Deep Reinforcement Learning by S. Levine

CS285 - UC Berkeley

Filippo GIRUZZI

January 21, 2020

Contents

1 Introduction

2 Supervised Learning of behaviors

2.1 Goal

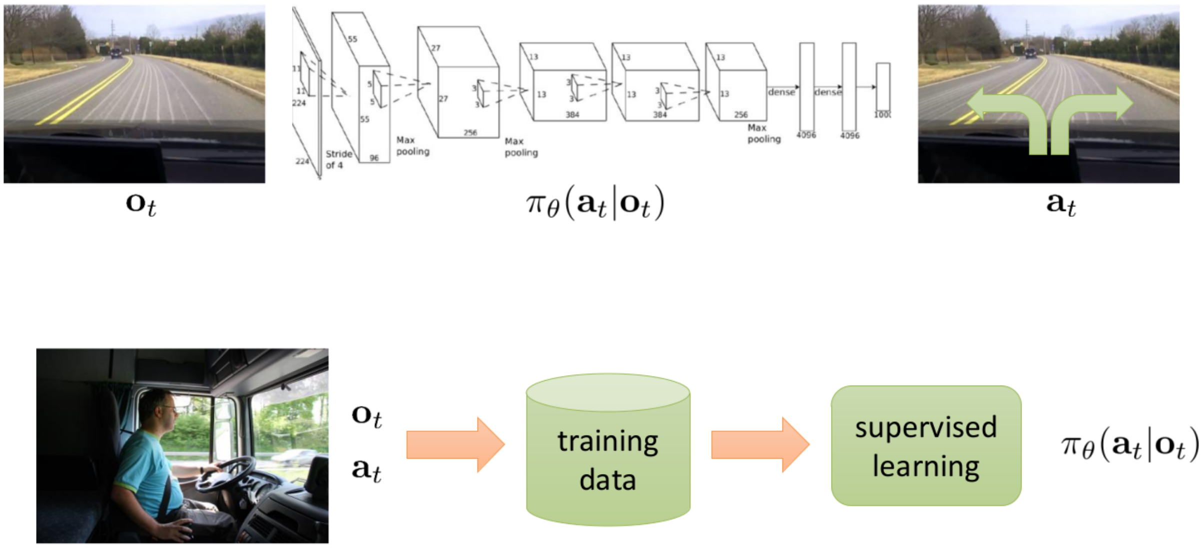

Figure 1: Imitation Learning objective

2.2 Algorithms

2.2.1 DAgger: Dataset Aggregation

Algorithm 1: DAgger

Loop

1. Train $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ from human data $\mathcal{D} = \{\mathbf{o}_1, \mathbf{a}_1, \dots, \mathbf{o}_N, \mathbf{a}_N\}$

2. Run $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ to get dataset $\mathcal{D}_\pi = \{\mathbf{o}_1, \dots, \mathbf{o}_M\}$

3. Ask human to label $\mathcal{D}_\pi$ with actions $\mathbf{a}_t$

4. Aggregate $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_\pi$

end

2.3 Tips & hacks

1. **Distributional shift:**
   - Use DAgger
   - Randomize input
2. **Non-Markovian behavior:** use RNN to model $\pi_\theta(\mathbf{a}_t|\mathbf{o}_1, \dots, \mathbf{o}_t)$
3. **Multi-modal behavior:**
   - Output a mixture of Gaussians
   - Use latent variable models
   - Use autoregressive discretization

3 Introduction to Reinforcement Learning

3.1 Goal

Definition 1 (POMDP: Partially Observed Markov Process)

$$\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \mathcal{E}, r\} \tag{1}$$

Definition 2 (Reinforcement Learning objective)

$$\max_{\theta} E_{(\mathbf{s},\mathbf{a})\sim p_{\theta}(\mathbf{s},\mathbf{a})} \left[r(\mathbf{s},\mathbf{a})\right] \tag{2}$$

$$\max_{\theta} \sum_{t=1}^{T} E_{(\mathbf{s}_t,\mathbf{a}_t)\sim p_{\theta}(\mathbf{s}_t,\mathbf{a}_t)} \left[r(\mathbf{s}_t,\mathbf{a}_t)\right] = \max_{\theta} E_{\tau\sim p_{\theta}(\tau)} \left[\sum_{t} r(\mathbf{s}_t,\mathbf{a}_t)\right] \tag{3}$$

Definition 3 (Q-function)

$$Q^{\pi}(\mathbf{s}_t,\mathbf{a}_t) = \sum_{t'=t}^{T} E_{\pi_{\theta}} \left[r(\mathbf{s}'_t,\mathbf{a}'_t)|\mathbf{s}_t,\mathbf{a}_t\right] \tag{4}$$

