Deep Reinforcement Learning

12 - Policy Learning

Prof. Dr. Malte Schilling

Autonomous Intelligent Systems Group

Recap – Convergence of Prediction Algorithms

On/Off-Policy	Algorithm	Table Lookup	Linear Approx.	Non-Linear
On-Policy	MC	✔	✔	✔
	TD	✔	✔	✘
Off-Policy	MC	✔	✔	✔
	TD	✔	✘	✘

(Silver 2015)

Recap – Deadly Triad

Three elements that can interfere with divergence: With only two present, we can avoid instability.

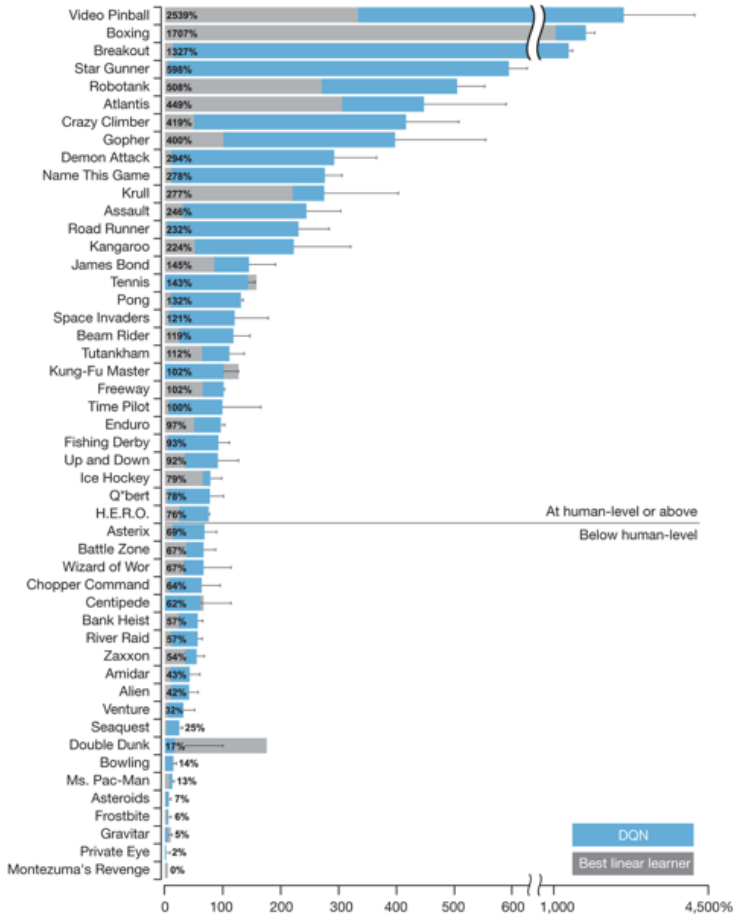
- Function approximation: Allows for generalization, required for realistic tasks.
- Bootstrapping: Update targets based on the existing estimates leads to more efficient learning.
- Off-Policy training: Crucial for Online Learning, meaning learning from a single stream of experience how to adapt many policies.

Countermeasures

We can't counter all these with at once, but DQN showed a way how to balance these issues (target networks for example). Other possibilities include, e.g., using n-step returns.

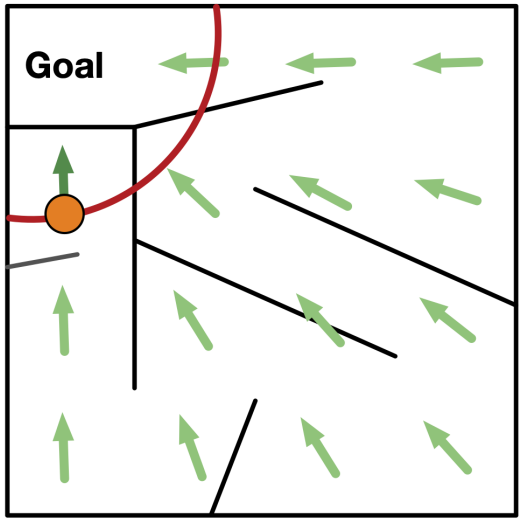
Recap – Drawbacks of DQN (and other DRL methods)

- Delayed Rewards (makes Credit Assignment even more difficult)
- Overfitting towards a specific niche and showing no generalization
- many real world scenarios are non-Markovian or non-stationary (e.g. when other agents are co-adapting)

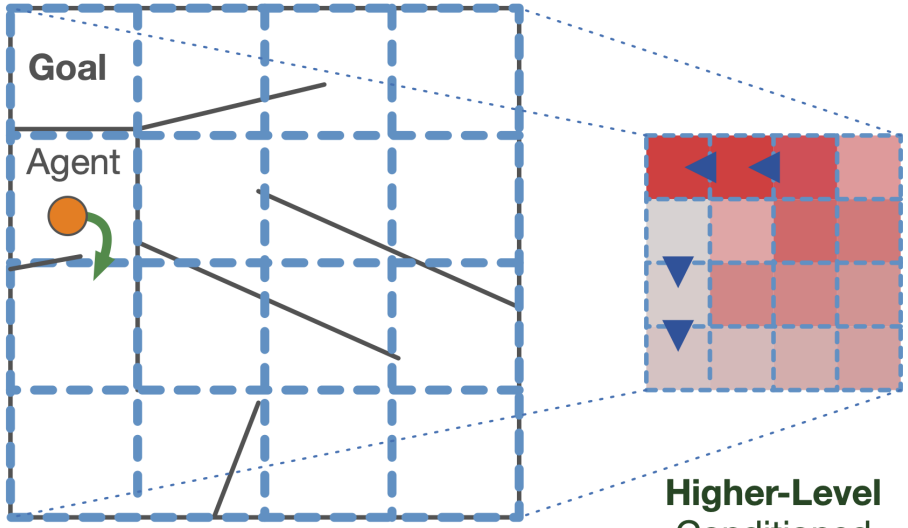


(Mnih u. a. 2015)

Recap – Deceptive Objectives



Distance-based Objective



Lower-level Motor Primitives

Higher-Level
Conditioned
on Location

Landscapes induced by objective functions are often deceptive – the objective function is misleading.

Often, stepping stones are required — initially, objective might get worse.

Overview Lecture

Policy Gradient Methods

- Stochastically sample variations of parameters
- Analytical Policy Gradient derivation and collecting
  - For Contextual PGs
  - For Episodic Cases
  - For the average reward case

Policy Learning

Policy-Based Reinforcement Learning

We already approximated the value or action-value function using parameters θ:

$$V_\theta(s) \approx V_\pi(s), Q_\theta(s,a) \approx Q_\pi(s,a)$$

A policy was generated directly from the value function, e.g., using ε-greedy.

But we can also directly parametrise the policy $\pi_\theta(s,a) = P(a|s,\theta)$

We will focus again on model-free reinforcement learning.

(Silver 2015)

Value-Based and Policy-Based RL

Value Based

- Learnt Value Function
- Implicit policy (e.g. ϵ-greedy)

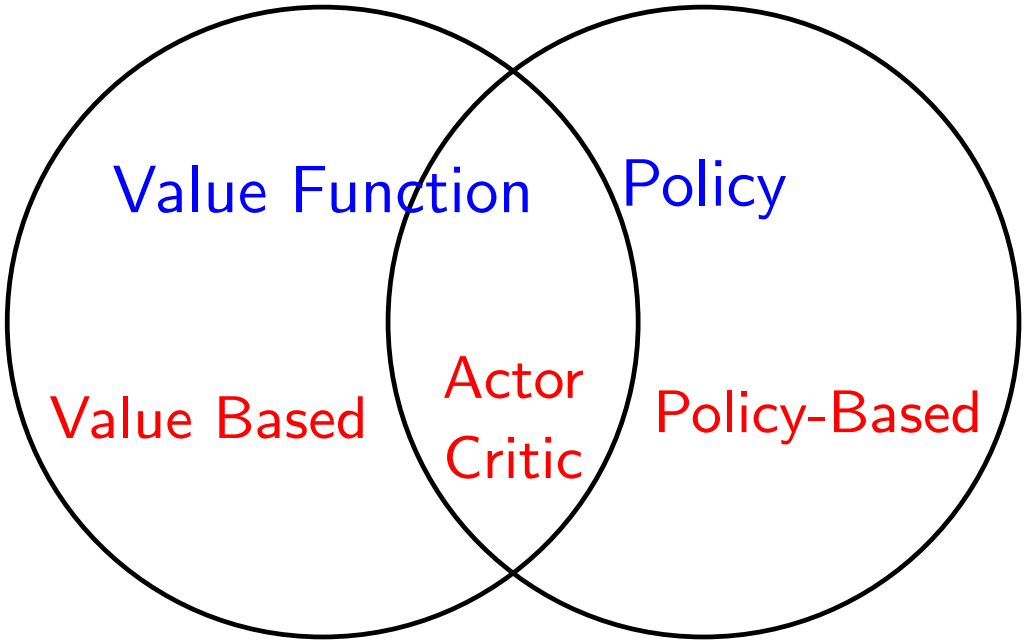
Policy Based

- No Value Function
- Learnt Policy

Actor-Critic

- Learnt Value Function
- Learnt Policy

Overview Approaches



Advantages of Policy-Based RL

Advantages:

- True objective
- Easy extended to high-dimensional or continuous action spaces
- Can learn stochastic policies
- Sometimes policies are simple while values and models are complex, e.g., complicated dynamics, but optimal policy is always “move forward”

Disadvantages:

- Could get stuck in local optima
- Obtained knowledge can be specific, does not always generalise well
- Does not necessarily extract all useful information from the data (when used in isolation)

(Silver 2015)

Policy Learning Objective

Policy Objective Functions

Goal: given policy $\pi_\theta(s, a)$, find best parameters θ

How do we measure the quality of a policy π_θ?

