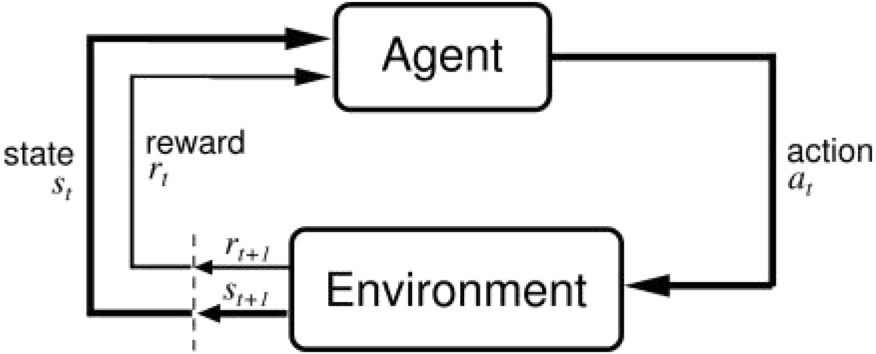
Q-Learning with Neural Networks & OpenAI Universe

Kevin Sun, Lior Hirschfeld, Henry Desai, and Jihoun Im

Reinforcement Learning Review

- Agent determines ideal behavior by interacting with environment
- Reward feedback as reinforcement signal
- Optimizes actions to maximize reward
- Main challenge: Exploration vs. Exploitation to find new strategies

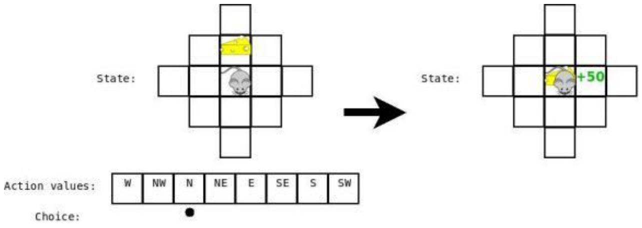


Tabular Q-Learning Overview

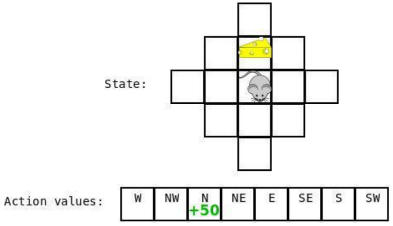
Tabular Q-Learning update algorithm:

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha[R_{t+1} + \gamma * \max Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]$$

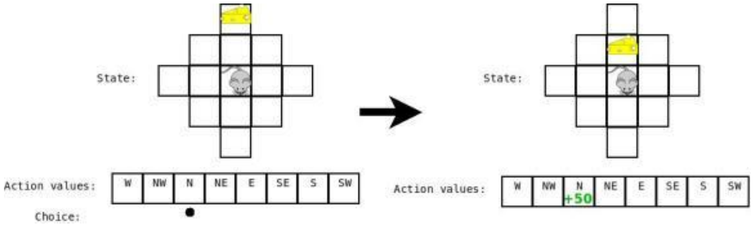
State A:



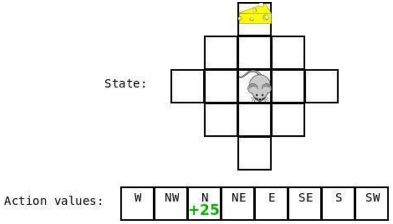
Result:



State B:



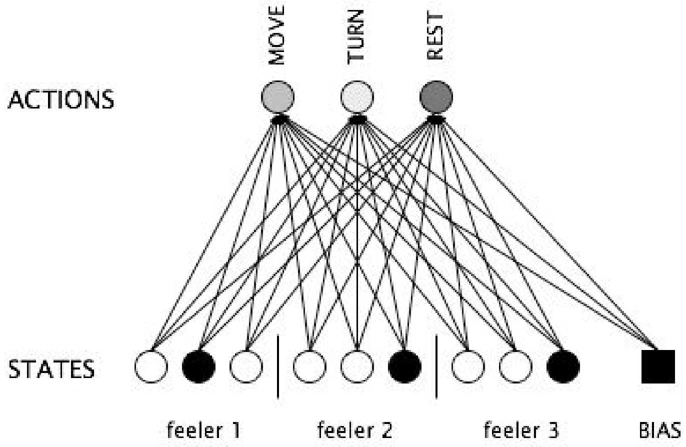
Result:



Why Not Tabular?

