

STATS 701 – Theory of Reinforcement Learning

Introduction

Ambuj Tewari

Associate Professor, Department of Statistics, University of Michigan
tewaria@umich.edu

<https://ambujtewari.github.io/stats701-winter2021/>

Slide Credits: Prof. M. Vidyasagar @ IIT Hyderabad, India

Winter 2021

Outline

- 1 About the Course
- 2 Introduction to Reinforcement Learning
 - What is Reinforcement Learning?
 - Introduction to Markov Decision Processes (MDPs)
 - Introduction to Reinforcement Learning
- 3 Some Examples of Reinforcement Learning
 - Backgammon
 - AlphaGo and AlphaZero
 - Beyond Board Games

Outline

- 1 About the Course
- 2 Introduction to Reinforcement Learning
 - What is Reinforcement Learning?
 - Introduction to Markov Decision Processes (MDPs)
 - Introduction to Reinforcement Learning
- 3 Some Examples of Reinforcement Learning
 - Backgammon
 - AlphaGo and AlphaZero
 - Beyond Board Games

Logistics

- Will record lectures and post links in Canvas
- Live sessions are for registered students only
 - 3 credits (default option)
 - 1.5 credits (need approval)
 - official audit (need approval)
- Unofficial auditors won't attend live sessions and will be added to Canvas as "observer"
- First part on "core" RL: traditional lecture format
- Second part: group discussions of recent (theoretical) RL papers

Logistics Contd.

- We have a slack workspace for this course
 - Go to oldest announcement on canvas for link to join
- Might give out homework but will not be graded (no exams)
- Grade based on paper presentation and course project:
 - 1-2 pages proposal 20%
 - 20-30 minutes paper presentation 30%
 - 4-8 pages report 50% (required only for 3 credit option students)

Course Project

- Teams of **two** students (email me if you want to work alone or want a larger team)
- Many kinds of projects possible:
 - Read a small set of related papers and summarize their findings including reproduction of some of the published results
 - Apply some RL algorithms to a problem in your field of research and make some conjectures that theory could address
 - Original research that is publishable in a top ML conference
- You should plan to work about 30-40 hours on the project (3-4 hrs per week \times 10 weeks)
- Get started early, form a team, pick a topic, and **submit a 1-2 page proposal by March 2, 2021**
- Ask me if you need help with project ideas

Overview of the Course

- Core
 - Introduction to MRPs, MDPs and RL
 - Solution methods for known MDPs
 - MC and TD methods for tabular MDPs
 - Online learning and bandits
 - Online learning in MDPs (focus on regret analysis)
- Advanced (flexible)
 - Function approximation (from simple cases like linear functions to more difficult cases like deep RL)
 - Continuous state and action spaces (e.g., LQR systems)
 - Offline learning and off-policy evaluation
 - Multi-task, transfer, lifelong, meta learning etc. in RL
 - Hierarchical RL
 - Multi-agent RL


Prerequisites

- This is a **theoretical** course with a **mathematical** flavor
- If you're primarily interested in RL **applications**, this may not be the right course for you
- We will look at many theorems, some with proofs
- Prior exposure to RL, at least MDPs, **strongly recommended**
- **Mathematical** maturity a must!
- Some math tools we will use
 - Markov chains (on a finite space)
 - Contraction Mapping Theorem
 - Concentration and Uniform Convergence of Empirical Means
 - Stochastic Approximation

References

- R. S. Sutton and A. G. Barto. Reinforcement Learning: An Introduction (2nd ed)¹. MIT Press, 2018. *A classic book now in its second edition, keeps the math very light*
- D. Bertsekas. Dynamic Programming and Optimal Control. Athena Scientific, 2017 (Vol I, 4th ed) and 2012 (Vol II, 4th ed). *Written from a control theory viewpoint, higher math level*
- T. Lattimore and C. Szepesvári. Bandit Algorithms². Cambridge University Press, 2020. *Bandits are a special, easier case of RL.*
- M. L. Puterman. Markov Decision Processes: Discrete Stochastic Dynamic Programming. John Wiley, 2005. *Standard reference on MDPs, no RL*

