A Gaussian Mixture Model Approach to Off-Policy Policy Gradient Estimation

Master-Thesis von Stephane Tekam Feudjo
Tag der Einreichung: 8. Juli 2020

1. Gutachten: Prof. Dr. Jan Peters
2. Gutachten: Prof. Dr. Alexandra Schwartz
3. Gutachten: M.Sc. Samuele Tosatto

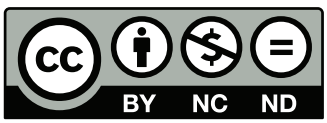
TECHNISCHE
UNIVERSITÄT
DARMSTADT

A Gaussian Mixture Model Approach to Off-Policy Policy Gradient Estimation

Vorgelegte Master-Thesis von Stephane Tekam Feudjo

1. Gutachten: Prof. Dr. Jan Peters
2. Gutachten: Prof. Dr. Alexandra Schwartz
3. Gutachten: M.Sc. Samuele Tosatto

Tag der Einreichung:



Erklärung zur Abschlussarbeit gemäß §23 Abs. 7 APB der TU Darmstadt

Hiermit versichere ich, Stephane Tekam Feudjo, die vorliegende Master-Thesis ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.
Mir ist bekannt, dass im Falle eines Plagiats (§38 Abs.2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.
Bei der abgegebenen Thesis stimmen die schriftliche und die zur Archivierung eingereichte elektronische Fassung überein.

Thesis Statement pursuant to §23 paragraph 7 of APB TU Darmstadt

I herewith formally declare that I, Stephane Tekam Feudjo, have written the submitted thesis independently. I did not use any outside support except for the quoted literature and other sources mentioned in the paper. I clearly marked and separately listed all of the literature and all of the other sources which I employed when producing this academic work, either literally or in content. This thesis has not been handed in or published before in the same or similar form.
I am aware, that in case of an attempt at deception based on plagiarism (§38 Abs. 2 APB), the thesis would be graded with 5,0 and counted as one failed examination attempt. The thesis may only be repeated once.
In the submitted thesis the written copies and the electronic version for archiving are identical in content.

Datum/Date 8. Juli 2020

Unterschrift des Autors/Signature of author

Abstract

Despite the significant advances in state-of-the-art reinforcement learning, sample efficiency remains an essential issue in the development of reinforcement learning algorithms. This problem turns out to be more evident when sampling is costly. In theory, Off-policy methods deliver a learning scheme capable of higher sample efficiency. However, the state-of-the-art off-policy gradient estimation either suffers high bias (semi-gradient approaches) or high variance (importance sampling). A recent approach, based on non-parametric density estimation, delivers a better bias/variance tradeoff. Still, non-parametric methods do not scale well with dimensionality. Hence they have limited applicability. Our proposed solution is to approach the density estimation via Gaussian Mixture Models, which scale better while avoiding the problems of importance sampling and semi-gradient techniques. We empirically analyze the quality of the gradient estimation on a set of classical control tasks.

Acknowledgments

I want to thank my thesis advisor M.Sc. Samuele Tosatto for his invaluable support and guidance. I would also like to acknowledge Prof. Dr. Alexandra Schwartz of the Department of Mathematics and Prof. Dr. Jan Peters of the Department of Computer Science. They made this thesis possible. I am grateful for their valuable comments on this thesis.

The experiments could not have been possible without the services of the Lichtenberg High Performance Computer of TU Darmstadt. I am grateful for the computing resources.

Finally I express my profound gratitude to my parents and to my friends for their continuous encouragement throughout my years of study.

