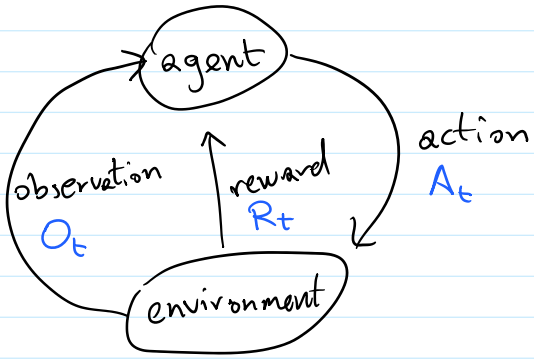
RL Basics

Monday, July 10, 2017 9:02 AM

Intro



- Markov property → the next state is only dependent on the current state, not past states

$$P(S_{t+1} \mid S_t) = P(S_{t+1} \mid S_1, \ldots, S_t)$$

- fully observable: $O_t = S_t$

- POMDPs → partially observable states

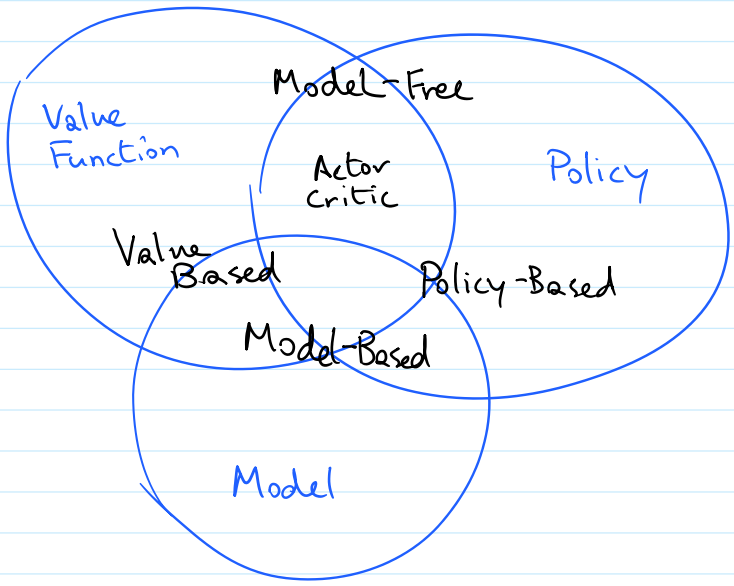
$$S_t = \sigma(S_{t-1} W_s + O_t W_s)$$

↳ use RNN

- Dynamics Model

$$M(s, a, s') = P(S_{t+1} = s' \mid S_t = s, A_t = a)$$
$$R(s, a) = E(R_{t+1} \mid S_t = s, A_t = a)$$

- RL Agent Taxonomy



- Fundamental Problems
  - Learning → learn unknown dynamics model
    → improve policy based on experience
  - Planning → given known dynamics, find optimal policy

- Exploration vs Exploitation → exploration may seem suboptimal but it might lead to finding even better states
  - Exploration → gain new info about environment
  - Exploitation → maximize reward w/ known info

- Prediction vs Control
  - Prediction → evaluate given policy
    → find true value of states
  - Control → find the best policy

- MDPs

  - gain → total reward w/ discount

$$G_t = R_{t+1} + \gamma R_{t+2} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+1+k}$$

- Markov Reward Process (MRP) → no actions
  - value function

$$V(s) = \mathbb{E}(G_t \mid S_t = s)$$

$$= \mathbb{E}(R_{t+1} + \gamma V(S_{t+1}) \mid S_t = s)$$

- Bellman Equation

$$v(s) = R_s + \gamma \sum_{s' \in S} M(s, s') v(s')$$

(prob going from s to s')

↳ matrix form

$$\bar{v} = \bar{R} + \gamma \hat{M} \bar{v}$$

$$\bar{v} = (I - \gamma \hat{M})^{-1} \bar{R} \rightarrow \text{solution}$$

- Markov Decision Process (MDP)

