Tensorflow Playground Activity

For this activity, you will be using a website called Tensorflow playground to experiment with simple neural network concepts and the problems they can solve. As you move through the activity, there will be questions for you to answer. Here is a [link](https://playground.tensorflow.org/#activation=linear&batchSize=20&dataset=gauss&regDataset=reg-plane&learningRate=0.03&regularizationRate=0&noise=5&networkShape=1&seed=0.93980&showTestData=false&discretize=false&percTrainData=50&x=true&y=true&xTimesY=false&xSquared=false&ySquared=false&cosX=false&sinX=false&cosY=false&sinY=false&collectStats=false&problem=classification&initZero=false&hideText=false) to get started. You should see something like this:

A screenshot of a computer

Description automatically generated

The problem type you will be exploring is classification. The model will try to identify which class each point belongs to by training a classifier on the input data. The blue and orange areas you see are visible description of the decision boundaries of the classifier, showing how the model determines the class of the point.

Let’s look at some of the different on-screen controls you will use with the model. First, there is a Reset, Play/Pause, and a Forward Step button at the top left. Playing will cause the model to train until you stop it, the Forward Step button will only advance the model a single step. Epochs refer to training steps and then number show will update with how many training steps have been done. The data section will determine the classification problem the model you will try to solve. We will explore two of them during the activity. There are feature selection options, but you will stick to the default X1 and X2 for this activity. Using the + and – buttons allow you to add hidden layers and more neurons to each hidden layer.

1. To begin, you will do some research into what some of these other options are all about. Briefly describe (1-2 sentences) the following concepts, either using online resources or your own understanding:
   1. Learning Rate:  
      *The learning rate is the rate which determines how fast the model learns data; it is a hyperparameter that determines the size of the steps taken during minimizing loss*
   2. Activation Function:  
      *An activation function is a mathematical operation applied to data to change its representation, e.g., sigmoid, tanh*
   3. Hidden Layer:  
      *A hidden layer is a layer of nodes that fall between the input layer and output layer*
   4. Batch Size:  
      *The batch size is how many data points are used in one iteration of the learning process*

We will start by examining the linearly separable data set:

A screen shot of a graph

Description automatically generated

1. Using the given starting parameters, press play and let the model run until both training and test loss are 0. Observe the model find the decision boundaries for the data set. How many epochs did it take to find a classifier that completely separates both classes?

*About 90 epochs*

1. If you were to increase the learning rate, do you expect the model to take less or more steps to find a solution? Why?

*Take less steps to find a solution because the model would be learning faster*

1. Try increasing the learning rate, were you correct in your prediction in number 3?  
   *Yes*

For the linearly separable dataset, a linear activation function with a single layer with a single neuron (this is known as a perceptron) is sufficient to find the solution. You will now try a different dataset and see the limits of the linear activation function and some ways to overcome those limitations.

Start by selecting the XOR data set:

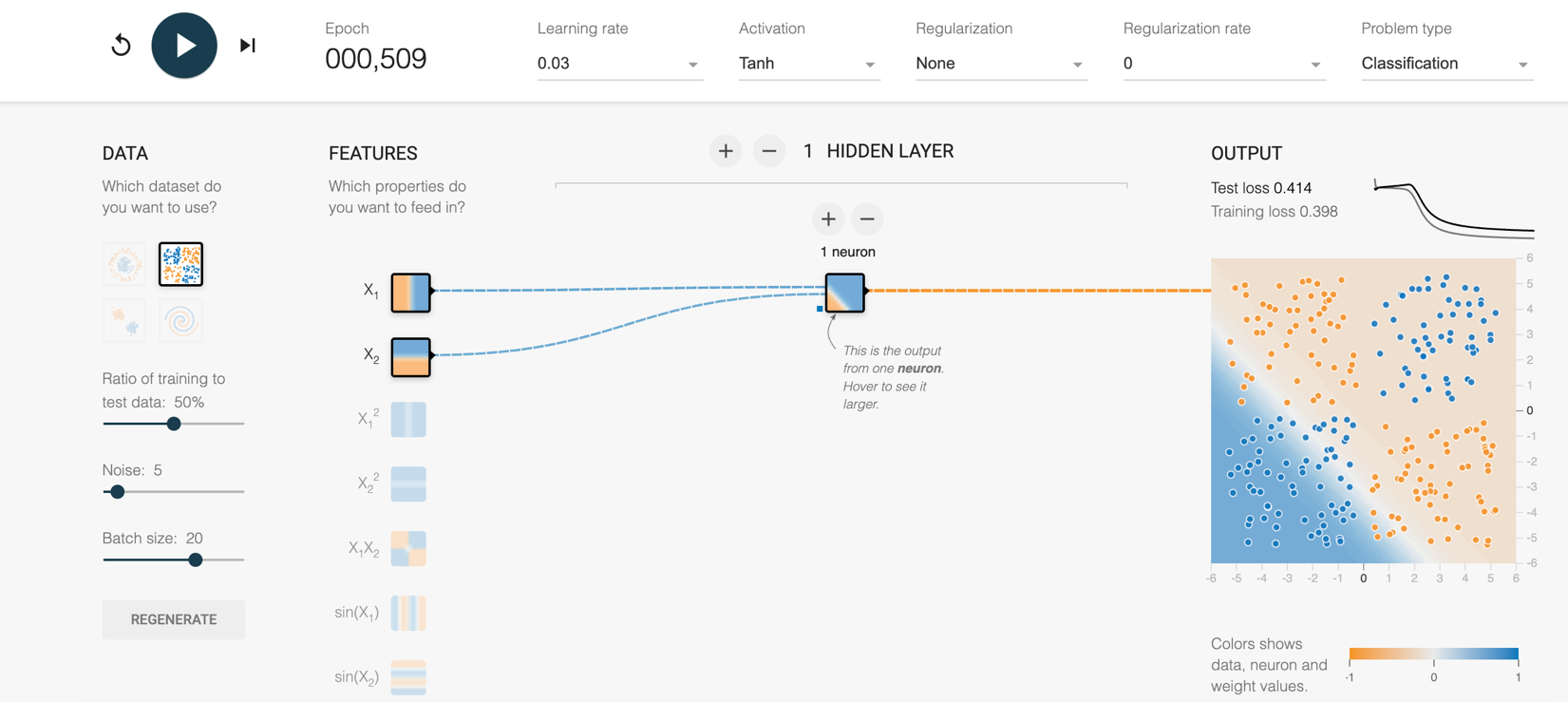
A screen shot of a graph

Description automatically generated

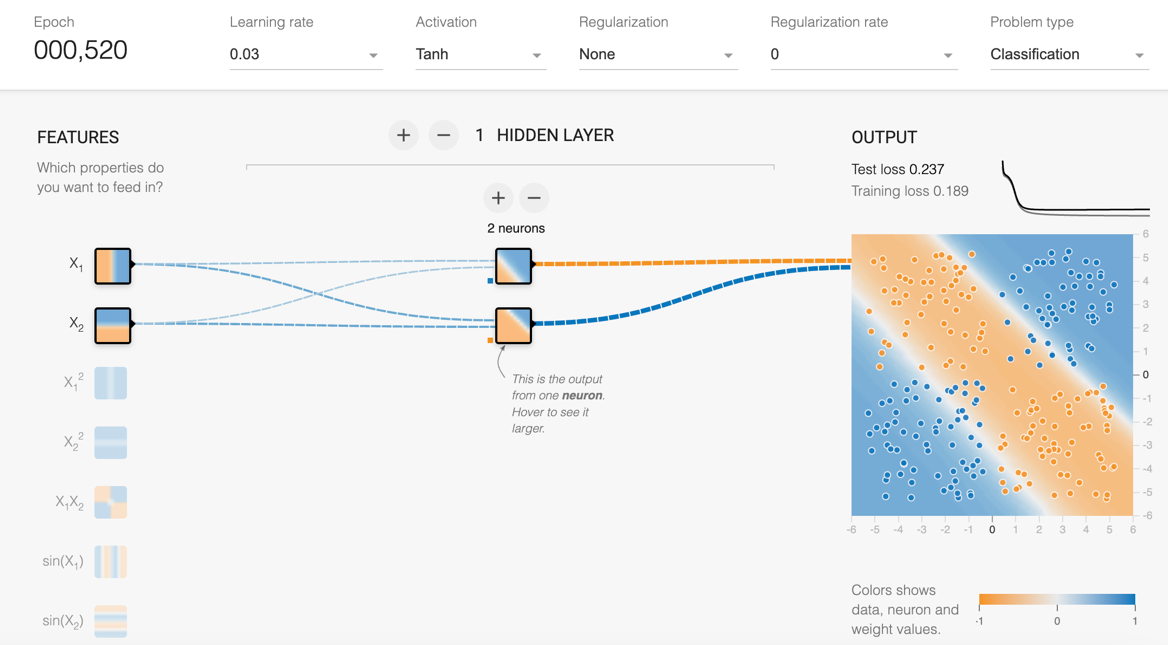
1. Press play and let the model try to converge with the linear activation function. Modify the learning rate, number of layers, and number of nodes to try and get the model to find the solution.   
     
