Reinforcement Learning
Lecture 7

Georg Martius

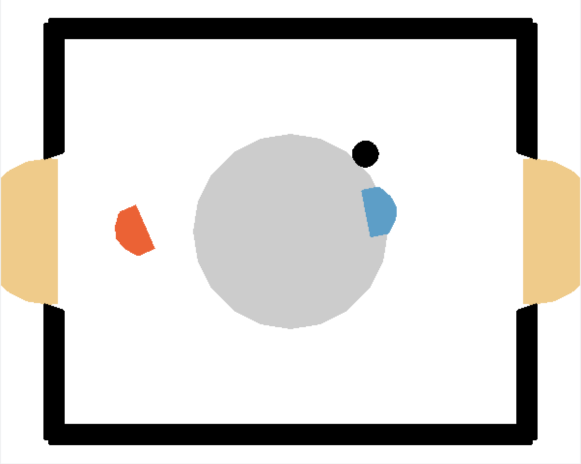
Distributed Intelligence / Autonomous Learning Group, Uni Tübingen, Germany

November 26, 2024

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Final Project Information

- You will implement an RL agent to play "Laserhockey"



- We will organize a competition: your agents play against each other
- Mark: report (60%), presentation as video (20%), performance (20%)

Final Project Information – cont.

- The report will have strict page limits, individual part for each person (latex template will be provided)
- Everyone needs to implement some addition to a standard algorithm and analyse it in a particular aspect
- Program performance is evaluated w.r.t. a reference agent
- Playing well in the tournament gives bonus points
- From tutorial 10 onwards: tutorial sessions for making progress towards final project
- You will get access to the TCML compute cloud
- Task description will be uploaded by 6th of December (end of next week)

Policy Gradient

Overview

- Last time: value estimation with function approximators
  1. DQN (Deep Q Networks): Q-learning with deep nets, replay buffers and target networks
- This lecture: policy gradient
  1. Finite Differences
  2. Monte-Carlo (REINFORCE)
- Next lecture: Actor-Critic

Following and adapting slides from David Silver

Recap Value-based Reinforcement Learning

So far looked at value-based RL:

- Estimate value function $v_\pi(s)$ or $q_\pi(s, a)$
- Policy is directly given by value function, e.g. using ϵ-greedy
- For large/continuous state space:
  - Use function approximation with parameters $\mathbf{w}$:

$$\hat{q}(s, a, \mathbf{w}) \approx q_\pi(s, a)$$

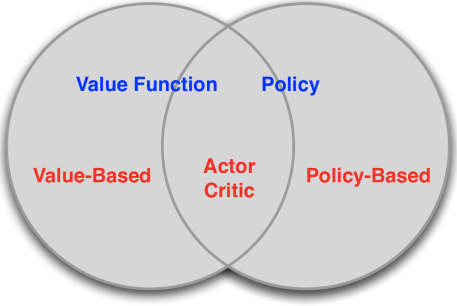
Now: parametrize policy explicitly with θ

$$\pi_\theta(s, a) = \mathbb{P}(a \mid s, \theta)$$

Remark on notation: to stay consistent with the literature
we use $\pi_\theta(s, a)$ instead of $\hat{\pi}(s, a, \theta)$
(which would be consistent with value function notation).

Value-Based and Policy-Based RL

- Value Based
  - Learnt Value Function
  - Implicit policy
- Policy Based
  - No Value Function
  - Learnt Policy
- Actor-Critic
  - Learnt Value Function
  - Learnt Policy



When is policy-based good?

