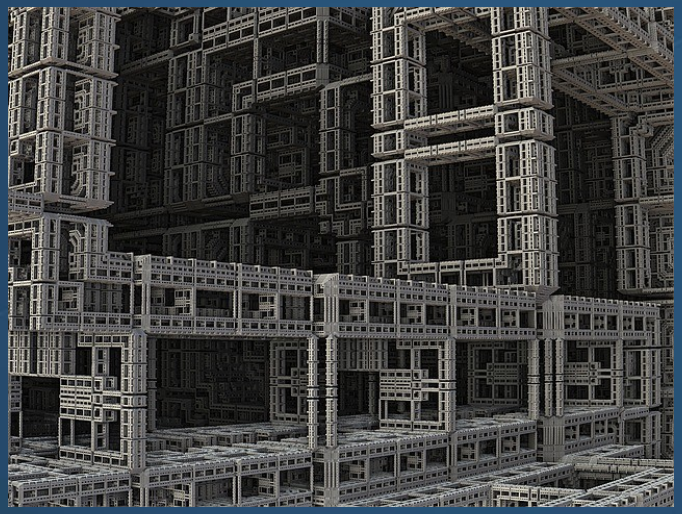
Deep Q Learning: From Paper to Code

Dealing with Continuous State Spaces with Deep Neural Networks

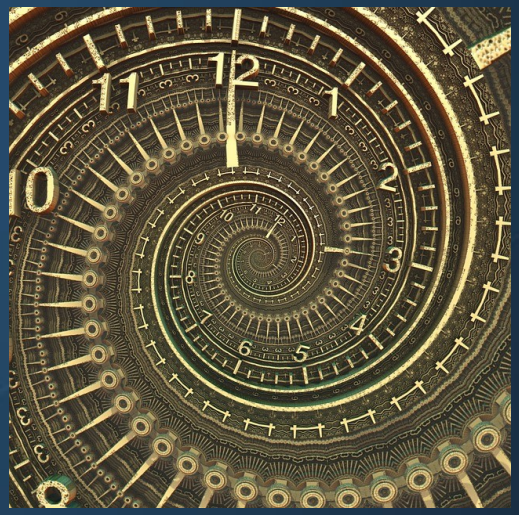
Last Time ...