   Did changing any of those tuning parameters cause the model to find the solution?  
     
   *No*
2. Is a linear activation function sufficient for non-linearly separable data? Why or why not?

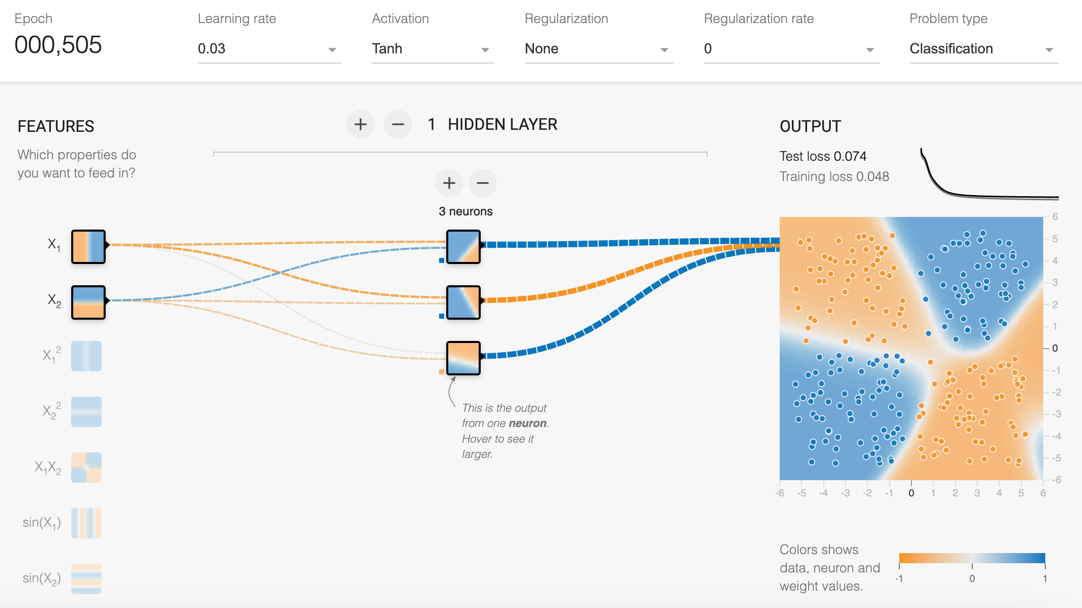
*No because the linear activation function should be used for data that is linearly separable, otherwise it seems to fail when trying to separate non-linearly separable data.*

Revert the changes you made to the model and return it to a single layer model with a learning rate of 0.03, but this time use the activation function Tanh.