- In episodic environments: We can use the average total return per episode
- In continuing environments: We can use the average reward per step.

(Hasselt und Borsa 2021)

Policy Objective Functions: Episodic Environments

Episodic-return objective:

$$
\begin{aligned}
J_G(\theta) &= \mathbb{E}_{S_0 \sim d_0, \pi_0}\left(\sum_{t=0}^{\infty} \gamma^t R_{t+1}\right) \\
&= \mathbb{E}_{S_0 \sim d_0, \pi_0}\left(G_0\right) \\
&= \mathbb{E}_{S_0 \sim d_0}\left(\mathbb{E}_{\pi_0}(G_t | S_t = S_0)\right) \\
&= \mathbb{E}_{S_0 \sim d_0}\left(v_{\pi_0}(G_t | S_t = S_0)\right)
\end{aligned}
$$

where d_0 is the start-state distribution. This objective equals the expected value of the start state.

(Hasselt und Borsa 2021)

Policy Objective Functions: Average Reward Objective

$$
\begin{aligned}
J_R(\theta) &= \mathbb{E}_{\pi_\theta}\Big(R_{t+1}\Big) \\
&= \mathbb{E}_{S_t \sim d_{\pi_\theta}}\Big(\mathbb{E}_{A_t \sim \pi_\theta(S_t)}(R_{t+1}|S_t)\Big) \\
&= \sum_s d_{\pi_\theta}(s) \sum_a \pi_\theta(s,a) \sum_r p(r|s,a) r
\end{aligned}
$$

where $d_\pi(s) = p(S_t = s|\pi)$ is the probability of being in state s in the long run (Think of it as the ratio of time spent in s under policy π).

(Hasselt und Borsa 2021)

Learning on Policy Objective Function (episodic case)

$$J(\theta) = \mathbb{E}_{S_0 \sim d_0, \pi}(G) = v_{\pi_\theta}(S_0)$$

For optimization: We want to use gradient ascent on j over θ. Why could this be difficult?

- When we change θ, action selection (π) is affected.
- But this further affects which states will be visited and how often (plus corresponding rewards). This depends on the environment!

Policy Gradients

Policy Optimisation

Policy based reinforcement learning is an optimization problem.

- Find θ that maximises $J(\theta)$
- We will focus on stochastic gradient ascent, which is often quite efficient (and easy to use with deep nets).
- Different approaches that do not use gradient
  - Hill climbing / simulated annealing
  - Genetic algorithms / evolutionary strategies

(Hasselt und Borsa 2021)

Policy Gradient

Approach: Ascent the gradient of the objective $J(\theta)$

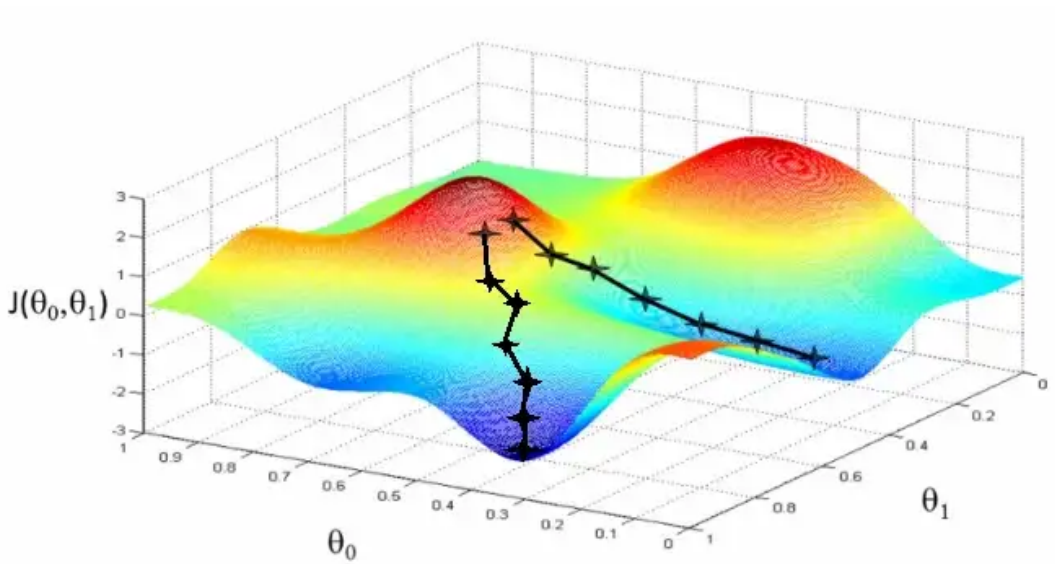
$$\Delta\theta = \alpha \nabla_\theta J(\theta)$$

- Policy Gradient

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

- α is a step-size parameter

Gradient Ascent



Stochastic policies help ensure that $J(\theta)$ is (mostly) smooth.

Gradients on parameterized policies

How to compute this gradient $\nabla_\theta J(\theta)$?

- Approximate stochastically.
- Assume policy π_θ is differentiable almost everywhere (e.g., neural net).

(Hasselt und Borsa 2021)

1) Approximate Gradients by Finite Differences

To evaluate policy gradient of $\pi_\theta(s, a)$

- For each dimension $k \in [1, n]$
  - Estimate k—th partial derivative of objective function w.r.t. θ_k
  - By perturbing θ_k by small amount ε in k—th dimension

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

- Uses n evaluations to compute policy gradient in n dimensions

Characteristics:

- Simple, noisy, inefficient - but sometimes effective
- Works for arbitrary policies, even if policy is not differentiable

(Silver 2015)

Example: Learning to Walk on AIBO

Goal: learn a fast walk on AIBO robot that can be applied in RoboCup

Parametrize AIBO walking policy and learn directly these parameters through reinforcement learning

(Kohl und Stone 2004)

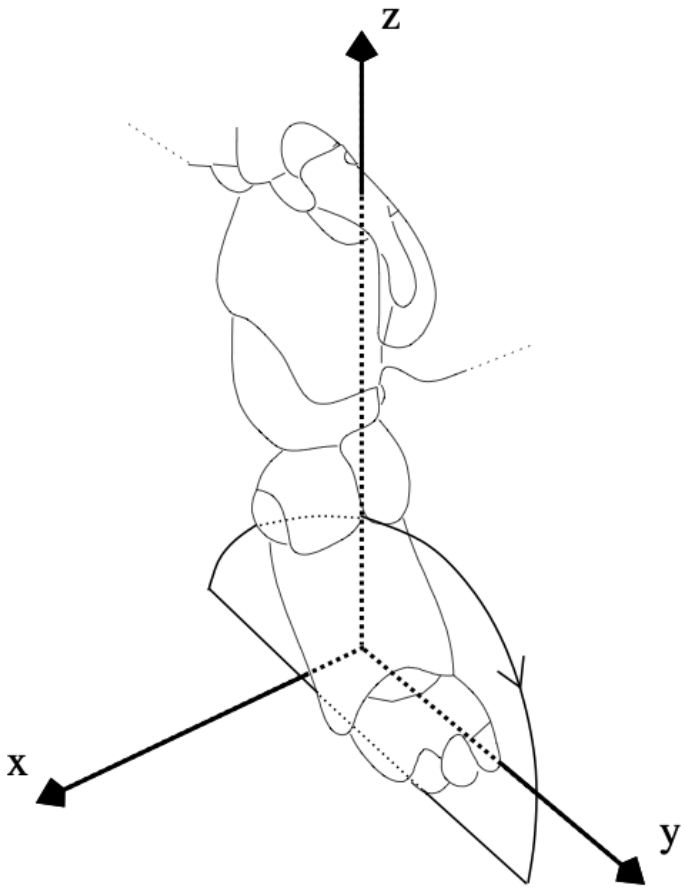
AIBO Parameters of gait for Stepping

Each leg is performing a half-elliptical locus. Each pair of diagonally opposite legs in phase with each other and perfectly out of phase with the other two.

