Reinforcement Learning

—

Exercise 10: Multi-armed Bandits

30.06.2023

Nico Meyer

Fraunhofer IIS

Fraunhofer-Institut für Integrierte Schaltungen IIS

Exercise Sheet 9

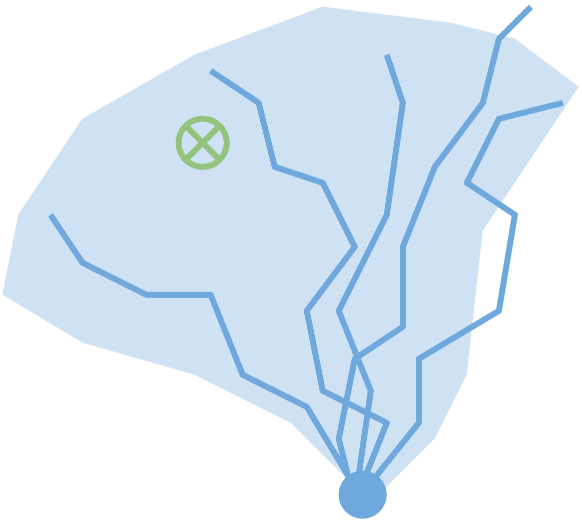
Cross-Entropy Method (CEM)

Online Planning with Continuous Actions

Cross Entropy Maximization

Sampling methods → Cross Entropy Maximization

- Gradient-free
- Population-based (like e.g., Genetic Algorithms), can escape local optima



Stulp et al (2012). Path Integral Policy Improvement with Covariance Matrix Adaptation.

Use Gaussian to sample around current parameter mean

Cross-Entropy Method (one iteration)

$$\boldsymbol{\theta}_{k=1\ldots K} \sim \mathcal{N}(\boldsymbol{\theta}, \Sigma) \quad \textbf{sample} \quad (1)$$

$$J_k = J(\boldsymbol{\theta}_k) \quad \textbf{eval.} \quad (2)$$

$$\boldsymbol{\theta}_{k=1\ldots K} \leftarrow \texttt{sort } \boldsymbol{\theta}_{k=1\ldots K} \texttt{ w.r.t } J_{k=1\ldots K} \quad \textbf{sort} \quad (3)$$

$$\boldsymbol{\theta}^{new} = \sum_{k=1}^{K_e} \frac{1}{K_e} \boldsymbol{\theta}_k \quad \textbf{update} \quad (4)$$

$$\Sigma^{new} = \sum_{k=1}^{K_e} \frac{1}{K_e} (\boldsymbol{\theta}_k - \boldsymbol{\theta})(\boldsymbol{\theta}_k - \boldsymbol{\theta})^\intercal \quad \textbf{update} \quad (5)$$

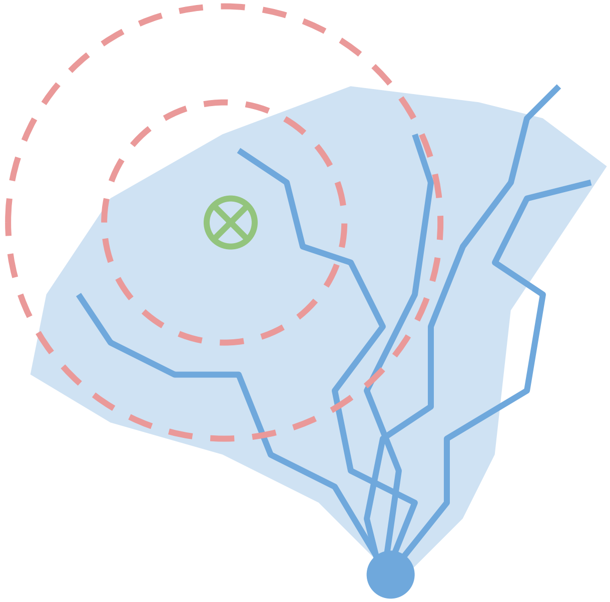
https://sites.google.com/view/mbrl-tutorial

Online Planning with Continuous Actions

Cross Entropy Maximization

Sampling methods → Cross Entropy Maximization

- Gradient-free
- Population-based (like e.g., Genetic Algorithms), can escape local optima



Stulp et al (2012). Path Integral Policy Improvement with Covariance Matrix Adaptation.

Evaluate **(using the model)** the sampled parameters and keep the top K samples

Cross-Entropy Method (one iteration)

$$\boldsymbol{\theta}_{k=1\ldots K} \sim \mathcal{N}(\boldsymbol{\theta}, \Sigma) \qquad \textbf{sample} \quad (1)$$

$$J_k = J(\boldsymbol{\theta}_k) \qquad \textbf{eval.} \quad (2)$$

$$\boxed{\boldsymbol{\theta}_{k=1\ldots K} \leftarrow \texttt{sort } \boldsymbol{\theta}_{k=1\ldots K} \texttt{ w.r.t } J_{k=1\ldots K} \qquad \textbf{sort} \quad (3)}$$

$$\boldsymbol{\theta}^{new} = \sum_{k=1}^{K_e} \frac{1}{K_e} \boldsymbol{\theta}_k \qquad \textbf{update} \quad (4)$$

$$\Sigma^{new} = \sum_{k=1}^{K_e} \frac{1}{K_e} (\boldsymbol{\theta}_k - \boldsymbol{\theta})(\boldsymbol{\theta}_k - \boldsymbol{\theta})^\intercal \qquad \textbf{update} \quad (5)$$