Contents

Figures and Tables

List of Figures

List of Tables

Abbreviations, Symbols and Operators

List of Abbreviations

Notation	Description
BR	Bayes rule
eqn	equation
GMM	Gaussian Mixture Model
GMR	Gaussian Mixture Regression
i.i.d.	independently and identically distributed
LQR	Linear Quadratic Regulator
MDP	Markov Decision Process
MRP	Markov Reward Process
RL	Reinforcement learning

List of Symbols

Notation	Description
$\boldsymbol{\theta}$	vector of parameters from a probability distribution

List of Operators

Notation	Description	Operator
$\ln$	the natural logarithm	$\ln(\bullet)$

1 Introduction

RL has made overwhelming progress in recent years [1, 2]. For example, it is being used to teach computers to control robots in simulation [3]. Moreover, advanced algorithms for strategic games were successfully designed using this approach [4]. However, the vast majority of RL approaches are successful in simulated tasks, where it is possible to retrieve a large number of samples and to safely interact with the environment. For real-world applications, state-of-the-art RL techniques are not yet able to deliver satisfying results. This limitation is due to the general sample inefficiency of RL, and to its incapability to deliver safe exploration in early stages.

The main reason is that the vast majority of RL algorithms are on-policy. On-policy techniques are constrained to use the optimization policy to interact with the system: this is the primary cause to sample inefficiency as after each policy update, one needs to further interact with the environment, and at the same time hinders safe interaction.

These limitations can be potentially overcome by off-policy RL. In off-policy techniques the behavioural policy is detached to the optimization policy, allowing a safe collection of the sample from an expert (either a human or a hand-crafted policy), and granting sample reuse in the policy update process.

One of the main techniques to obtain off-policy updates, is via policy gradient methods, which update the policy using a gradient ascent algorithm. However the off-policy gradient estimation is non-trivial. The current state-of-the-art can be divided in two main categories: semi-gradient estimation (SG), and path-wise importance sampling (PWIS). SG techniques, like OffPAC [5] and Deep Deterministic Policy Gradient [6, 7] deliver biased estimate of the gradient. Such bias is critical and causes failure in more complex off-policy datasets [8]. On the other hand, the estimate delivered by PWIS is affected by high variance, and further more, importance sampling requires known stochastic behavioral policies, impeding human demonstrations for data collections.

In this thesis we develop a method which overcomes both these limitations. In detail, we propose a full-gradient estimation of the objective function, which is less subject to both bias and variance. We build this estimation on a closed form solution of the value function, which allows to express the full-gradient w.r.t. the policy's parameters.

We test our method both on classic benchmarks such as the swing-up pendulum as well as a 2-dimensional LQR problem.

2 Foundations

2.1 Reinforcement Learning

RL is a decision problem where an agent is interacting in an environment, and at each interaction, it observes a reward. The goal is to maximize cumulative rewards. More in detail, at each time step, the agent interacts with the environment by performing an *action a*. At each time step, it interacts with the environment by performing an action, which results to a transition in the *next state s'*, and a reward (which is a scalar valued function). This reward measures the performance of the agent at each step. The agent decides which action to take by using a function that describes its behaviour (*policy*). A policy can be deterministic or stochastic. A deterministic policy takes a state as input and outputs a single action, whereas a stochastic policy outputs a distribution over actions. The agent tries to find the maximum reward while interacting with the environment through numerous trials and error. The main objective of RL is to define the best sequence of decisions that allow the agent to solve a problem while maximizing the long term reward (i.e. to find an optimal policy that maximizes the numerical reward). A Rl can be viewed as a sequential decision problem under uncertainty. At each sequence, the agent decides on an action to take, thus the numerous trials and errors results to uncertainty. The figure below illustrates the interaction of the agent with the environment.

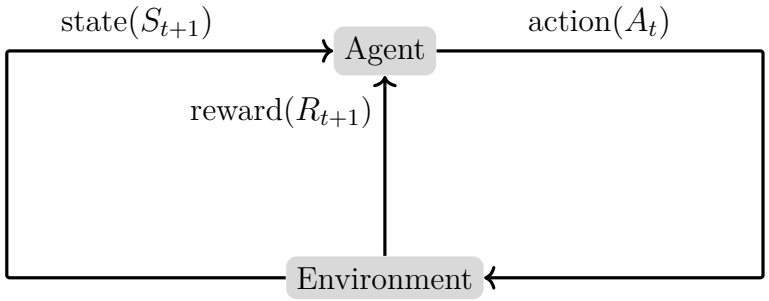


Figure 2.1.: A general RL framework.