Definition 4 (Value function)

$$V^{\pi}(\mathbf{s}_t) = \sum_{t'=t}^{T} E_{\pi_{\theta}} \left[r(\mathbf{s}'_t,\mathbf{a}'_t)|\mathbf{s}_t\right] = E_{\mathbf{a}_t\sim\pi(\mathbf{a}_t|\mathbf{s}_t} \left[Q^{\pi}(\mathbf{s}_t,\mathbf{a}_t)\right] \tag{5}$$

$E_{\mathbf{s}_1\sim p(\mathbf{s}_1)} \left[V^{\pi}(\mathbf{s}_1)\right]$ *is the Reinforcement Learning objective.*

3.2 Algorithms

3.2.1 Global structure

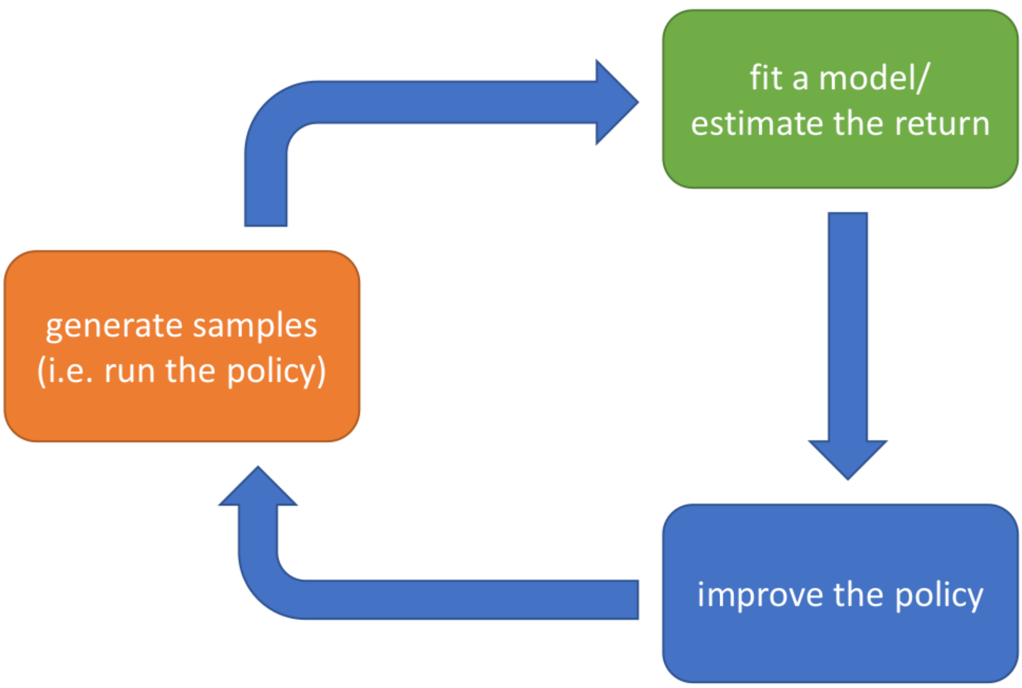

Figure 2: Global RL algorithm structure

3.2.2 Exemples

1. **Policy Gradients:** directly differentiate the RL objective
   - REINFORCE
   - Natural Policy Gradient
   - Trust region policy optimization
2. **Value-based:** estimate the Value function or the Q-function of the optimal policy
   - Q-learning, DQN
   - Temporal difference learning
   - Fitted Value iteration
3. **Actor-critic:** estimate the Value function or the Q-function of the current policy, and use it to improve the policy
   - A3C: Asynchronous advantage Actor-critic
   - SAC: Soft Actor-critic
4. **Model-based:** estimate a transition model, and use it for planning, to improve a policy or else
   - DYNA
   - Guided policy search

4 Policy Gradients

4.1 Algorithms

4.1.1 REINFORCE

Algorithm 2: REINFORCE

Loop

1. Run the policy and sample $\{\tau^i\}$ from $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$

2. $\nabla_\theta J(\theta) \approx \dfrac{1}{N}\displaystyle\sum_{i=1}^{N}\left(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(\mathbf{a}_t^i|\mathbf{s}_t^i)\right)\left(\sum_{t=1}^{T} r(\mathbf{s}_t^i, \mathbf{a}_t^i)\right)$

3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

end

4.2 Tips & hacks

1. **Variance reduction:**
   - Causality & reward to-go: $\nabla_\theta J(\theta) \approx \dfrac{1}{N}\displaystyle\sum_{i=1}^{N}\left(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(\mathbf{a}_t^i|\mathbf{s}_t^i)\right)\hat{Q}_t^i$ where the reward to-go is $\hat{Q}_t^i = \displaystyle\sum_{t'=t}^{T} r(\mathbf{s}_{t'}^i, \mathbf{a}_{t'}^i)$