Four parameters define this elliptical locus:

- length of the ellipse;
- height of the ellipse;
- position of the ellipse on x axis;
- position of the ellipse on y axis.

$= 12$ param. (front, rear, $+$ height, timing)



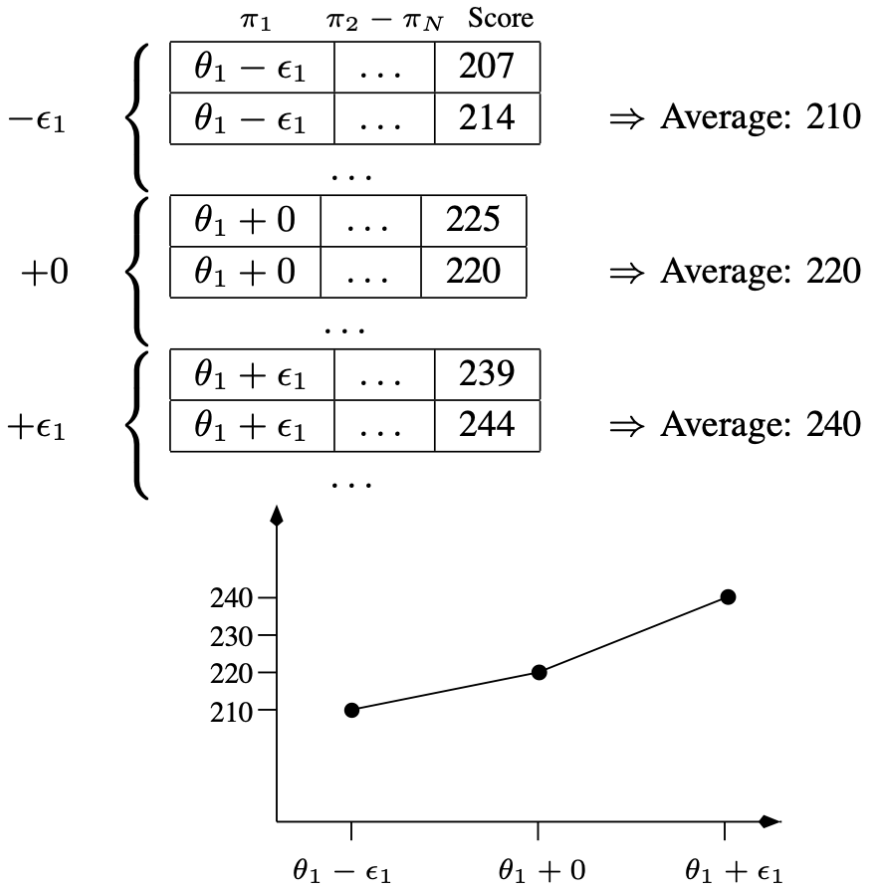
(Kohl und Stone 2004)

Example: Policy Gradient Approach

Goal: optimize forward speed as the sole objective function.

Approach: Policy gradient reinforcement learning – consider possible sets of parameter assignments that define a policy which is then executed on the robot.

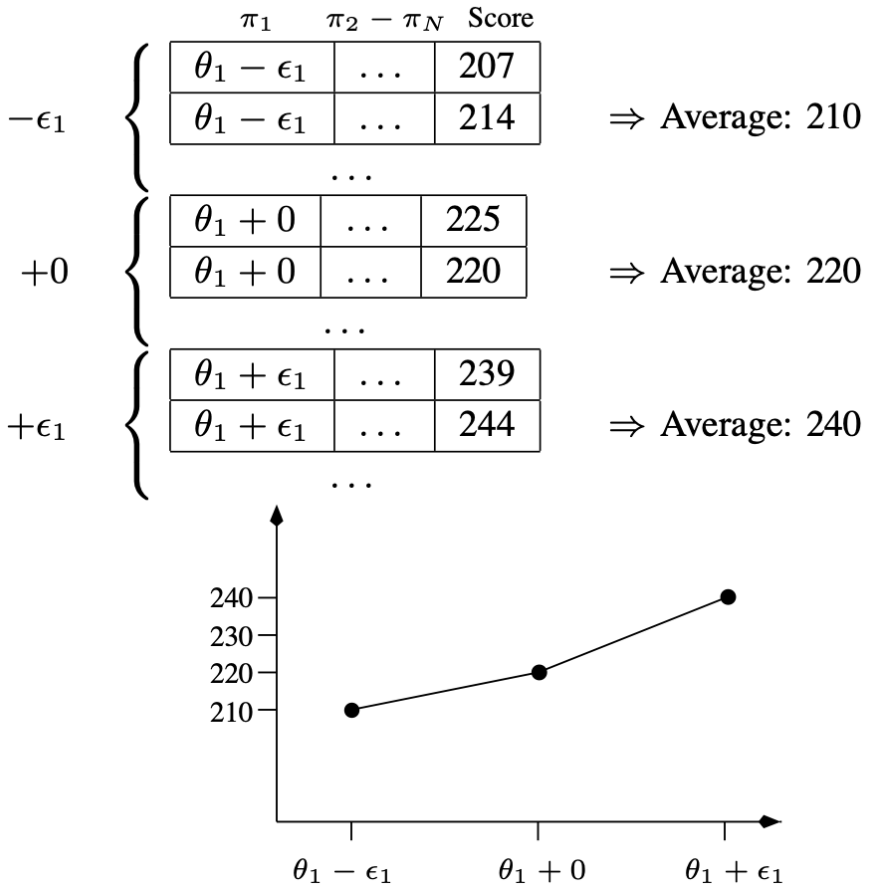
Gradient is estimated in parameter space, and then moved towards an optimum.



(Kohl und Stone 2004)

Example: Policy Gradient Approach

- Start with t random policies $R_1, \ldots, R_t$ near initial policiy π: $R_i = \theta_1 + \Delta_1, \ldots, \theta_N + \Delta_N$ with Δ_j chosen randomly from $+\epsilon_j, 0, -\epsilon_j$ (ϵ_j is a fixed value that is small relative to θ_j)
- Evaluate all policies on actual robot.
- Estimate gradient in each dimension: averaging over score variations wrt. variation in that parameter.



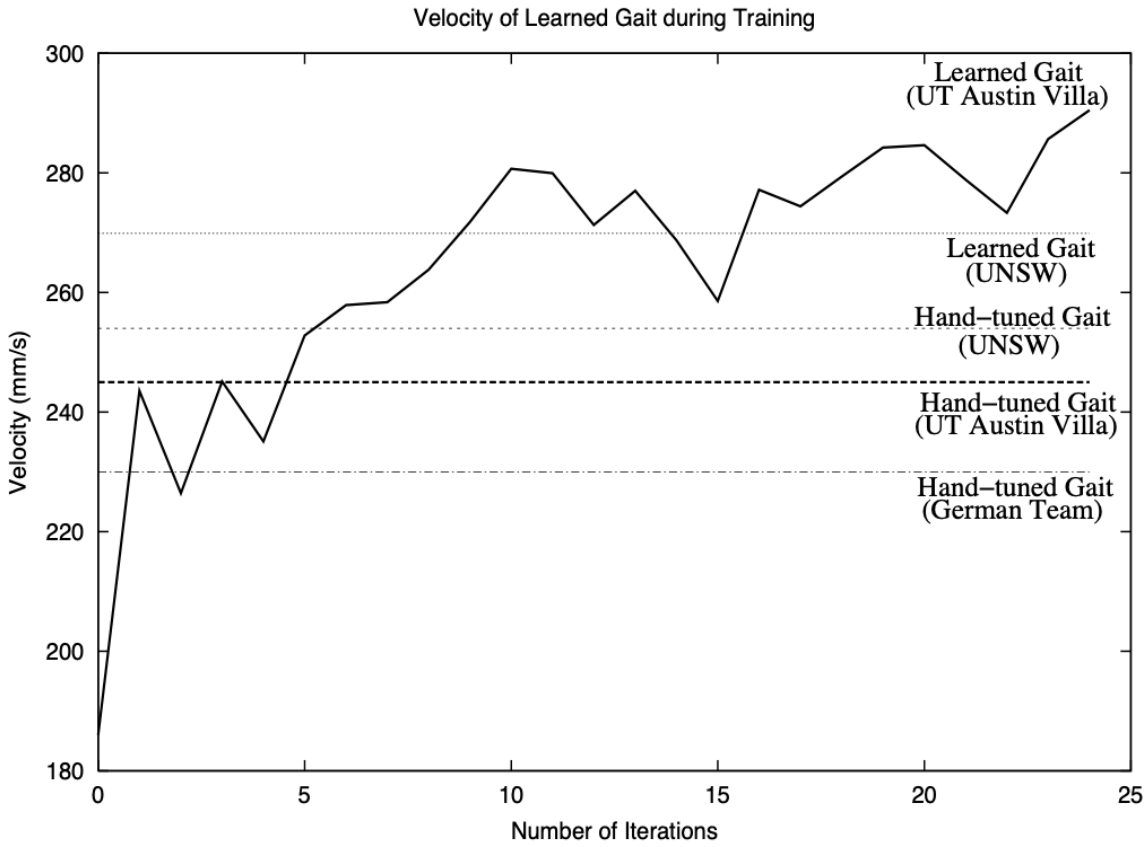
(Kohl und Stone 2004)