Advantages:

- Better convergence properties
- Effective in high-dimensional or continuous action spaces
- Can learn stochastic policies

Disadvantages:

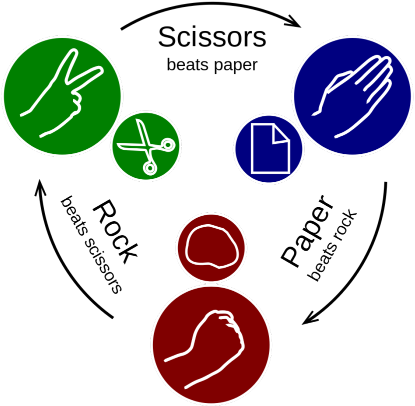
- Typically converge to a local rather than global optimum
- Evaluating a policy is typically inefficient and high variance

When do I need a stochastic policy?

- In non-stationary and
- partially observable systems

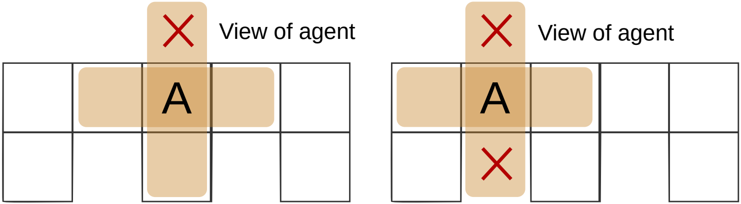
would be good to parameterise policy
if it is stochastic
deterministic policy is optimal for an MDP,
but there are cases when this isn't the case

Example: Rock-Paper-Scissors

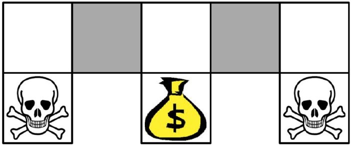


- Two-player game of rock-paper-scissors
- In iterated rock-paper-scissors:
  - A deterministic policy is easily exploited
  - A uniform random policy is optimal (i.e. Nash equilibrium)

Example: Aliased Gridworld



Agent cannot distinguish gray states:



- Deterministic Policy:
  - Either moves right in both gray states, or left in both (get's stuck)
- Stochastic Policy:
  - can move randomly in gray states, goes to cash every time

Objective Functions for Policy Optimization

Given: parametrized policy: π_θ

How to measure the quality of a policy π_θ?

- episodic: value of start state

$$J_1(\theta) = v^{\pi_\theta}(s_1) = \mathbb{E}_{\pi_\theta}[G_1]$$

- continuing: average value

$$J_{\text{avg}V}(\theta) = \sum_s d^{\pi_\theta}(s) v^{\pi_\theta}(s)$$

- or: average reward per time-step

$$J_{\text{avg}R}(\theta) = \sum_s d^{\pi_\theta}(s) \sum_a \pi_\theta(s, a) \mathcal{R}_s^a$$

where $d^{\pi_\theta}(s)$ is stationary distribution of Markov chain for π_θ

Policy Optimization

How do we optimize $J(\theta)$?

we might need to use algms for optimising functions when we don't know much about the function (for eg gradient free methods)

Find a good θ by:

Gradient-free methods (zero-order optimization):

- Hill climbing
- Simplex/ Nelder Mead
- Genetic algorithms and Evolutionary Strategies

First order methods: (often better efficiency)

- Gradient descent
- Conjugate gradient
- Quasi-Newton

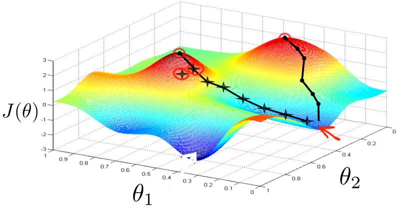
This lecture: gradient based methods + expoiting sequential structure

Policy gradient algorithms search for a local maximum in $J(\theta)$ by ascending the gradient of the policy, w.r.t. parameters θ

$$\Delta\theta = \alpha \nabla_\theta J(\theta)$$

Policy gradient: $\nabla_\theta J(\theta)$

$$\nabla_\theta J(\theta) = \begin{pmatrix} \frac{\partial J(\theta)}{\partial \theta_1} \\ \vdots \\ \frac{\partial J(\theta)}{\partial \theta_n} \end{pmatrix}$$



Finite Difference Method

- Computing the gradients is not easy!
- How does the cost depend on a change in the policy?