- Computationally expensive
- Instead, approximate Q function by a function approximator that generalizes and pattern matches between states
- Neural Networks are less stable, but are much more flexible

$$
R = \begin{array}{c} \\ \text{State} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}
\begin{array}{c} \text{Action} \\ \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\
\begin{bmatrix}
-1 & -1 & -1 & -1 & 0 & -1 \\
-1 & -1 & -1 & 0 & -1 & 100 \\
-1 & -1 & -1 & 0 & -1 & -1 \\
-1 & 0 & 0 & -1 & 0 & -1 \\
0 & -1 & -1 & 0 & -1 & 100 \\
-1 & 0 & -1 & -1 & 0 & 100
\end{bmatrix}
\end{array}
$$



Q-Learning with Neural Networks

- Three major components to Reinforcement Learning
  - Policy: maps a certain state to an action (behavior of the learning agent)
    - Value based RL
  - **Value: takes in the state and the action and outputs a reward value (Q function)**
    - **Policy based RL**
  - Model: learning agent makes decisions on future situations without experiencing them
    - Model based RL
- Two types of Neural Networks:
  - Batch: need all the data at once
  - Incremental: one piece of data at a time

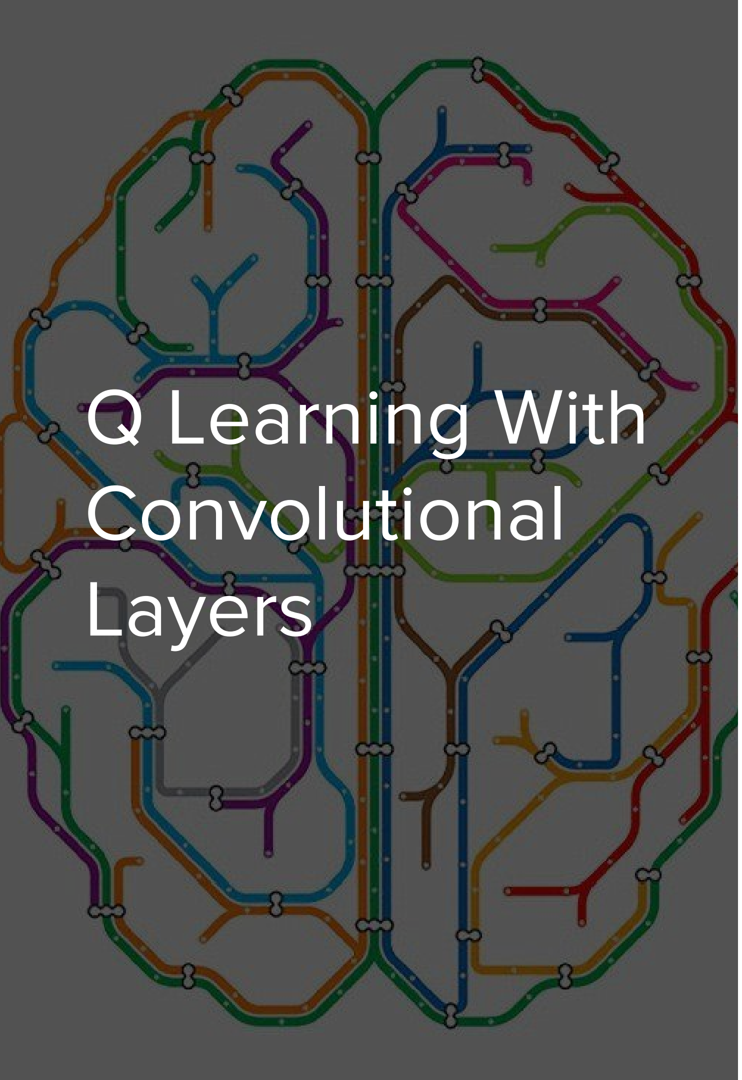
Q-Learning with Neural Networks (Cont.)

- Neurons: for fitting linear forms (y = a+bi, where “a” and “b” are constants and “i” is a state)
- Backpropagation: for fitting non-linear forms
- Algorithm of an incremental neuron
  - Compute the output
    - Output = sum of all w(j) * x(j), where w(j) and x(j) is the jth weight and input, respectively
  - Update the weights
    - w(i) = w(i) + a[target - output] * x(i) where target is the Q-factor

Q-Learning with Neural Networks (Cont.)