Example: Results – Learning of Gaits

Use $t = 15$ policies per iteration.

As there was significant noise in each evaluation, each set of parameters was evaluated three times.

Training was stopped after reaching a peak policy at 23 iterations, which amounted to just over 1000 field traversals in about 3 hours.

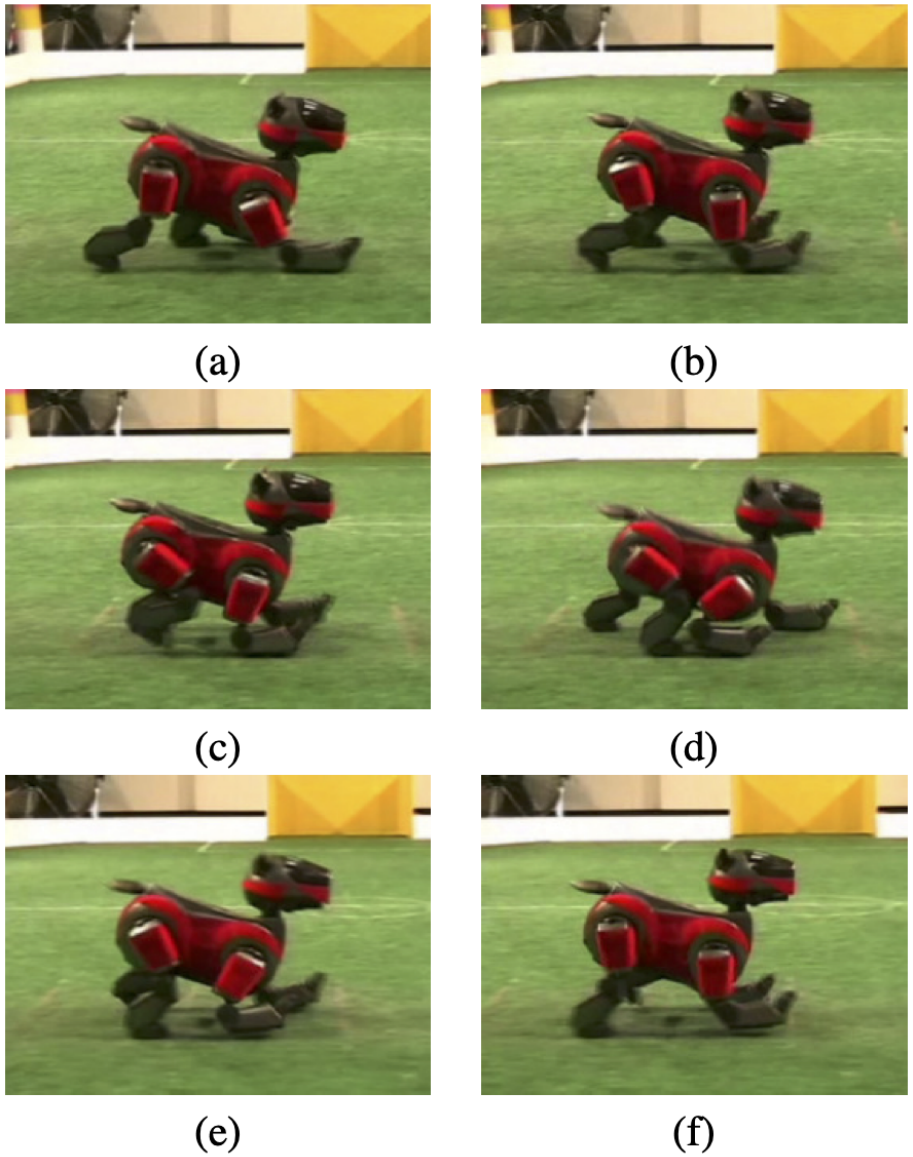


(Kohl und Stone 2004)

Aibo Performance

Velocity (given in mm/s) from different teams as of 2004.

Team	Hand-Tuned Gaits	Learned Gaits
CMU (2002)	200	
German Team	230	
UT Austin Villa	245	**291**
UNSW	**254**	
Hornby (1999)		170
UNSW		270

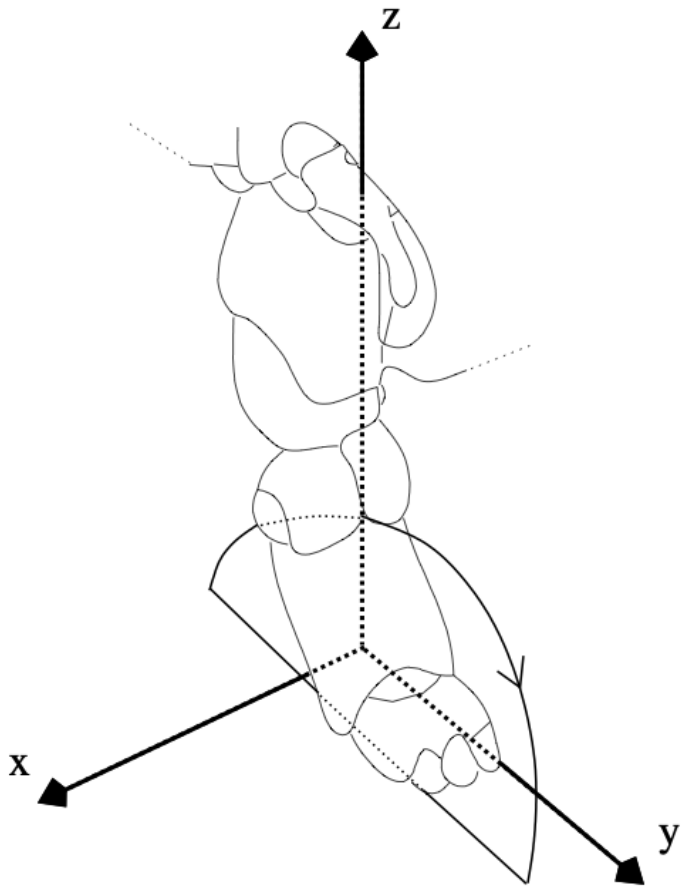


(Kohl und Stone 2004)

Results – AIBO Parameters for Stepping

Found Parameters

Parameter	Initial V.	ϵ	Best V.
Front locus:			
(height)	4.2	0.35	4.081
(x offset)	2.8	0.35	0.574
(y offset)	4.9	0.35	5.152
Rear locus:			
(height)	5.6	0.35	6.02
(x offset)	0.0	0.35	0.217
(y offset)	-2.8	0.35	-2.982
Locus length	4.893	0.35	5.285
Locus skew mult.	0.035	0.175	0.049
Front height	7.7	0.35	7.483
Rear height	11.2	0.35	10.843
Cycle time	0.704	0.016	0.679
Time on ground	0.5	0.05	0.430