As an example, let us consider the pendulum task whose target is to keep a frictionless pendulum standing. In this problem, the pendulum's angle and the angular velocity capture the *state s* of the problem. The *action u* consist of a continuous torque $u \in [-2, 2]$. The reward is a scalar objective function that depends on the pendulum's angle, velocity and action. A function π that generates the torque based on the current angle and angular velocity is called the *policy* . The optimal policy will produce a sequence of actions that will keep the pendulum upright.

In the section that follows, we introduce the Markov Decision Process (MDP), which is the formal definition of the RL problem.

2.2 Markov Decision Process

The Markov decision process (MDP) allows us to model how the state of a stochastic system changes when an agent acts or controls this system by selecting and applying an action. But before defining an MDP it is important to get some intuition by defining a Markov reward process (MRP). Both an MDP and an MRP satisfy the Markov decision property.

Definition 1. (Markov Property) A state S_t satisfies the markov property if and only if

$$P(S_{t+1} \mid S_t) = P(S_{t+1} \mid S_1, \cdots, S_t)$$

In other words, all the information required to make a decision is included in the present state, not in the past.

Definition 2. A Markov reward process is a tuple $< \mathscr{S}, \mathscr{R}, P, \gamma, \mu_0 >$

- $\mathscr{S}$ is the state space.
- $P : \mathscr{S} \times \mathscr{S} \to [0, 1]$ is the transition function.
- R: $\mathscr{S} \to \mathbb{R}$ is the reward function.
- $\gamma \in [0, 1)$ is a discount factor.

where $\mathscr{S}$ is a set of all possible states of the environment; $P(s'|s)$ represents the probability of observing the state s' given that the agent is in state s. $R(s)$ is a random mapping between the state and the reward signal.

The Markov decision process formally describes the RL problem. It is an MRP with actions. For a fixed policy, a MDP becomes a MRP.

Definition 3. An MDP is a tuple $< \mathscr{S}, \mathscr{A}, R, P, \gamma, \mu_0 >$ where $\mathscr{S}$ is a set of all possible states of the environment; $\mathscr{A}$ a set of actions that are available to the agent; $P(s'|s, a)$ represents the probability of observing the state s' after the application of the action a in state s

$$P(s'|s, a) = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]. \tag{2.1}$$

$R(s, a)$ is the stochastic mapping between state-action pairs and the real-valued reward signal

$$\begin{aligned} R : \mathscr{S} \times \mathscr{A} &\mapsto \mathbb{R} \\ (s, a) &\mapsto R(s, a) = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]. \end{aligned} \tag{2.2}$$

The discount factor $\gamma \in [0, 1)$ determines the importance of immediate versus future rewards. If this factor is close to 0 the agent is concerned with only maximizing the immediate rewards. A value of γ close to 1 leads to a strong consideration of future rewards (the agent becomes far-sighted). μ_0 represents the initial state distribution. Associated to an MDP is the maximum number of time steps $T \in \mathbb{N} \cup \{\infty\}$. When $T \in \mathbb{N}$, the setting is said to be a finite horizon setting. Else it is referred to as an infinite horizon setting. Furthermore, the state and action spaces can either be continuous or discrete.