- Use baselines: $\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N} \nabla_\theta \log \pi_\theta(\tau)(r(\tau) - b)$ where the baseline can be $b = \frac{1}{N}\sum_{i=1}^{N} r(\tau)$

2. **Off-policy Policy Gradient:** use importance sampling

3. **Discount factor:** $\hat{Q}_t^i = \sum_{t'=t}^{T} \gamma^{t'-t} r(\mathbf{s}_{t'}^i, \mathbf{a}_{t'}^i)$

5 Actor-Critic algorithms

5.1 Algorithms

5.1.1 Batch Actor-Critic

Algorithm 3: Batch Actor-Critic

Loop

1. Run the policy and sample $\{\mathbf{s}_i, \mathbf{a}_i\}$ from $\pi_\theta(\mathbf{a}|\mathbf{s})$

2. Fit $\hat{V}_\phi^\pi(\mathbf{s})$ to $y_t^i = \sum_{t'=t}^{T} \gamma^{t'-t} r(\mathbf{s}_{t'}^i, \mathbf{a}_{t'}^i) = r(\mathbf{s}_t^i, \mathbf{a}_t^i) + \gamma \hat{V}_\phi^\pi(\mathbf{s}_{t+1}^i)$

3. Evaluate $\hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i) = r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \hat{V}_\phi^\pi(\mathbf{s}_i') - \hat{V}_\phi^\pi(\mathbf{s}_i)$

4. $\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N} \nabla_\theta \log \pi_\theta(\mathbf{a}_i|\mathbf{s}_i) \hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i)$

5. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

end

5.1.2 Online Actor-Critic

Algorithm 4: Online Actor-Critic

Loop

1. Take an action $\mathbf{a} \sim \pi_\theta(\mathbf{a}|\mathbf{s})$ and get $(\mathbf{s}, \mathbf{a}, \mathbf{s}', r)$

2. Update $\hat{V}_\phi^\pi$ using target $r + \gamma \hat{V}_\phi^\pi(\mathbf{s}')$

3. Evaluate $\hat{A}^\pi(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma \hat{V}_\phi^\pi(\mathbf{s}') - \hat{V}_\phi^\pi(\mathbf{s})$

4. $\nabla_\theta J(\theta) \approx \nabla_\theta \log \pi_\theta(\mathbf{a}|\mathbf{s}) \hat{A}^\pi(\mathbf{s}, \mathbf{a})$

5. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

end

5.2 Tips & hacks

1. **Architecture design:** use two networks or a single MTL network to fit $\pi_\theta(\mathbf{a}|\mathbf{s})$ and $\hat{V}_\phi^\pi(\mathbf{s})$

2. **Parallel workers:** use parallel workers to batch steps **2.** and **4.** of the Online algorithm, either synchronized or asynchronous

3. **n-step returns:** $\hat{A}^{\pi}_{n}(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^{t+n} \gamma^{t'-t} r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) + \gamma^n \hat{V}^{\pi}_{\phi}(\mathbf{s}_{t+n}) - \hat{V}^{\pi}_{\phi}(\mathbf{s}_t)$ with $n > 1$

4. **Generalized Advantage Estimation:** $\hat{A}^{\pi}_{GAE}(\mathbf{s}_t, \mathbf{a}_t) = \sum_{n=1}^{\infty} w_n \hat{A}^{\pi}_{n}(\mathbf{s}_t, \mathbf{a}_t)$ with $w_n \propto \lambda^{n-1}$ such that $\hat{A}^{\pi}_{GAE}(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^{\infty} (\gamma\lambda)^{t'-t} \delta_{t'}$ where $\delta_{t'} = r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) + \gamma \hat{V}^{\pi}_{\phi}(\mathbf{s}_{t'+1}) - \hat{V}^{\pi}_{\phi}(\mathbf{s}_{t'})$

6 Value function methods

6.1 Algorithms

6.1.1 Policy iteration

Algorithm 5: Policy iteration

Loop

1. Evaluate $A^{\pi}(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma E\left[V^{\pi}(\mathbf{s}')\right] - V^{\pi}(\mathbf{s})$

2. $\pi \leftarrow \pi'$ where $\pi'(\mathbf{a}_t|\mathbf{s}_t) = \begin{cases} 1 \text{ if } \mathbf{a}_t = \text{argmax}_{\mathbf{a}_t} A^{\pi}(\mathbf{s}_t, \mathbf{a}_t) \\ 0 \text{ otherwise} \end{cases}$

end

6.1.2 Policy iteration with Dynamic programming

Algorithm 6: Policy iteration with Dynamic programming

Loop

1. Evaluate $V^{\pi}(\mathbf{s}) = r(\mathbf{s}, \pi(\mathbf{s})) + \gamma E_{\mathbf{s}' \sim p(\mathbf{s}'|\mathbf{s}, \pi(\mathbf{s}))}\left[V^{\pi}(\mathbf{s}')\right]$