Finite Difference Method

Compute gradient by perturbations:

$$\frac{\partial J(\theta)}{\partial \theta_k} \approx \frac{J(\theta + \epsilon u_k) - J(\theta)}{\epsilon}$$

where u_k is unit vector with 1 in k-th component.

empirically calculate gradient by evaluating difference in J for small changes in theta

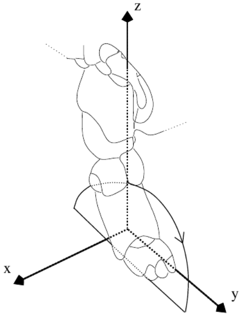
no need for J to be differentiable

- How many evaluations needed?
- For n dimensions n additional evaluations
- Simple, noisy, inefficient – but sometimes effective
- Policy does not need to be differentiable

noisy because assumes nothing else changes in world if theta changes

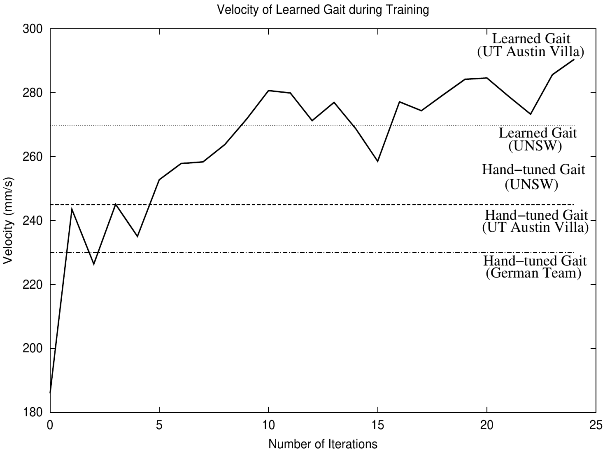
Example: Training Aibo to Walk faster

- Learn a fast gait for AIBO ((was) useful in Robocup)
- Gait is defined by 12 numbers
- Adapt parameters by finite difference
- Policy evaluation: time for field traversal



[Kohl & Stone, ICRA, 2004]

Aibo Walk policies



Show videos

Break

Policy Gradient

Consider a simple class of one-step MDPs

- Starting in state $s \sim \mu(s)$
- Terminating after one time-step with reward $r = \mathcal{R}_{s,a}$

$$
\begin{aligned}
J(\theta) &= \mathbb{E}_{\pi_\theta}[r] \\
&= \sum_{s \in \mathcal{S}} \mu(s) \sum_{a \in \mathcal{A}} \pi_\theta(a \mid s) \mathcal{R}_{s,a}
\end{aligned}
$$

sum over all possible starting states and possible actions. And we assume one step here (no returns)

$$
\nabla_\theta J(\theta) = \sum_{s \in \mathcal{S}} \mu(s) \sum_{a \in \mathcal{A}} \nabla_\theta \pi_\theta(a \mid s) \mathcal{R}_{s,a}
$$

gradient only based on pi (mu doesnt depend on theta)

How to compute these sums?

Score function – Likelihood ratios trick

$$
\begin{aligned}
\nabla_\theta \pi_\theta(a \mid s) &= \pi_\theta(a \mid s) \frac{\nabla_\theta \pi_\theta(a \mid s)}{\pi_\theta(a \mid s)} \\
&= \pi_\theta(a \mid s) \nabla_\theta \log \pi_\theta(a \mid s)
\end{aligned}
$$

The score function: $\nabla_\theta \log \pi_\theta(a \mid s)$

The score function is the gradient that would be used if maximing log-likelihood in a supervised setup.

