Neural Networks for Machine Learning
tensorflow playground
TensorFlow Playground

• Great javascript app demonstrating many basic neural network concepts
• Doesn’t use TensorFlow software, but a lightweight js library
• Runs in a Web browser
• See http://playground.tensorflow.org/
• Code also available on GitHub
• Try the playground exercises in Google’s machine learning crash course
Tinker With a **Neural Network** Right Here in Your Browser.
Don’t Worry, You Can’t Break It. We Promise.

DATA
Which dataset do you want to use?

![Dataset options](image)

Ratio of training to test data: 50%
Noise: 0
Batch size: 10

REGENERATE

FEATURES
Which properties do you want to feed in?

![Feature options](image)

4 neurons
2 neurons

OUTPUT
Test loss 0.435
Training loss 0.432

The outputs are mixed with varying **weights**, shown by the thickness of the lines.

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

Colors show data, neuron and weight values.

Show test data  □  
Discretize output □
Datasets

• Six datasets, each with 500 \((x,y)\) points on a plane where \(x\) and \(y\) between -5 and +5

• Points have *labels* of positive (orange) or negative (blue)

• Two possible machine learning *tasks*:
  – Classification: Predict class of test points
  – Regression: find function to separate classes

• *Evaluation*: split dataset into training and test, e.g., 70% training, 30% test
Available Input features

\( X_1 \)  Point’s x value
\( X_2 \)  Point’s y value
\( X_1^2 \)  Point’s x value squared
\( X_2^2 \)  Point’s y value squared
\( X_1X_2 \)  Product of point’s x & y values
\( \sin(X_1) \)  Sine of point’s x value
\( \sin(X_2) \)  Sine of point’s y value
• Divide training data into batches of instances (e.g., batch size = 10)

• For each epoch:
  – For each batch:
    • Instances run through network, noting difference between predicted and actual value
    • Backpropagation used to adjust connection weights
  – Stop when training loss flatten out

• If test loss is high, then try
  – Adding additional hidden layers
  – Adding more features to inputs
  – Adjusting hyperparameters (e.g., learning rate)
  – Get more training data
Hyperparameters

• Parameters whose values are set before the learning process begins
• Basic neural network hyperparameters
  – Learning rate
  – Activation function
  – Regularization
  – Regularization rate
Learning rate

- **Gradient descent** used in backpropagation to adjust weights to minimize the loss function
- Learning rate determines how much weights are adjusted
- If too high, we may miss some or most minima
- If too low, learning will take too long
Activation Function

• Determines a node’s output given its inputs
• The ReLu (rectified linear unit) is simple and a good choice for most networks
• Returns zero for negative values and its input for positive ones
  \[ f(x) = \max(0, x) \]
Regularization

• Parameter to control overfitting, i.e. when the model does well on training data but poorly on new, unseen data
• L2 regularization is the most common
• Using dropout is another common way of controlling overfitting in neural networks
  – At each training stage, some hidden nodes are temporarily removed