2. $\pi \leftarrow \pi'$ where $\pi'(\mathbf{a}_t|\mathbf{s}_t) = \begin{cases} 1 \text{ if } \mathbf{a}_t = \text{argmax}_{\mathbf{a}_t} A^{\pi}(\mathbf{s}_t, \mathbf{a}_t) \\ 0 \text{ otherwise} \end{cases}$

end

6.1.3 Value iteration

Algorithm 7: Value iteration

Loop

1. Set $Q^{\pi}(\mathbf{s}, \mathbf{a}) \leftarrow r(\mathbf{s}, \mathbf{a}) + \gamma E_{\mathbf{s}' \sim p(\mathbf{s}'|\mathbf{s}, \mathbf{a})}\left[V^{\pi}(\mathbf{s}')\right]$

2. Set $V^{\pi}(\mathbf{s}) \leftarrow \max_{\mathbf{a}} Q^{\pi}(\mathbf{s}, \mathbf{a})$

end

6.1.4 Fitted Value iteration

Algorithm 8: Fitted Value iteration

Loop
- **1.** Collect dataset $\{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)\}$ using some policy
- **Loop**
  - **2.** Set $y_i \leftarrow \max_{\mathbf{a}_i}(r(\mathbf{s}_i, \mathbf{a}_i) + \gamma E\left[V_\phi(\mathbf{s}'_i)\right])$
  - **3.** Set $\phi \leftarrow \operatorname{argmin}_\phi \frac{1}{2}\sum_i ||V_\phi(\mathbf{s}_i) - y_i||^2$
- **end**

end

6.1.5 Fitted Q-iteration

Algorithm 9: Fitted Q-iteration

Loop
- **1.** Collect dataset $\{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)\}$ using some policy
- **Loop**
  - **2.** Set $y_i \leftarrow r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \max_{\mathbf{a}'_i} Q_\phi(\mathbf{s}'_i, \mathbf{a}'_i)$
  - **3.** Set $\phi \leftarrow \operatorname{argmin}_\phi \frac{1}{2}\sum_i ||Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - y_i||^2$
- **end**

end

6.1.6 Online Q-iteration

Algorithm 10: Online Q-iteration

Loop
- **1.** Take some action $\mathbf{a}_i$ and observe $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)$
- **2.** $y_i = r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \max_{\mathbf{a}'_i} Q_\phi(\mathbf{s}'_i, \mathbf{a}'_i)$
- **3.** $\phi \leftarrow \phi - \alpha \frac{dQ_\phi}{d\phi}(\mathbf{s}_i, \mathbf{a}_i)(Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - y_i)$

end

6.2 Tips & hacks

1. **Exploration:** Use a different policy to improve exploration

   - Espilon-greedy exploration: $\pi(\mathbf{a}_t|\mathbf{s}_t) = \begin{cases} 1-\epsilon \text{ if } \mathbf{a}_t = \operatorname{argmax}_{\mathbf{a}_t} Q_\phi(\mathbf{s}_t, \mathbf{a}_t) \\ \frac{\epsilon}{|\mathcal{A}|-1} \text{ otherwise} \end{cases}$
   - Boltzmann exploration: $\pi(\mathbf{a}_t|\mathbf{s}_t) \propto \exp\left(Q_\phi(\mathbf{s}_t, \mathbf{a}_t)\right)$

7 Deep Reinforcement Learning with Q-functions

7.1 Algorithms

7.1.1 Q-learning with replay buffer

Algorithm 11: Q-learning with replay buffer

Loop
- **1.** Collect dataset $\{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)\}$ using some policy, and add it to $\mathcal{B}$
- **Loop**
  - **2.** Sample a batch $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)$ from $\mathcal{B}$
  - **3.** $y_i = r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \max_{\mathbf{a}'_i} Q_\phi(\mathbf{s}'_i, \mathbf{a}'_i)$
  - **4.** $\phi \leftarrow \phi - \alpha \sum_i \frac{dQ_\phi}{d\phi}(\mathbf{s}_i, \mathbf{a}_i)(Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - y_i)$
- **end**

