularization



→ This screenshot displays the type of dataset someone can choose from, the ratio of training to test data, amount of noise, and batch size (each Epoch is divided into a certain amount of batches)

-This screen shot displays the hidden layers (nodes/neurons) as well as the input features that can be decided by the individual. Various properties can be fed into the layers.

→ This screenshot is representative of the output result of a neural network that an individual creates using the TensorFlow playground. The test loss is displayed as well as the training loss.



The test data can also be shown in order display comparisons.

6. Additional Resources:

1. Additional information on deep learning
2. Deep Learning Fundamental Concepts
3. Understanding Tensorflow Playground
4. SlideShow adopted from MIT Deep Learning Course
5. In-Depth Deep Learning
6. Deep Learning Library

7. References:

Smilkov, Daniel, and Shan Carter. “Tensorflow - Neural Network Playground.” *A Neural Network Playground*, playground.tensorflow.org/#activation=tanh&batchSize=10&dataset=xor®Dataset=reg-plane&learningRate=0.03®ularizationRate=0&noise=0&networkShape=4,2&seed=0.56691&showTestData=false&discretize=false&percTrainData=50&x=false&y=false&xTimesY=false&xSquared=false&ySquared=true&cosX=false&sinX=false&cosY=false&sinY=false&collectStats=false&problem=classification&initZero=false&hideT.