Reinforcement Learning

By
Saptarashmi Bandyopadhyay
First Year PhD Student
Department of Computer Science and Engineering
Pennsylvania State University, University Park

Outline

- Reinforcement Learning Book
  - Introduction (Chapter 1)
  - Multi-armed bandit problem (Chapter 2)
  - Finite Markov decision Process (Chapter 3)
- A Course in Machine Learning
  - Imitation Learning (Chapter 18)
- Journal paper on Human-level control through deep reinforcement learning

Introduction

- Re-inforcement Learning- Goal directed learning from interaction with the environment
- Learning how to map situations to actions to maximize a numerical reward signal
- Trial-and-error search to find which actions yield the most reward
- Delayed reward
  - Short-term reward
  - Long-term reward

Learning agents

- Able to sense state of the environment
- Able to take actions based on affecting the state
- Goal or goals related to the state of the environment
- Complete interactive goal-seeking agent
- Can be a component of a bigger behaving system

Difference with Supervised Learning

- Supervised learning: learning from a trained set of labeled known samples
- System to generalize for situations outside training dataset
- But in interactive problems, impractical to get desired behavior that are both correct and reflects all situations of the agent's action space
- Learning from experience in reinforcement learning

Difference with Unsupervised Learning

- Unsupervised learning: learning of structures hidden in unlabeled data
- Reinforcement learning(RL) maximizes the goal reward instead of trying to find hidden structures
- Uncovering the structure useful but does not address the core objective of RL

Trade-off between exploration vs. exploitation

- Exploitation of experience in order to obtain reward
- Exploration to select better actions
- Neither exploration nor exploitation can be

done solely without task failure

Illustration in chess

- Move of an expert chess player
- Choice by planning possible actions and counter-responses in the long term
- immediate, intuitive assessment of whether particular positions and moves are more favorable

Elements of Reinforcement Learning (RL)

- Policy defines the behavior of learning agent
- Reward signal defines the goal of RL