end

7.1.2 Q-learning with replay buffer and target network

Algorithm 12: Q-learning with replay buffer

Loop
- **1.** Save network parameters $\phi' \leftarrow \phi$
- **Loop**
  - **2.** Collect dataset $\{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)\}$ using some policy, and add it to $\mathcal{B}$
  - **Loop**
    - **3.** Sample a batch $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)$ from $\mathcal{B}$
    - **4.** $y_i = r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \max_{\mathbf{a}'_i} Q_{\phi'}(\mathbf{s}'_i, \mathbf{a}'_i)$
    - **5.** $\phi \leftarrow \phi - \alpha \sum_i \frac{dQ_\phi}{d\phi}(\mathbf{s}_i, \mathbf{a}_i)(Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - y_i)$
  - **end**
- **end**

end

7.1.3 DQN: classic Deep Q-learning

Algorithm 13: DQN: classic Deep Q-learning

Loop
- **1.** Take some action $\mathbf{a}_i$, observe $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)$ and add it to $\mathcal{B}$
- **2.** Sample a mini-batch $(\mathbf{s}_j, \mathbf{a}_j, \mathbf{s}'_j, r_j)$ from $\mathcal{B}$ uniformly
- **3.** Compute $y_j = r(\mathbf{s}_j, \mathbf{a}_j) + \gamma \max_{\mathbf{a}'_j} Q_{\phi'}(\mathbf{s}'_j, \mathbf{a}'_j)$ using the target network $Q_{\phi'}$
- **4.** $\phi \leftarrow \phi - \alpha \sum_j \frac{dQ_\phi}{d\phi}(\mathbf{s}_j, \mathbf{a}_j)(Q_\phi(\mathbf{s}_j, \mathbf{a}_j) - y_j)$
- **5.** Update ϕ': copy ϕ every N steps or using Polyak averaging
  $\phi' \leftarrow \tau\phi' - (1 - \tau)\phi$

end

7.1.4 DDPG: Q-learning for continuous actions

Algorithm 14: DDPG: Deep Deterministic Policy Gradient

Loop

1. Take some action $\mathbf{a}_i$, observe $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)$ and add it to $\mathcal{B}$

2. Sample a mini-batch $(\mathbf{s}_j, \mathbf{a}_j, \mathbf{s}'_j, r_j)$ from $\mathcal{B}$ uniformly

3. Compute $y_j = r(\mathbf{s}_j, \mathbf{a}_j) + \gamma \max_{\mathbf{a}'_j} Q_{\phi'}(\mathbf{s}'_j, \mu_{\theta'}(\mathbf{s}'_j))$ using target nets $Q_{\phi'}$, $\mu_{\theta'}$

4. $\phi \leftarrow \phi - \alpha \sum_j \frac{dQ_\phi}{d\phi}(\mathbf{s}_j, \mathbf{a}_j)(Q_\phi(\mathbf{s}_j, \mathbf{a}_j) - y_j)$

5. $\theta \leftarrow \theta + \beta \sum_j \frac{d\mu}{d\theta}(\mathbf{s}_j) \frac{dQ_\phi}{d\mathbf{a}}(\mathbf{s}_j, \mathbf{a})$

6. Update ϕ' and θ'

end

7.2 Tips & hacks

1. **Double Q-learning:** use two networks to avoid overestimation
   - Standard Q-learning: $y = r + \gamma Q_{\phi'}(\mathbf{s}', \text{argmax}_{\mathbf{a}'} Q_{\phi'}(\mathbf{s}', \mathbf{a}'))$
   - Double Q-learning: $y = r + \gamma Q_{\phi'}(\mathbf{s}', \text{argmax}_{\mathbf{a}'} Q_{\phi}(\mathbf{s}', \mathbf{a}'))$
2. **Multi-step returns:** $y_t^j = \sum_{t'=t}^{t+N-1} \gamma^{t'-t} r_{t'}^j + \gamma^N \max_{\mathbf{a}_{t+N}^j} Q_{\phi'}(\mathbf{s}_{t+N}^j, \mathbf{a}_{t+N}^j)$
3. **General tips:**
   - Test on easy, reliable tasks
   - Use large replay buffers
   - Start with high exploration, then gradually reduce
   - Clip gradients or use Huber loss
   - Use double Q-learning
   - Use N-step returns
   - Schedule exploration and learning rates
   - Run multiple random seeds
   - Use DDPG for continuous actions, or other methods

8 Advanced Policy Gradients

9 Model-based planning

10 Model-based Reinforcement Learning

10.1 Algorithms

10.1.1 Model-based Reinforcement Learning version 0.5

Algorithm 15: Model-based RL version 0.5

1. Run some base policy $\pi_0(\mathbf{a}_t|\mathbf{s}_t)$, and collect $\mathcal{D} = \{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i)\}$
2. Learn the dynamics model $f(\mathbf{s}, \mathbf{a})$ to minimize $\sum_i ||f(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{s}'_i||^2$
3. Plan through $f(\mathbf{s}, \mathbf{a})$ to choose actions

