

Reinforcement Learning : Closing Notes

Easwar Subramanian

TCS Innovation Labs, Hyderabad

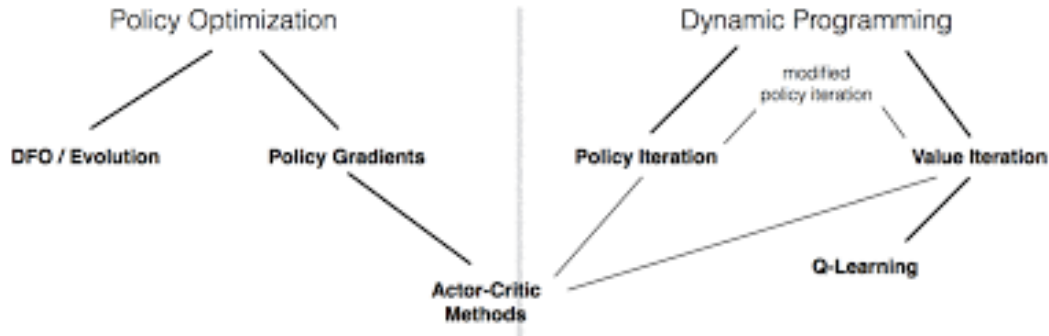
Email : easwar.subramanian@tcs.com / cs5500.2020@iith.ac.in

Novemer 18, 2022

Overview of this Lecture

- 1 Landscape, Summary and References
- 2 Other Topics
- 3 Practical Tips – Based on John Schulman's talk on Nuts and Bolts of Deep RL

Landscape, Summary and References



Markov Property, transition probabilities, Markov reward process, Markov decision process

Three Key entities of RL

- ▶ Value Function - V
- ▶ Action Value Function - Q
- ▶ Policy - π

Optimal policies, notion of greedy policy, Bellman equations (evaluation and optimality)

Lecture Numbers : 1 to 5

Reference : David Silver's Lecture on RL

Key Algorithms

- ▶ Value Iteration
- ▶ Policy Iteration

Drawbacks

- ▶ Requires full prior knowledge of the dynamics of the environment
- ▶ Can be implemented only on small, discrete state spaces

Lecture Numbers : 6 to 8

Reference : David Silver's Lecture on RL

Proofs on convergence available at : <https://runzhe-yang.science/2017-10-04-contraction/>

Notion of bootstrap, lookahead and backup

Evaluation Algorithms

- ▶ Monte-Carlo methods (First Visit and Every Visit MC)
- ▶ Temporal difference methods
- ▶ TD- λ methods

Control Algorithms

- ▶ SARSA
- ▶ Watkin's Q-learning algorithm

Drawback : Not extendible to high dimensional state and action spaces

Lecture Numbers : 9 to 12

**References : David Silver's Lecture on RL and Relevant Chapters on Sutton
and Barto Book**

Use of neural nets as function approximators, convergence of NN based algorithms

Algorithms

- ▶ Monte Carlo based value function estimation
- ▶ Fitted V iteration and Q iteration
- ▶ Deep Q networks

Lecture Numbers : 13 and 14

References : Deep RL course in Berkeley (2017,2018), Deep RL Bootcamp, Minh (2015), Riedmiller(2006)

Notion of Policy gradients, derivation of policy gradient expression, temporal structure, baseline and discounting for variance reduction, advantage function, deterministic policy gradient

Key Algorithms

- ▶ Actor-critic algorithms A2C and A3C
- ▶ DDPG

Lecture Numbers : 15 to 18

References : Deep RL course in Berkeley (2017,2018), Deep RL Bootcamp, Minh (2016), Lillicrap(2016)

Different approach to policy gradients by looking at distance in policy space;
Surrogate loss function; Constrained policy optimization

Key Algorithms

- ▶ Natural Policy Gradient
- ▶ TRPO
- ▶ PPO

Lecture Numbers : 22 to 23

References : Deep RL course in Berkeley (2017,2018), Deep RL course in Berkeley, Joshua Aicham lecture by the same topic, NPG(Kakade, 2001), TRPO (Schulman2015) PPO (Schulman, 2017)




Bandit Concepts :

Naive Exploration, Optimistic Initialization, Optimism in the face of Uncertainty

Key Algorithms

- ▶ UCB and Thompson Sampling
- ▶ Monte Carlo Tree Search (Tree search methods)

Lecture Numbers : 19 to 21 References : David Silver's Lecture on RL and Relevant Chapters on Sutton and Barto Book