problem

- Value function highlighting long-term

desirability of the state space

- Model defining interaction with the

environment

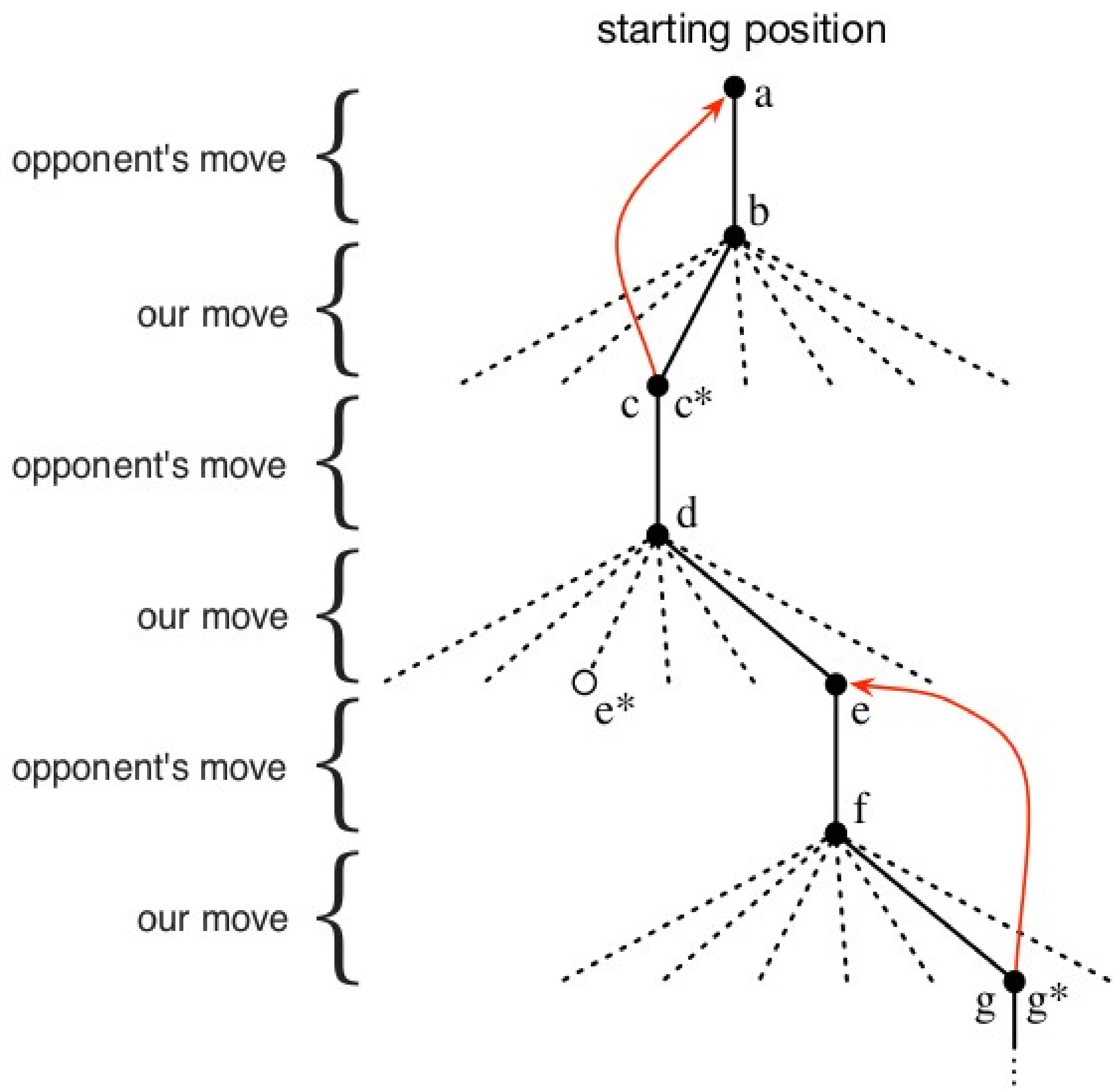
Limitations of Reinforcement Learning

- Highly dependent on the idea of state as input to the policy, value function and input and output of the model
  - Current focus on decision-making
- Problems can be unstructured around estimation of value functions like evolutionary methods e.g. genetic algorithms
  - Multiple static policies interacting over a long time with separate instances of the environment

Illustration in tic-tac-toe game

- Mini-max algorithm assumes the playing process of the opponent
- Dynamic programming can compute optimal solution but requires complete specification of the opponent
- Information often unavailable apriori but can be estimated from experience
- $V(S_t) \leftarrow V(S_t) + \alpha * [V(S_{t+1}) - V(S_t)]$
  - S_t :State before greedy move
  - S_{t+1} :State after greedy move

State space representation of tic-tac-toe game



Multi-armed bandits problem

- Non-associative, evaluative feedback problem
- Choice from k different options
- After each choice to maximize numerical reward from a stationery probability distribution based on selected action
- $q_*(a) = E[R_t \mid A_t = a]$
- $q_*(a)$: value of an arbitrary action a
- A_t is action at time t and R_t is corresponding reward

Action value methods

- Methods to estimate value of actions for action-selection decisions
- Averaging the rewards
- Greedy action selection method
  - $A_t = \text{argmax}\ Q_t(a)$
  - Where $Q_t(a)$ is estimated action at time t

10-armed testbed

- Set of 2000 randomly generated k-armed

bandit problems with k=10

- Comparison to find how effective greedy and ε-greedy selection methods are
- ε-greedy selection
  - greedy behavior mostly
  - but sometimes with a small probability ε random selection from among all actions with equal probability, independent of the action-value estimates

Incremental Implementation

- Computationally efficient method for average

in estimating action

- $Q_{n+1} = Q_n + (1/n) * [R_n - Q_n]$
  - Given Q_n and the nth reward R_n
- General formula is
- New estimate ← old estimate + step-size*(target - old estimate)

Non-stationary problem

- In stationary bandit problems, reward probabilities do not change over time
- In non-stationary problems, more weightage to recent rewards than past rewards
- Exponential recency weighted average

Optimistic initial value

- Previous methods biased by initial action value estimate $Q_1(a)$
- Sample-average methods, bias disappears on selection of all subjects at least once
- Bias permanent for constant α methods
- Exploration by action value methods to find optimistic initial values.

