Reinforcement Learning: From Games to Robotics

Sanjiban Choudhury
The story thus far ...
But what if the dynamics are unknown?

\[ \langle S, A, C, T \rangle \]

\( s, a \rightarrow s', T \)
Exploration vs Exploitation

From Dan Klein
Doors

$\alpha^1$

$\alpha^2$

$\alpha^3$

Round 1  Round 2  Round 3

Money

$+100$

$+1$

$-1000$
Activity!
Think-Pair-Share

Think (30 sec): What strategy would you pick doors?

Pair: Find a partner

Share (45 sec): Partners exchange ideas
What if we played the game over multiple time steps?
\[ t = 1 \quad \text{and} \quad t = 2 \]
Don’t see how good a door is until the end of the episode
When we know the MDP: Dynamic Programming!

$$V(S_t) \leftarrow \mathbb{E}_\pi [R_{t+1} + \gamma V(S_{t+1})]$$
When we don’t know MDP: Estimate Values

Monte Carlo

\[ V(S_t) \leftarrow V(S_t) + \alpha (G_t - V(S_t)) \]

Temporal Difference

\[ V(S_t) \leftarrow V(S_t) + \alpha (R_{t+1} + \gamma V(S_{t+1}) - V(S_t)) \]
Two Ingredients of RL

Exploration Exploitation

Estimate Values $Q(s, a)$
What is an application where we *really* need RL?
Games!
Why are Games perfect for RL?

You know the cost function
* +1 for winning and -1 for losing

You don’t know the transition function
* Don’t know what opponent will do, stochasticity in dice rolls etc.

Perfect for learning from trial and error (by playing itself!)
Name the Game!

One of the biggest success stories of RL

An agent trained via self-play and neural networks

Beat the world champion

Discovered totally new moves
The game of Backgammon
A backgammon position

white pieces move counterclockwise
black pieces move clockwise
TD-Gammon

Initialised with random weights
Trained by games of self-play
Using non-linear temporal-difference learning

$$\delta_t = v(S_{t+1}, w) - v(S_t, w)$$

$$\Delta w = \alpha \delta_t \nabla_w v(S_t, w)$$

Greedy policy improvement (no exploration)
Algorithm always converged in practice
Why was TD Gammon such a big deal?

1) Power of self-play
2) First time system could deal with stochastic uncertainty in dynamics (this broke everything in the deep blue style chess engines that were first super human)
3) Actually learned a value function: first "human" like behavior rather than just deep search with some heuristic values.
4) Changed elements of how backgammon is played because it demonstrated that certain positions were more valuable than self play
5) Demonstrated the TD idea could be scaled to super human performance at a game previously unaddressable.
6) Use a "deep" (i.e. Neural) representation within RL algorithms
7) Saw the benefit of imitation learning, but eventually got better than imitation
8) Showed boosting performance by some explicit forward search
Two Ingredients of RL: TD-Gammon

No exploration, was just greedy

Used temporal difference to estimate values

Exploration Exploitation

Estimate Values
Okay, but what about games with more complex representations?
Circa 2013

Playing Atari with Deep Reinforcement Learning

Volodymyr Mnih    Koray Kavukcuoglu    David Silver    Alex Graves    Ioannis Antonoglou

Daan Wierstra    Martin Riedmiller

DeepMind Technologies

{vlad,koray,david,alex.graves,ioannis,daan,martin.riedmiller} @ deepmind.com
Use NN to learn Q-function and then use to infer the optimal policy, $\pi(s)$.
Two Ingredients of RL: DQN

Epsilon-Greedy

Used Q-learning with Deep CNN

Exploration Exploitation

Estimate Values
Okay, but what about games that require some level of planning?
Mastering the game of Go without human knowledge

David Silver¹, Julian Schrittwieser³, Karen Simonyan¹, Ioannis Antonoglou¹, Aja Huang¹, Arthur Guez¹, Thomas Hubert¹, Lucas Baker¹, Matthew Lai¹, Adrian Bolton¹, Yutian Chen¹, Timothy Lillicrap¹, Fan Hui¹, Laurent Sifre¹, George van den Driessche¹, Thore Graepel² & Demis Hassabis¹
AlphaGoZero

Figure 1 | Self-play reinforcement learning in AlphaGo Zero. 

(a) The program plays a game $s_1, \ldots, s_T$ against itself. In each position $s_t$, an MCTS $\alpha_t$ is executed (see Fig. 2) using the latest neural network $f_\theta$. Moves are selected according to the search probabilities computed by the MCTS, $a_t \sim \pi_t$. The terminal position $s_T$ is scored according to the rules of the game to compute the game winner $z$. 

(b) Neural network training. The neural network takes the raw board position $s_t$ as its input, passes it through many convolutional layers with parameters $\theta$, and outputs both a vector $p_t$, representing a probability distribution over moves, and a scalar value $v_t$, representing the probability of the current player winning in position $s_t$. The neural network parameters $\theta$ are updated to maximize the similarity of the policy vector $p_t$ to the search probabilities $\pi_t$, and to minimize the error between the predicted winner $v_t$ and the game winner $z$ (see equation (1)). The new parameters are used in the next iteration of self-play as in (a).
Two Ingredients of RL: AlphaGoZero

Monte Carlo Tree Search

Deep value network

+ Deep policy network

Exploration Exploitation

Estimate Values
So what happens when we replace the human with a robot?
Learning Strategies in Table Tennis using Inverse Reinforcement Learning

Katharina Muelling · Abdeslam Boularias · Betty Mohler · Bernhard Schölkopf · Jan Peters
Reinforcement Learning in Robotics: A Survey

Jens Kober*†      J. Andrew Bagnell‡      Jan Peters§†

email: jkober@cor-lab.uni-bielefeld.de, dbagnell@ri.cmu.edu, mail@jan-peters.net

1992
(a) OBELIX robot

1994
(b) Zebra Zero robot

2001
(c) Autonomous helicopter

1996
(d) Sarcos humanoid DB
Curses in RL for Robotics

Curse of Real-World Samples

Curse of Reward Specification

Curse of Dimensionality

Curse of Model Uncertainty
No Silver Bullet!
Ingredients for Practical RL

Leveraging Demonstrations
Kakade et al., Bagnell and Scheider

Reward Shaping
Ng et al.

Conservatism and Trust
Kakade and Langford, Schulman et al.

Appropriate Policy Representation
Locally linear (Kolter and Ng),
Dynamic Motor Primitives (Schaal)

Plannable Models
Abbeel and Ng
What about Deep RL?
Position 2
real time
autonomous execution
How to Train Your Robot with Deep Reinforcement Learning – Lessons We’ve Learned

Julian Ibarz¹, Jie Tan¹, Chelsea Finn¹,³, Mrinal Kalakrishnan², Peter Pastor², Sergey Levine¹,⁴

Levine et al 2016

Harnoja et al 2019

Kalashnikov et al. 2018

(a) block stacking

Harnoja et al 2018

(b) door opening

Gu et al 2017
There is no Deep RL
There is no “Deep RL”

There is Deep Learning

There is Reinforcement Learning

Better representations for state / value function