Bayesian Optimization in Reinforcement Learning

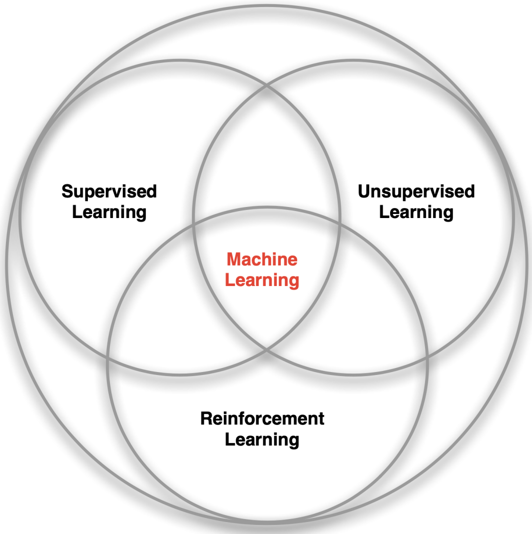
Kushagra Khatwani
Supervised by: Prabuchandran K J, Tejas Bodas

Indian Institute of Technology Dharwad

April 12, 2022

Overview

Branches of Machine Learning



Overview

Reinforcement Learning

What is different in reinforcement learning from other branches of machine learning?

- There is no supervisor, only a reward signal
- Feedback is delayed, not instantaneous
- Time matters (sequential not i.i.d data)
- Agent's choice of action affect future data and scenarios

Examples to motivate Reinforcement Learning

- Humanoid robot walk
- Playing Atari games better than humans
- Self-driving cars
- Alpha Zero

Overview

Basic Terminology

Agent

The component that decides which action to take

Basic Terminology

Agent

The component that decides which action to take

Environment

The component which provides the agent with observations

Basic Terminology

Agent

The component that decides which action to take

Environment

The component which provides the agent with observations

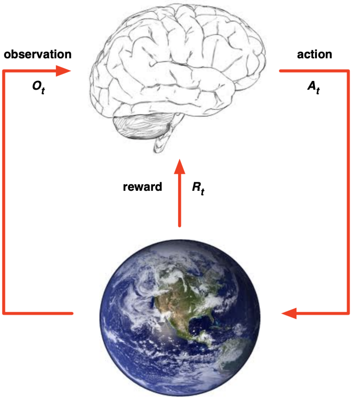
Rewards

- Scalar feedback signal
- Indicates how well the agent is perfoming in the environment

Agent and Environment

At each time step t:

- Agent
  - Execute action a_t
  - Receive observation O_t
  - Receive scalar reward R_t
- Environment
  - Receive action a_t
  - Provide observation O_{t+1}
  - Give scalar reward R_{t+1}



Types of Environments

Types of Environments

Fully Observable Environments

- agent directly observes environment state
- This is what we call Markov Decision Process(MDP)

Types of Environments

Fully Observable Environments

- agent directly observes environment state
- This is what we call Markov Decision Process(MDP)

Partially Observable Environments

- agent indirectly observes environment state
- This is what we call Partially Observable Markov Decision Process(POMDP)

RL Agent

RL agent may include one or more of the below components:

- Policy
- Value function
- Model

Policy

- It is a mapping from state to actions
- Policy determines the agent's behaviour
- Example:
  - Deterministic policy: $a = \pi(s)$
  - Stochastic policy: $\pi(a|s) = P(A_t = a | S_t = s)$

Value function

- Value function is a prediction of future rewards
- Is usually used to evaluate the states

$$V_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots | S_t = s]$$

Model

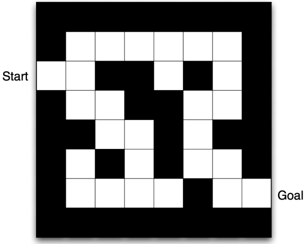
- Model predicts what the environment will do next depending on agent's decision
- P for action prediction
- R for immediate rewards