Upper-confidence-bound action selection

- $A_t = \text{argmax}[Q_t(a) + c * \text{sqrt}\,[\ln(t)/N_t(a)]$
  - $N_t(a)$ it is the number of times, a has been selected prior to time t
- Square root term denotes the uncertainty or variance in the estimate of a's value
- Maximum argument is an upper bound on the possible true value of action a.

Gradient bandit problem

- Relative preference of one action over another
- Determined by a soft-max distribution

Contextual bandit

- Associative search
- Several k-armed bandit problems
- At each step, one of the problems chosen at random
- Policy associating each task with the best action
- Actions affecting the next situation and reward giving the full reinforcement learning problem

Finite Markov Decision Process

- Formalizes the problem of reinforcement learning
- Formalizes sequential decision making
- Actions influence immediate rewards and future rewards

Agent-environment interface

- Agent: Learner and decision maker
- Environment: the thing agents interact with
- The probabilities completely characterize the environment's dynamics
- $p(s', r | s, a) = Pr\{S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\}$
  - Dynamics function of MDP
- Markov property: state has to include all aspects of the past agent-environment interaction

Goals and rewards

- Reward hypothesis
- Goals: maximization of the expected value of the cumulative sum of an obtained scalar signal called reward

Returns and episodes

- Return is some specific function on the reward sequence
- Objective to maximize return
- Episodes: when agent-environment interaction naturally breaks into subsequences
- All episodes can end in the same terminal state with different rewards for different results
- Episodic task: Tasks with such episodes

Unified notation for episodic and continuing tasks

- Episodic tasks need extra notation
- Rather than one long sequence of time steps, series of episodes with finite sequence of time steps

Policies and value functions

- Policy: mapping of states to probabilities of selecting each possible action
- Value function of a state s and policy π: expected return when starting in s and following π thereafter.
- State value function for policy π
- Action value function for policy π

Optimal policies and optimal value functions

- Optimal policy: Policy better than or equal to

all other policies

- Optimal state-value function:

$v_*(s) = \max v_\pi (s)$

- Optimal action-value function:

$q_*(s,a) = \max q_\pi (s,a)$

Optimality and approximation

- Extremely high computation cost
- Constraint of available memory
- Approximation e.g. tesauro's backgammon

game

- online nature of reinforcement learning
- Approximates optimal policies
  - To learn to make good decisions for frequent states
  - Less effort fore infrequent states

References

Reinforcement learning book by Richard S. Sutton and Andrew G. Barto, Second Edition, 2018

Imitation learning

- Learning by demonstration
- Access to an expert
- Goal: to learn function f that maps data from

environmental interaction to actions

- Close relationship with reinforcement

learning

- Function f is called a policy

Certain nuances

- Conversion of expert data to multi-class classification problems
- Goodness of the expert
- Goodness of the multi-class classification algorithm
- Upper bound of loss suffered by supervised imitation learning algorithm is a function of
    - Expert quality
    - Error rate of learned classifier

Failure Analysis

- Challenge is recovery from failure
- Compounding error
- Expert can easily get zero loss
- Upper and lower bounds of loss pretty far apart

Dataset aggregation

- Ideally generalistic imitation learning
- Practically impossible to get expert data from all possible combinations in the world
- Train model based on configurations of the expert ?!
- So, dagger (dataset aggregation algorithm)
- Using expert to generate dataset fo recovery
- Direct contact with experts
- But less compounding errors

Expensive algorithms as experts

→ Game playing : computationally expensive semi-optimal behavior computed with brute force search during simulation

