Planning and Optimal Control

8. Policy Gradient Methods

Lars Schmidt-Thieme

Information Systems and Machine Learning Lab (ISMLL)
Institute for Computer Science
University of Hildesheim, Germany

Syllabus

		A. Models for Sequential Data
Tue. 22.10.	(1)	1. Markov Models
Tue. 29.10.	(2)	2. Hidden Markov Models
Tue. 5.11.	(3)	3. State Space Models
Tue. 12.11.	(4)	3b. (ctd.)
		B. Models for Sequential Decisions
Tue. 19.11.	(5)	1. Markov Decision Processes
Tue. 26.11.	(6)	1b. (ctd.)
Tue. 3.12.	(7)	1c. (ctd.)
Tue. 10.12.	(8)	2. Monte Carlo and Temporal Difference Methods
Tue. 17.12.	(9)	3. Q Learning
Tue. 24.12.	—	— *Christmas Break* —
Tue. 7.1.	(10)	4. Policy Gradient Methods
Tue. 14.1.	(11)	tba
Tue. 21.1.	(12)	tba
Tue. 28.1.	(13)	8. Reinforcement Learning for Games
Tue. 4.2.	(14)	Q&A

Outline

1. The Policy Gradient Theorem
2. Monte Carlo Policy Gradient (REINFORCE)
3. TD Policy Gradients: Actor-Critic Methods
4. Deterministic Policy Gradients

Outline

1. The Policy Gradient Theorem

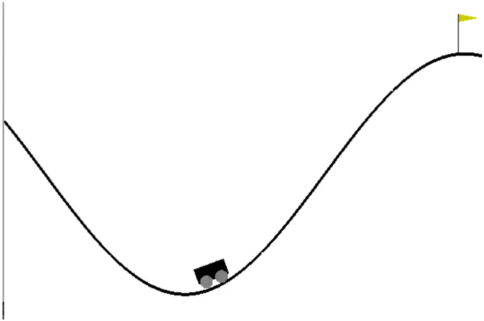
2. Monte Carlo Policy Gradient (REINFORCE)

3. TD Policy Gradients: Actor-Critic Methods

4. Deterministic Policy Gradients

- Q Learning: learn the action value function.
- Policy Gradient Methods

Example: Continuous Mountain Car



$$
\begin{aligned}
S &:= [-1.2, 0.5] \times [-0.07, 0.07], \quad s =: (x, v) \quad \text{(position and velocity)} \\
A &:= [-1, +1] \quad \text{(acceleration)} \\
x_{t+1} &:= \text{clip}(x_t + v_t) \\
v_{t+1} &:= \text{clip}(v_t + 0.001 a_t - 0.0025 \cos(3 x_t)) \\
p(x_0) &:= \text{unif}([-0.6, -0.4]), \quad v_0 := 0
\end{aligned}
$$

Idea

- let π be a parametrized policy with parameters θ:

$$\pi(a \mid s; \theta)$$

  - i.e., a neural network with input s and output a

- view its value function V^π of the unique starting state s_0 as a function of θ:

$$V(\theta) := V^\pi(s_0; \theta)$$

- finding the optimal policy $\rightsquigarrow$ maximize V w.r.t. θ
  - e.g., by gradient ascent:

$$\theta^{(t+1)} := \theta^{(t)} + \alpha_t \nabla_\theta V(\theta^{(t)})$$

State Distribution of a Policy

state visiting frequencies of a policy π:

$$\eta^\pi(s) := \mathbb{E}(|\{S_t = s \mid t = 1 : T\}|)$$

consistency:

$$\eta^\pi(s) = p(S_0 = s) + \sum_{s' \in S} \eta^\pi(s') \sum_{a \in A} \pi(a \mid s')\, p(s \mid s', a)$$

$$\rightsquigarrow \eta^\pi = (I - ((P^\pi)^T)^{-1}) p_0$$

with $\quad P^\pi := (\sum_{a \in A} \pi(a \mid s')\, p(s \mid s', a))_{(s', s) \in S^2} \quad$ state transition under π

$\quad p_0 := (p(S_0 = s))_{s \in S} \quad$ initial states

state distribution of policy π (**on-policy distribution**):

$$\mu^\pi(s) := \frac{\eta^\pi(s)}{\sum_{s' \in S} \eta^\pi(s')}$$

Note: With I the identity matrix.