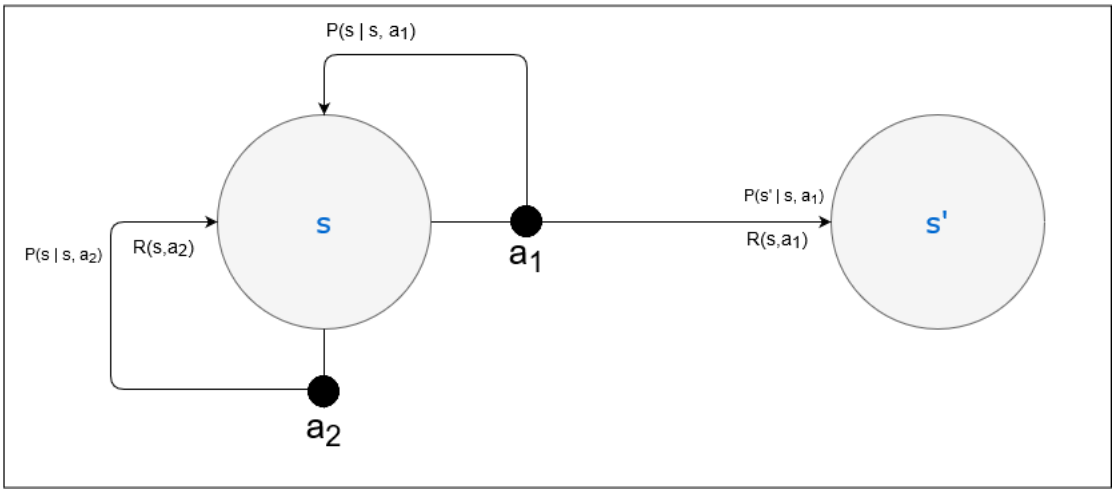


Figure 2.2.: An example of a simple Markov decision process.

In this section we represented the Markov reward process and the Markov decision process. The former captures an environment in which no actions are taken, and the later allows us to model the evolution dynamics of a stochastic system controlled by selecting and applying actions.
It should be noted that an MDP models the problem, not the solution. A policy π is the solution to an MDP. A stochastic policy is defined as

$$\begin{aligned} \pi : \mathscr{S} &\to \mathscr{P}(\mathscr{A}) \\ s &\mapsto \pi(a \mid s) = \mathbb{P}(A_t = a \mid S_t = s), \end{aligned}$$

where $\mathscr{P}$ is the probability over the set of actions. While a deterministic policy is given by

$$\begin{aligned} \pi : \mathscr{S} &\to \mathscr{A} \\ s &\mapsto \pi(s). \end{aligned}$$

An agents behaviour is determined by the policy, which encodes a deterministic or a stochastic mapping between the state space and the action space.

2.3 Value Functions and Bellman Equations

The discounted return of a policy in the infinite horizon case at time step t is given by,

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}.$$

The discount factor guarantees the convergence of the infinite sum [9]. Solving an MDP is equivalent to searching for an optimal policy (i.e. a policy that maximizes the expected discounted return). The expectation here accounts for the stochasticity in the environment as well as in the policy

$$J_\pi = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}\right] \tag{2.3}$$

where $A_t \sim \pi(S_t)$, $S_{t+1} \sim P(\cdot|S_t, A_t)$ and $R_t \sim R(S_t, A_t)$.

Value Functions

Value functions are an essential concept for determining optimal policies. For a fix policy π, the state value function $V_\pi : \mathscr{S} \to \mathbb{R}$ of an MDP is the expected return when an agent starts from a state s, and follows the policy π,

$$V_\pi(s) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s\right].$$

Likewise, for some fix policy π, the action-value function $Q_\pi : \mathscr{S} \times \mathscr{A} \to \mathbb{R}$ describes the average discounted cumulative reward when the agent starts in state s, takes action a and then act according to policy π,

$$Q_\pi(s,a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a\right].$$

Following [10] we define $\mu_\pi(s) = \sum_{t=0}^{\infty} \gamma^t \mathbf{Pr}(S_t = s|S_0, \pi)$ with $S_0 \sim \mu_0$, as the state distribution function induced by policy π.
The state value function and the action-value function respectively indicate how good it is to be in a state and how good it is to perform an action in a state, while taking into account the future. The Bellman equations can be used for an efficient computation of the value functions.