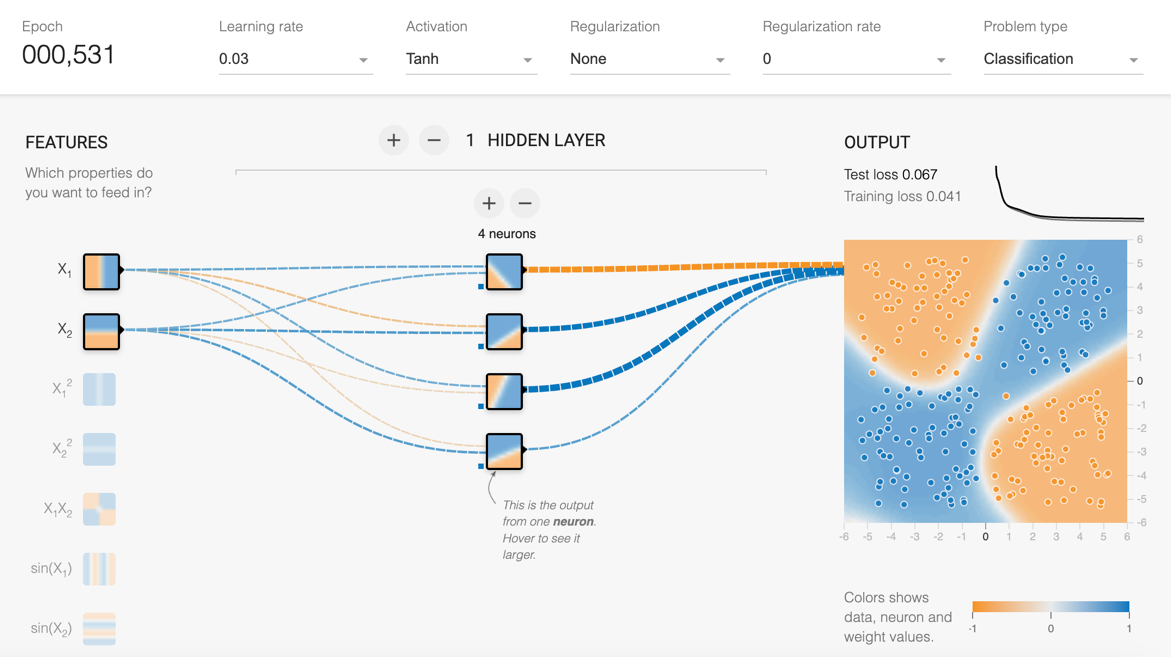
1. Using online resources or your own knowledge, what is the Tanh function and how does it work?  
   *The tanh function is hyperbolic tangent, it looks similar to a sigmoid function.*
2. Run the model with one neuron and take a screenshot of the best decision boundaries it was able to come up with after 500 epochs.   
     
     
   
3. If you were to increase the number of neurons to two, what would you expect to happen to the decision boundary? Explain your reasoning.

*I would think that it becomes more accurate when increasing the number of neurons since there is more computation being done to improve the result.*

1. Add the second neuron, did the model behave how you expected? Paste a screenshot of your results after 500 epochs.  
   
2. Repeat this experiment 2 more times by adding a third and fourth neuron. Post a separate screenshot for each experiment.