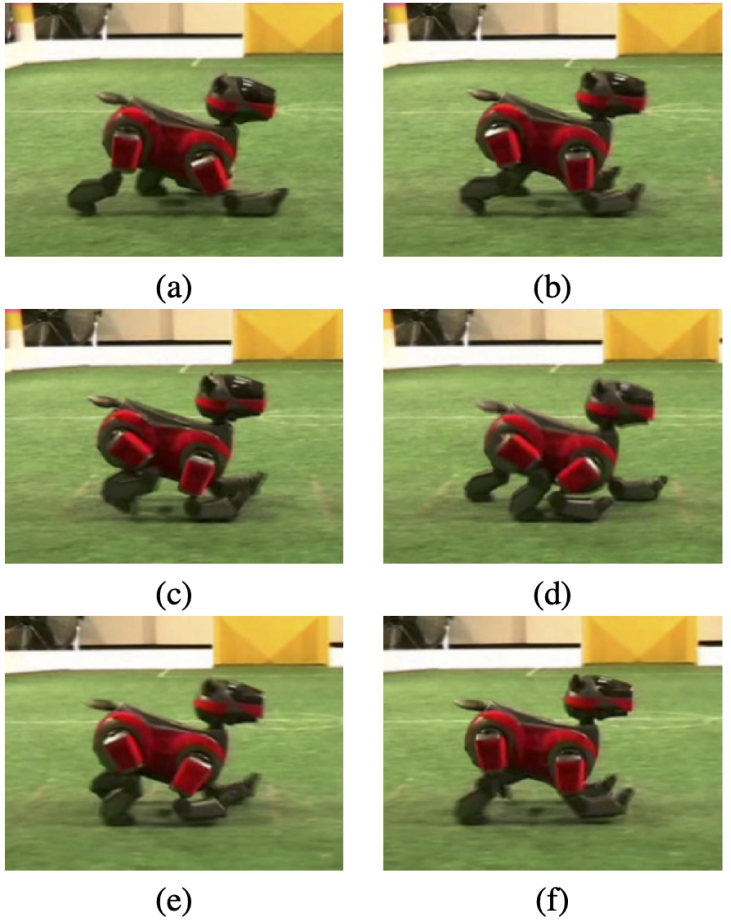
(Kohl und Stone 2004)

Example: Learned AIBO Gaits

Initial Gait

Learned Gait

Learned Gait Images



(Kohl und Stone 2004)

Recap – Gradients on parameterized policies

How to compute this gradient $\nabla_\theta J(\theta)$?

- Approximate stochastically.
- Assume policy π_θ is differentiable almost everywhere (e.g., neural net).

Reward

For average reward $\nabla_\theta J(\theta) = \nabla\theta\mathbb{E}_{\pi_\theta}(R)$

Remember: $\mathbb{E}(R)$ depend on θ because of policy, but also the state distribution is affected.

(Hasselt und Borsa 2021)

Policy Gradient Theorem – Contextual Bandits

Recap – Policy Learning in Multi-Armed Bandits

Consider action selection as a probability distribution:

- For each action: Consider an (estimated) preference $H_t(a)$ of that action which
- can be directly used to express a probability for selecting that action (as a soft-max distribution)

$$p(A_t = a) = \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}} = \pi_t(a)$$

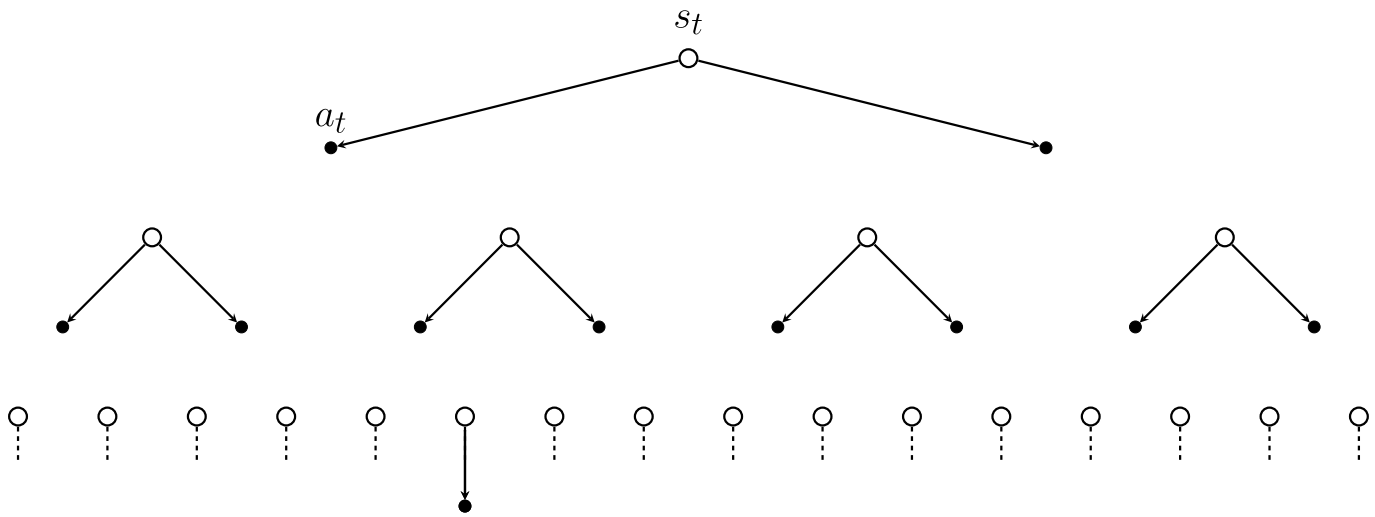
Recap – Bandits with States – Contextual bandits

How can we extend the multiarmed bandit approach to multiple states? Consider bandits with different states

- but episodes are still one step
- actions do not affect state transitions
- no long-term consequences

Then, we want to estimate:

$$q(s, a) = \mathbb{E}(R_{t+1}|S_t = s, A_t = a)$$



(Hasselt und Borsa 2021)

Recap – Monte Carlo Sampling

Gradient Bandit Algorithm

Learn / adapt the action preference function directly using stochastic gradient ascent:

$$H_{t+1}(A_t) = H_t(A_t) + \alpha(R_t - \bar{R}_t)(1 - \pi_t(A_t)), \quad \text{and}$$
$$H_{t+1}(a) = H_t(a) - \alpha(R_t - \bar{R}_t)\pi_t(a) \quad \text{for all } a \neq A_t$$

Following (Richard S. Sutton und Barto 2018).

Contextual Bandits Policy Gradient

For the one-step case (a contextual bandit) we use $J(\theta) = \mathbb{E}_{\pi_\theta}(R(S, A))$. (Expectation is over d (states) and π (actions) and, for now, d does not depend on π).

- We cannot sample R_{t+1} and then take a gradient as R_{t+1} is just a number and does not depend on θ!

Contextual Bandits Policy Gradient

For the one-step case (a contextual bandit) we use $J(\theta) = \mathbb{E}_{\pi_\theta}(R(S, A))$. (Expectation is over d (states) and π (actions) and, for now, d does not depend on π).

- We cannot sample R_{t+1} and then take a gradient as R_{t+1} is just a number and does not depend on θ!
- Instead, we use the identity:

$$\nabla_\theta \mathbb{E}_{\pi_\theta}(R(S, A)) = \mathbb{E}_{\pi_\theta}(R(S, A) \nabla_\theta \log \pi(A|S)).$$

The right-hand side gives an expected gradient that can be sampled.

Proof for Identity

Let $r_{sa} = \mathbb{E}(R(S,A)|S=s, A=a)$

$$\nabla_\theta \mathbb{E}_{\pi_\theta}(R(S,A)) = \nabla_\theta \sum_s d(s) \sum_a \pi_\theta(a|s) r_{sa}$$

$$= \sum_s d(s) \sum_a r_{sa} \nabla_\theta \pi_\theta(a|s)$$

$$= \sum_s d(s) \sum_a r_{sa} \pi_\theta(a|s) \frac{\nabla_\theta \pi_\theta(a|s)}{\pi_\theta(a|s)}, (\log f(x))' = \frac{f'(x)}{f(x)}$$

$$= \sum_s d(s) \sum_a r_{sa} \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s)$$

$$= \mathbb{E}_{d,\pi_\theta}(R(S,A) \nabla_\theta \log \pi_\theta(A|S))$$

(Hasselt und Borsa 2021)

Contextual Bandit Policy Gradient

$$\nabla_\theta \mathbb{E}_{\pi_\theta}(R(S,A)) = \mathbb{E}_{d,\pi_\theta}(R(S,A)\nabla_\theta \log \pi_\theta(A|S))$$

This is something we can sample!

- Our stochastic policy-gradient update is then

$$\theta_{t+1} = \theta_t + \alpha R_{t+1} \nabla_\theta \log \pi_{\theta_t}(A_t|S_t).$$

- In the expectation, this is following the actual gradient – so this is a pure (unbiased) stochastic gradient algorithm.
- Intuition: increase probability for actions with high rewards.