Bellman Equation

Value functions can be computed in a recursive way. Because of the Bellman Equation, the state value function at a specific state can be expressed as a function of the value function computed in the next state [11]. Given a MDP $< \mathscr{S}, \mathscr{A}, R, P, \gamma, \mu_0 >$ and a policy π, we obtain an MRP with

$$\begin{aligned} R_\pi(s) &= \int_{a \in \mathscr{A}} \pi(a \mid s) R(s,a) da, \\ P_\pi(s' \mid s) &= \int_{a \in \mathscr{A}} \pi(a \mid s) P(s' \mid s, a) da. \end{aligned} \tag{2.4}$$

Theorem 1. *(The rule of iterated Expectations) [12] For random variables X and Y, assuming the expectations exist, we have that*

$$\mathbb{E}\left[\mathbb{E}\left[Y \mid X\right]\right] = \mathbb{E}\left[Y\right].$$

Generally, for any function $r(x,y)$ we have

$$\mathbb{E}\left[\mathbb{E}\left[r(X,Y) \mid X\right]\right] = \mathbb{E}\left[r(X,Y)\right].$$

Proof. See [12]. □

We then proceed to derive the Bellman equation as in [11].

$$\begin{aligned}
V_\pi(s) &= \mathbb{E}\left[G_t \mid S_t = s\right] \\
&= \mathbb{E}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \mid S_t = s\right] \\
&= \mathbb{E}\left[R_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k R_{t+k+2} \mid S_t = s\right] \\
&= \mathbb{E}\left[R_{t+1} + \gamma G_{t+1} \mid S_t = s\right] \\
&= \mathbb{E}\left[R_{t+1} \mid S_t = s\right] + \gamma \mathbb{E}\left[G_{t+1} \mid S_t = s\right].
\end{aligned}$$

From Eqn 2.4 we obtain:

$$\mathbb{E}\left[R_{t+1} \mid S_t = s\right] = \int_{a \in \mathscr{A}} \pi(a \mid s) \mathbb{E}\left[R_{t+1} \mid S_t = s, A_t = a\right] da.$$

By the rule of iterated expectations, we have

$$\mathbb{E}\left[G_{t+1} \mid S_t = s\right] = \mathbb{E}\left[\mathbb{E}\left[G_{t+1} \mid S_{t+1} = s'\right] \mid S_t = s\right].$$

Hence

$$\begin{aligned}
\mathbb{E}\left[G_{t+1} \mid S_t = s\right] &= \mathbb{E}\left[V_\pi(s') \mid S_t = s\right] \\
&= \int_{s' \in \mathscr{S}} V_\pi(s') P_\pi(s' \mid s) ds' \\
&= \int_{a \in \mathscr{A}} \pi(a \mid s) \int_{s' \in \mathscr{S}} V_\pi(s') P(s' \mid s, a) ds' da.
\end{aligned}$$

It follows that

$$\begin{aligned}
V_\pi(s) &= \int_{\mathscr{A}} \pi(a \mid s) \left(R(s,a) + \gamma \int_{\mathscr{S}} V_\pi(s') P(s' \mid s, a) ds'\right) da \\
&= R_\pi(s) + \gamma \int_{s' \in \mathscr{S}} V_\pi(s') P_\pi(s' \mid s) ds'. \qquad (2.5)
\end{aligned}$$

Similarly we derive the Bellman equation for the action-value function as follows:

$$\begin{aligned}
Q_\pi(s,a) &= \mathbb{E}\left[G_t \mid S_t = s, A_t = a\right] \\
&= \mathbb{E}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \mid S_t = s, A_t = a\right] \\
&= \mathbb{E}\left[R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t = a\right] \\
&= R(s,a) + \gamma \mathbb{E}\left[G_{t+1} \mid S_{t+1} = s'\right] \\
&= R(s,a) + \gamma \int_{a' \in \mathscr{A}} \pi(a' \mid s') \mathbb{E}\left[G_{t+1} \mid S_{t+1} = s', A_{t+1} = a'\right] da' \\
&= R(s,a) + \gamma \int_{s' \in \mathscr{S}} \int_{a' \in \mathscr{A}} \pi(a' \mid s') Q_\pi(s', a') da ds' \qquad (2.6)
\end{aligned}$$

