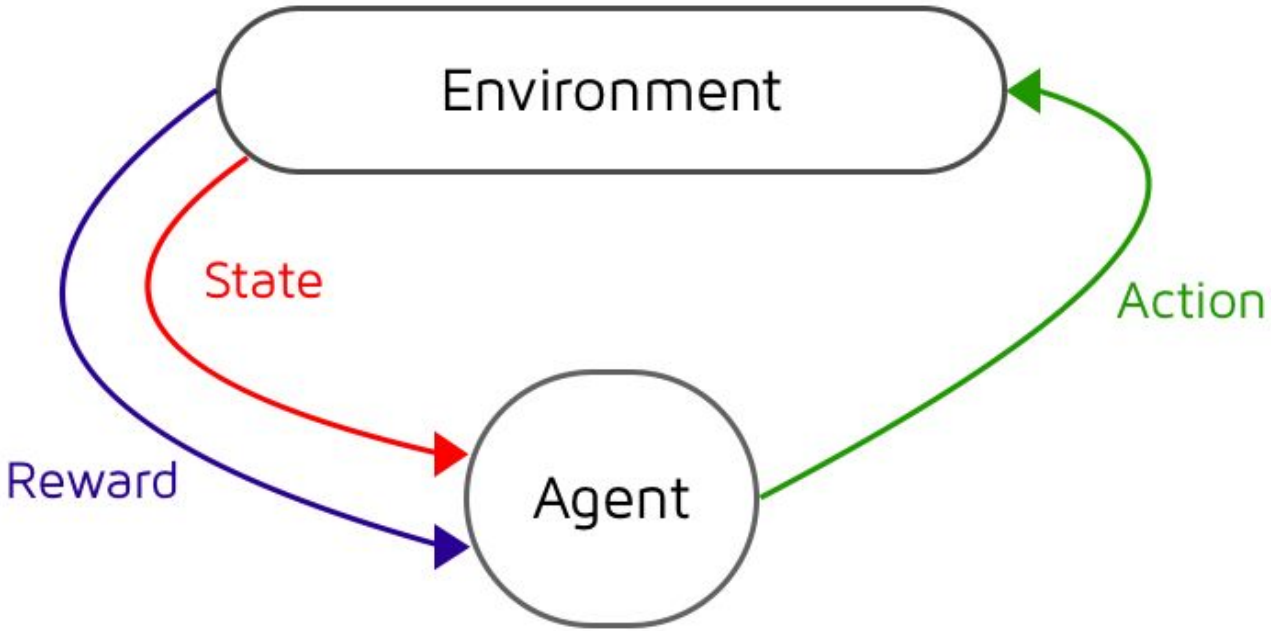
Huntsville Data Science, Artificial Intelligence, & Machine Learning

May 02, 2018

Deep Learning Deep Dive - Reinforcement Learning



What is it?

Wikipedia:

Reinforcement learning (RL) is an area of machine **learning** inspired by behaviourist psychology, concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward.

Russell & Norvig:

The task of reinforcement learning is to use observed rewards to learn an optimal (or nearly optimal) policy for the environment.

Beginnings

Alan Turing - Computing Machinery and Intelligence (1950) - I didn't even have to change this line from our last presentation.

The basic concept of "learning by trial and error" goes back to the "Law of Effect" from Edward Thorndike (search & memory)

The term "Reinforcement" and "Reinforcement Learning" were used in 1965 in a paper by Waltz & Fu

"Optimal Control" - Richard Bellman - mid 1950's

- Dynamic Programming
- Markovian Decision Processes

Intelligent Agents

Markov Decision Processes

1. **A finite set of states.**
2. **A set of actions available in each state.**
3. **Transitions between states.**
4. **Rewards associated with each transition.**
5. **A discount factor γ between 0 and 1.**
6. **Memorylessness.** Once the current state is known, the history can be erased because the current Markov state contains all useful information from the history. In other words, “the future is independent of the past given the present”.