(Hasselt und Borsa 2021)

Policy gradients: reduce variance

Note that, in general

$$\mathbb{E}(b\nabla_\theta \log \pi(A_t|S_t)) = \mathbb{E}\Big(\sum_a \pi(a|S_t) b \nabla_\theta \log \pi(a|S_t)\Big), (\log f(x))' = \frac{f'(x)}{f(x)}$$

$$= \mathbb{E}\Big(b\nabla_\theta \sum_a \pi_\theta(a|S_t)\Big)$$

$$= \mathbb{E}(b\nabla_\theta 1) = 0$$

- This is true if b does not depend on the action (but it can depend on the state).
- Implies, we can subtract a baseline to reduce variance

$$\theta_{t+1} = \theta_t + \alpha(R_{t+1} - b(S_t))\nabla_\theta \log \pi_{\theta_t}(A_t|S_t).$$

(Hasselt und Borsa 2021)

Example: Softmax Policy

- Consider a softmax policy on action preferences $h(s, a)$ as an example
- Probability of action is proportional to exponentiated weight

$$\pi_\theta(a|s) = \frac{e^{h(s,a)}}{\sum_b e^{h(s,b)}}$$

- The gradient of the log probability is

$$\nabla_\theta \log \pi_\theta(A_t|S_t) = \underbrace{\nabla_\theta h(S_t, A_t)}_{\text{gradient of preference}} - \underbrace{\sum_a \pi_\theta(a|S_t) \nabla_\theta h(S_t, a)}_{\text{expected gradient of preference}}$$

(Hasselt und Borsa 2021)

Policy Gradient Theorem

Policy Gradient Theorem

The policy gradient approach also applies to (multi-step) MDPs.

- Replaces reward R with long-term return G_t or value $q_\pi(s, a)$
- There are actually two policy gradient theorems (R. S. Sutton u. a. 2000):
  - average return per episode
  - average reward per step

(Hasselt und Borsa 2021)

Policy gradient theorem (episodic)

PG Theorem

For any differentiable policy $\pi_\theta(s, a)$, let d_0 be the starting distribution over states in which we begin an episode.

Then, the policy gradient of $J(\theta) = \mathbb{E}(G_0 | S_0 \sim d_0)$ is

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\Big(\sum_{t=0}^{T} \gamma^t q_{\pi_\theta}(S_t, A_t) \nabla_\theta \log \pi_\theta(A_t | S_t) | S_0 \sim d_0 \Big)$$

where

$$\begin{aligned} q_\pi(s, a) &= \mathbb{E}_\pi(G_t | S_t = s, A_t = a) \\ &= \mathbb{E}_\pi(R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) | S_t = s, A_t = a) \end{aligned}$$

(Hasselt und Borsa 2021)

Policy gradients on trajectories

Observation / Important as step from Bandit to MDP case:

- Policy gradients do not need to know the MDP dynamics
- Kind of surprising: shouldn’t we know how the policy influences the states?

Reminder: Score function trick from the Bandit Case

Let $r_{sa} = \mathbb{E}(R(S,A)|S=s, A=s)$

$$\begin{aligned}
\nabla_\theta \mathbb{E}_{\pi_\theta}(R(S,A)) &= \nabla_\theta \sum_s d(s) \sum_a \pi_\theta(a|s) r_{sa} \\
&= \sum_s d(s) \sum_a r_{sa} \nabla_\theta \pi_\theta(a|s) \\
&= \sum_s d(s) \sum_a r_{sa} \pi_\theta(a|s) \frac{\nabla_\theta \pi_\theta(a|s)}{\pi_\theta(a|s)}, (\log f(x))' = \frac{f'(x)}{f(x)} \\
&= \sum_s d(s) \sum_a r_{sa} \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s) \\
&= \mathbb{E}_{d,\pi_\theta}(R(S,A) \nabla_\theta \log \pi_\theta(A|S))
\end{aligned}$$

(Hasselt und Borsa 2021)

Episodic policy gradients: proof 1

Consider trajectory $\tau = S_0, A_0, R_1, S_1, A_1, R_2, S_2, \ldots$ with return $G(\tau)$.

$$\nabla_\theta J_\theta(\pi) = \nabla_\theta \mathbb{E}(G(\tau)) = \mathbb{E}(G(\tau) \nabla_\theta \log p(\tau))$$

We used the score function "trick":

$$\begin{aligned}
\nabla_\theta \log p(\tau) &= \nabla_\theta \log \big(p(S_0)\pi(A_0|S_0)p(S_1|S_0,A_0)\pi(A_1|S_1)\ldots\big) \\
&= \nabla_\theta \big(\log p(S_0) + \log \pi(A_0|S_0) + \log p(S_1|S_0,A_0) + \log \pi(A_1|S_1) + \ldots\big) \\
&= \nabla_\theta \big(\log \pi(A_0|S_0) + \log \pi(A_1|S_1) + \ldots\big)
\end{aligned}$$

Therefore:

$$\nabla_\theta J_\theta(\pi) = \mathbb{E}_\pi\big(G(\tau) \nabla_\theta \sum_{t=0}^{T} \log \pi(A_t|S_t)\big)$$

(Hasselt und Borsa 2021)

Episodic policy gradients: proof 2

$$
\begin{aligned}
\nabla_\theta J_\theta(\pi) &= \mathbb{E}_\pi\Big(G(\tau) \sum_{t=0}^{T} \nabla_\theta \log \pi(A_t|S_t)\Big) \\
&= \mathbb{E}_\pi\Big(\sum_{t=0}^{T} G(\tau) \nabla_\theta \log \pi(A_t|S_t)\Big) \\
&= \mathbb{E}_\pi\Big(\sum_{t=0}^{T} \Big(\sum_{k=0?}^{T} \gamma^k R_{k+1}\Big) \nabla_\theta \log \pi(A_t|S_t)\Big) \\
&= \mathbb{E}_\pi\Big(\sum_{t=0}^{T} \Big(\sum_{k=t}^{T} \gamma^k R_{k+1}\Big) \nabla_\theta \log \pi(A_t|S_t)\Big)
\end{aligned}
$$

(Hasselt und Borsa 2021)

Episodic policy gradients: proof 3

$$
\begin{aligned}
\nabla_\theta J_\theta(\pi) &= \mathbb{E}_\pi\Big(\sum_{t=0}^{T}\Big(\sum_{k=t}^{T}\gamma^k R_{k+1}\Big)\nabla_\theta \log \pi(A_t|S_t)\Big) \\
&= \mathbb{E}_\pi\Big(\sum_{t=0}^{T}\Big(\gamma^t\sum_{k=t}^{T}\gamma^{k-t} R_{k+1}\Big)\nabla_\theta \log \pi(A_t|S_t)\Big) \\
&= \mathbb{E}_\pi\Big(\sum_{t=0}^{T}(\gamma^t G_t)\nabla_\theta \log \pi(A_t|S_t)\Big) \\
&= \mathbb{E}_\pi\Big(\sum_{t=0}^{T}\gamma^t q_\pi(S_t, A_t)\nabla_\theta \log \pi(A_t|S_t)\Big)
\end{aligned}
$$

(Hasselt und Borsa 2021)

Episodic policy gradients

$$\nabla_\theta J_\theta(\pi) = \mathbb{E}_\pi\Big(\sum_{t=0}^{T} \gamma^t q_\pi(S_t, A_t) \nabla_\theta \log \pi(A_t|S_t)\Big)$$

- We can sample this, given a whole episode.
- Typically, people pull out the sum, and split up this into separate gradients, e.g., $\Delta\theta_t = \gamma^t G_t \nabla_\theta \log \pi(A_t|S_t)$ such that $\mathbb{E}_\pi(\sum_t \Delta\theta_t) = \nabla_\theta J_\theta(\pi)$
- Typically, people ignore the γ^t term, use $\Delta\theta_t = G_t \nabla_\theta \log \pi(A_t|S_t)$
- This is a (doable) approximation as we pretend on each step that we could have started an episode in that state instead (alternative view: as a slightly biased gradient)

(Hasselt und Borsa 2021)

Policy gradient theorem (average reward)

PG Theorem (Average Reward)

For any differentiable policy $\pi_\theta(s, a)$, the policy gradient of $J(\theta) = \mathbb{E}(R|\pi)$ is

$$\nabla_\theta J(\theta) = \mathbb{E}_\pi\Big(q_{\pi_\theta}(S_t, A_t)\nabla_\theta \log \pi_\theta(A_t|S_t)\Big)$$

where

$q_\pi(s, a) = \mathbb{E}_\pi(R_{t+1} - \rho + q_\pi(S_{t+1}, A_{t+1})|S_t = s, A_t = a)$