¹<http://incompleteideas.net/book/RLbook2020.pdf>

²<http://tor-lattimore.com/downloads/book/book.pdf> 

Other Resources

- Prof. M. Vidyasagar's notes on RL available on Canvas look under Files -> MV-notes.pdf
- 4 workshops from the Theory of RL program at the Simons Institute, UC Berkeley, Fall 2020
<https://simons.berkeley.edu/programs/r120>
- Courses elsewhere and some other resources
<https://ambujtewari.github.io/stats701-winter2021/resources.html>

Outline

1 About the Course

2 Introduction to Reinforcement Learning

- What is Reinforcement Learning?
- Introduction to Markov Decision Processes (MDPs)
- Introduction to Reinforcement Learning

3 Some Examples of Reinforcement Learning

- Backgammon
- AlphaGo and AlphaZero
- Beyond Board Games

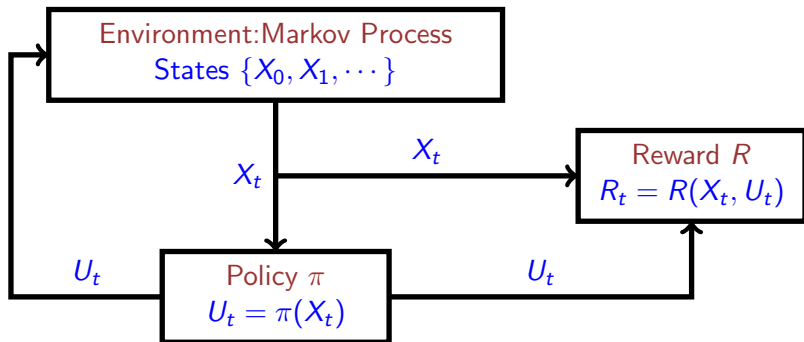
Outline

- 1 About the Course
- 2 Introduction to Reinforcement Learning
 - What is Reinforcement Learning?
 - Introduction to Markov Decision Processes (MDPs)
 - Introduction to Reinforcement Learning
- 3 Some Examples of Reinforcement Learning
 - Backgammon
 - AlphaGo and AlphaZero
 - Beyond Board Games

What is Reinforcement Learning?

Reinforcement learning (RL) is about learning to take “optimal” (or nearly optimal) decisions in an uncertain environment.

“Optimal decisions”: those that maximize accumulated rewards



Ingredients of Reinforcement Learning

- Three ingredients: Environment, Action, Reward
- Menu of actions \mathcal{U} available to the agent at each time.
 - Assumed to be a finite set $\mathcal{U} = \{u_1, \dots, u_m\}$.
 - Assumed to be **the same set** at each instant of time.
- The “environment” characterized by a “state” $X_t \in \mathcal{X}$.
 - Again, \mathcal{X} is assumed to be finite. Say $\mathcal{X} = \{x_1, \dots, x_n\}$.
 - Current state X_t and action U_t at time t define the statistics of the next state X_{t+1} .

Ingredients of Reinforcement Learning (Cont'd)

- The “reward” $R(X_t, U_t) \in \mathbb{R}$:
 - Can be a deterministic or a random function of X_t and U_t .
 - Can be given at time t or time $t + 1$.
 - If reward is given at time $t + 1$, then R can have a “joint distribution” with X_{t+1} . We can talk about $\Pr\{X_{t+1}, R_{t+1} | X_t, U_t\}$.
 - If random, $R(X_t, U_t)$ can take values in a continuum, e.g., $[0, M]$.
Need to watch out for technicalities, so often R is also assumed to take values in a discrete set.
- The “discount factor” $\gamma \in (0, 1)$.
 - The case $\gamma = 1$ usually causes difficulties.

Some Consequences of RL Model

- The same action $u_i \in \mathcal{U}$ taken at different times need not produce the same reward each time (because the state may be different).
- If reward R is a random function of (X_t, U_t) , then the same state-action pair can produce different rewards at different times.
- This is true even if the model of how X_{t+1} and the reward depend on U_t is perfectly known.
- If even this dependence is not known but has to be “learned,” then the situation is still more complicated.