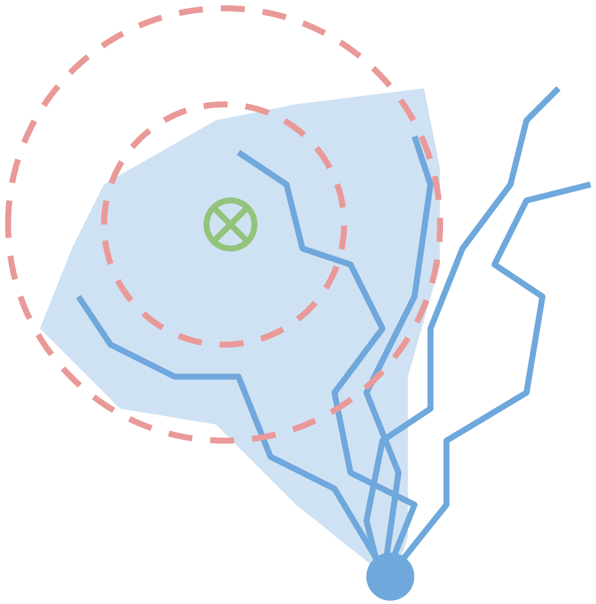
https://sites.google.com/view/mbrl-tutorial

Online Planning with Continuous Actions

Cross Entropy Maximization

Sampling methods → Cross Entropy Maximization

- Gradient-free
- Population-based (like e.g., Genetic Algorithms), can escape local optima



Re-fit the sampling Gaussian
using the top K samples

Cross-Entropy Method (one iteration)

$$\boldsymbol{\theta}_{k=1\ldots K} \sim \mathcal{N}(\boldsymbol{\theta}, \Sigma) \qquad \textbf{sample} \quad (1)$$

$$J_k = J(\boldsymbol{\theta}_k) \qquad \textbf{eval.} \quad (2)$$

$$\boldsymbol{\theta}_{k=1\ldots K} \leftarrow \texttt{sort}\ \boldsymbol{\theta}_{k=1\ldots K}\ \texttt{w.r.t}\ J_{k=1\ldots K} \qquad \textbf{sort} \quad (3)$$

$$\boldsymbol{\theta}^{new} = \sum_{k=1}^{K_e} \frac{1}{K_e} \boldsymbol{\theta}_k \qquad \textbf{update} \quad (4)$$

$$\Sigma^{new} = \sum_{k=1}^{K_e} \frac{1}{K_e} (\boldsymbol{\theta}_k - \boldsymbol{\theta})(\boldsymbol{\theta}_k - \boldsymbol{\theta})^{\intercal} \qquad \textbf{update} \quad (5)$$

https://sites.google.com/view/mbrl-tutorial

Stulp et al (2012). Path Integral Policy Improvement with Covariance Matrix Adaptation.

Cross-Entropy Method (CEM)

Pseudocode

Initialize $\mu \in \mathbb{R}^d, \sigma \in \mathbb{R}^d$
for iteration $= 1, 2, \ldots$ **do**
  Collect n samples of $\theta_i \sim N(\mu, \text{diag}(\sigma))$
  Perform a noisy evaluation $R_i \sim \theta_i$
  Select the top $p\%$ of samples (e.g. $p = 20$), which we'll
    call the **elite set**
  Fit a Gaussian distribution, with diagonal covariance,
    to the elite set, obtaining a new μ, σ.
end for
Return the final μ.

MLSS 2016 on Deep Reinforcement Learning by John Schulman

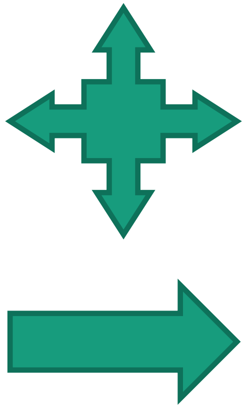
Fraunhofer IIS

Exploration vs. Exploitation

Exploration vs. Exploitation

Why and How

- Both definitions stem from the same problem:
  - **Exploration**: do things you haven’t done before (in the hopes of getting even higher reward) → increase knowledge
  - **Exploitation**: do what you know to yield highest reward → maximize performance based on knowledge



Multi-armed bandits (1-step stateless RL problems)	Contextual bandits (1-step RL problems)	Small, finite MDPs (e.g., tractable planning, model-based RL)	Large, infinite MDPs (e.g., continuous spaces)