Score Function

Score function:
$\nabla_\theta \log \pi_\theta(a \mid s)$

- Intuitively, the score function tell us how to increase the likelyhood of the what we plug in.
- In particular, how to increase probability of choosing action a (in state s)

Policy Gradient

Consider a simple class of one-step MDPs

- Starting in state $s \sim \mu(s)$
- Terminating after one time-step with reward $r = \mathcal{R}_{s,a}$

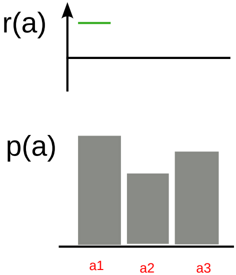
$$
\begin{aligned}
J(\theta) &= \mathbb{E}_{\pi_\theta}[r] \\
&= \sum_{s \in \mathcal{S}} \mu(s) \sum_{a \in \mathcal{A}} \pi_\theta(a \mid s) \mathcal{R}_{s,a} \\
\nabla_\theta J(\theta) &= \sum_{s \in \mathcal{S}} \mu(s) \sum_{a \in \mathcal{A}} \nabla_\theta \pi_\theta(a \mid s) \mathcal{R}_{s,a} \\
&= \underbrace{\sum_{s \in \mathcal{S}} \mu(s) \sum_{a \in \mathcal{A}} \pi_\theta(a \mid s)}_{\text{occurance under } \pi_\theta} \nabla_\theta \log \pi_\theta(a \mid s) \mathcal{R}_{s,a} \\
&= \mathbb{E}_{\pi_\theta} \left[\nabla_\theta \log \pi_\theta(a \mid s)\, r \right]
\end{aligned}
$$

Policy Gradient – Intuition

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a \mid s)\, r\right]$$

- score function: how to change parameters to get higher probability for action a
- consider simple case:
  - $r > 0$ for good outcomes: **increase** probability of choosing the action a
  - $r < 0$ for bad outcomes: **decrease** probability of choosing action a

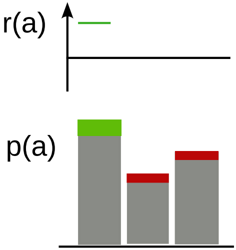
Example: $p(a) = \pi(a \mid s)$



$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a \mid s)\, r\right]$$

- score function: how to change parameters to get higher probability for action a
- consider simple case:
  - $r > 0$ for good outcomes: **increase** probability of choosing the action a
  - $r < 0$ for bad outcomes: **decrease** probability of choosing action a

Example: $p(a) = \pi(a \mid s)$

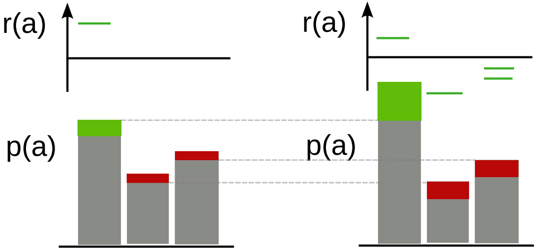


Policy Gradient – Intuition

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta} \left[\nabla_\theta \log \pi_\theta(a \mid s)\, r \right]$$

- score function: how to change parameters to get higher probability for action a
- consider simple case:
  - $r > 0$ for good outcomes: **increase** probability of choosing the action a
  - $r < 0$ for bad outcomes: **decrease** probability of choosing action a

Example: $p(a) = \pi(a \mid s)$



- The policy gradient theorem generalises the likelihood ratio approach to multi-step MDPs
- Replaces instantaneous reward r with long-term value $q^\pi(s, a)$

Theorem (Policy Gradient)

For any differentiable policy $\pi_\theta(a \mid s)$ for any of the policy objective functions $J = J_1, J_{\text{avg}R}$, or $\frac{1}{1-\gamma} J_{\text{avg}V}$, the policy gradient is

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a \mid s) q^{\pi_\theta}(s, a)\right]$$

Policy Gradient – Trajectory View

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a \mid s) q^{\pi_\theta}(s, a)\right]$$

Why does it also work for general MDPs?