How is RL different from SL

- A lot of times when people say “Machine Learning” they mean “Supervised Learning” (SL)
- RL is a harder problem than SL
- **Very brief** discussion of supervised learning to make the point
- In SL, learner gets immediate and (mostly) accurate feedback.
- In RL, the learner gets feedback that is:
 - **partial**: rewards of actions not taken are not observed
 - **evaluative not prescriptive**: agent is just given rewards and is not explicitly told what to do
- Nobody would claim that SL provides a general framework for building AI agents
- But perhaps RL does provide such a framework

Supervised Learning

In “supervised learning” systems, the objective is to “train” a machine on a set of “labelled training” data. After that, the machine is “tested” by being asked to predict the correct label for previously unseen data.

“Generalization” quantifies the quality of the prediction:

- In a classification problem where the label is binary, the fraction of correct labels.
- in a regression problem where the label is a real number, a least-squares measure.

Deep Learning and Deep RL

- Artificial neural networks (ANNs) are a class of non-linear functions approximators
- Have seen a huge resurgence in AI and ML since about 2005 rebranded as “deep learning”
- “Deep” = many layers in the network
- “Deep RL” = RL plus use of deep ANNs as function approximators (popular 2015 onwards)

Outline

- 1 About the Course
- 2 Introduction to Reinforcement Learning
 - What is Reinforcement Learning?
 - Introduction to Markov Decision Processes (MDPs)
 - Introduction to Reinforcement Learning
- 3 Some Examples of Reinforcement Learning
 - Backgammon
 - AlphaGo and AlphaZero
 - Beyond Board Games

Introduction to Markov Decision Processes (MDPs)

To give some “structure” to the previous vague description of RL, we formulate RL as Markov Decision Processes (MDPs) with unknown dynamics and reward function.

(Proper review of Markov processes in later lectures.)

- The sequence of states $\{X_t\}_{t \geq 0}$ is a Markov process, where the state transition matrix at each time t depends on the action U_t .
- The reward $R(X_t, U_t)$ can be a deterministic or a random function of (X_t, U_t) , and can be paid at time t or time $t + 1$.
- The discount factor γ is typically assumed to be strictly less than one.

Introduction to MDPs – Con'td

Key idea in MDPs: **Policy**: A policy π is a map from \mathcal{X} into \mathcal{U} . Mostly it is deterministic, though probabilistic policies are possible.

Expected discounted future reward:

$$V_{\pi} = E \left[\sum_{t=0}^{\infty} \gamma^t R(X_t, U_t) \mid U_t = \pi(X_t) \right].$$

Objective: Choose the **optimal** policy that maximizes V_{π} .

Outline

- 1 About the Course
- 2 Introduction to Reinforcement Learning
 - What is Reinforcement Learning?
 - Introduction to Markov Decision Processes (MDPs)
 - Introduction to Reinforcement Learning
- 3 Some Examples of Reinforcement Learning
 - Backgammon
 - AlphaGo and AlphaZero
 - Beyond Board Games

What if an MDP has Unknown Dynamics

- Can the dynamics of an “unknown” Markov process be inferred from its state sequence?
- Can the dynamics of an MDP and its reward function be inferred from the sequence $\{X_t, U_t, R_t\}_{t \geq 0}$?
- How can this inference be used to construct an optimal policy?

Exploration vs. Exploitation

At each time t , we can update our model of the MDP and the reward, compute the optimal policy for this model, and implement **one step** of the optimal policy; then repeat.

This is called the “**certainty equivalence**” approach. But it doesn’t work well.

It is better, at least initially, to choose a “suboptimal policy” (based on the current model) with some nonzero probability, and then slowly decrease to zero this probability of choosing a suboptimal policy.