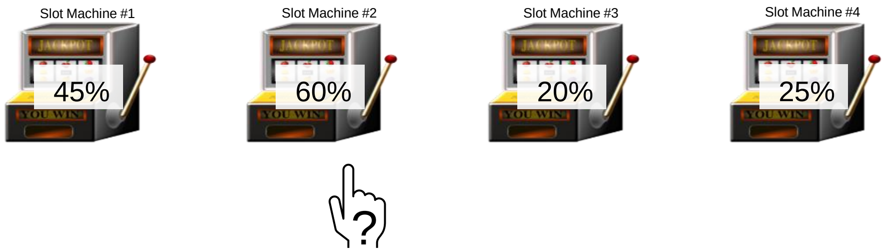
theoretically tractable ←→ theoretically intractable

(illustration adapted from Sergey Levine’s CS285 class from UC Berkeley)

Exploration vs. Exploitation

Multi-Armed Bandits and Regret

- The multi-armed-bandit problem is a classic problem used to study the exploration vs. exploitation dilemma
- Imagine you are in a casino with multiple slot machines, each configured with an unknown reward probability:



- Under the assumption of an infinite number of trials:

→ ***What is the best strategy to achieve highest long-term rewards?***

- Our loss function is the total regret we might have by not select the optimal action up to the time step T:

what we did

per-action regret

$$\mathcal{L}_T = \mathbb{E}\left[\sum_{t=1}^{T}\left(\theta^* - Q(a_t)\right)\right] = \sum_{a\in\mathcal{A}} N_T(a)\Delta_a$$

what we should have been doing

action-selection counter

Fraunhofer IIS

Exploration vs. Exploitation

Straightforward but usually bad: Greedy or ε-greedy

- Greedy may select a suboptimal action forever
  → Greedy has hence linear expected total regret

- ϵ-greedy continues to explore forever
  - with probability $1 - \epsilon$ it selects $a = \arg\max_{a \in \mathcal{A}} Q_T(a)$
  - with probability ϵ it selects a random action

- Will hence continue to select all suboptimal actions with (at least) a probability of $\frac{\epsilon}{|\mathcal{A}|}$
  → ϵ-greedy, with a constant ϵ has a linear expected total regret

- **Option #1: decrease ϵ over course of training might work**
  - It is not easy to tune the parameters

- **Option #2: be optimistic with options of high uncertainty**
  - Prefer actions for which you do not have a confident value estimation yet
    → Those have a great potential to be high-rewarding!
  - This idea is called **Upper Confidence Bounds**

Exploration vs. Exploitation

Upper Confidence Bounds (UCB1)

- Idea: estimate an upper confidence $U_t(a)$ for each action value, such that with a high probability we satisfy

$$Q(a) \leq \hat{Q}_t(a) + U_t(a)$$

- Next, we select the action that maximizes the upper confidence bound:

$$a_t^{UCB} = \arg\max_{a \in \mathcal{A}} [Q_t(a) + U_t(a)]$$

Large $N_t(a)$ → small bound $U_t(a)$ *(estimated value is certain/accurate)*

Small $N_t(a)$ → large bound $U_t(a)$ *(estimated value is uncertain)*

- The vanilla **UCB1** algorithm uses $p = t^{-4}$:

$$U_t(a) = \sqrt{\frac{2 \log t}{N_t(a)}} \quad \text{and} \quad a_t^{UCB} = \arg\max_{a \in \mathcal{A}} Q(a) + \sqrt{\frac{2 \log t}{N_t(a)}}$$

Derived from Hoeffding's Inequality:
$P(\mathbb{E}[X] \geq \bar{X}_t + u) \leq e^{-2tu^2}$

  - This ensures that we always keep exploring
  - But we select the optimal action much more often as $t \rightarrow \infty$

Exploration vs. Exploitation

Probability Matching via Thompson Sampling

We can also try the idea of directly sampling the action

- Select action a according to probability that a is the optimal action (given the history of everything we observed so far):

$$\pi_t(a|h_t) = P[Q(a) > Q(a'), \forall a' \neq a|\, h_t]$$
$$= \mathbb{E}_{r|h_t}\left[\mathbb{I}\left(a = \arg\max_{a \in \mathcal{A}} Q(a)\right)\right]$$

Probability matching via Thompson Sampling:

1. Assume $Q(a)$ follows a Beta distribution for the Bernoulli bandit
   - As $Q(a)$ is the success probability of θ
   - Beta(α, β) is within $[0,1]$,and α and β relate to the counts of success/failure
2. Initialize prior (e.g., $\alpha = \beta = 1$ or something different/what we think it is)
3. At each time step t we sample an expected reward $\hat{Q}(a)$ from the prior Beta(α_i, β_i) for every action
   - We select and execute the best action among the samples: $a_i^{TS} = \arg\max_{a \in \mathcal{A}} \hat{Q}(a)$
4. With the newly observed experience we update the Beta distribution:

$$\alpha_i \leftarrow \alpha_i + r_i \mathbb{I}[a_t^{TS} = a_i]$$
$$\beta_i \leftarrow \beta_i + (1 - r_i)\mathbb{I}[a_t^{TS} = a_i]$$

Exercise Sheet 10

Bandits

Fraunhofer

IIS

Fraunhofer-Institut für Integrierte Schaltungen IIS

Thank you for your attention!