Discounted State Distribution of a Policy

discounted state visiting frequencies of a policy π:

$$\eta^\pi(s) := \mathbb{E}(\sum_t \gamma^t \, \mathbb{I}(S_t = s))$$

- visiting a state at time t contributes weight γ^t.
- $=$ visiting frequency for $\gamma = 1$.

consistency:

$$\eta^\pi(s) = p(S_0 = s) + \sum_{s' \in S} \eta^\pi(s') \gamma \sum_{a \in A} \pi(a \mid s') \, p(s \mid s', a)$$

$$\rightsquigarrow \eta^\pi = (I - \gamma((P^\pi)^T)^{-1}) p_0$$

discounted state distribution of policy π:

$$\mu^\pi(s) := \frac{\eta^\pi(s)}{\sum_{s' \in S} \eta^\pi(s')}$$

Policy Gradient Theorem

$$\nabla_\theta V^\pi(s_0) \propto \sum_s \mu^\pi(s) \sum_a \nabla \pi(a \mid s; \theta) Q^\pi(s, a)$$

Policy Gradient Theorem / Proof

$$
\begin{aligned}
\nabla V^\pi(s) &= \nabla \sum_a \pi(a \mid s) Q^\pi(s,a) \\
&= \sum_a \nabla \pi(a \mid s) Q^\pi(s,a) + \pi(a \mid s) \nabla Q^\pi(s,a) \\
&= \sum_a \nabla \pi(a \mid s) Q^\pi(s,a) + \pi(a \mid s) \nabla \sum_{s',r} p(s', r \mid s, a)(r + \gamma V^\pi(s')) \\
&= \sum_a \nabla \pi(a \mid s) Q^\pi(s,a) + \pi(a \mid s) \sum_{s',r} p(s' \mid s, a) \gamma \nabla V^\pi(s') \\
&\underset{\text{rec.}}{=} \sum_a \nabla \pi(a \mid s) Q^\pi(s,a) + \pi(a \mid s) \sum_{s'} p(s' \mid s, a) \gamma \\
&\qquad (\sum_{a'} \nabla \pi(a' \mid s') Q^\pi(s',a') + \pi(a' \mid s') \sum_{s''} p(s'' \mid s', a') \gamma \nabla V^\pi(s'')) \\
&\underset{\text{rec.}}{=} \sum_{s'} \sum_{k=0}^{\infty} \Pr(s \rightarrow s', k, \pi) \gamma^k \sum_a \nabla \pi(a \mid s') Q^\pi(s', a)
\end{aligned}
$$

Note: γ is missing in [Sutton and Barto, 2018, p.325].

Policy Gradient Theorem / Proof

$$\nabla V^\pi(s) = \sum_{s'} \sum_{k=0}^{\infty} \Pr(s \rightarrow s', k, \pi)\gamma^k \sum_a \nabla\pi(a \mid s')Q^\pi(s', a)$$

$$\begin{aligned}
\nabla V^\pi(s_0) &= \sum_{s} \sum_{k=0}^{\infty} \Pr(s_0 \rightarrow s, k, \pi)\gamma^k \sum_a \nabla\pi(a \mid s)Q^\pi(s, a) \\
&= \sum_s \eta^\pi(s) \sum_a \nabla\pi(a \mid s)Q^\pi(s, a) \\
&\propto \sum_s \mu^\pi(s) \sum_a \nabla\pi(a \mid s)Q^\pi(s, a)
\end{aligned}$$

