An Overview of Reinforcement Learning

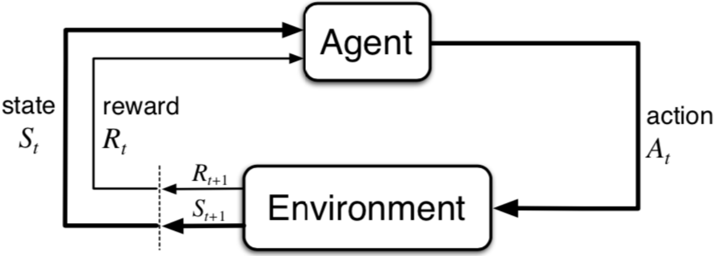
Yue Zhao

November 27, 2019

Content

1. Fundamental Knowledge
2. Value-based RL
   - Monte Carlo
     - MCTS
     - *AlphaGo Zero
   - Time Difference
     - DQN
3. Policy-based RL
   - Policy Gradient
   - Trust Region Policy Optimization

1.1. Framework



1.2. Definitions

- Markov Decision Process: $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$
- Policy π:
  - $\pi(a|s) = \mathbb{P}\left[A_t = a | S_t = s\right]$
- Return G_t:
  - $G_t = R_{t+1} + \gamma R_{t+2} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$
- State-value function $v_\pi(s)$:
  - $v_\pi(s) = \mathbb{E}_\pi\left[G_t | S_t = s\right]$
- Action-value function $q_\pi(s, a)$:
  - $q_\pi(s, a) = \mathbb{E}_\pi\left[G_t | S_t = s, A_t = a\right]$

1.3. Properties

- $v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) q_\pi(s, a)$
- $q_\pi(s, a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi\left(s'\right)$
- $v_\pi(s) = \mathbb{E}_\pi\left[R_{t+1} + \gamma v_\pi\left(S_{t+1}\right) | S_t = s\right]$
- $q_\pi(s, a) = \mathbb{E}_\pi\left[R_{t+1} + \gamma q_\pi\left(S_{t+1}, A_{t+1}\right) | S_t = s, A_t = a\right]$

1.4. Categorizing

- Value-based or Policy-based or AC
- On-policy or Off-policy
- Model-based or Model-free

2. Value-based

$$v_\pi(s) = E_\pi\left[G_t | S_t = s\right] = E_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s\right]$$

$$q_\pi(s, a) = E_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a\right]$$

2.1. Monte Carlo

- $v(s) = \frac{G_{11}(s)+G_{21}(s)+\cdots}{N(s)}$
- $v(s) = \frac{G_{11}(s)+G_{12}(s)+\cdots+G_{21}(s)+\cdots}{N(s)}$

2.1. Monte Carlo

On-policy first-visit MC control (for ε-soft policies), estimates $\pi \approx \pi_*$

Algorithm parameter: small $\varepsilon > 0$

Initialize:

  $\pi \leftarrow$ an arbitrary ε-soft policy

  $Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

  $Returns(s,a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Repeat forever (for each episode):

  Generate an episode following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

  $G \leftarrow 0$

  Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

    $G \leftarrow G + R_{t+1}$

    Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:

      Append G to $Returns(S_t, A_t)$

      $Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

evaluation

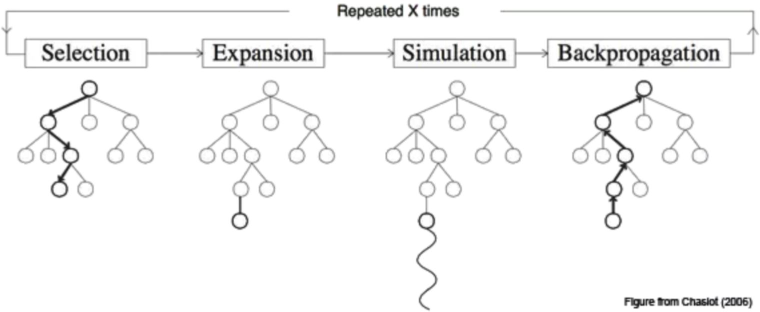
      $A^* \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken arbitrarily)

      For all $a \in \mathcal{A}(S_t)$:

