Learning through trial & error

Team 2-leaf clover: Devon Kenzie & Isabel Zorrilla

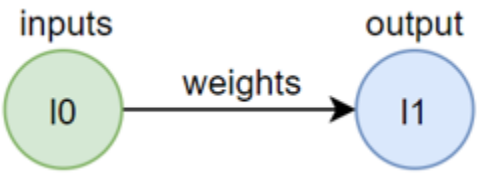
Summary:

After learning the fundamentals of perceptrons and neural networks in lecture, we were interested in delving into the technical details of how they work. Our project was exploratory, experimenting with different network designs & applications, testing the capabilities of different networks with varying success.

- Inspired by Rogers & McLelland, Clark, Gurney, and other assigned class readings
- we designed 2 neural networks
  - a simple network which determines whether simple 3x3 mazes have a solution
  - a more complex network which can provide solutions to mazes of any sizes
- we tested different learning speeds, varied number of training iterations, different data representations and different training data sizes

A Simple 2-layer Network

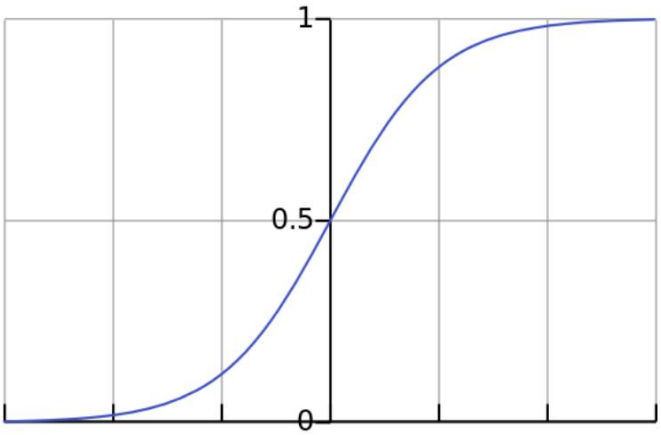
Goal: given a maze, determine whether a solution exists



Predictions after 2 training cycles:

[…, 0.00201651, 0.99977125, 0.00201651, 0.00201651, 0.99977125, 0.00201651, …]

- 1.0 is a solvable maze and 0.0 is unsolvable – these values represent the probability that each maze has a solution

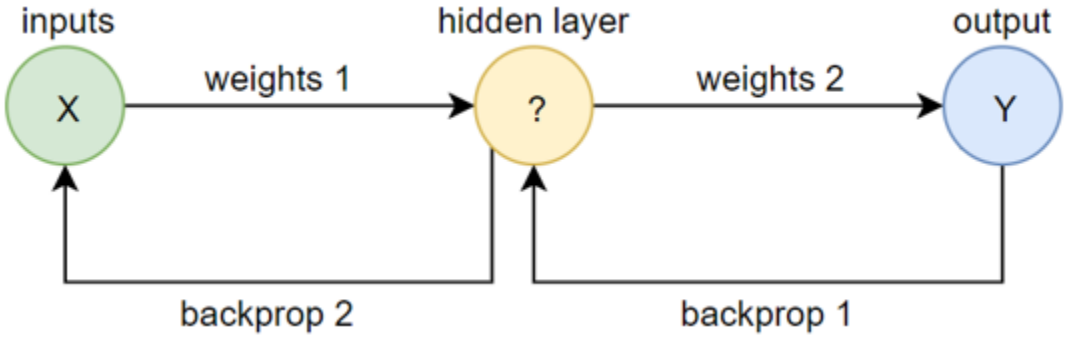


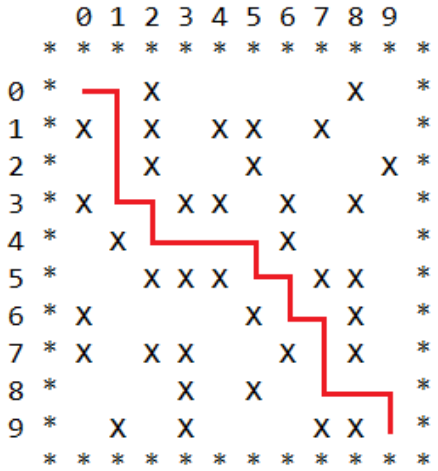
Activation function:

We used a Sigmoid activation, which maps any number to a value between 0 and 1. We use the sigmoid activation function to map our inputs onto the probability that the perceptron will activate.

A 3-layer Network

Goal: Given a maze, find a solution path. Then, given the solution path, determine which move is the best move to make from any given cell.





Path prediction after 150 training cycles:
[up, down, down, down, right, down, right, right, right, down, right, down, right, down, down, up, right, down]

Actual path solution:
[right, down, down, down, right, down, right, right, right, down, right, down, right, down, down, right, right, down]

Learning through 150 cycles:

Cycle 0: Success = 5.0%
Cycle 1: Success = 0.0%
Cycle 2: Success = 45.0%
Cycle 3: Success = 45.0%
Cycle 4: Success = 0.0%
...

Cycle 51: Success = 85.0%
Cycle 52: Success = 85.0%
Cycle 53: Success = 85.0%
Cycle 54: Success = 85.0%
Cycle 55: Success = 85.0%
...

Cycle 145: Success = 65.0%
Cycle 146: Success = 65.0%
Cycle 147: Success = 65.0%
Cycle 148: Success = 65.0%
Cycle 149: Success = 65.

References

iamtrask. (2015, July 12). A Neural Network in 11 lines of Python. Retrieved from http://iamtrask.github.io/2015/07/12/basic-python-network/

Loy, J. (2019, May 14). How to build your own Neural Network from scratch in Python. *Towards Data Science*. Retrieved from https://towardsdatascience.com/how-to-build-your-own-neural-network-from-scratch-in-python-68998a08e4f6

Updating weights:

1. Determine the degree of error from predicted output to the expected output
2. Multiply the degree of error by the sigmoid function
    a. This determines the *error weighted derivative*
3. Calculate new weights by adding multiplying the o
4. riginal inputs by the error weighted derivative, and adding to the old weights
5. repeat