- Coded our first Q learning agent
- 70% win rate in a simple environment

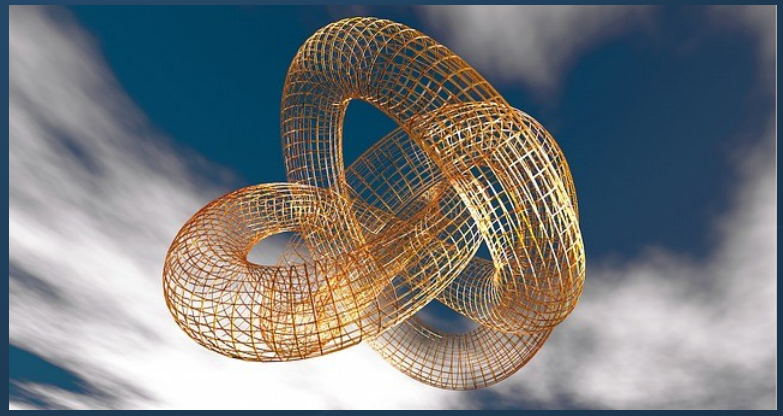
Large or Continuous State Spaces



Large state spaces problematic



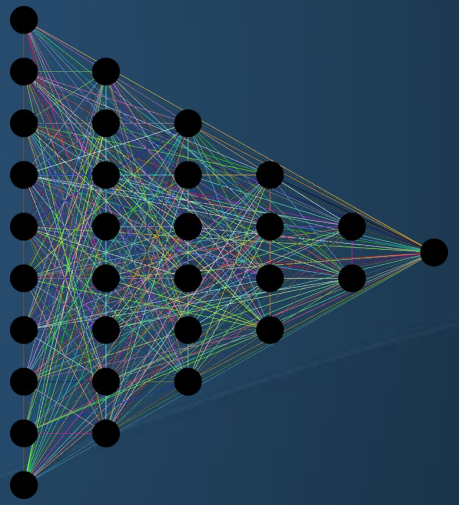
Not enough time



Continuous spaces are infinite

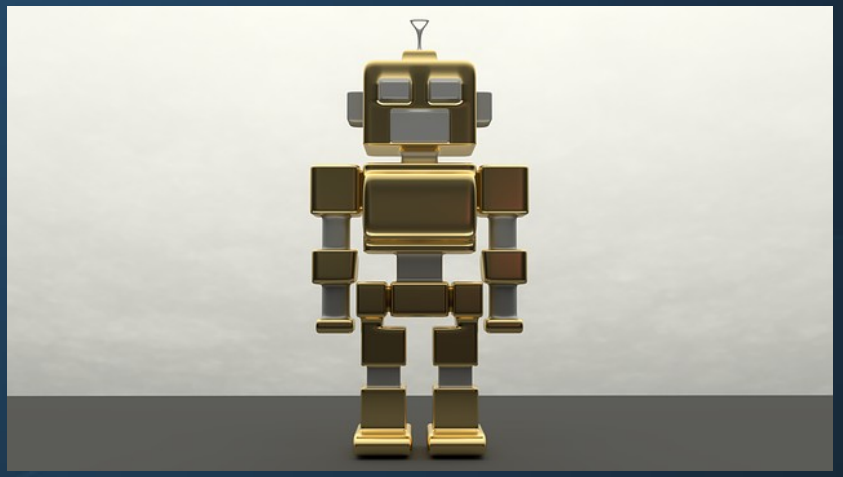
Deep Learning to the Rescue

Tables will not work



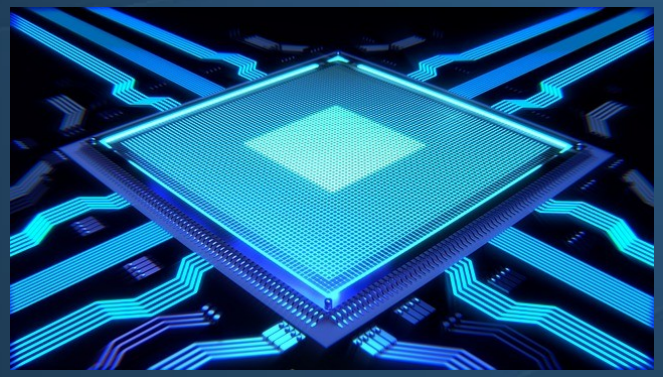
Deep neural networks

Universal function approximator

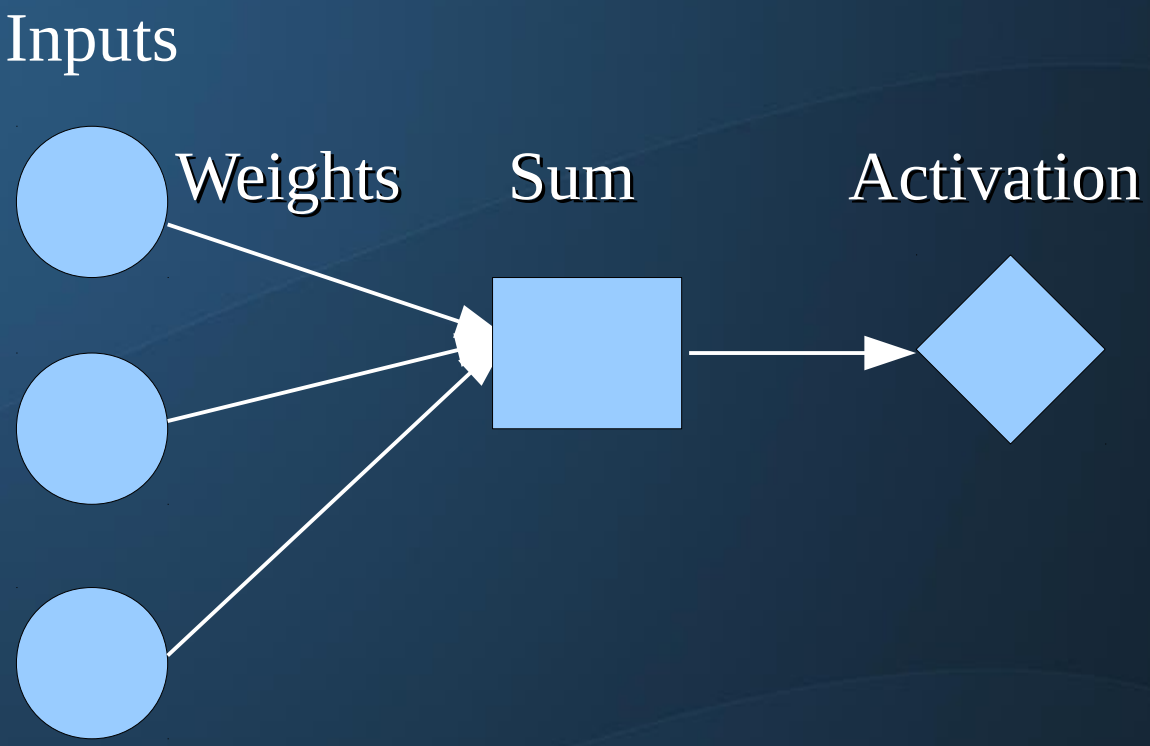


Continuous Q function

A Crash Course in Neural Networks



Perceptron



A Crash Course in Neural Networks

Car?

Four wheels → W

Streering wheel → S

$$\text{Sum} = w_0 \times bias + w_1 \times W + w_2 \times S$$

$$\text{Class} = \sigma(\text{Sum})$$

Let σ Be the unit step function: $\sigma(x) \stackrel{\text{def}}{=} \text{car if } x \geq 0 \text{ else not car}$

A Crash Course in Neural Networks

$$\text{Sum} = w_0 \times bias + w_1 \times W + w_2 \times S$$

$$\text{Class} = \sigma(\text{Sum})$$