$$\mathbb{P}^{a}_{ss'} = P[S_{t+1} = s^{'} | S_t = s, A_t = a]$$

$$\mathbb{R}^{a}_{ss'} = \mathbb{E}[R_{t+1} | S_t = s, A_t = a]$$

Overview

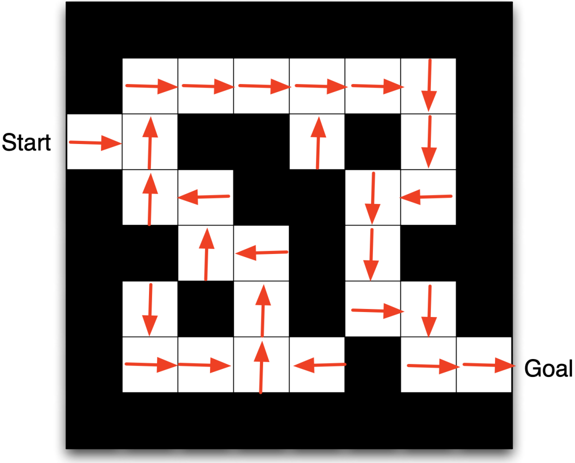
Maze Example



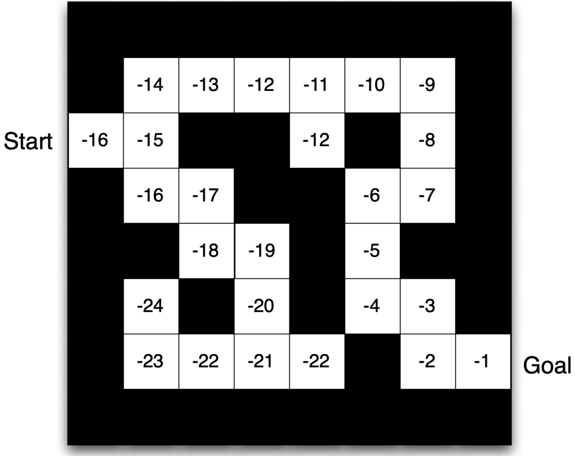
At each time step t:

- Rewards: -1 per timestep
- Actions: up,down,left,right
- State: Agent's location

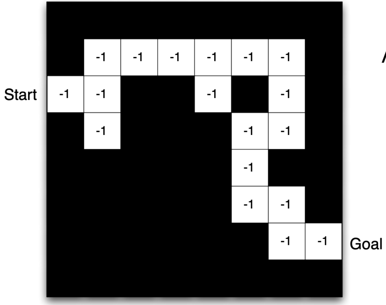
Maze Example: Policy



Maze Example: Value Function



Maze Example: Model

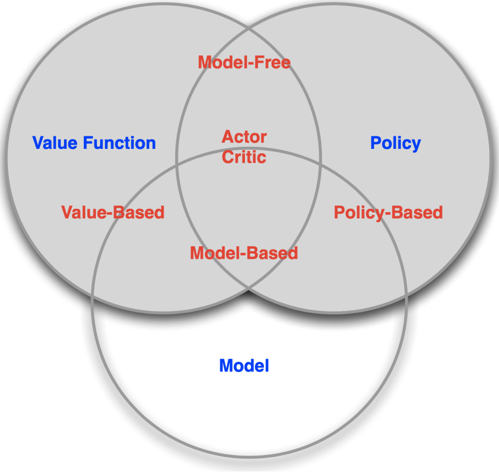


At each time step t:

- Grid Layout represents transition model $\mathbb{P}^a_{ss'}$
- Numbers inside cell represent immediate reward $\mathbb{R}^a_s$ for each state s.

Overview

Division of RL agent



Overview

Policy based methods

- Goal:Given a policy $\pi_\theta(s, a)$ parameterised by θ, find best θ
- can be treated as an optimisation problem
- find θ that will maximize $J(\theta)$
- Gradient Descent can be used as follows:

$$\triangle \, \theta = \alpha \nabla_\theta J(\theta)$$

where $\nabla_\theta J(\theta)$ is policy gradient

Policy based methods

Episodic RL tasks

- S, A are state space and action space respectively
- $P(j|i, a)$, is the probabilty of transition to state j given that we take action a from i
- $R(i, a, j)$, is the immediate reward we get if we enter j by taking action a from i
- $d(.)$ is the initial distribution for choosing first state
- In episodic RL there is a special state where $P(s^*|a, s^*) = 1$ and $R(s*, a, i) = 0$ known as the terminal state
- $\tau = (s_0, a_0, s_1, a_1, \ldots, a_{T-1}, s_T)$ produced by π_θ and $d(.)$ is known as a trajectory