Outline

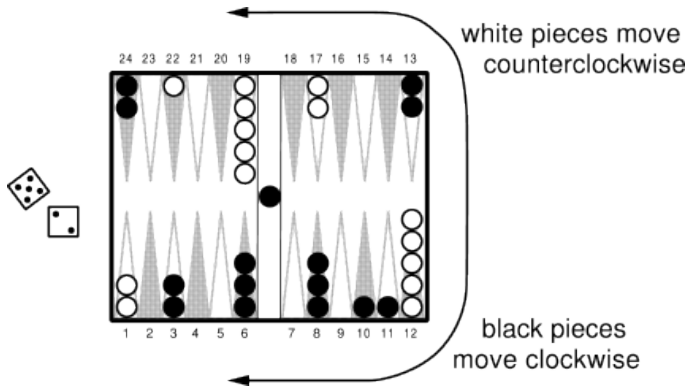
- 1 About the Course
- 2 Introduction to Reinforcement Learning
 - What is Reinforcement Learning?
 - Introduction to Markov Decision Processes (MDPs)
 - Introduction to Reinforcement Learning
- 3 Some Examples of Reinforcement Learning
 - Backgammon
 - AlphaGo and AlphaZero
 - Beyond Board Games

Outline

- 1 About the Course
- 2 Introduction to Reinforcement Learning
 - What is Reinforcement Learning?
 - Introduction to Markov Decision Processes (MDPs)
 - Introduction to Reinforcement Learning
- 3 Some Examples of Reinforcement Learning
 - Backgammon
 - AlphaGo and AlphaZero
 - Beyond Board Games

A Backgammon Board

Backgammon is a game that combines chance (throwing two dice) and strategy (what do after the dice are thrown). The board is shown below.



Solution of Backgammon Using TD Learning

- As with most board games with two or more players, the number of possibilities is enormous
- Approximate assessment of possible policies is the only way
- Gerald Tesauro solved this problem using a RL approach called “temporal difference (TD) learning.”
- He developed a series of programs called TD-Gammon
- Using [self-play](#) TD-Gammon eventually learned to play at, or higher than, the highest human level
 - Project Idea: Despite wide-spread use, self-play is not understood well theoretically
- An important precursor of Tesauro’s work was Arthur Samuel’s work on checker playing programs
- Detailed discussion in the book by Sutton & Barto, Section 16.1

Outline

- 1 About the Course
- 2 Introduction to Reinforcement Learning
 - What is Reinforcement Learning?
 - Introduction to Markov Decision Processes (MDPs)
 - Introduction to Reinforcement Learning
- 3 Some Examples of Reinforcement Learning
 - Backgammon
 - AlphaGo and AlphaZero
 - Beyond Board Games

Go: A Challenging Board Game

Go is played on a 19×19 board, with 181 white pieces and 180 black pieces.



AlphaGo

- In 2016, the Go-playing program AlphaGo developed by Deep Mind (since acquired by Google) defeated Lee Sedol, eighteen times World Go champion (though he was not the champion at the time).
- In 2017 an improved version defeated the sitting world champion Ji Kie.
- This caused a lot of excitement at the time.

AlphaZero

- AlphaZero, also developed by Deep Mind, was a “general-purpose” game-playing program that taught itself how to play a variety of games using “pure RL” augmented with MCTS (Monte Carlo tree search) and ANNs
- Starting from a common architecture, AlphaZero could train itself to play chess, Go and Shogi (Japanese chess).
- AlphaZero defeated AlphaGo at Go, Stockfish at chess and Elmo at Shogi.
- AlphaZero suggested that “prior domain knowledge” may not be an advantage!
- Detailed discussion in the book by Sutton & Barto, Section 16.6

Outline

- 1 About the Course
- 2 Introduction to Reinforcement Learning
 - What is Reinforcement Learning?
 - Introduction to Markov Decision Processes (MDPs)
 - Introduction to Reinforcement Learning
- 3 Some Examples of Reinforcement Learning
 - Backgammon
 - AlphaGo and AlphaZero
 - Beyond Board Games

Some Recent Applications of RL

- Autonomous Greenhouse: MSR's Sonoma Project <https://www.microsoft.com/en-us/research/project/sonoma/>
- Autonomous Soaring: MSR's Project Frigatebird <https://www.microsoft.com/en-us/research/project/project-frigatebird-ai-for-autonomous-soaring/>
- Alphabet's (Google's parent company) Loon: Autonomous navigation of stratospheric balloons using reinforcement learning <https://doi.org/10.1038/s41586-020-2939-8>
- From my own group: TorsionNet: A reinforcement learning approach to sequential conformer search <https://arxiv.org/abs/2006.07078>