Outline

1. The Policy Gradient Theorem

2. Monte Carlo Policy Gradient (REINFORCE)

3. TD Policy Gradients: Actor-Critic Methods

4. Deterministic Policy Gradients

Monte Carlo Policy Gradient (REINFORCE)

$$\begin{aligned}
\nabla V^\pi(s_0) &\propto \sum_s \mu^\pi(s) \sum_a \nabla\pi(a \mid s) Q^\pi(s,a) \\
&= \mathbb{E}(\sum_a \nabla\pi(a \mid S_t) Q^\pi(S_t, a)) \\
&= \mathbb{E}(\sum_a \frac{\nabla\pi(a \mid S_t)}{\pi(a \mid S_t)} \pi(a \mid S_t) Q^\pi(S_t, a)) \\
&= \mathbb{E}(\frac{\nabla\pi(A_t \mid S_t)}{\pi(A_t \mid S_t)} V_t) \\
&= \mathbb{E}(V_t \nabla \log \pi(A_t \mid S_t))
\end{aligned}$$

$\rightsquigarrow$ update rule:

$$\theta := \theta + \alpha v_t \nabla \log \pi(a_t \mid s_t)$$

Note: Observed value V_t also called **return** in the literature.

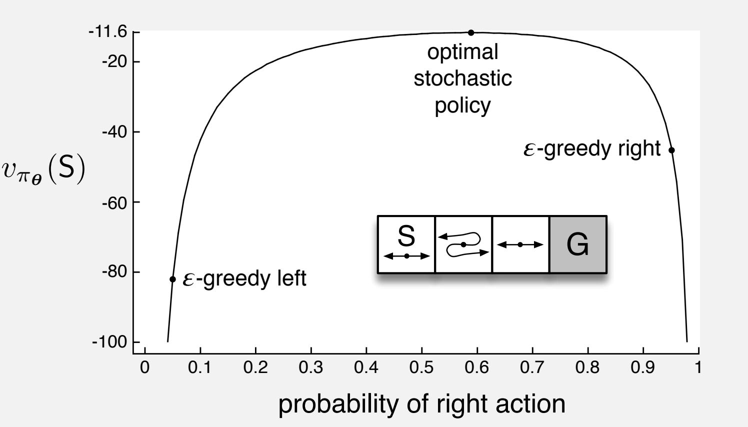
Monte Carlo Policy Gradient (REINFORCE) / Alg.

learn-opt-policy-discounted-mc-policygrad$(S, A, \gamma, s_{\text{term}}, N, \alpha)$:
  initialize parameters θ of policy π
  for $n := 1, \ldots, N$:
    $(s, a, r, T) :=$ generate-episode$(S, A, s_{\text{term}}, \pi)$
    for $t := 0, \ldots, T-1$:
      $v_t := \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$
      $\theta := \theta + \alpha v_t \nabla \log \pi(a_t \mid s_t)$
  return π

where

- s_{term} terminal state with zero reward.
- α **learning rate**
- returning π actually means to return its parameters θ

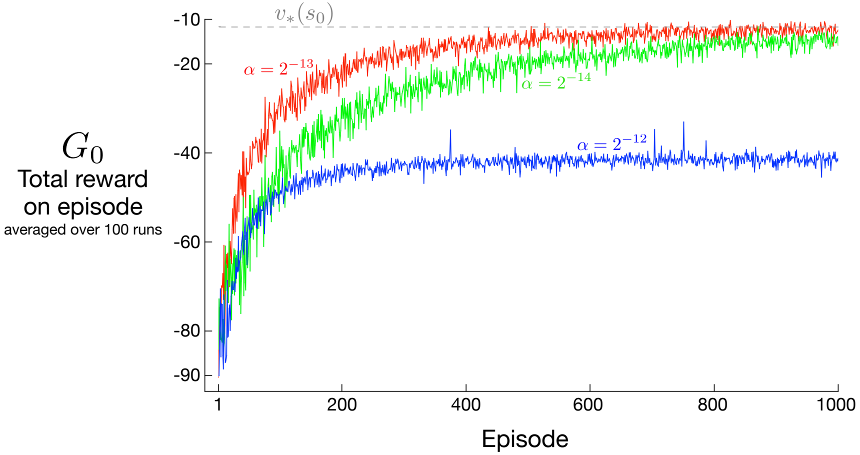
Example



[source: [Sutton and Barto, 2018, p.323]]

- assume all states are indistinguishable; reward -1 per step
- $\epsilon := 0.1$