Policy based methods

Applying on episodic RL

- Using earlier definitions:

$$\begin{aligned} J(\theta) &= \mathbb{E}_{\tau\sim\pi_\theta}[R_\tau] \\ &= \sum_\tau P(\tau|\theta)R_\tau \\ \nabla J(\theta) &= \sum_\tau \nabla P(\tau|\theta)R_\tau \\ &= \sum_\tau P(\tau|\theta)\nabla log(P(\tau|\theta))R_\tau \\ &= \mathbb{E}_{\tau\sim\pi_\theta}[\nabla log(P(\tau|\theta))R_\tau] \end{aligned}$$

Policy based methods

Applying on episodic RL

Now,

$$
\begin{aligned}
P(\tau|\theta) &= d(s_0)\prod_{t=1}^{T} P(s_t|s_{t-1}, a_{t-1})\pi_\theta(a_{t-1}|s_{t-1}) \\
\nabla J(\theta) &= \mathbb{E}_{\tau\sim\pi_\theta}[\nabla log(P(\tau|\theta))R_\tau] \\
&= \mathbb{E}_{\tau\sim\pi_\theta}[\sum_{t=1}^{T}\nabla log(\pi_\theta(a_{t-1}|s_{t-1})R_\tau] \\
\triangle\ \theta &= \alpha\nabla J(\theta)
\end{aligned}
$$

where α is the step size.

Overview

BO in RL

Why apply BO in RL?

Difficulties with previous approach:

- Convergence to local optimum
- choice of step size α
- can be slow and still not provide global optima

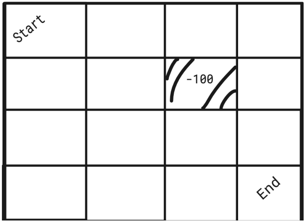
BO in RL

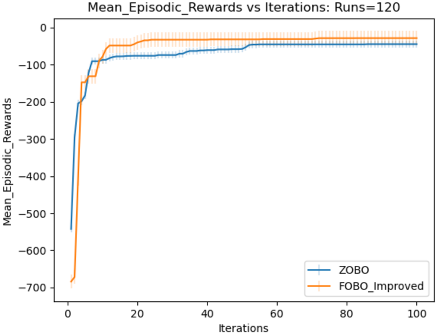
BORL settings

- Policy treated as evaluation points for BO black box
- Output of black box is the expected return i.e the expected total reward collected by the end of the episode
- For FOBO methods gradient of the expected return is also returned by the black box
- Averaging over multiple sample trajectories is done to obtain estimates of expected return and gradients
- Finally we can apply policy gradient methods described before for episodic tasks

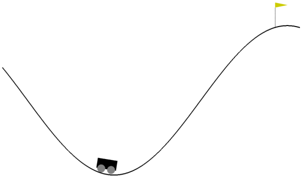
Overview

Simple Gridworld



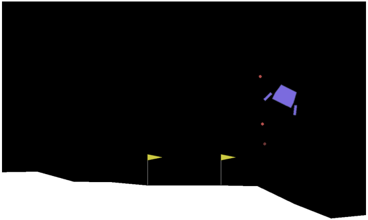


MountainCar-v0



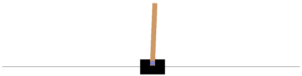
- State Space:2
- Action Space:3
- Rewards: -1 per time step

LunarLander-v2



- State Space:8
- Action Space:4
- Rewards:
  - Leg ground contact:+10 or -10
  - Fire main engine:-0.3 per frame
  - Fire side engines:-0.03 per frame
  - If solved then +200

Carpole-v1



- State Space:4
- Action Space:2
- Rewards:+1 per timestep balanced

Settings considered

Existing methods:

- ZOBO (Zero Order Bayesian Optimization)
- FOBO (First Order Bayesian Optimization)

Modified methods:

- FOBO_Improved (First Order Bayesian Optimization with modified acquisition function)
- FOBO_Improved with NG (Using Natural gradients instead of gradients for FOBO)
- FOBO_topK(using FOBO with top K acquistion function)

Modified methods are compared with existing methods and only the methods which show best results are shown in next few slides.