10.1.2 Model-based Reinforcement Learning version 1.0

Algorithm 16: Model-based RL version 1.0

1. Run some base policy $\pi_0(\mathbf{a}_t|\mathbf{s}_t)$, and collect $\mathcal{D} = \{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i)\}$
Loop
  2. Learn the dynamics model $f(\mathbf{s}, \mathbf{a})$ to minimize $\sum_i ||f(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{s}'_i||^2$
  3. Plan through $f(\mathbf{s}, \mathbf{a})$ to choose actions
  4. Execute those actions and add the data to $\mathcal{D}$
end

10.1.3 Model-based Reinforcement Learning version 1.5

Algorithm 17: Model-based RL version 1.5

1. Run some base policy $\pi_0(\mathbf{a}_t|\mathbf{s}_t)$, and collect $\mathcal{D} = \{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i)\}$
Loop *(every N steps)*
  2. Learn the dynamics model $f(\mathbf{s}, \mathbf{a})$ to minimize $\sum_i ||f(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{s}'_i||^2$
  Loop
    3. Plan through $f(\mathbf{s}, \mathbf{a})$ to choose actions
    4. Execute the first planned action, and observe $\mathbf{s}'$ (MPC)
    5. Append $(\mathbf{s}, \mathbf{a}, \mathbf{s}')$ to $\mathcal{D}$
  end
end

10.1.4 Model-based Reinforcement Learning with latent space models

Algorithm 18: Model-based RL with latent space models

1. Run some base policy $\pi_0(\mathbf{a}_t|\mathbf{o}_t)$, and collect $\mathcal{D} = \{(\mathbf{o}_i, \mathbf{a}_i, \mathbf{o}'_i)\}$

Loop *(every N steps)*

  2. Learn $p_\phi(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$, $p_\phi(\mathbf{r}_t|\mathbf{s}_t)$, $p(\mathbf{o}_t|\mathbf{s}_t)$, $g_\psi(\mathbf{o}_t)$ to maximize

$$\max_{\phi,\psi} \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} \underbrace{\log p_\phi(g_\psi(\mathbf{o}^i_{t+1})|g_\psi(\mathbf{o}^i_t), \mathbf{a}^i_t)}_{\text{latent space dynamics}} + \underbrace{\log p_\phi(\mathbf{o}^i_t|g_\psi(\mathbf{o}^i_t))}_{\text{image reconstruction}} + \underbrace{\log p_\phi(r^i_t|g_\psi(\mathbf{o}^i_t))}_{\text{reward model}}$$

  Loop

    3. Plan through the model to choose actions

    4. Execute the first planned action, and observe $\mathbf{o}'$

    5. Append $(\mathbf{o}, \mathbf{a}, \mathbf{o}')$ to $\mathcal{D}$

  end

end

10.2 Tips & hacks

1. **Uncertainty-aware models:**

   - Use output entropy in the networks
   - Estimate the uncertainty of the networks
     - Use Bayesian networks
     - Use bootstrap ensembles, where each model is trained on a subset sampled with replacement from the dataset

11 Model-based Policy Learning

11.1 Algorithms

11.1.1 Model-based Reinforcement Learning version 2.0

Algorithm 19: Model-based RL version 2.0

1. Run some base policy $\pi_0(\mathbf{a}_t|\mathbf{s}_t)$, and collect $\mathcal{D} = \{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i)\}$

Loop

  2. Learn the dynamics model $f(\mathbf{s}, \mathbf{a})$ to minimize $\sum_i ||f(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{s}'_i||^2$

  3. Backpropagate through $f(\mathbf{s}, \mathbf{a})$ into the policy to optimize $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$

  4. Run $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ and add the visited $(\mathbf{s}, \mathbf{a}, \mathbf{s}')$ to $\mathcal{D}$

end

11.1.2 DYNA: online Q-learning model-free Reinforcement Learning with a model

Algorithm 20: DYNA

Loop
- **1.** Given $\mathbf{s}$, pick $\mathbf{a}$ using some exploration policy, and observe $(\mathbf{s}, \mathbf{a}, \mathbf{s}', r)$
- **2.** Update the models $\hat{p}(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ and $\hat{r}(\mathbf{s}, \mathbf{a})$ using $(\mathbf{s}, \mathbf{a}, \mathbf{s}')$
- **3.** Update $Q(\mathbf{s}, \mathbf{a}) \leftarrow Q(\mathbf{s}, \mathbf{a}) + \alpha E_{\mathbf{s}',r}\left[r + \max_{\mathbf{a}'} Q(\mathbf{s}', \mathbf{a}') - Q(\mathbf{s}, \mathbf{a})\right]$
- **Loop**
  - **4.** Sample $(\mathbf{s}, \mathbf{a}) \sim \mathcal{B}$ from the buffer
  - **5.** Update $Q(\mathbf{s}, \mathbf{a}) \leftarrow Q(\mathbf{s}, \mathbf{a}) + \alpha E_{\mathbf{s}',r}\left[r + \max_{\mathbf{a}'} Q(\mathbf{s}', \mathbf{a}') - Q(\mathbf{s}, \mathbf{a})\right]$
- **end**