Example



[source: [Sutton and Barto, 2018, p.328]]

Monte Carlo Policy Gradient (REINFORCE w. Baseline)

- **baseline** $b : S \rightarrow \mathbb{R}$, not depending on actions a.

$$\nabla V^\pi(s_0) \propto \sum_s \mu^\pi(s) \sum_a \nabla\pi(a \mid s) Q^\pi(s,a)$$
$$= \sum_s \mu^\pi(s) \sum_a \nabla\pi(a \mid s)(Q^\pi(s,a) - b(s))$$

as

$$\sum_a \nabla\pi(a \mid s) b(s) = b(s) \nabla \sum_a \pi(a \mid s) = b(s) \nabla 1 = 0$$

Monte Carlo Policy Gradient (REINFORCE w. Baseline)

$$
\begin{aligned}
\nabla V^\pi(s_0) &\propto \sum_s \mu^\pi(s) \sum_a \nabla \pi(a \mid s)(Q^\pi(s,a) - b(s)) \\
&= \mathbb{E}(\sum_a \nabla \pi(a \mid S_t)(Q^\pi(S_t,a) - b(S_t))) \\
&= \mathbb{E}(\sum_a \frac{\nabla \pi(a \mid S_t)}{\pi(a \mid S_t)} \pi(a \mid S_t)(Q^\pi(S_t,a) - b(S_t))) \\
&= \mathbb{E}(\frac{\nabla \pi(A_t \mid S_t)}{\pi(A_t \mid S_t)}(V_t - b(S_t))) \\
&= \mathbb{E}((V_t - b(S_t)) \nabla \log \pi(A_t \mid S_t))
\end{aligned}
$$

$\leadsto$ update rule:

$$
\theta := \theta + \alpha(v_t - b(s_t)) \nabla \log \pi(a_t \mid s_t)
$$

Monte Carlo Policy Gradient (REINFORCE w. Baseline)

- often a current estimate of the value function $\hat{V}$ is used as baseline:

$$b(s) := \hat{V}(s; \eta)$$

- updates for parameters η of $\hat{V}$:

$$\eta := \eta + \alpha_2(v_t - \hat{V}(s_t))\nabla\hat{V}(s_t)$$

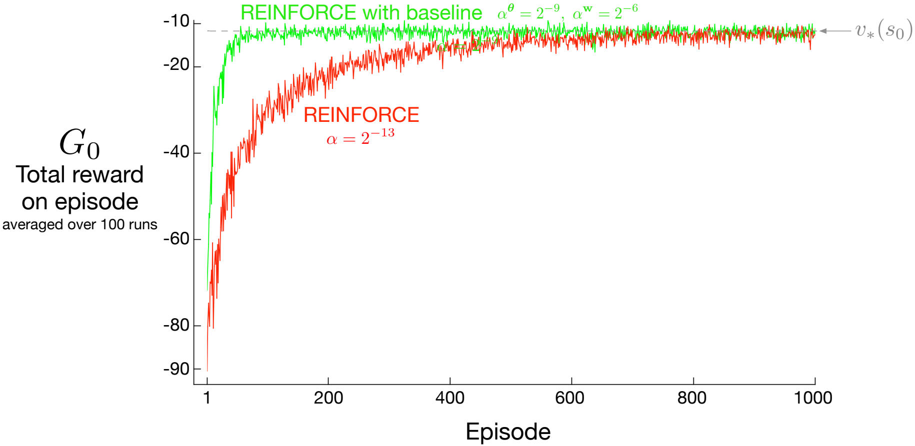
Monte Carlo Policy Gradient (REINFORCE w. Basline) / Alg.

```
learn-opt-policy-discounted-mc-policygrad-base(S, A, γ, s_term, N, α):
  initialize parameters η of value function V̂
  initialize parameters θ of policy π
  for n := 1, ..., N:
    (s, a, r, T) := generate-episode(S, A, s_term, π)
    for t := 0, ..., T − 1:
      v_t := Σ_{t'=t}^{T} γ^{t'−t} r_t
      v̂_t := V̂(s_t)
      η := η + α_2 (v_t − v̂_t) ∇V(s_t)
      θ := θ + α_1 (v_t − v̂_t) ∇ log π(a_t | s_t)
  return π
```

where

- s_{term} terminal state with zero reward.
- α_1, α_2 **learning rates** for π and $\hat{V}$, resp.
- returning π actually means to return its parameters θ

