Randal, @RKOUTNIK

What’s the deal with neural networks?

How would you tell a computer this image is Seinfeld? If we can’t tell a computer how to this, we can’t write software then. Or can we?

What we CAN do is tell a computer how to learn.

Game of Go and chess. Tons of possible positions. Chess AI was basically just brute forcing the best positions. Couldn’t beat Go, but a neural network was trained and beat the Go champ for the first time last year.

Self driving cars - teach a computer to learn to drive...

What you do need to know:

- Add
- Subtract
- Multiply
- Division

XOR Gate - logic gate, 2 inputs = 1 output. If any input is on, it will output 1, otherwise 0.

Garbage in, garbage out.

“On two occasion I have been asked, ‘Pray, Mr. Babbage, if you put into the machine wrong figures, will good data…’"

XOR Gate. Feed forward neural network - every node is connected to another node. Feed forward -> it’s connected alot.

A neuron takes some number of inputs and outputs a single value. Outputs are always between -1 to 1. Each input is given a weight. The node has a bias (how good am I at this computation?)

X1 -> Weight1 -> +bias -> neuron
X2 -> Weight2 -^

Perceptron function. 0 or 1 of output doesn’t allow for ambiguity.

Sigmoid function - allows for some ambiguity in input and output. Recommended to use this when ever possible.

Weight and biases, when you start a network, have been randomly weight them all (this is from the library that is chosen). Training then SETS these values.

Each neuron provides a 0 or 1 as output. Need a neuron for EACH possible result.

Takes 300-400 passes with 4 inputs and 2 possible answers to get the right values.

Gradient discent -> indicates error rate. Goal is to find the lowest error rate to use for our model.

Learning rate (how much the neuron tweaks in each pass)

Calculords - math game that only requires 3 symbols. Wanted to predict who would win a lane in the game.

MNIST - training set. This is a set of tons of hand written digits. MNIST collected this to train a computer.

How do you design a network to solve a problem?

For digits, we know there are 10 possible outputs (0-10). Each image is exactly the same size.

784 input neurons, 100 hidden neurons, 10 output neurons.

For every neuron we add to a feed forward network, the longer training takes. GPUs are used since it has more cores than a CPU (but the cores are not as smart/powerful).

Back propagation - errors and successes sent back to the neuron network

Neural Warrior (WarriorJS is one library used)

1. Collect data
2. Format the data

   1. matrix used
3. What is good?
   1. Lose after 1000 moves
   2. Lose when die
   3. Win when find the exit
4. Build the network
   1. Input layer
      1. We know our input
   2. Deep neural network = multiple neural networks used in the calculation
   3. Output layer
      1. Direction
      2. Action
5. Training
   1. Run all inputs for all turns of the game
   2. Save these
   3. Then evaluate the results

Python - start with TensorFlow -> TFLearn. TFLearn is simplified and allows jumping into training and such.