*3 Neurons*  
  


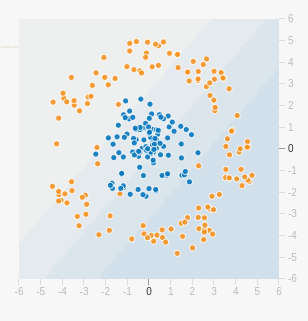
*4 Neurons*



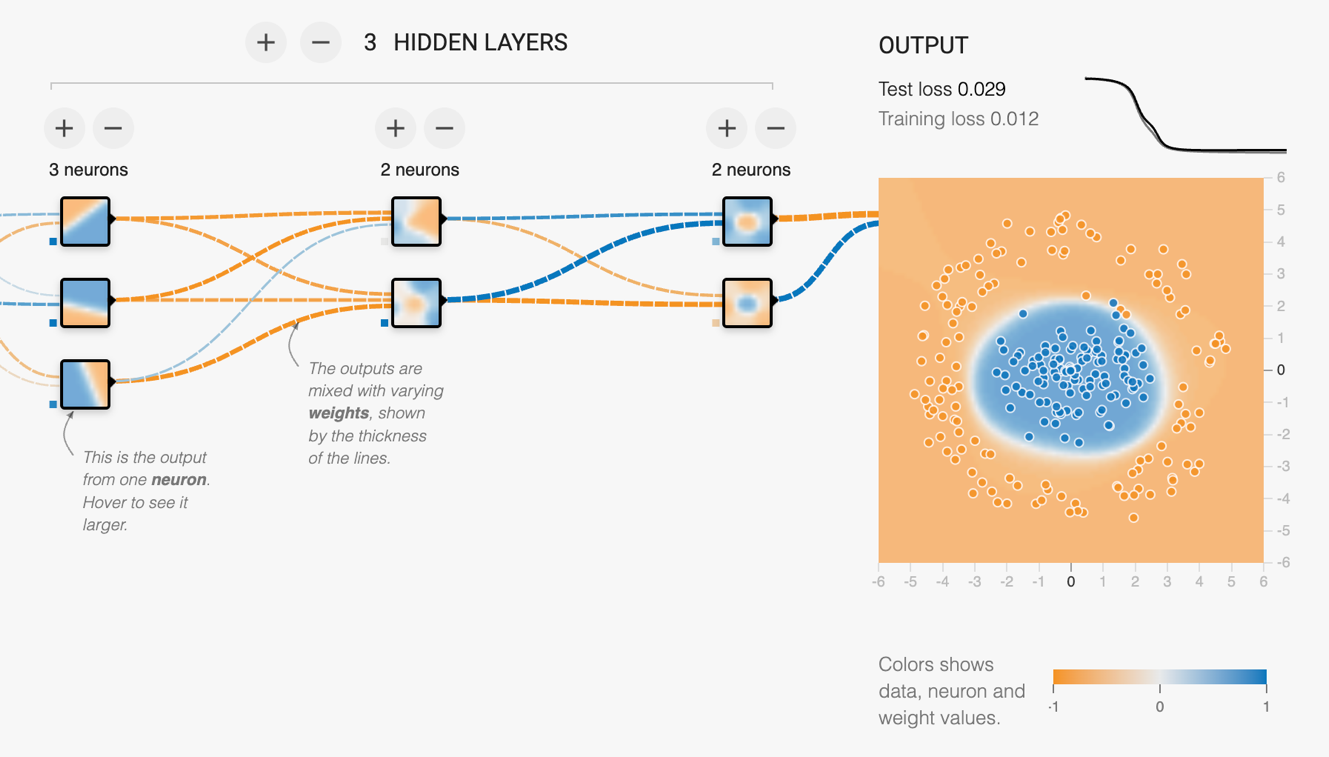
1. Why does adding neurons increase the model’s performance?   
   *Each neuron can contribute more to the model’s accuracy by improving the boundary conditions for the data.*
2. If you were to add one extra hidden layer to model, what do you expect to happen?  
     
   *I would expect the result to improve*
3. Add a second hidden layer. You should now have a two-layer model, one layer with 4 neurons, and the other with 2. Run the model for around 500 epochs and observe the results. Post a screenshot here. Did the model behave how you expected?  
     
   A screenshot of a computer

   Description automatically generated  
   *Yes, the model improved and behaved as I expected*
4. Why does adding a second layer improve performance?   
     
   *My answer: The first layer is only so good at performing actual classification in the context of this problem. What it can do is identify the key features. Once those are identified, a second layer is able to use those features to preform the classification.*

As a final experiment we will look at one other data set. It looks like a target:



Reset the model back to one neuron, you can leave the learning rate and activation function as is.

1. Experiment with different learning rates, number of neurons, number of layers, etc similar to what you did with the XOR dataset. Find the minimum number of neurons required for the model to fit a sufficient decision boundary. How many neurons did you find to be the minimum and why do you think that number is sufficient?  
     
     
     
   

*I found that 7 total neurons were necessary to fit the decision boundary. The first layer has 3 neurons, the second layer has 2, and the third layer has 2. Removing one neuron in any layer led to an inaccurate decision boundary. I believe that 7 neurons is sufficient because each neuron can identify some key features, and then the subsequent neurons can perform the classification.*