Example



[source: [Sutton and Barto, 2018, p.330]]

Outline

1. The Policy Gradient Theorem

2. Monte Carlo Policy Gradient (REINFORCE)

3. TD Policy Gradients: Actor-Critic Methods

4. Deterministic Policy Gradients

TD Policy Gradients: Actor-Critic Methods

To derive MC policy gradient / REINFORCE:

$$\mathbb{E}(\sum_a \pi(a \mid S_t) Q^\pi(S_t, a)) = \mathbb{E}(V_t)$$

To derive TD policy gradient / Actor Critic:

$$\mathbb{E}(\sum_a \pi(a \mid S_t) Q^\pi(S_t, a)) = \mathbb{E}(R_t + \gamma V_{t+1})$$

then plug in $\hat{V}(S_{t+1})$ for V_{t+1}:

$$\nabla V^\pi(s_0) \underset{approx.}{\propto} \mathbb{E}((R_t + \gamma \hat{V}(S_{t+1}))\nabla \log \pi(A_t \mid S_t))$$

and subtract baseline $\hat{V}(S_t)$:

$$\nabla V^\pi(s_0) \underset{approx.}{\propto} \mathbb{E}((R_t + \gamma \hat{V}(S_{t+1}) - \hat{V}(S_t))\nabla \log \pi(A_t \mid S_t)$$

$\rightsquigarrow$ update rule:

$$\theta := \theta + \alpha(r_t + \gamma \hat{V}(s_{t+1}) - \hat{V}(s_t))\nabla \log \pi(a_t \mid s_t)$$

TD Policy Gradients (Actor Critic)

learn-opt-policy-discounted-td-policygrad$(S, A, \gamma, s_{\text{term}}, N, \alpha)$:
  initialize parameters η of value function $\hat{V}$
  initialize parameters θ of policy π
  for $n := 1, \ldots, N$:
    $s :=$ new_process()
    $\hat{v} := \hat{V}(s)$
    while $s \neq s_{\text{term}}$:
      $a := \pi(s)$
      $(r, s') :=$ execute_action(s, a)
      $\hat{v}' := \hat{V}(s')$
      $\delta := r + \gamma \hat{v}' - \hat{v}$
      $\eta := \eta + \alpha_2 \delta \nabla V(s)$
      $\theta := \theta + \alpha_1 \delta \nabla \log \pi(a \mid s)$
      $s := s', \hat{v} := \hat{v}'$
  return π

where

- new_process() sets up a new process.
- execute_action(s, a) executes action a in process in state s.

limitations:

- online $\rightsquigarrow$ replay memory
- gradient descent $\rightsquigarrow$ general model update algorithm

TD Policy Gradients (Actor Critic) / Replay Memory

learn-opt-policy-discounted-td-policygrad$'(S, A, \gamma, s_{\text{term}}, N, \alpha)$:
  initialize parameters η of value function $\hat{V}$
  initialize parameters θ of policy π
  $\mathcal{D} := \emptyset$
  for $n := 1, \ldots, N$:
    $s :=$ new_process()
    while $s \neq s_{\text{term}}$:
      $a := \pi(s)$
      $(r, s') :=$ execute_action(s, a)
      $\mathcal{D} := \mathcal{D} \cup \{(s, a, r, s')\}$
      $\mathcal{D}'_1 := \{((s, a), r + \gamma \hat{V}(s')) \mid (s, a, r, s') \in \mathcal{D}\}$
      $\mathcal{D}'_2 := \{(s, a, \text{caseweight} = r + \gamma \hat{V}(s') - \hat{V}(s)) \mid (s, a, r, s') \in \mathcal{D}\}$
      $\hat{V} :=$ update-model$(\hat{V}, \mathcal{D}'_1)$
      $\pi :=$ update-model$(\pi, \mathcal{D}'_2)$
      $s := s'$
  return π

where

- new_process() sets up a new process.
- execute_action(s, a) executes action a in process in state s.