Algorithm of Q learning with a Neuron

- Assume there are two actions
- Initialize the weights to each action
- Let “i” be denoted as first state and “j” be denoted as the next state
- Let Q-old = w(1, a) + w(2, a)i
- Q-next(1) = w(1, 1) + w(2, 1)j ; Q-next(2) = w(2, 1) + w(2, 2)j
- Find the max of Q-next(1) and Q-next(2)
- Update relevant Q-factor: (1 - a)Q-old + a[immediate reward + gamma * Q-next]
- Update weights using the algorithm of the incremental Neuron

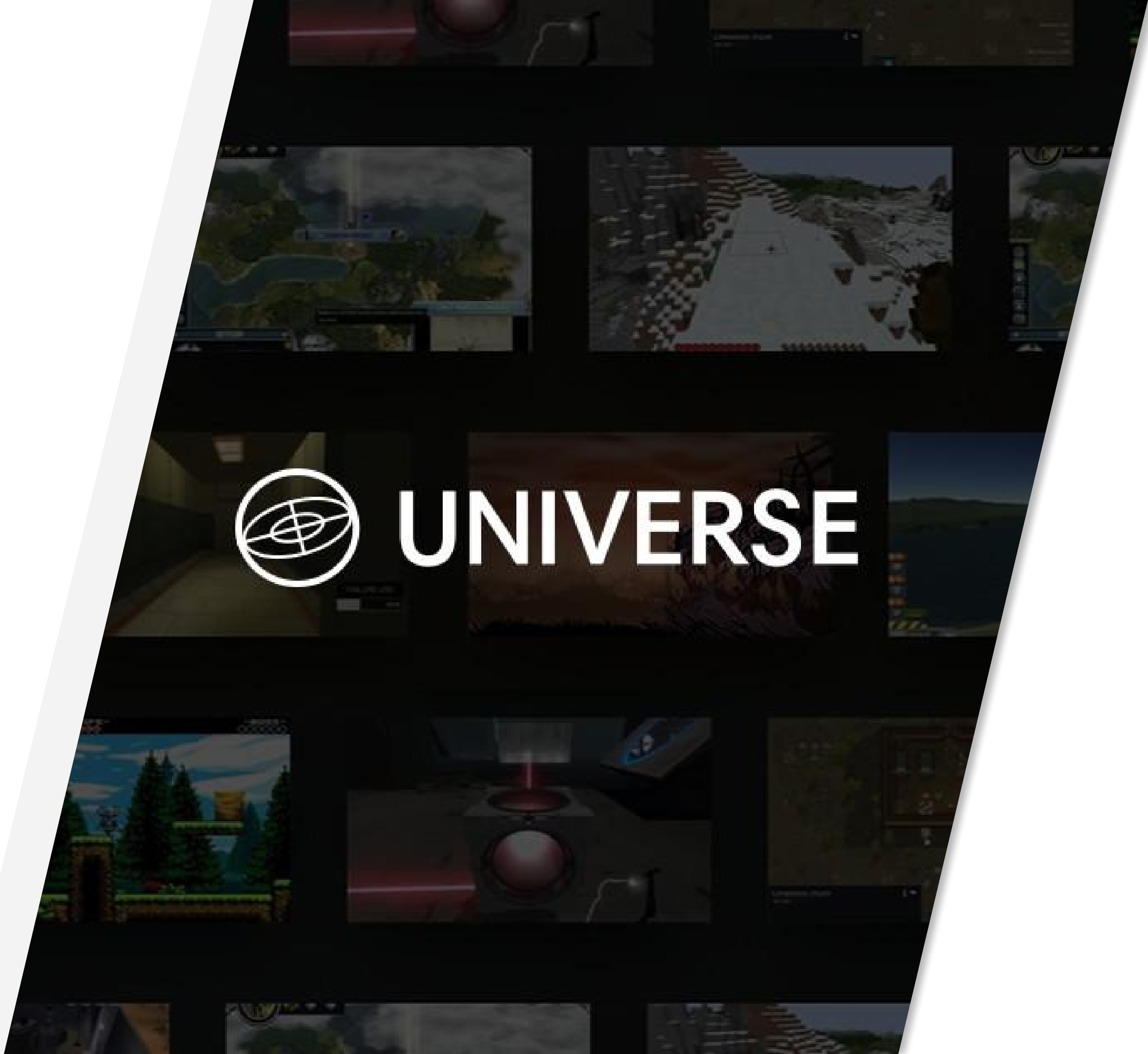


Q Learning With Convolutional Layers

- Make sense of game’s screen output
- Instead of considering each pixel, convolutional layers:
  - Allow agent to consider regions of an image
  - Maintain spatial relationships while sending info to higher levels of network
  - Similar to the primate visual cortex

What is OpenAI Universe?