Let σ Be the unit step function: $\sigma(x) \stackrel{\text{def}}{=} \text{car if } x \geq 0 \text{ else not car}$

- Bias → best guess in absence of input
- Weights → importance of features S, W
- Start with some arbitrary numbers

$$bias = 1, \quad w_0 = -3, w_1 = 2, w_2 = 2$$

A Crash Course in Neural Networks

$$\text{Sum} = w_0 \times bias + w_1 \times W + w_2 \times S$$

$$\text{Class} = \sigma(\text{Sum})$$

Let σ Be the unit step function: $\sigma(x) \stackrel{\text{def}}{=} \text{car if } x \geq 0 \text{ else not car}$

$$bias = 1, \quad w_0 = -3, w_1 = 2, w_2 = 2$$

$$\text{Sum} = -3 \times 1 + 2 \times 0 + 2 \times 1 = -1$$

$$\text{Class} = \sigma(\text{-1}) = \text{not a car}$$

A Crash Course in Neural Networks

$$\text{Sum} = w_0 \times bias + w_1 \times W + w_2 \times S$$

$$\text{Class} = \sigma(\text{Sum})$$

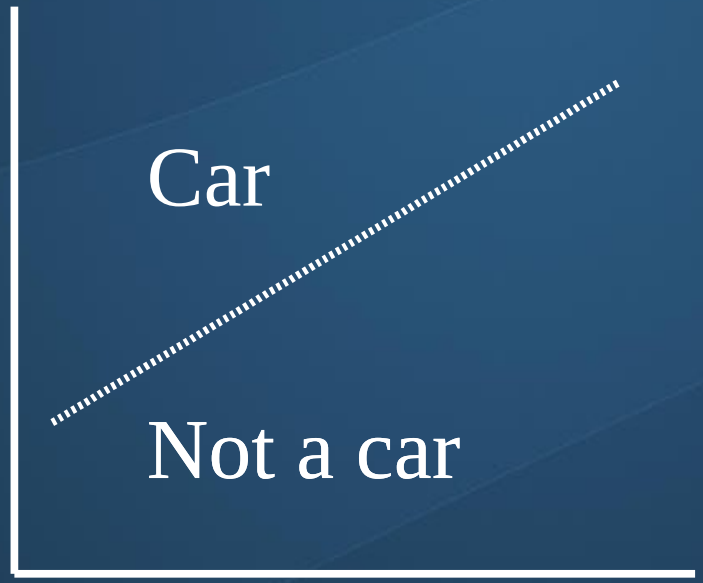
Let σ Be the unit step function: $\sigma(x) \stackrel{\text{def}}{=} \text{car if } x \geq 0 \text{ else not car}$