- policy

$$\pi(a \mid s) = P(A_t = a \mid S_t = s)$$

↳ MRP given a policy

$$M^{\pi}(s, s') = \sum_{a \in A} \pi(a \mid s) M(s, a, s')$$

$$R^{\pi}(s) = \sum_{a \in A} \pi(a \mid s) R(s, a)$$

- Q-function → action-value

$$Q_{\pi}(s, a) = E_{\pi}(G_t \mid S_t = s, A_t = a)$$

- Bellman Expectation Equation

$$V_{\pi}(s) = \sum_{a \in A} \pi(a \mid s) Q_{\pi}(s, a)$$

$$Q_{\pi}(s, a) = R(s, a) + \gamma \sum_{s' \in S} M(s, a, s') V_{\pi}(s')$$

$s' \in S$

- Bellman Optimality Equation

$$V^*(s) = \max_a Q^*(s,a)$$

$$Q^*(s,a) = R(s,a) + \sum_{s' \in S} M(s,a,s')\, V^*(s')$$

- Additional Topics → ∞ MDPs
  → POMDPs
  → Ergodic MDPs

- Dynamic Programming

- DP → the problem must have a sequential component
  → breakdown problem into subproblems & combine solutions
- this is all assuming agent knows $M(s,a,s')$ & $R(s,a)$
  ↳ how to solve for $V^*(s)$ & $Q^*(s,a)$ when S is large

- Policy Evaluation

$$V_{k+1}(s) = \sum_{a \in A} \pi(a|s)\left(R(s,a) + \gamma \sum_{s' \in S} M(s,a,s')\, V_k(s') \right)$$

$$\bar{V}^{k+1} = \bar{R}^{\pi} + \gamma \hat{M}^{\pi} \bar{V}^{k}$$

- Improving Policy → greedily

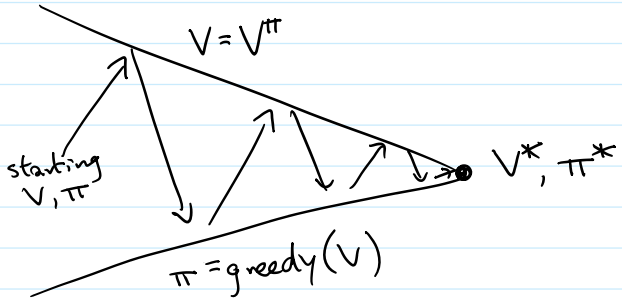
$$\pi'(s|a) = \underset{a \in A}{\operatorname{argmax}} \left[R(s,a) + \gamma \sum_{s' \in S} M(s,a,s')\, V_{\pi}(s') \right]$$

- Policy Iteration

- algorithm: initialize π (arbitrarily)
  repeat until convergence ($\pi = \pi'$)
  estimate V^{π} by evaluating policy

improve policy $\pi' = \text{greedy}(V^\pi)$

$\pi' \leftarrow \pi$



- Value Iteration → skip policy eval

$$V_{k+1}(s) = \max_{a \in A} \left[R(s,a) + \gamma \sum_{s' \in S} M(s,a,s') V_k(s') \right]$$

- DP Algorithms

Problem	Bellman Eqn.	Algorithm
Prediction	Expectation	Policy Eval
Control	Expectation + Greedy Improvement	Policy Iteration
Control	Optimality	Value Iteration

- Async DP
  - in-place DP → analogue to SGD
  - prioritized sweeping → update based on Bellman error
  - real-time DP → update only those states agent visits
- Problem Size → only works up to $|S| \sim 10^6$
  - For larger problems use sample backups instead full-backups

Prediction

- Monte Carlo → update V(s) after episode

$$V(S_t) \leftarrow V(S_t) + \alpha\left(G_t - V(S_t)\right)$$

$$G_t = \sum_{episode} R_t$$

- Temporal Difference → adjust $V(s)$ after each action → bootstrapping

$$V(S_t) \leftarrow V(S_t) + \alpha\left(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right)$$

- MC vs TD

TD	MC
bootstraps	learns at the end of an episode
online or episodic	episodic only
learn from incomplete sequences	high variance ← change depends on many random actions
works in non-terminating systems	no bias ← starting state doesn't matter
low variance	converges to min mean squared error of $\sum\sum \left(G_t^k - V(S_t^k)\right)^2$
some bias	
converges to Markov model solution	
uses Markov property	

- TD(λ) → generalizing MC & TD

- use eligibility function to rank how much you can learn about other states based on your current experience
  - ↳ using both a frequency & recency heuristic

$$E_0(s) \leftarrow 0$$

$$E_{t+1}(s) \leftarrow \gamma\lambda E_t(s) + 1(S_t = s)$$

- now adjust all states each step according to eligibility

$$\delta_t = R_t + \gamma V(S_{t+1}) - V(S_t)$$ ← TD error

$$V(s) \leftarrow V(s) + \alpha \delta_t E_t(s)$$

- MC is equivalent to TD(1)
- TD is equivalent to TD(0)