$\rho = \mathbb{E}_\pi(R_{t+1})$, Note: global average, not conditioned on state or action

(Expectation is over both states and actions)

(Hasselt und Borsa 2021)

Policy gradient theorem (average reward) 2

Alternatively (but equivalently):

For any differentiable policy $\pi_\theta(s, a)$, the policy gradient of $J(\theta) = \mathbb{E}(R|\pi)$ is

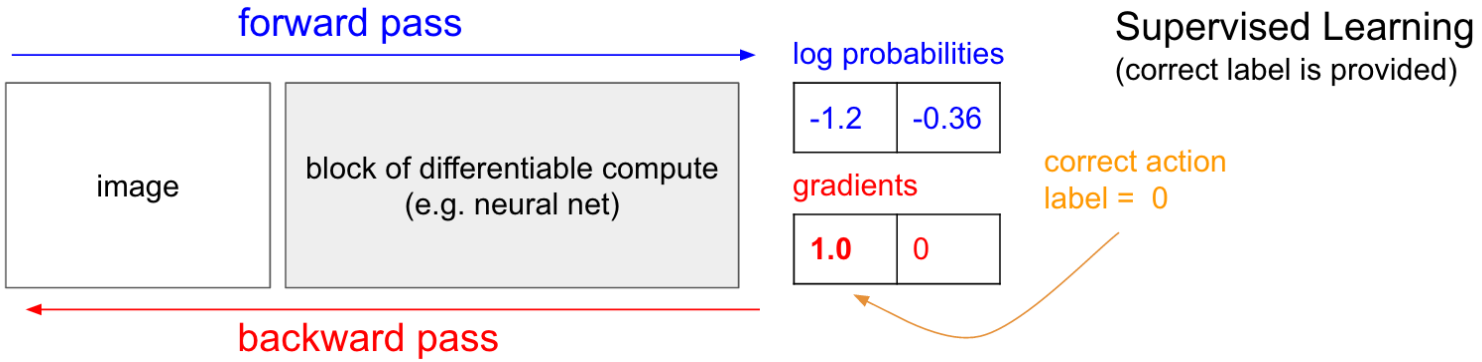
$$\nabla_\theta J(\theta) = \mathbb{E}_\pi \left(R_{t+1} \sum_{n=0}^{\infty} \nabla_\theta \log \pi_\theta(A_{t-n}|S_{t-n}) \right)$$

(Expectation is over both states and actions)

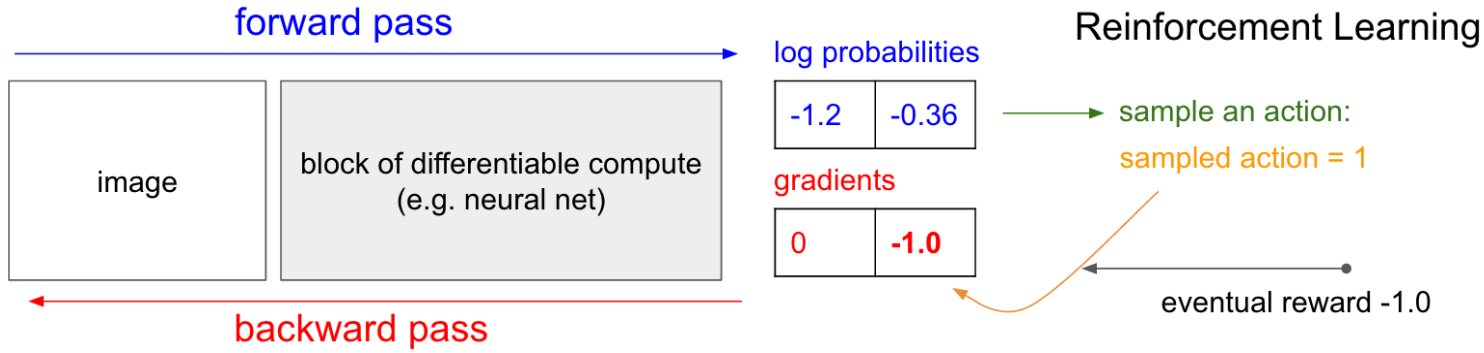
(Hasselt und Borsa 2021)

Policy Gradient Training

For comparison: Training of a NN using supervised learning:



Training a policy network in reinforcement learning:



(Karpathy 2016), for more information and overview of PG algorithms see (Weng 2018).

Policy Gradient Theorem – General Outline

Problem: Estimate f and optimize using gradient ascent. How to estimate the gradient ?

$$\nabla_\theta E_x[f(x)] = \nabla_\theta \sum_x p(x) f(x) \qquad \text{definition of expectation}$$

$$= \sum_x \nabla_\theta p(x) f(x) \qquad \text{swap sum and gradient}$$

$$= \sum_x p(x) \frac{\nabla_\theta p(x)}{p(x)} f(x) \qquad \text{both multiply and divide by } p(x)$$

$$= \sum_x p(x) \nabla_\theta \log p(x) f(x) \qquad \text{use the fact that } \nabla_\theta \log(z) = \frac{1}{z} \nabla_\theta z$$

$$= E_x[f(x) \nabla_\theta \log p(x)] \qquad \text{definition of expectation}$$

$p(x) = p(a \mid \text{Image})$ will be our policy - note: gradient comes from policy.

(Karpathy 2016), for more information and overview of PG algorithms see (Weng 2018).

Actor Critics

Recall – Policy gradients: reduce variance

Note that, in general

$$\mathbb{E}(b\nabla_\theta \log \pi(A_t|S_t)) = \mathbb{E}\Big(\sum_a \pi(a|S_t) b \nabla_\theta \log \pi(a|S_t)\Big), (\log f(x))' = \frac{f'(x)}{f(x)}$$

$$= \mathbb{E}\Big(b\nabla_\theta \sum_a \pi_\theta(a|S_t)\Big)$$

$$= \mathbb{E}(b\nabla_\theta 1) = 0$$

- This is true if b does not depend on the action (but it can depend on the state).
- Implies, we can subtract a baseline to reduce variance

$$\theta_{t+1} = \theta_t + \alpha(R_{t+1} - b(S_t))\nabla_\theta \log \pi_{\theta_t}(A_t|S_t).$$

(Hasselt und Borsa 2021)

Policy gradients: reduce variance

- Recall $\mathbb{E}_\pi(b(S_t)\nabla \log \pi(A_t|S_t)) = 0$ for any $b(S_t)$ that does not depend on A_t
- A common baseline is the value function $v_\pi(S_t)$

$$\nabla_\theta J_\theta(\pi) = \mathbb{E}_\pi\Big(\sum_{t=0}^{T} \gamma^t (q_\pi(S_t, A_t) - v_\pi(S_t)) \nabla_\theta \log \pi(A_t|S_t)\Big)$$

- Typically, we estimate $v_w(s) \approx v_\pi(s)$ explicitly, and sample $q_\pi(S_t, A_t) \approx G_t$
- We can minimise variance further by bootstrapping, e.g., $G_t = R_{t+1} + \gamma v_w(S_{t+1})$

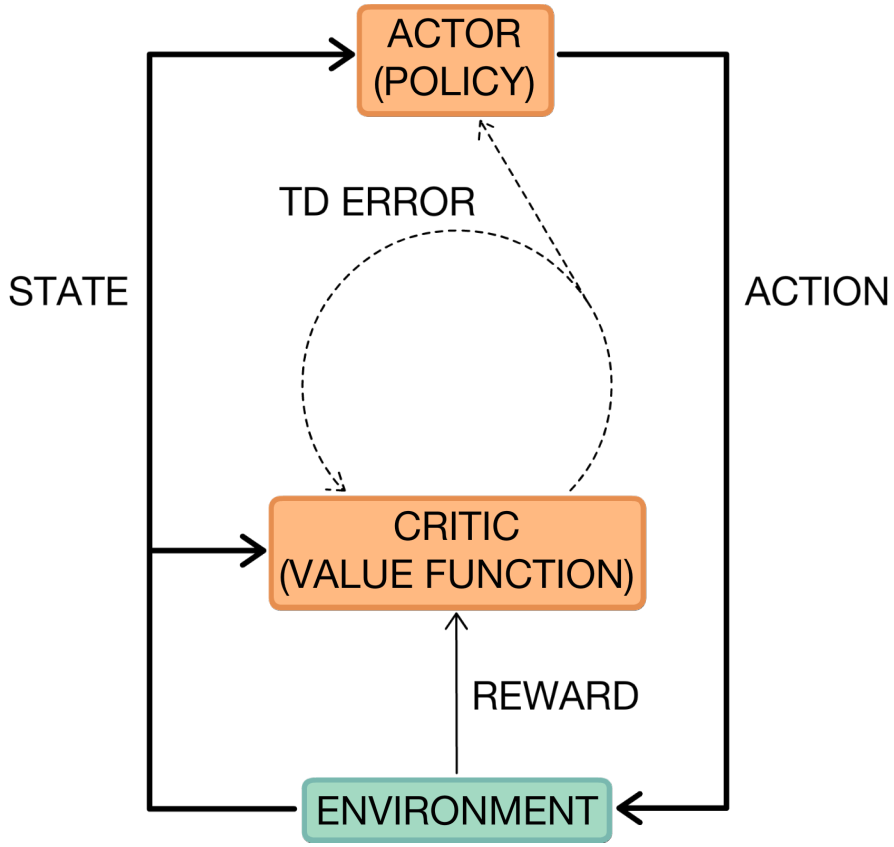
(Hasselt und Borsa 2021)

Actor-Critic Method

Combination of both methods is used in **Actor-Critic** approaches, learning both:

- an actor policy allowing to use Policy Gradients and
- a value-based function that allows to do the updates during each timestep using bootstrapping.