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(S_t)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(S_t)| & \text{if } a \neq A^* \end{cases}$$

improvement

2.1.2. MCTS



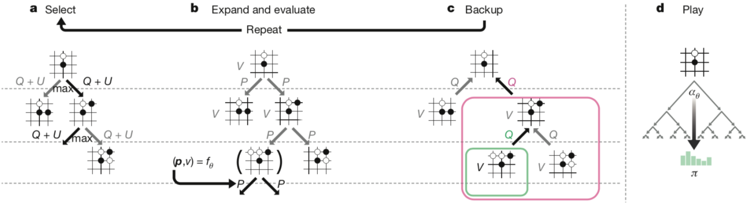
Repeated X times
Selection
Expansion
Simulation
Backpropagation
Figure from Chaslot (2006)

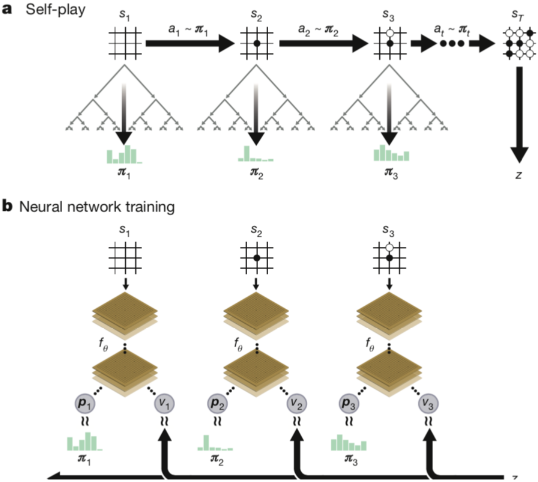
- **NN:**

$$(\boldsymbol{p}, v) = f_\theta(s)$$

- **Loss Function:**

$$l = (z - v)^2 - \boldsymbol{\pi}^T log(\boldsymbol{p}) + c||\theta||^2$$





a Self-play
s1
a1 ~ π1
s2
a2 ~ π2
s3
at ~ πt
sT
π1
π2
π3
z
b Neural network training
s1
s2
s3
fθ
fθ
fθ
p1
v1
p2
v2
p3
v3
π1
π2
π3
z

2.2. Time Difference

- MC: $V(S_t) \leftarrow V(S_t) + \alpha (G_t - V(S_t))$
- TD: $V(S_t) \leftarrow V(S_t) + \alpha (R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$

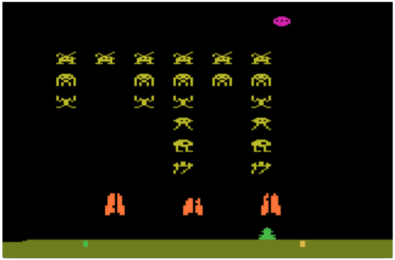
2.2. Q Learning

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

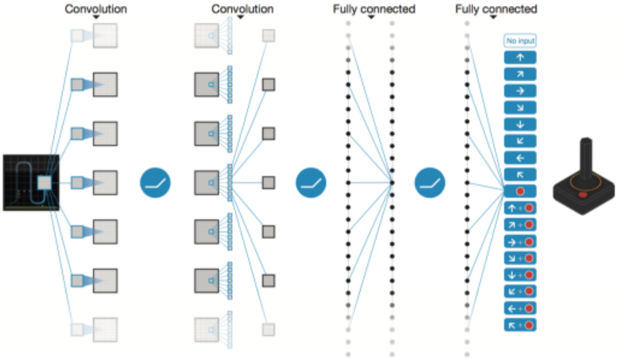
Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
    Initialize S
    Loop for each step of episode:
        Choose A from S using policy derived from Q (e.g., ϵ-greedy)
        Take action A, observe R, S'
        $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
        $S \leftarrow S'$
    until S is terminal

- Raw Pixel Input
- NN: $Q(s, a) \approx f(s, a, w)$



2.2. DQN



Convolution
Convolution
Fully connected
Fully connected
No input

2.2. DQN

- **Contributions:**
  - Raw Pixel Input
  - Experience Replay

2.2. DQN

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}_{s,a\sim\rho(\cdot);s'\sim\mathcal{E}}\left[\left(r + \gamma \max_{a'} Q(s',a';\theta_{i-1}) - Q(s,a;\theta_i)\right)\nabla_{\theta_i} Q(s,a;\theta_i)\right]$$