-  Reinforcement Learning : Sutton and Barto
-  Dynamic Programming and Optimal Control (I and II) by Bertsekas
-  Reinforcement Learning and Optimal Control, Bertsekas and Tsitsiklis
-  David Silver's course on Reinforcement Learning
-  Stanford course on Deep RL
-  Deep RL BootCamp (Pieter Abeel)
-  John Schulman's lectures in Policy Gradient Methods
-  ... and many others

Other Topics

- ▶ **Central Question :** How can we make decisions better if we know system dynamics ? (possibly when state and action space is high dimensional or continuous)
 - ★ Games, navigating car etc, simulated environments
- ▶ If system dynamics in not known, can we identify them ?
 - ★ System identification - fit unknown parameters to a known model
 - ★ Learning - fit a general purpose model to observed transitions

Forward Reinforcement Learning

Given states $s \in \mathcal{S}$, actions $a \in \mathcal{A}$, reward function $\mathcal{R}(s, a)$ and possibly transition probabilities $P(s'|s, a)$ and

- Learn policy $\pi^*(a|s)$

Inverse Reinforcement Learning

Given states $s \in \mathcal{S}$, actions $a \in \mathcal{A}$, a policy $\pi(a|s)$ and possibly transition probabilities $P(s'|s, a)$ and

- Learn reward function $\mathcal{R}(s, a)$

and then use it learn $\pi^*(a|s)$

- ▶ Forward transfer : train on one task, transfer to a new task
- ▶ Multi-task transfer : train on many task, transfer to a new task
- ▶ Multi-task meta learning : learn to learn from many tasks

Question

How can we better utilize our computational resources to accelerate RL progress ?

Examples

- ▶ DQN and its variants (Large scale RL) (2013)
- ▶ GORILLA (2015)
- ▶ A3C (2016)
- ▶ IMPALA (2018)

- ▶ Topics like Hierarchical RL, Feudal RL etc
- ▶ Imitation Learning
- ▶ Partially Observable MDPs
- ▶ Multi-agent RL

- ▶ Repository of multitude of environments
- ▶ Baseline implementation of several popular algorithms

Practical Tips – Based on John Schulman's talk on Nuts and Bolts of Deep RL

- ▶ Test on small use cases and then run on medium-sized problems
- ▶ Interpret and visualize learning process: state visitation, value function, etc
- ▶ Construct toy problems where your idea will be strongest and weakest, where you have a sense of what it should do

- ▶ Progressively increase the state and action space formulation
- ▶ Reward shaping is crucial to test the working of algorithm

- ▶ Explore sensitivity to each parameter
- ▶ Health indicators
 - ★ Quality of value function
 - ★ Entropy of the policy
 - ★ KL diagnostics
- ▶ Run a battery of benchmarks

- ▶ Compare against baselines
 - ★ Cross entropy method
 - ★ Well tuned policy gradient method
 - ★ Well tuned Q-learning or SARSA based method
- ▶ Use multiple random seeds
- ▶ Don't be deterred by published works
 - ★ TRPO on Atari: 100K timesteps per batch for $KL=0:01$
 - ★ DQN on Atari: update freq=10K, replay buffer size=1M

- ▶ DQN converges slowly - for ATARI it is often necessary to wait for 10-40 million frames (couple of hours to a day of training on GPU) to see results significantly better than random policy. **Be Patient**
- ▶ Optimize memory usage carefully: you'll need it for replay buffer
- ▶ Learning rate and exploration schedules are vital
- ▶ Do use Double DQN with prioritized experience replay – significant improvement

- ▶ Policy Initialization : More important than in supervised learning: determines initial state visitation
- ▶ KL spike \rightarrow drastic loss of performance
- ▶ Not recommended to use DDPG when you have discrete action set

- ▶ Automate your experiments;
- ▶ Techniques from supervised learning don't necessarily work in RL: batch norm, dropout, big networks
- ▶ Read older textbooks and theses, not just conference papers

- ▶ Games
- ▶ Robotics
- ▶ Wealth Management
- ▶ Supply Chain Management
- ▶ Control Systems Applications

- ▶ Risk Sensitive RL
- ▶ Optimizing Expected Reward with Constraints
- ▶ Multi Agent Systems
- ▶ Transfer and Meta Learning in RL
- ▶ Improving Data efficiency in RL

- ▶ Things that we can all do (Walking) (Evolution, may be)
- ▶ Things that we learn (driving a bicycle, car etc)
- ▶ We learn a huge variety of things (music, sport, arts etc)
- ▶ We can learn 'difficult' tasks as well

We are still far from building a 'reasonable' intelligent system

- ▶ We are taking baby steps towards the goal of building intelligent systems
- ▶ **Reinforcement Learning (RL) is one of the important paradigm towards that goal**

Thank You and Good Luck