end

11.1.3 General DYNA-style model-based Reinforcement Learning

Algorithm 21: General DYNA-style model-based RL

Loop
- **1.** Collect some data $(\mathbf{s}, \mathbf{a}, \mathbf{s}', r)$
- **2.** Learn the model $\hat{p}(\mathbf{s}'|\mathbf{s}, \mathbf{a})$, and optionally $\hat{r}(\mathbf{s}, \mathbf{a})$
- **Loop**
  - **3.** Sample $\mathbf{s} \sim \mathcal{B}$ from the buffer
  - **4.** Choose action $\mathbf{a}$ (from $\mathcal{B}$, from π or randomly)
  - **5.** Simulate $\mathbf{s}' \sim \hat{p}(\mathbf{s}'|\mathbf{s}, \mathbf{a})$, and $r = \hat{r}(\mathbf{s}, \mathbf{a})$
  - **6.** Train on $(\mathbf{s}, \mathbf{a}, \mathbf{s}', r)$ with model-free RL
  - **7.** (Optional) Take N more model-based steps
- **end**

end

11.1.4 MBA: Model-based Acceleration – MVE: Model-based Value Expansion – MBPO: Model-based Policy Optimization

Algorithm 22: MBA – MVE – MBPO

Loop
- **1.** Take some action $\mathbf{a}_i$, observe $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)$ and add it to $\mathcal{B}$
- **2.** Sample a mini-batch $\{\mathbf{s}_j, \mathbf{a}_j, \mathbf{s}'_j, r_j\}$ from $\mathcal{B}$ uniformly
- **3.** Use $\{\mathbf{s}_j, \mathbf{a}_j, \mathbf{s}'_j\}$ to update the model $\hat{p}(\mathbf{s}'|\mathbf{s}, \mathbf{a})$
- **4.** Sample $\{\mathbf{s}_j\}$ from $\mathcal{B}$
- **5.** For each $\mathbf{s}_j$, perform model-based rollout with $\mathbf{a} = \pi(\mathbf{s})$
- **6.** Use all transitions $(\mathbf{s}, \mathbf{a}, \mathbf{s}', r)$ along the rollout to update the Q-function

end

11.1.5 Divide and Conquer Reinforcement Learning

Algorithm 23: Divide and Conquer RL

Loop

1. Optimize each local policy $\pi_{\theta_i}(\mathbf{a}_t|\mathbf{s}_t)$ on initial state $\mathbf{s}_0^i$ w.r.t $\widetilde{r}_k^i(\mathbf{s}_t, \mathbf{a}_t)$

2. Use samples from step 1. to train $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ to mimic each $\pi_{\theta_i}(\mathbf{a}_t|\mathbf{s}_t)$

3. Update reward $\widetilde{r}_{k+1}^i(\mathbf{s}_t, \mathbf{a}_t) = r(\mathbf{s}_t, \mathbf{a}_t) + \lambda_{k+1}^i \log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$

end

11.2 Tips & hacks

1. **Local models:** use complex controllers as local models

2. **Dynamics models:** use Bayesian linear regression

3. **Multi-task learning:**

    - Use ensemble models
    - Use soft targets
    - Use distillation: train a global policy with supervised learning to mimic each policy

12 Variational Inference & Generative models

13 Control as inference

13.1 Algorithms

13.1.1 Soft Q-learning

Algorithm 24: Soft Q-learning

Loop

1. Take some action $\mathbf{a}_i$, observe $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)$ and add it to $\mathcal{B}$

2. Sample a mini-batch $\{\mathbf{s}_j, \mathbf{a}_j, \mathbf{s}'_j, r_j\}$ from $\mathcal{B}$ uniformly

3. Compute $y_j = r(\mathbf{s}_j, \mathbf{a}_j) + \gamma \text{softmax}_{\mathbf{a}'_j} Q_{\phi'}(\mathbf{s}'_j, \mathbf{a}'_j)$ using the target network $Q_{\phi'}$

4. $\phi \leftarrow \phi - \alpha \sum_j \frac{dQ_\phi}{d\phi}(\mathbf{s}_j, \mathbf{a}_j)(Q_\phi(\mathbf{s}_j, \mathbf{a}_j) - y_j)$