→ rather to use the search during training to

learn a fast policy mimicking the search

→ then tested

→ discrete optimizers: computing shortest paths offline as training data and testing it

Structured prediction via Imitation Learning

- Structured prediction can be solved via imitation learning using sequence labeling
- e.g. 'Monsters eat tasty bunnies'

  noun verb adjective noun
- Policy to learn noun then verb then adjective then noun

Reference

A Course in Machine Learning by Haul Daumé III

http://ciml.info/

Human-level control through deep reinforcement learning

- 19 authors from Google DeepMind, London
- 3 equal contribution authors Volodymyr

Mnih,Koray Kavukcuoglu, David Silverwith

only one first author

- 3735 citations

Background

- RL theory gives a normative account
- Psychological and neuro-scientific perspectives on animal behavior
- Efficient representation of environment from high dimensional sensory inputs
- General past experience with new situations

Problems with reinforcement learning agents

It is restricted to domains with

- Hand-crafted useful features
- Full observed, low dimensional state spaces

Work done

- Deep Q network, a novel artificial agent
- Learns successful policies from high

dimensional sensory inputs using end to end

reinforcement learning

- Testing on Atari 2600 games
- Pixels and game scores as inputs
- Result
  - Better performance than all previous algorithms
  - Comparable level with a professional human game tester over a set of 49 games

Atari 2600

- Home video game console developed in 1977
- Use of general purpose CPUs in game console hardware
- Game code distributed through cartridges
- Arcade games like PACMAN ported to console
- Simple hardware limiting the game complexity
- Obtainable short-term progress in learning, modeling, and planning

Architecture used

- Deep convolutional neural network
- Exploits local spatial correlation present in images
- Robust to transformations such as changes of viewpoints or scales
- Approximates optimal action value function

Optimal action-value function

$$Q^*(s,a) = \max_\pi \mathbb{E}\left[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots \middle| s_t = s,\ a_t = a,\ \pi\right],$$

maximum sum of rewards r_t discounted by γ at each time-step t, obtained by a behaviour policy π = P(a|s), after an observation (s) and executing an action (a)

Reasons for instability of reinforcement learning

- Unstable or diverges when non linear function approximator represents the action value function
- Reasons:-
  - Correlations in the sequence of observations
  - Small updates to Q may change the policy and the data distribution
  - Correlations between the action values and target values

Solution: Novel variant of Q Learning

- Experience replay randomizes over the data
  - Reducing correlation in observation sequence
  - Smoothing changes in data distribution
- Iterative update of action values towards