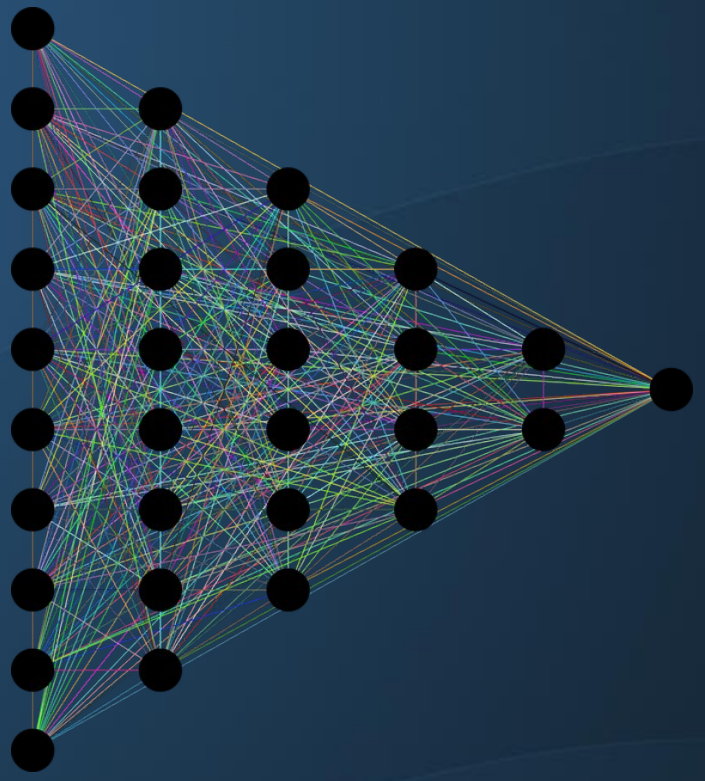
$$bias = 1, \quad w_0 = -3, w_1 = 2, w_2 = 2$$