A critic is a value function, learnt via policy evaluation: What is the value v_{π_θ} of policy π_θ for current parameters θ?



(Arulkumaran u. a. 2017)

Actor-Critic

Critic – Update parameters $\mathbf{w}$ of $v_{\mathbf{w}}$ by TD (e.g., one-step) or MC

Actor – Update θ by policy gradient

(Hasselt und Borsa 2021)

function One-step Actor Critic

Initialize s, θ

for $t = 0, 1, \ldots$ **do**

  Sample $A_t \sim \pi_\theta(S_t)$

  Sample R_{t+1} and S_{t+1}

  $\Delta_t = R_{t+1} + \gamma v_{\vec{w}}(S_{t+1}) - v_{\vec{w}}(S_t)$, one step TD-error or advantage

  $\vec{w} \leftarrow \vec{w} + \beta \Delta_t \nabla_{\vec{w}} v_{\vec{w}}(S_t)$, TD(0)

  $\theta \leftarrow \theta + \alpha \Delta_t \nabla_\theta \log \pi_\theta(A_t | S_t)$, Policy gradient update (ignoring γ^t)

end

Policy gradient variations

- Many extensions and variants exist
- Take care: bad policies lead to bad data
- This is different from supervised learning (where learning and data are independent)

(Hasselt und Borsa 2021)

Increasing robustness with trust regions

One way to increase stability is to **regularise**.

- A popular method is to limit the difference between subsequent policies, e.g., use the Kullbeck-Leibler divergence:

$$KL(\pi_{\text{old}} \| \pi_\theta) = \mathbb{E}\Big(\int \pi_{\text{old}}(a|S) \log \frac{\pi_\theta(a|S)}{\pi_{\text{old}}(a|S)} \mathrm{d}a \Big)$$

  (Expectation is over states)
- A divergence is like a distance between distributions.
- Then maximise $J(\theta) - \eta KL(\pi_{\text{old}} \| \pi_\theta)$, for some hyperparameter η.

(Hasselt und Borsa 2021) and see as examples, TRPO (Schulman u. a. 2015) or PPO (Schulman u. a. 2017).

Continuous Action Spaces

Continuous actions

- Pure value-based RL can be non-trivial to extend to continuous action spaces
  - How to approximate $q(s, a)$?
  - How to compute $\max_a q(s, a)$?
- When directly updating the policy parameters, continuous actions are easier
- Most algorithms discussed today can be used for discrete and continuous actions

Note: exploration in high-dimensional continuous spaces can be challenging.

(Hasselt und Borsa 2021)

Example: Gaussian policy

As example, consider a Gaussian policy,

- for example: mean is some function of state $\mu_\theta(s)$
- For simplicity, lets consider fixed variance of σ^2 (can be parametrized as well)
- Policy is Gaussian, $A_t \sim \mathcal{N}(\mu_\theta(S_t), \sigma^2)$ (here μ_θ is the mean — not to be confused with the behaviour policy!)
- The gradient of the log of the policy is then

$$\nabla_\theta \log \pi_\theta(s, a) = \frac{A_t - \mu_\theta(S_t)}{\sigma^2} \nabla \mu_\theta(s)$$

- This can be used, for instance, in REINFORCE / actor critic.

(Hasselt und Borsa 2021)

Example: Policy gradient with Gaussian policy 2

Gaussian policy gradient update:

$$\begin{aligned}\theta_{t+1} &= \theta_t + \beta(G_t - v(S_t))\nabla_\theta \log \pi_\theta(A_t|S_t) \\ &= \theta_t + \beta(G_t - v(S_t))\frac{A_t - \mu_\theta(S_t)}{\sigma^2}\nabla \mu_\theta(S_t)\end{aligned}$$

Intuition: if return was high, move $\mu_\theta(S_t)$ toward A_t

(Hasselt und Borsa 2021)

Gradient ascent on value

Policy gradients work well, but do not strongly exploit the critic

If values generalise well, perhaps we can rely on them more?

1. Estimate $q_{\mathbf{w}} \approx q_\pi$, e.g., with Sarsa
2. Define deterministic actor: $A_t = \pi_\theta(S_t)$
3. Improve actor (policy improvement) by gradient ascent on the value:

$$\Delta\theta \propto \frac{\partial Q_\pi(s,a)}{\partial\theta} = \frac{\partial Q_\pi(s,\pi_\theta(S_t))}{\partial\pi_\theta(S_t)} \frac{\partial\pi_\theta(S_t)}{\partial\theta}$$

Known nowadays as “Deterministic policy gradient” (DPG) (Silver u. a. 2014) and it’s a form of policy iteration.

(Hasselt und Borsa 2021)

Comparison: Advantages of Methods

Value-based Methods

- Simple – can be realized as tables, still convergence guarantees.
- Efficiency and Speed – bootstraping speeds up learning

Policy Gradient Methods

- Applicable in large and continuous action spaces
- Employ stochastic policies

Further considerations:

- Do you want to access directly a value, e.g. for other methods?
- The state representation of the problem might lends itself more easily to either a value function or a policy function.

References

Arulkumaran, Kai, Marc P. Deisenroth, Miles Brundage, und Anil A. Bharath. 2017. „Deep Reinforcement Learning: A Brief Survey“. *IEEE Signal Processing Magazine* 34 (6).

Hasselt, Hado van, und Diana Borsa. 2021. „Reinforcement Learning Lecture Series 2021“. https://www.deepmind.com/learning-resources/reinforcement-learning-lecture-series-2021.

Karpathy, Andrej. 2016. „Deep Reinforcement Learning: Pong from Pixels“. http://karpathy.github.io/2016/05/31/rl/.

Kohl, Nate, und Peter Stone. 2004. „Policy Gradient Reinforcement Learning for Fast Quadrupedal Locomotion.“ In *ICRA*, 2619–24. IEEE.

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, u. a. 2015. „Human-level control through deep reinforcement learning“. *Nature* 518 (7540): 529–33. http://dx.doi.org/10.1038/nature14236.

Schulman, John, Sergey Levine, Philipp Moritz, Michael I. Jordan, und Pieter Abbeel. 2015. „Trust Region Policy Optimization“. *CoRR* abs/1502.05477. http://arxiv.org/abs/1502.05477.

Schulman, John, Filip Wolski, Prafulla Dhariwal, Alec Radford, und Oleg Klimov. 2017. „Proximal policy optimization algorithms“. *arXiv preprint arXiv:1707.06347*.

Silver, David. 2015. „UCL Course on RL UCL Course on RL UCL Course on Reinforcement Learning“. http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html.

Silver, David, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, und Martin Riedmiller. 2014. „Deterministic Policy Gradient Algorithms“. In *Proceedings of the 31st International Conference on Machine Learning*, herausgegeben von Eric P. Xing und Tony Jebara, 32:387–95. Proceedings of Machine Learning Research 1. Bejing, China: PMLR. https://proceedings.mlr.press/v32/silver14.html.

Sutton, R. S., D. Mcallester, S. Singh, und Y. Mansour. 2000. „Policy gradient methods for reinforcement learning with function approximation“. In *Advances in Neural Information Processing Systems 12*, 12:1057–63. MIT Press.

Sutton, Richard S., und Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction*. Second. The MIT Press.

Weng, Lilian. 2018. „Policy Gradient Algorithms“. https://lilianweng.github.io/lil-log/2018/04/08/policy-gradient-algorithms.html.