Outline

1. The Policy Gradient Theorem

2. Monte Carlo Policy Gradient (REINFORCE)

3. TD Policy Gradients: Actor-Critic Methods

4. Deterministic Policy Gradients

Policy Value

- for non-finite MDPs, use expected value over starting states (policy value) to define optimality of a policy:

$$V(\pi) := \mathbb{E}_{s_0 \sim p}(V^\pi(s_0)) = \int_{s_0} V^\pi(s_0) p(s_0) ds_0$$
$$= \mathbb{E}_{s \sim \mu^\pi, a \sim \pi}(r(s,a))$$

- policy gradient theorem is for **stochastic policies**

$$\pi : S \times A \to [0,1]$$

- is there a corresponding result for **deterministic policies**?

$$\pi : S \to A$$

Deterministic Policy Gradients

- stochastic policy gradient:

$$\nabla V(\pi) = \mathbb{E}_{s\sim\eta^\pi, a\sim\pi}(\nabla_\theta \log \pi(s,a;\theta) Q^\pi(s,a))$$

- deterministic policy gradient:

$$\nabla V(\pi) = \mathbb{E}_{s\sim\eta^\pi}(\nabla_\theta \pi(s;\theta) \nabla_a Q^\pi(s,a)|_{a=\pi(s)})$$

- determinisitc policy gradient is limit of stochastic one:

$$\lim_{\pi'\to\pi} \nabla_\theta V(\pi') = \nabla_\theta V(\pi), \quad \pi' \text{ stochastic policy}, \pi \text{ deterministic policy}$$

- see Silver et al. [2014]

Overview

what to learn / *from what to learn*

		from episodes	from transitions
value function	V^π	Monte Carlo for V^π	Temporal Differences TD
action value function	Q^π	Monte Carlo for Q^π	SARSA
optimal action value function	Q^{π^*}	.	Q Learning
optimal policy	π^*	Monte Carlo Policy Gradient / REINFORCE	TD Policy Gradient / Actor Critic

Summary

- **Policy gradient methods** parametrize the policy directly: $\pi(s;\theta)$
  - instead of parametrizing the action-value function $Q(s,a;\theta)$ and then deriving the policy as $\pi(s) := \arg\max_a Q(s,a;\theta)$.
- **Gradients of the value function** can be computed from gradients of the policy via the **policy gradient theorem**:

$$\nabla_\theta V^\pi(s_0) \propto \sum_s \mu^\pi(s) \sum_a \nabla\pi(a \mid s;\theta) Q^\pi(s,a)$$

- Combined with observed values (Monte Carlo approach), gradient ascent for the (implicit) value function leads to a simple update rule for the policy parameters θ called **REINFORCE**:

$$\theta := \theta + \alpha v_t \nabla \log \pi(a_t \mid s_t)$$

- any **baseline** for the observed value can be used and often accelerates convergence
  - esp. using a current estimate for the value function as baseline.

$$\theta := \theta + \alpha(v_t - \hat{V}(s_t))\nabla \log \pi(a_t \mid s_t)$$

Summary (2/2)

- Instead of using observed values (MC approach), one also can combine policy gradients with temporal differences (called **actor critic methods**):

$$\theta := \theta + \alpha(r_t + \gamma \hat{V}(s_{t+1}) - \hat{V}(s_t)) \nabla \log \pi(a_t \mid s_t)$$

  - then besides the policy model (actor),
    a second model for the value function (critic) is required.

Further Readings

- Reinforcement Learning:
  - Olivier Sigaud, Frederick Garica (2010): *Reinforcement Learning*, ch. 1 in Sigaud and Buffet [2010].
  - Sutton and Barto [2018]

References

Olivier Sigaud and Olivier Buffet, editors. *Markov Decision Processes in Artificial Intelligence*. Wiley, 2010.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *ICML*, 2014.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning*. MIT Press, 2nd edition edition, 2018.