Q Learning

Q Learning - evaluates which action to take based on an **action-value**

$$Q(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \left(\overbrace{\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}}}^{\text{learned value}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \right)$$

As the program chooses an action to take when changing state, the Q value of that state is updated based on the reward received, the learning rate, the discount factor, and the old value.

Discount factor = the difference between an immediate reward, and future rewards.

Temporal Difference (TD)

Temporal Difference (TD) Learning methods can be used to estimate these value functions. If the value functions were to be calculated without estimation, the agent would need to wait until the final reward was received before any state-action pair values can be updated. Once the final reward was received, the path taken to reach the final state would need to be traced back and each value updated accordingly.

On the other hand, with TD methods, an estimate of the final reward is calculated at each state and the state-action value updated for every step of the way.

The TD method is called a "bootstrapping" method, because the value is updated partly using an existing estimate and not a final reward.

Policy Learning

Another option - Policy Learning - learns a policy function, which is a map from each state to the best corresponding action at that state.

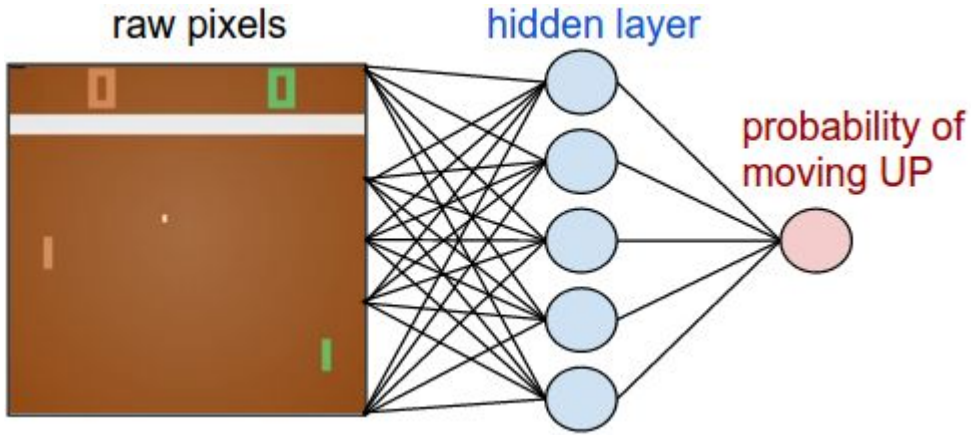
Most of the material I have been consuming lately references learning a “policy” with respect to Reinforcement Learning.

Any thoughts on a mechanism to learn a complex function?