target values that are periodically updated,

reducing correlation with the targets

Training algorithm

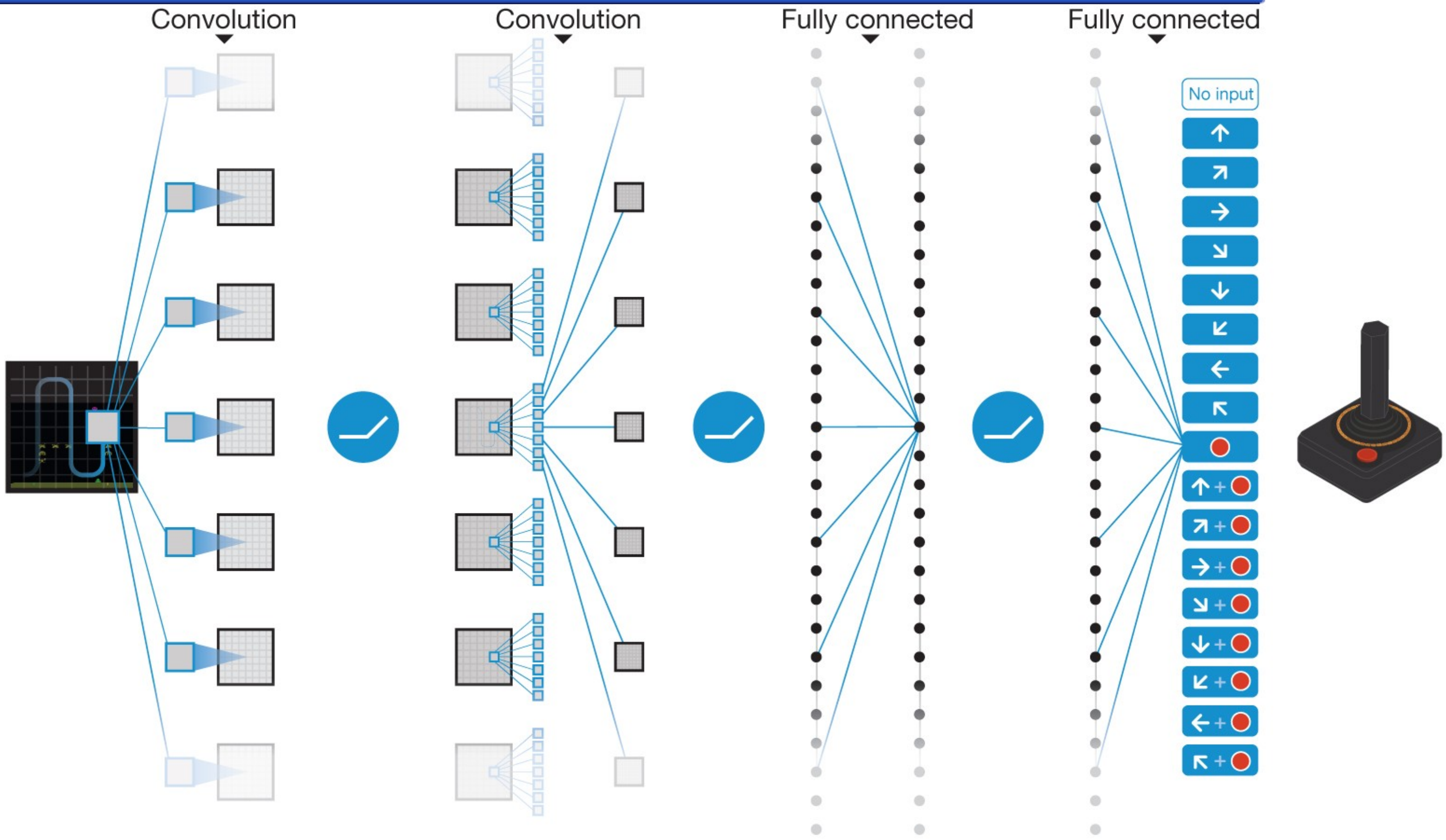
- Parameterizing an objective value function using deep neural network
- Action replay done by storing agent's experiences $e_t = (s_t, a_t, r_t, s_{t+1})$ at each time step in a dataset

Loss function

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s') \sim \mathrm{U}(D)}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta_i^-) - Q(s, a; \theta_i)\right)^2\right]$$

γ is the discount factor controlling agent's horizon, θ_i are the parameters of the Q-network at iteration i also the network parameters used to compute the target at iteration i.
Target network parameters are only updated with the Q-network parameters (θi) every C steps and fixed between each update

Schematic illustration (from Nature slides)



V Mnih *et al. Nature* **518**, 529-533 (2015) doi:10.1038/nature14236

nature

References

1. Human-level control through deep reinforcement learning by V Mnih, K Kavukcuoglu, D Silver, A Rusu, J Veness, M G Bellemare, A Graves, M Riedmiller, A Fidjeland, G Ostrovski, S Petersen, C Beattie, A Sadik, I Antonoglou, H King, D Kumaran, D Wierstra, S Legg, D Hassabis, Nature journal, Volume 545, pp. 529-533, February 2015, DOI:10.1038/nature14236

References

2. The arcade learning environment: An evaluation platform for general agents by Marc G. Bellemare, Yavar Naddaf, Joel Veness, Michael Bowling, Journal of Artificial Intelligence Research 47, pages 253-279, DOI: 10.1613/jair.3912

arXiv:1207.4708