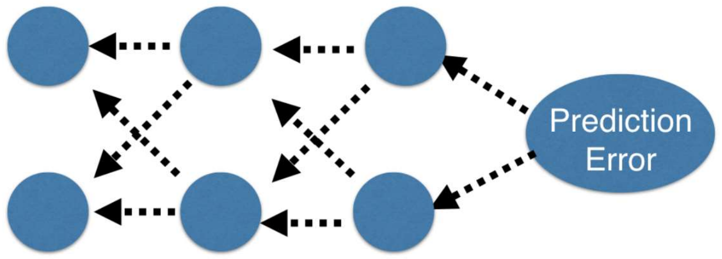
Backpropagation

INTRODUCTION TO DEEP LEARNING IN PYTHON

Dan Becker

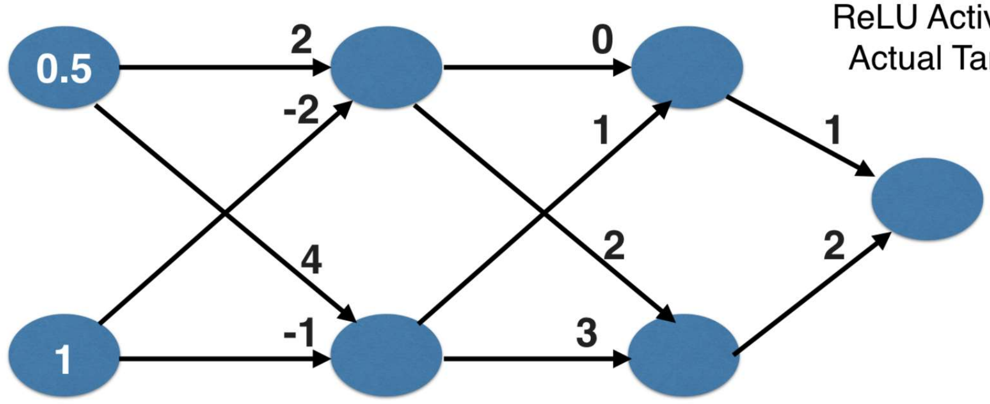
Data Scientist and contributor to Keras and TensorFlow libraries

Backpropagation



- Allows gradient descent to update all weights in neural network (by getting gradients for all weights)
- Comes from chain rule of calculus
- Important to understand the process, but you will generally use a library that implements this

Backpropagation process

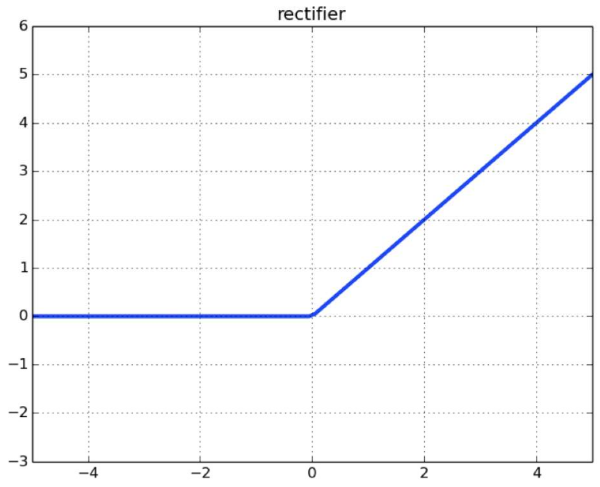


ReLU Activation Function
Actual Target Value = 4

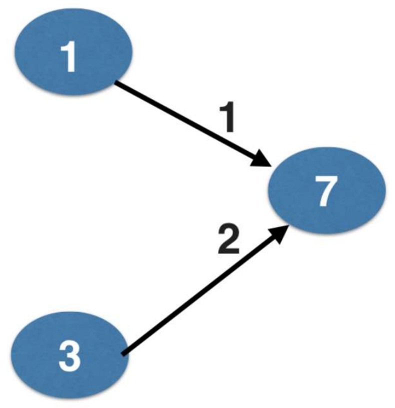
Backpropagation process

- Go back one layer at a time
- Gradients for weight is product of:
  1. Node value feeding into that weight
  2. Slope of loss function w.r.t node it feeds into
  3. Slope of activation function at the node it feeds into

ReLU Activation Function



Backpropagation



ReLU Activation Function
Actual Target Value = 4
Error = 3

Calculating slopes associated with any weight

- Gradients for weight is product of:
  1. Node value feeding into that weight
  2. Slope of activation function for the node being fed into
  3. Slope of loss function w.r.t output node

Backpropagation: Recap

- Start at some random set of weights
- Use forward propagation to make a prediction
- Use backward propagation to calculate the slope of the loss function w.r.t each weight
- Multiply that slope by the learning rate, and subtract from the current weights
- Keep going with that cycle until we get to a flat part

Stochastic gradient descent

- It is common to calculate slopes on only a subset of the data (a *batch*)
- Use a different batch of data to calculate the next update
- Start over from the beginning once all data is used
- Each time through the training data is called an epoch
- When slopes are calculated on one batch at a time: stochastic gradient descent