Deep Reinforcement Learning

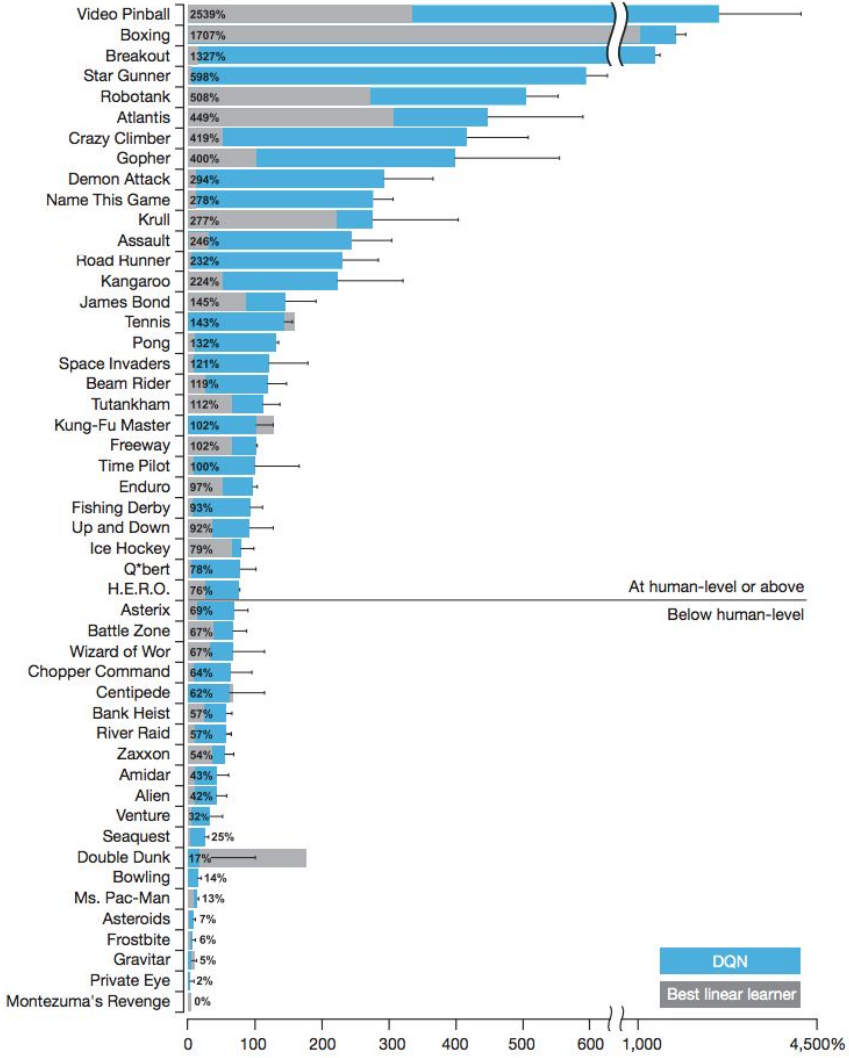
Deep Neural Networks began entering the conversation of Reinforcement Learning some time around 2014/2015.

For example - here's an architecture for Andrej Karpathy's Pong from Pixels implementation:



Deep Q Networks (DQN)

In 2015, DeepMind used a method called **deep Q-networks (DQN)**, an approach that approximates Q-functions using deep neural networks, to beat human benchmarks across many Atari games


Video Pinball 2539%
Boxing 1707%
Breakout 1327%
Star Gunner 598%
Robotank 508%
Atlantis 449%
Crazy Climber 419%
Gopher 400%
Demon Attack 294%
Name This Game 278%
Krull 277%
Assault 246%
Road Runner 232%
Kangaroo 224%
James Bond 145%
Tennis 143%
Pong 132%
Space Invaders 121%
Beam Rider 119%
Tutankham 112%
Kung-Fu Master 102%
Freeway 102%
Time Pilot 100%
Enduro 97%
Fishing Derby 93%
Up and Down 92%
Ice Hockey 79%
Q*bert 78%
H.E.R.O. 76%
Asterix 69%
Battle Zone 67%
Wizard of Wor 67%
Chopper Command 64%
Centipede 62%
Bank Heist 57%
River Raid 57%
Zaxxon 54%
Amidar 43%
Alien 42%
Venture 32%
Seaquest 25%
Double Dunk 17%
Bowling 14%
Ms. Pac-Man 13%
Asteroids 7%
Frostbite 6%
Gravitar 5%
Private Eye 2%
Montezuma's Revenge 0%
At human-level or above
Below human-level
DQN
Best linear learner
0
100
200
300
400
500
600
1,000
4,500%

Recurrent Neural Networks (RNNs) augmenting DQNs.

When an agent can only see its immediate surroundings (e.g. robot-mouse only seeing a certain segment of the maze vs. a birds-eye view of the whole maze), the agent needs to remember the bigger picture so it remembers where things are. This is similar to how humans babies develop object permanence to know things exist even if they leave the baby's visual field. RNNs are "recurrent", i.e. they allow information to persist on a longer-term basis.

Questions/Comments?

Note - most of the material for this presentation was sourced from:

Machine Learning for Humans

https://medium.com/machine-learning-for-humans/reinforcement-learning-6eacf258b265