- consider the expected return of trajectories $J_1(\theta) = v^{\pi_\theta}(s_1) = \mathbb{E}_{\pi_\theta}[G_1]$
- G is return of trajectory τ: need to know how to increase probability of τ

$$\begin{aligned}
\nabla_\theta \log p(\tau; \theta) &= \nabla_\theta \log \left[\prod_{t=0}^{H} \underbrace{p(s_{t+1} \mid s_t, a_t)}_{\text{dynamics model}} \cdot \underbrace{\pi_\theta(a_t \mid s_t)}_{\text{policy}}\right] \\
&= \nabla_\theta \left[\sum_{t=0}^{H} \log p(s_{t+1} \mid s_t, a_t) + \sum_{t=0}^{H} \log \pi_\theta(a_t \mid s_t)\right] \\
&= \nabla_\theta \sum_{t=0}^{H} \log \pi_\theta(a_t \mid s_t) \\
&= \sum_{t=0}^{H} \underbrace{\nabla_\theta \log \pi_\theta(a_t \mid s_t)}_{\text{no dynamics model needed}}
\end{aligned}$$

The **policy gradient** **increases** **probability** of trajectories with **high** **reward**

Policy Parametrizations and Score Function

Softmax Policy: Softmax of linear combination of features

$$\pi_\theta(a \mid s) \propto e^{\phi(s,a)^\top \theta}$$

Discrete actions:

$$\pi_\theta(a \mid s) = \frac{e^{\phi(s,a)^\top \theta}}{\sum_{a' \in \mathcal{A}} e^{\phi(s,a')^\top \theta}}$$

$$\begin{aligned}
\nabla_\theta \log \pi_\theta(a \mid s) &= \nabla_\theta \log e^{\phi(s,a)^\top \theta} - \nabla_\theta \log \sum_{a' \in \mathcal{A}} e^{\phi(s,a')^\top \theta} \\
&= \phi(s,a) - \frac{1}{\sum_{a' \in \mathcal{A}} e^{\phi(s,a')^\top \theta}} \nabla_\theta \sum_{a' \in \mathcal{A}} e^{\phi(s,a')^\top \theta} \\
&= \phi(s,a) - \frac{1}{\sum_{a' \in \mathcal{A}} e^{\phi(s,a')^\top \theta}} \sum_{a' \in \mathcal{A}} e^{\phi(s,a')^\top \theta} \phi(s,a') \\
&= \phi(s,a) - \sum_{a' \in \mathcal{A}} \pi_\theta(a' \mid s) \phi(s,a') \\
&= \phi(s,a) - \mathbb{E}_{\pi_\theta}[\phi(s,\cdot)]
\end{aligned}$$

score function

Monte-Carlo Policy Gradient (REINFORCE)

- Update parameters by stochastic gradient ascent
- Using policy gradient theorem
- Using return G_t as an unbiased sample of $q^{\pi_\theta}(s_t, a_t)$

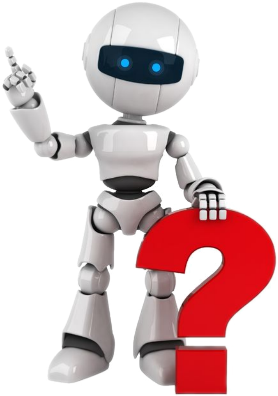
$$\Delta\theta_t = \alpha \nabla_\theta \log \pi_\theta (s_t, a_t) G_t$$

REINFORCE

1. Initialise θ arbitrarily
2. for each episode:
   - **2.1** $\tau \leftarrow \{s_1, a_1, r_2, \ldots, s_{T-1}, a_{T-1}, r_T\} \sim \pi_\theta$
   - **2.2** for $t = 1$ to $T - 1$:
     - **2.2.1** $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} r_k$
     - **2.2.2** $\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta (s_t, a_t) G$
3. return θ

this method is v sensitive to step size

Questions?



[Image: globalrobots.com]