Control

- now learn best policy
  - ↳ naive approach

evaluate π (using prediction)
↑
improve π (greedily)

- greedy policy improvement → given dynamics & value function

$$\pi'(s) = \underset{a \in A}{\text{argmax}} \; R(s,a) + M(s,a,s') V(s')$$

  - ↳ model-free

$$\pi'(s) = \underset{a \in A}{\text{argmax}} \; Q(s,a)$$

- Exploration vs Exploitation → ε-greedy action selection
  - w/ prob. $1-\varepsilon$ choose greedy action
  - w/ prob. ε choose random action
- Monte-Carlo Control → policy iteration
  - policy eval → MC $Q \leftarrow Q^\pi$
  - policy improvement → ε-Greedy

- gradually ↓ ε in ∞ limit → GLIE

↳ in k^{th} episode w/ π

↳ for each S_t & A_t in episode

$$N(S_t, A_t) \leftarrow N(S_t, A_t) + 1$$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{N(S_t, A_t)}\left(G_t - Q(S_t, A_t)\right)$$

$$\varepsilon \leftarrow \frac{1}{k}$$

$$\pi(s) \leftarrow \underset{a \in A}{\operatorname{argmax}}\ Q(s, a)$$

- SARSA → TD(0) control
  → on policy → choose future actions using π

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Choose A from S using policy derived from Q (e.g., ε-greedy)
  Repeat (for each step of episode):
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ε-greedy)
    $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma Q(S',A') - Q(S,A)]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

- SARSA(λ) → analogue of TD(λ)
  → eligibility list to learn in advance

Initialize $Q(s,a)$ arbitrarily, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$
Repeat (for each episode):
  $E(s,a) = 0$, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$
  Initialize S, A
  Repeat (for each step of episode):
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ε-greedy)
    $\delta \leftarrow R + \gamma Q(S',A') - Q(S,A)$
    $E(S,A) \leftarrow E(S,A) + 1$
    For all $s \in \mathcal{S}, a \in \mathcal{A}(s)$:
      $Q(s,a) \leftarrow Q(s,a) + \alpha\delta E(s,a)$
      $E(s,a) \leftarrow \gamma\lambda E(s,a)$
    $S \leftarrow S'$; $A \leftarrow A'$
  until S is terminal

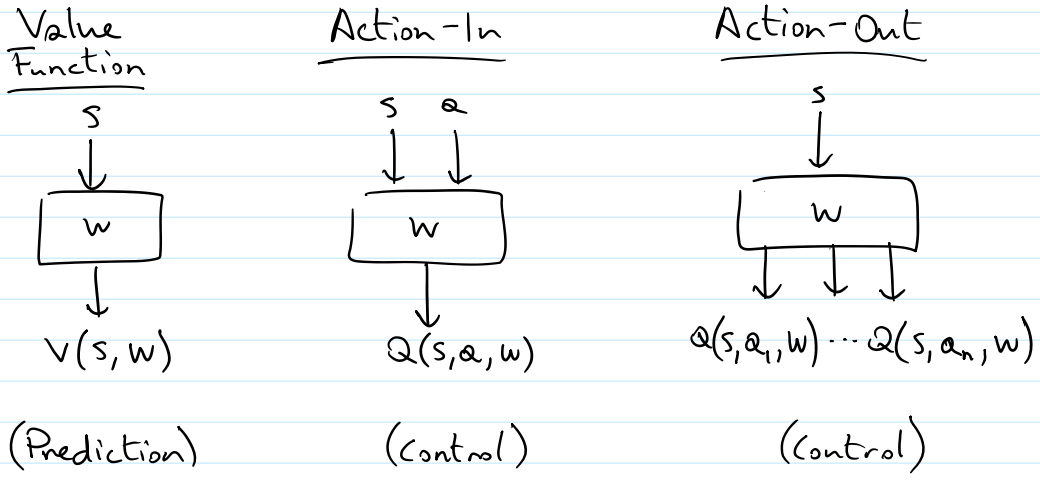
- Q-Learning → on or off policy

$$Q(S,A) \leftarrow Q(S,A) + \alpha\left(R + \max_{a'} \gamma Q(S', a') - Q(S,A)\right)$$