- Platform to train AI on games
- Uses a computer like a human does
- Doesn't need special access to program internals, source code, or bot APIs.

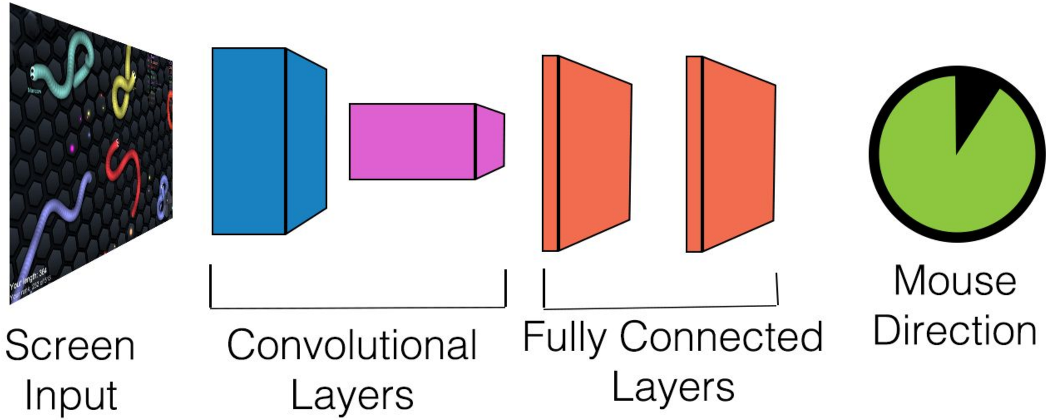


Our Example: Slither.io

- Use Q-learning with neural networks to train agent how to play slither.io
- Agent's actions: use angles
- Button click

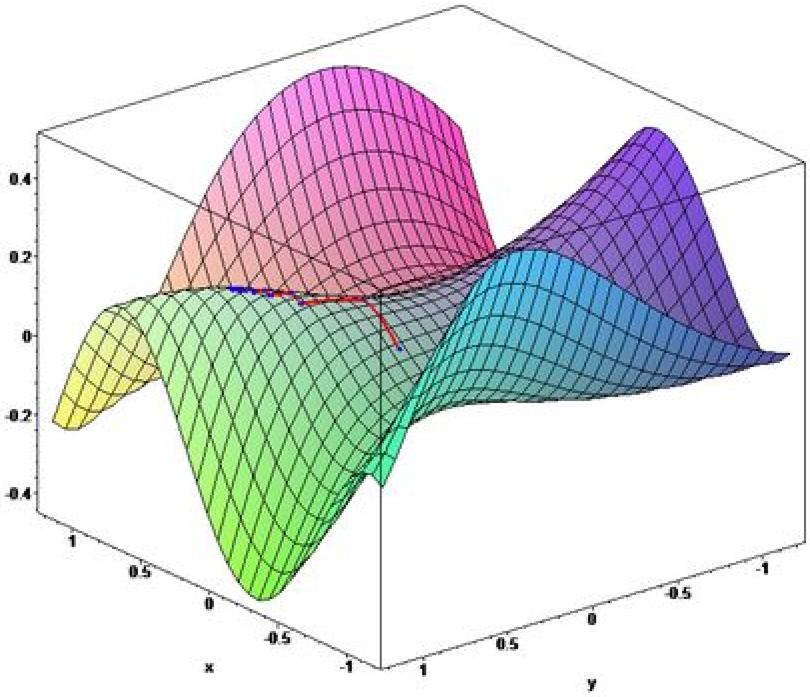
Building the Neural Network

- Keras
- Convolutional layers
- Fully connected layers
- Dimensionality



Training

- Each action is a training epoch
- Continuous play
- Network vs. Local
- Lag and processing power



Catastrophic Forgetting

- A model risks “forgetting” how to perform an action if it encounters an unlikely negative scenario.
- Easiest to explain by example.
- Consider a very simple situation where a mouse is placed between a pit and a block of cheese and it must choose whether to go left or right.

Initial Expected Rewards

-1 1

Assume when the mouse is flanked by objects, this setup will occur 90% of the time, and 10% of the time the objects will be reversed.

Thank You

References

http://outlace.com/Reinforcement-Learning-Part-3/

Image Sources

http://mnemstudio.org/path-finding-q-learning-tutorial.htm

http://www.cs.indiana.edu/~gasser/Smarts/learning.html

https://webdocs.cs.ualberta.ca/~sutton/book/ebook/node28.html

https://neuro.cs.ut.ee/demystifying-deep-reinforcement-learning/