Algorithm 1 Deep Q-learning with Experience Replay

```
Initialize replay memory D to capacity N
Initialize action-value function Q with random weights
for episode = 1, M do
    Initialise sequence s_1 = {x_1} and preprocessed sequenced φ_1 = φ(s_1)
    for t = 1, T do
        With probability ε select a random action a_t
        otherwise select a_t = max_a Q*(φ(s_t), a; θ)
        Execute action a_t in emulator and observe reward r_t and image x_{t+1}
        Set s_{t+1} = s_t, a_t, x_{t+1} and preprocess φ_{t+1} = φ(s_{t+1})
        Store transition (φ_t, a_t, r_t, φ_{t+1}) in D
        Sample random minibatch of transitions (φ_j, a_j, r_j, φ_{j+1}) from D
        Set y_j = { r_j                                   for terminal φ_{j+1}
                  { r_j + γ max_a' Q(φ_{j+1}, a'; θ)      for non-terminal φ_{j+1}
        Perform a gradient descent step on (y_j − Q(φ_j, a_j; θ))^2 according to equation 3
    end for
end for
```

Improvements since Nature DQN

- Double DQN: Remove upward bias caused by $\max_a Q(s, a, \mathbf{w})$
  - Current Q-network $\mathbf{w}$ is used to select actions
  - Older Q-network $\mathbf{w}^-$ is used to evaluate actions

$$l = \left(r + \gamma Q(s', \operatorname*{argmax}_{a'} Q(s', a', \mathbf{w}), \mathbf{w}^-) - Q(s, a, \mathbf{w}) \right)^2$$

- Prioritised replay: Weight experience according to surprise
  - Store experience in priority queue according to DQN error

$$\left| r + \gamma \max_{a'} Q(s', a', \mathbf{w}^-) - Q(s, a, w) \right|$$

- Duelling network: Split Q-network into two channels
  - Action-independent value function $V(s, v)$
  - Action-dependent advantage function $A(s, a, \mathbf{w})$

$$Q(s, a) = V(s, v) + A(s, a, \mathbf{w})$$

- **Core Concept:** $\pi(a|s, \boldsymbol{\theta}) = \Pr\{A_t = a | S_t = s, \boldsymbol{\theta}_t = \boldsymbol{\theta}\}$

- **Target:** $\max J(\boldsymbol{\theta})$
- **Update:** $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \widehat{\nabla J(\boldsymbol{\theta}_t})$

Policy Gradient Theorem

For any MDP, in either episodic cases or continuing cases,

$$\nabla J(\boldsymbol{\theta}) \propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla_{\boldsymbol{\theta}} \pi(a|s,\boldsymbol{\theta})$$

Proof.

3.1. Policy Gradient

function REINFORCE
    Initialise θ arbitrarily
    for each episode $\{s_1, a_1, r_2, ..., s_{T-1}, a_{T-1}, r_T\} \sim \pi_\theta$ **do**
        for $t = 1$ to $T - 1$ **do**
            $\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(s_t, a_t) v_t$
        end for
    end for
    return θ
end function

3.1. Policy Gradient

REINFORCE with Baseline

$$\nabla J(\boldsymbol{\theta}) \propto \sum_s \mu(s) \sum_a \left(q_\pi(s,a) - b(s)\right) \nabla_{\boldsymbol{\theta}} \pi(a|s,\boldsymbol{\theta})$$

- **Update:** $\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha \left(G_t - b\left(S_t\right)\right) \frac{\nabla_{\boldsymbol{\theta}} \pi(A_t|S_t,\boldsymbol{\theta}_t)}{\pi(A_t|S_t,\boldsymbol{\theta}_t)}$

- **Key Point:** Monotonic Improvement Guarantee.

3.2. Trust Region Policy Optimization

Lemma 1. *Given two policies $\pi, \tilde{\pi}$,*

$$\eta(\tilde{\pi}) = \eta(\pi) + \mathbb{E}_{\tau \sim \tilde{\pi}} \left[\sum_{t=0}^{\infty} \gamma^t A_\pi(s_t, a_t) \right] \tag{19}$$

This expectation is taken over trajectories $\tau := (s_0, a_0, s_1, a_0, \dots)$, and the notation $\mathbb{E}_{\tau \sim \tilde{\pi}}[\dots]$ indicates that actions are sampled from $\tilde{\pi}$ to generate τ.