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma \max_a Q(S',a) - Q(S,A)\right]$
    $S \leftarrow S'$;
  until S is terminal

- Value Function Approximation
  - for problems w/ too many states for table look-up → eg. continuous

- Types



- Types of Approximators
  - linear combinations of features
  - neural network
  - decision trees
  - nearest neighbor
  - fourier/wavelet bases

} differentiable, non-stationary, non-iid

- Linear Function Approximation

$$v(s, \bar{w}) = \bar{x}(s) \cdot \bar{w}$$

- Loss

$$L(\bar{w}) = E_\pi\left[\left(v_\pi(s) - \bar{x}(s) \cdot \bar{w}\right)^2\right]$$

($v_\pi(s)$: true value)

- update rule

$$\nabla_w v(S, w) = \bar{x}(S)$$
$$\Delta \bar{w} = \alpha \left(V_\pi(S) - v(S, \bar{w}) \right) \bar{x}(S)$$

- incremental method → approximate V_π by experience

$$\Delta \bar{w} = \alpha \left(R_{t+1} + \gamma v(S_{t+1}, \bar{w}) - v(S_t, \bar{w}) \right) \bar{\nabla}_w v(S, \bar{w})$$

- same idea w/ Q function

- Convergence Issues

- prediction

gradient TD follows true ∇ of Bellman eqn.

On/Off-Policy	Algorithm	Table Lookup	Linear	Non-Linear
On-Policy	MC	✓	✓	✓
	TD	✓	✓	✗
	Gradient TD	✓	✓	✓
Off-Policy	MC	✓	✓	✓
	TD	✓	✗	✗
	Gradient TD	✓	✓	✓

- control

Algorithm	Table Lookup	Linear	Non-Linear
Monte-Carlo Control	✓	(✓)	✗
Sarsa	✓	(✓)	✗
Q-learning	✓	✗	✗
Gradient Q-learning	✓	✓	✗

(✓) = chatters around near-optimal value function

- Experience Replay DQN → batch method → reuse training data

- take action → ε-greedy
- store transition in replay memory $(s_t, a_t, r_{t+1}, s_{t+1})$
- sample random mini batches from memory
- use 2 networks → fix one to ↑ stability
  → train the other w/ Q values from fixed network
- optimize MSL loss w/ SGD

— fixed

- optimize MSL loss w/ SGD

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

fixed

- Linear Analytical Solution

$$w = \left(\bar{x}(s_i)\, \bar{x}(s_j)\right)^{-1} \bar{x}(s_i)\, v_i^\pi$$

↳ don't know v_t^π → MC → $v_t^\pi \approx G_i$

→ TD(0) → $v_t^\pi \approx R_{t+1} + \gamma V(S_{t+1}, w)$

→ TD(λ) → G_t^λ

- convergence

On/Off-Policy	Algorithm	Table Lookup	Linear	Non-Linear
On-Policy	MC	✓	✓	✓
	LSMC	✓	✓	-
	TD	✓	✓	✗
	LSTD	✓	✓	-
Off-Policy	MC	✓	✓	✓
	LSMC	✓	✓	-
	TD	✓	✗	✗
	LSTD	✓	✓	-

- LSTDQ → least squares q-learning

LSTDQ algorithm: solve for total update = zero

$$0 = \sum_{t=1}^{T} \alpha(R_{t+1} + \gamma \hat{q}(S_{t+1}, \pi(S_{t+1}), \mathbf{w}) - \hat{q}(S_t, A_t, \mathbf{w}))\mathbf{x}(S_t, A_t)$$

$$\mathbf{w} = \left(\sum_{t=1}^{T} \mathbf{x}(S_t, A_t)(\mathbf{x}(S_t, A_t) - \gamma \mathbf{x}(S_{t+1}, \pi(S_{t+1})))^\top\right)^{-1} \sum_{t=1}^{T} \mathbf{x}(S_t, A_t) R_{t+1}$$

function **LSPI-TD**$(\mathcal{D}, \pi_0)$
  $\pi' \leftarrow \pi_0$
  repeat
    $\pi \leftarrow \pi'$
    $Q \leftarrow$ **LSTDQ**$(\pi, \mathcal{D})$
    for all $s \in \mathcal{S}$ **do**
      $\pi'(s) \leftarrow \underset{a \in \mathcal{A}}{\text{argmax}}\ Q(s, a)$
    end for
  until $(\pi \approx \pi')$
  return π
end function