5. Update ϕ': copy ϕ every N steps or using Polyak averaging $\phi' \leftarrow \tau\phi' - (1-\tau)\phi$

end

14 Inverse Reinforcement Learning

14.1 Algorithms

14.1.1 Maximum Entropy Inverse Reinforcement Learning

Algorithm 25: MaxEnt IRL

Loop

1. Given ψ, compute backward message $\beta(\mathbf{s}_t, \mathbf{a}_t) = p(\mathcal{O}_{t:T}|\mathbf{s}_t, \mathbf{a}_t)$

2. Given ψ, compute forward message $\alpha(\mathbf{s}_t) = p(\mathbf{s}_t|\mathcal{O}_{1:t-1})$

3. Compute $\mu_t(\mathbf{s}_t, \mathbf{a}_t) \propto \beta(\mathbf{s}_t, \mathbf{a}_t)\alpha(\mathbf{s}_t)$

4. Evaluate

$$\nabla_\psi \mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\psi r_\psi(\mathbf{s}_t^i, \mathbf{a}_t^i) - \sum_{t=1}^{T}\iint \mu_t(\mathbf{s}_t, \mathbf{a}_t)\nabla_\psi r_\psi(\mathbf{s}_t, \mathbf{a}_t) d\mathbf{s}_t d\mathbf{a}_t$$

5. $\psi \leftarrow \psi + \eta \nabla_{psi}\mathcal{L}$

end

14.2 Tips & hacks

1. **Loss computation:**

   - $\nabla_\psi \mathcal{L} \approx \frac{1}{N}\sum_{i=1}^{N}\nabla_\psi r_\psi(\tau_i) - \frac{1}{M}\sum_{j=1}^{M}\nabla_\psi r_\psi(\tau_j)$
   - With importance sampling: $\nabla_\psi \mathcal{L} \approx \frac{1}{N}\sum_{i=1}^{N}\nabla_\psi r_\psi(\tau_i) - \frac{1}{\sum_j w_j}\sum_{j=1}^{M} w_j \nabla_\psi r_\psi(\tau_j)$

     where $w_j = \frac{p(\tau)\exp r_\psi(\tau_j)}{\pi(\tau_j)} = \frac{\exp \sum_t r_\psi(\mathbf{s}_t, \mathbf{a}_t)}{\prod_t \pi(\mathbf{a}_t|\mathbf{s}_t)}$

2. IRL can be treated as a GAN

15 Transfer & Multi-task Learning

15.1 Tips & hacks

1. **Forward transfer:** train on one task, transfer to a new task

   - Finetune
   - Randomize the source domains

2. **Multi-task transfer:** train on many tasks, transfer to a new task

   - Generate highly randomized source domains
   - Model-based RL
   - Model distillation
   - Contextual policies
   - Modular policy networks

3. **Multi-task meta-learning:** learn to learn from many tasks
    - RNN-based meta-learning
    - Gradient-based meta-learning

16 Distributed Reinforcement Learning

17 Exploration

17.1 Algorithms

17.1.1 Pre-train & finetune

Algorithm 26: Pre-train & finetune

1. Collect demonstration data $(\mathbf{s}_i, \mathbf{a}_i)$

2. Initialize π_θ as $\max_\theta \sum_i \log \pi_\theta(\mathbf{a}_i|\mathbf{s}_i)$

Loop

  3. Run π_θ to collect experience

  4. Improve π_θ with any RL algorithm

end

17.2 Tips & hacks

1. **Exploration bonus:** $r^+(\mathbf{s}, \mathbf{a}) + \mathcal{B}(N(\mathbf{s}))$
    - UCB: $\mathcal{B}(N(\mathbf{s})) = \sqrt{\frac{2 \ln n}{N(\mathbf{s})}}$
    - MBIE-EB: $\mathcal{B}(N(\mathbf{s})) = \frac{1}{\sqrt{N(\mathbf{s})}}$
    - BEB: $\mathcal{B}(N(\mathbf{s})) = \frac{1}{N(\mathbf{s})}$
    - Use pseudo-counts and GANs
    - Counts encoding into hashes
    - Density modeling
    - Heuristic estimation of counts
2. **Thomson sampling**
3. **Bootstrap**
4. **Information gain**
5. **Data initialization:** initialize data (e.g for Q-learning) with demonstrations as off-policy data
    - Use good and bad demonstrations

- Use importance sampling

6. **Hybrid objective:** $\underbrace{E_{\pi_\theta}\left[r(\mathbf{s}, \mathbf{a})\right]}_{RL} + \lambda \underbrace{\sum_{(\mathbf{s},\mathbf{a}) \in \mathcal{D}_{demos}} \log \pi_\theta(\mathbf{a}|\mathbf{s})}_{IL}$

18 Meta-learning

19 Information theory