Overview

Results

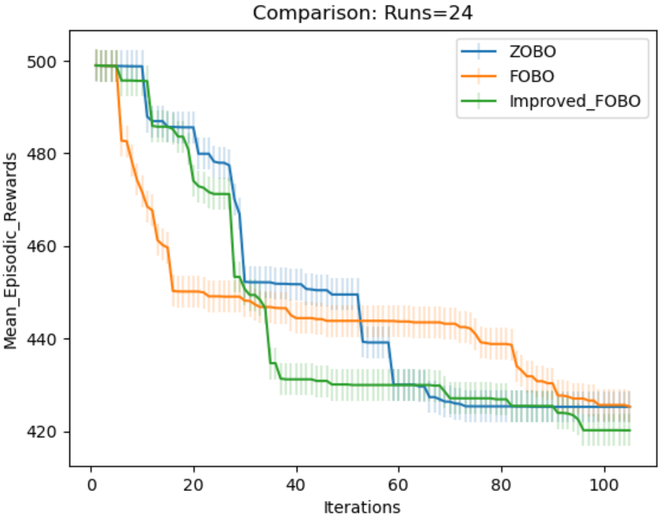


Figure: Mountain Car Comparison

Results

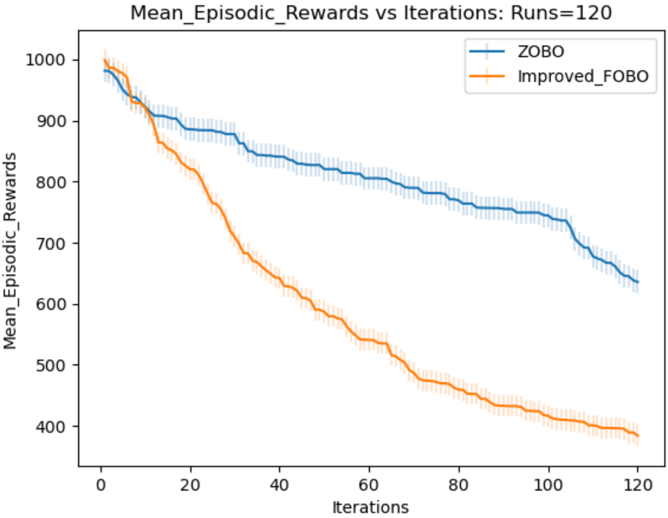


Figure: Mountain Car Comparison

Results

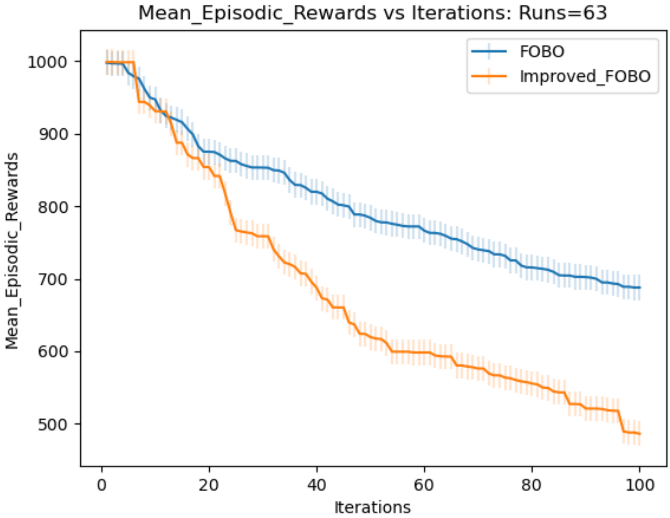


Figure: Mountain Car Comparison

Results

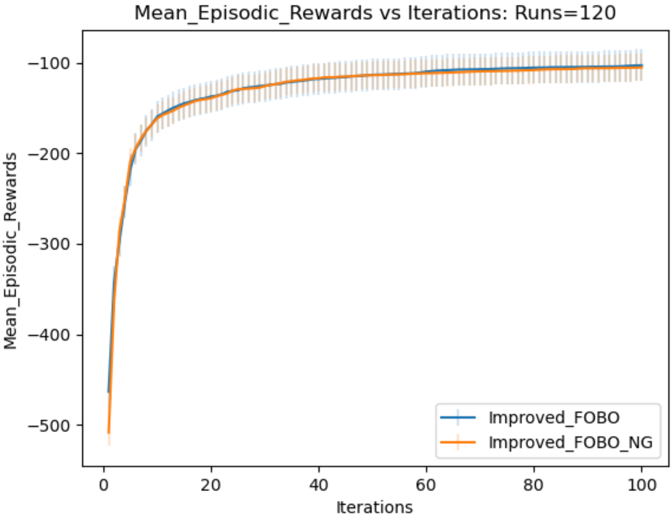


Figure: Lunar Lander Comparison

Results

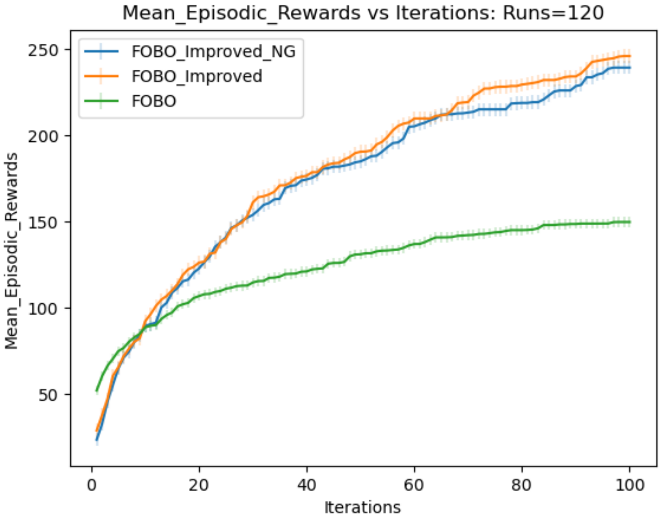


Figure: CartPole Comparison

Results

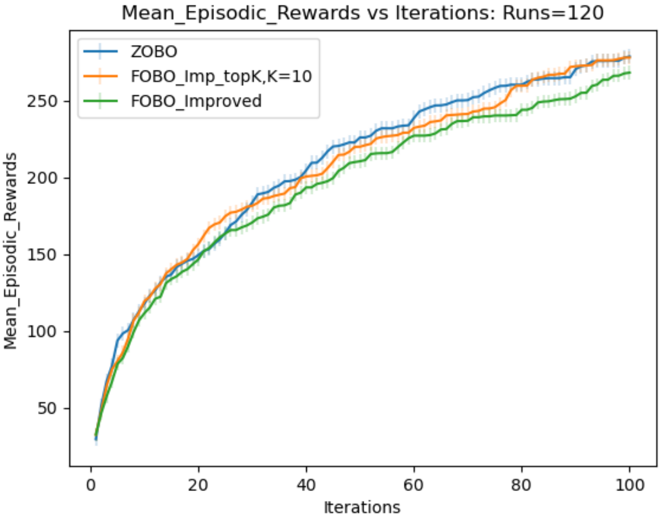


Figure: CartPole Comparison

Results

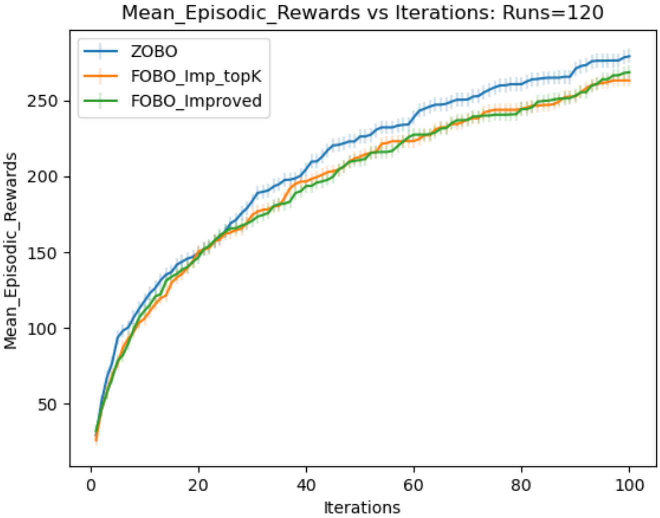


Figure: CartPole Comparison

Overview

Inferences

Other experiments that we tried:

- Tried tweaking architecture used for policy approximation
- Different values of K for topK method
- Different runs to average out rewards
- Varied length and number of trajectories in different tasks

Last Thoughts

- Using gradient information to improve existing methods is a right step ahead which can be helpful in many RL tasks
- The main problem lies with the gradients becoming too small for many of the tasks tried
- Still, we have been able to extract some information from the gradients and utilise it to be atleast at par or better than ZOBO(Zero order Bayesian Opitmization)
- We are still figuring out ways which can help improve information extracted from gradients by using better acquisition functions, which will further help improve performance

The End