RL Traffic Signal Control

David Sanwald

May 8, 2018

Table of contents

Introduction

Reinforcement Learning

Machine Learning

- Supervised Learning
    - Classification
    - Regression
- Unsupervised Learning
    - Clustering
    - ...
- Reinforcement Learning

Agent Environment

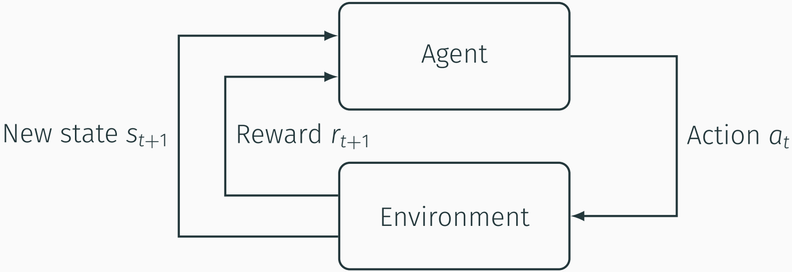


Figure 1: Agent environment interface

Markov Decission Process

MARKOV DECISSION PROCESS is defined by quatuple $\langle \mathcal{S}, \mathcal{P}, \mathcal{R}, \mathcal{A} \rangle$

- $\mathcal{S}$, a set of states
- $\mathcal{P}$, a state transition matrix defining the probabilities of some possible next state s' given any state s
  $\mathcal{P}^a_{ss'} = \mathbb{P}[S_{t+1} = s' | S_t = s, A_t = a]$
- a reward function $\mathcal{R}^a_s = \mathbb{E}[R_{t+1} | S_t = s, A_t = a]$
- $\mathcal{A}$, a set of actions

Policy

- specifies agent's behaviour
- mapping of state to action

$$\pi(s) = a$$

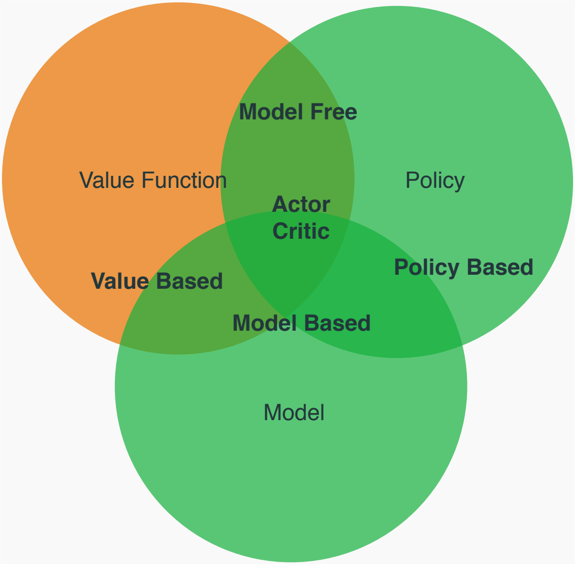
$$\mathbb{P}(a|s) = \pi(a|s)$$

Markov Property

- The future is conditionally independent of the past given the presence
- implies memorylessnes

$$\mathbb{P}[S_{t+1}|S_1, \ldots, S_t] = \mathbb{P}[S_{t+1}|S_t]$$

Taxonomy of RL



Value Function

Expected return

- from state s and action a
- given policy π

$$Q^\pi(s,a) = \mathbb{E}[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots | s, a]$$

- decomposable into

$$Q^\pi(s,a) = \mathbb{E}[r + \gamma Q^\pi(s', a') | s, a]$$

Optimal Value Function

- optimal value function

$$Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a) = Q^{\pi^*}(s,a)$$

- optimal policy

$$\pi^*(s) = \operatorname*{argmax}_{a} Q^*(s,a)$$

- decomposition into

$$Q^*(s,a) = \mathbb{E}_{s'}[r + \gamma \max_{a'} Q^*(s',a')|s,a]$$

TD Learning

Off Policy learning

Q-learning

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(S_t, A_t))]$$

On Policy learning

Sarsa

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(s_{t+1}, A_{t+1}) - Q(S_t, A_t))]$$

Demo

Function Approximation

Why Function Approximation?

- large state spaces
- slow learning
- need for generalization

Naive Function Approximation

$$Q(s, a, \theta) \approx Q(s, a)$$

$$\mathcal{L}(\theta) = \mathbb{E}\left[\left(r + \gamma \max_{a'} Q(s', a', \theta) - Q(s, a, \theta)\right)^2\right]$$

$$\frac{\partial \mathcal{L}(\theta)}{\partial \theta} = \mathbb{E}\left[\left(r + \gamma \max_{a'} Q(s', a', \theta) - Q(s, a, \theta)\right)\frac{\partial Q(s, a, \theta)}{\partial \theta}\right]$$

Deadly Triad

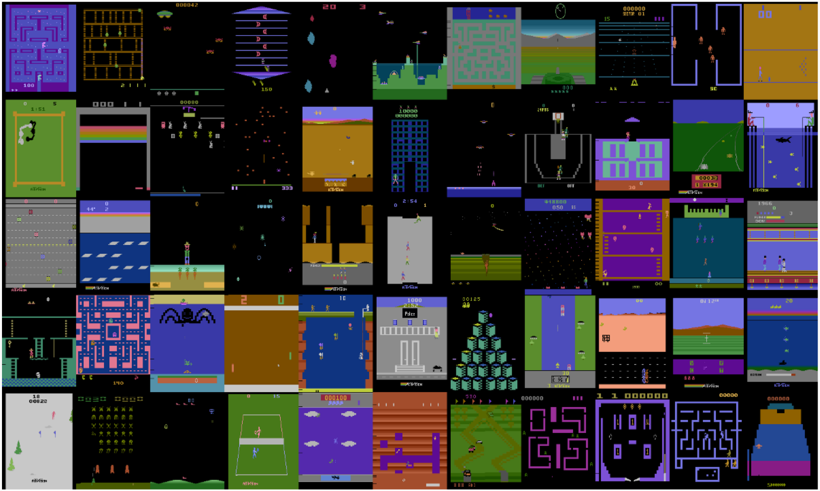
Deadly Triad

- function approximation
- off policy learning
- bootstrapping

Deadly Triad

- function approximation
- off policy learning
- bootstrapping

atari arcade games



DQN

Human-level control through deep reinforcement learning[1]

- (almost) raw pixel input
- one agent/set of network weights
- comparable to human performance on 29 of 49 games

[1] nature february 2015

DQN

experience replay

- decorrelates
- sample efficiency

target network

- inhibits loops

error clipping

- limits gradient magnitude

experience replay

- store experience $e_t = (s_t, a_t, r_t, s_{t=1})$ in $D_t = \{e_1, \ldots, e_t\}$
- at timestep t update $(s, a, r, s') \sim U(D)$

fixed target network

- separate target network $\tilde{Q}(s, a, \theta^-)$ and online network $Q(s, a, \theta)$
- TD error becomes $r + \gamma \max_{a'} Q(s', a', \theta^-) - Q(s, a, \theta)$

DQN architecture