- $$\eta(\widetilde{\pi}) = \eta(\pi) + \sum_s \rho_{\tilde{\pi}}(s) \sum_a \tilde{\pi}(a|s) A^\pi(s, a)$$

- $$L_\pi(\widetilde{\pi}) = \eta(\pi) + E_{s-\rho_{old}, a-\pi_{old}} \left[\frac{\tilde{\pi}_\theta(a|s)}{\pi_{\theta_{old}}(a|s)} A_{\theta_{old}}(s, a) \right]$$

3.2. Trust Region Policy Optimization

Theorem 1. *Let* $\alpha = D_{\mathrm{TV}}^{\max}(\pi_{\mathrm{old}}, \pi_{\mathrm{new}})$. *Then the following bound holds:*

$$\eta(\pi_{\mathrm{new}}) \geq L_{\pi_{\mathrm{old}}}(\pi_{\mathrm{new}}) - \frac{4\epsilon\gamma}{(1-\gamma)^2}\alpha^2$$
$$\textit{where } \epsilon = \max_{s,a} |A_\pi(s,a)| \tag{8}$$

3.2. Trust Region Policy Optimization

- $\underset{\theta}{\text{maximize}}\ L_{\theta_{\text{old}}}(\theta)$
  subject to $D_{\mathrm{KL}}^{\max}(\theta_{\text{old}}, \theta) \leq \delta$
- $\underset{\theta}{\text{maximize}}\ L_{\theta_{\text{old}}}(\theta)$
  subject to $\bar{D}_{\mathrm{KL}}^{\rho_{\text{old}}}(\theta_{\text{old}}, \theta) \leq \delta$
- $\underset{\theta}{\text{maximize}}\ \mathbb{E}_{s\sim\rho_{\theta_{\mathrm{old}}}, a\sim q}\left[\frac{\pi_\theta(a|s)}{q(a|s)} Q_{\theta_{\mathrm{old}}}(s,a)\right]$
  subject to $\mathbb{E}_{s\sim\rho_{\theta_{\mathrm{old}}}}\left[D_{\mathrm{KL}}\left(\pi_{\theta_{\mathrm{old}}}(\cdot|s) \| \pi_\theta(\cdot|s)\right)\right] \leq \delta$

3.2. Trust Region Policy Optimization

$$\underset{\theta}{\text{maximize}} \left[\nabla_\theta L_{\theta_{\text{old}}}(\theta) \big|_{\theta=\theta_{\text{old}}} \cdot (\theta - \theta_{\text{old}}) \right] \tag{17}$$

$$\text{subject to } \frac{1}{2}(\theta_{\text{old}} - \theta)^T A(\theta_{\text{old}})(\theta_{\text{old}} - \theta) \leq \delta,$$

$$\text{where } A(\theta_{\text{old}})_{ij} =$$

$$\frac{\partial}{\partial \theta_i} \frac{\partial}{\partial \theta_j} \mathbb{E}_{s \sim \rho_\pi} \left[D_{\text{KL}}(\pi(\cdot|s, \theta_{\text{old}}) \parallel \pi(\cdot|s, \theta)) \right] \big|_{\theta=\theta_{\text{old}}}.$$

The update is $\theta_{\text{new}} = \theta_{\text{old}} + \frac{1}{\lambda} A(\theta_{\text{old}})^{-1} \nabla_\theta L(\theta) \big|_{\theta=\theta_{\text{old}}}$,

3.2. Trust Region Policy Optimization

	B. Rider	*Breakout*	*Enduro*	*Pong*	*Q*bert*	*Seaquest*	*S. Invaders*
Random	354	1.2	0	-20.4	157	110	179
Human (Mnih et al., 2013)	7456	31.0	368	-3.0	18900	28010	3690
Deep Q Learning (Mnih et al., 2013)	4092	168.0	470	20.0	1952	1705	581
UCC-I (Guo et al., 2014)	5702	380	741	21	20025	2995	692
TRPO - single path	1425.2	10.8	534.6	20.9	1973.5	1908.6	568.4
TRPO - vine	859.5	34.2	430.8	20.9	7732.5	788.4	450.2