$$\text{Sum} = -3 \times 1 + 2 \times 1 + 2 \times 1 = 1$$

$$\text{Class} = \sigma(1) = \text{is a car}$$

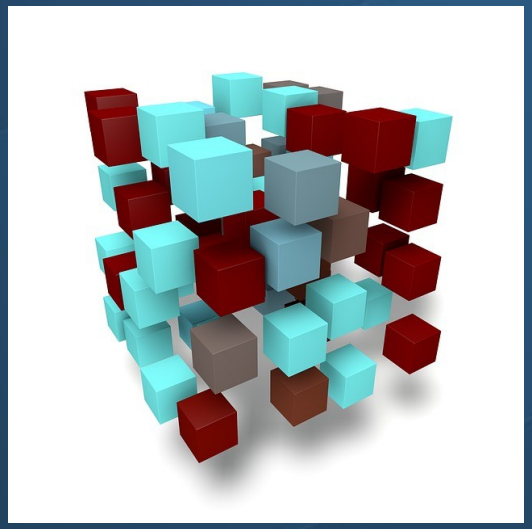
A Crash Course in Neural Networks





Fully connected neurons

Enter GPUs

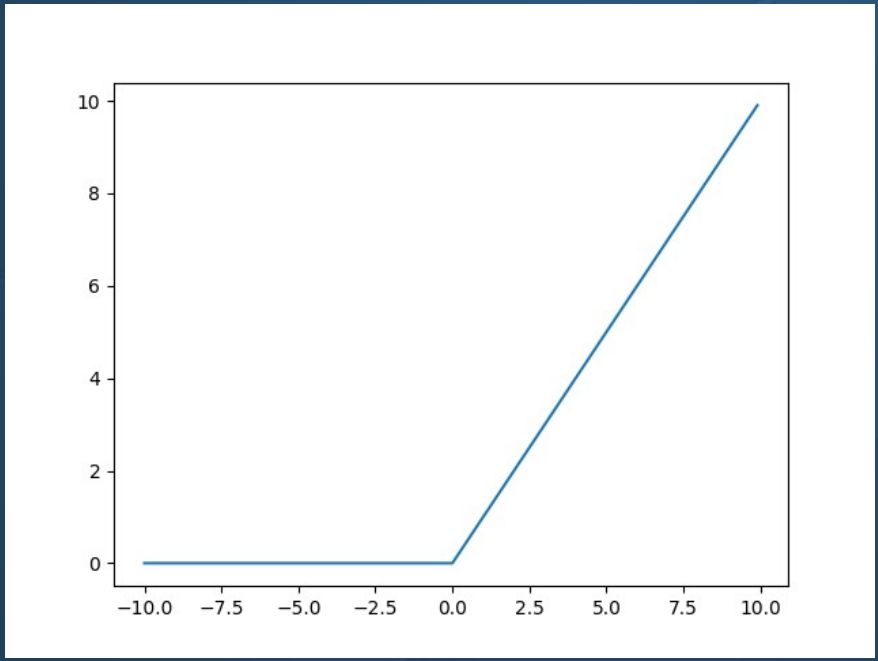


Matrix representation

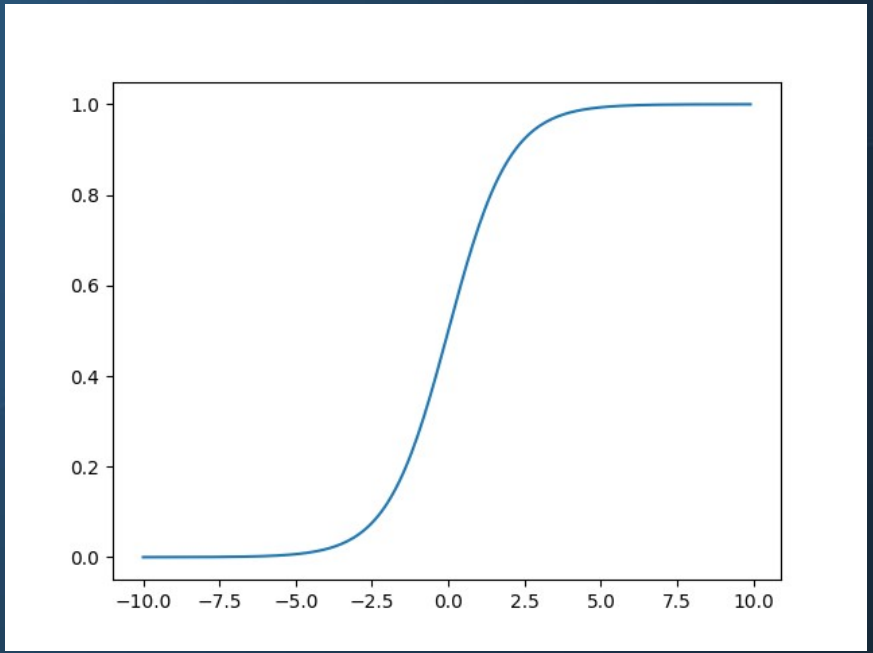
$$\begin{pmatrix} w_{00} & w_{01} & \dots & w_{0n} \\ w_{10} & w_{11} & \dots & w_{1n} \\ \dots & \dots & & \dots \\ w_{m0} & w_{m1} & & w_{mn} \end{pmatrix} \times \begin{pmatrix} x_0 \\ x_1 \\ \dots \\ x_m \end{pmatrix}$$

Feed forward

Activation Functions



$$relu(x) \stackrel{\text{def}}{=} x \text{ if } x \geq 0 \text{ else } 0$$



$$sigmoid(x) \stackrel{\text{def}}{=} \frac{1}{1+e^{-x}}$$

Neural Nets in Practice

- Input is fed into a layer and activated
- Result is then fed into next layer, and activated
- All the way through to the output
- Output compared to some target to get cost
- Weights changed to minimize cost (back propagation)
- Repeat process → profit

Summary

- Large and continuous state spaces are problematic
- Require deep neural networks
- Inputs are fed forward to generate output and cost

Up Next