The relation between the action-value function and the state value function is given by the following equations.

$$V_\pi(s) = \int_{a \in \mathscr{A}} \pi(a \mid s) Q_\pi(s,a) da, \qquad (2.7)$$

$$Q_\pi(s,a) = R(s,a) + \gamma \int_{s' \in \mathscr{S}} V_\pi(s') P(s \mid s', a) ds'. \qquad (2.8)$$

Proof. $V_\pi(s)$ is the expected return, from a fixed state s. Let the set of actions that can be taken in state s be $\mathscr{A}(s)$. Associated to each $a \in \mathscr{A}(s)$ is an action value $Q(s,a)$, and the probability of choosing the action. This probability is defined by the policy $\pi(a \mid s)$. It follows that the value of being in state s is the average over the actions. Hence

$$V_\pi(s) = \int_{a \in \mathscr{A}} \pi(a \mid s) Q_\pi(s,a) da.$$

Conversely, Eqn 2.8 follows directly from Eqn 2.6 and Eqn 2.7:

$$\begin{aligned} Q_\pi(s,a) &= R(s,a) + \gamma \int_{s' \in \mathscr{S}} \int_{a' \in \mathscr{A}} \pi(a' \mid s') Q_\pi(s',a') da ds' \\ &= R(s,a) + \gamma \int_{s' \in \mathscr{S}} V_\pi(s') P(s \mid s', a) ds'. \end{aligned}$$

□

Consider for simplicity a finite state space of dimension d_s , and a deterministic policy π. Then the value function can be viewed as a vector space [13] with coordinates $[V_\pi(s_1), \cdots V_\pi(s_{d_s})]$. The Bellman operator for the policy π is defined by

$$\begin{aligned} L_\pi &: \mathbb{R}^{d_s} \to \mathbb{R}^{d_s} \\ (L_\pi) V_\pi(s) &= R(s, \pi(s)) + \gamma \sum_{s' \in \mathscr{S}} P(s' \mid s, \pi(s)) V_\pi(s') \qquad s \in \mathscr{S}. \end{aligned}$$

From Eqn 2.5, $V_\pi(s) = R(s,\pi(s)) + \gamma \sum_{s' \in \mathscr{S}} V_\pi(s') P(s' \mid s, \pi(s))$.
Hence we obtain a vector form of the Bellman equation $L_\pi V_\pi = V_\pi$ which is a linear system of equations, and can directly be solved using methods such as conjugate gradient,

$$\begin{aligned} V_\pi &= R_\pi + \gamma P_\pi V_\pi, \\ V_\pi &= (I - \gamma P_\pi)^{-1} R_\pi, \end{aligned}$$

where

$$V_\pi = \begin{bmatrix} V_\pi(S_1) \\ \vdots \\ V_\pi(S_{d_s}) \end{bmatrix}, \qquad R_\pi = \begin{bmatrix} R_\pi(S_1) \\ \vdots \\ R_\pi(S_{d_s}) \end{bmatrix}, \qquad P_\pi = \begin{bmatrix} P_\pi(S_1 \mid S_1) & \cdots & P_\pi(S_n \mid S_1) \\ \vdots & & \\ P_\pi(S_1 \mid S_n) & \cdots & P_\pi(S_n \mid S_n) \end{bmatrix}.$$

The goal is to compute the optimal policy, not just the value function for a given policy. If we know the optimal value function, we can easily derive the optimal policy.
The optimal state value function is given by

$$V^*(s) = \sup_{a \in \mathscr{A}} \{ R(s,a) + \gamma \sum_{s' \in \mathscr{S}} V^*(s') P(s' \mid s, a) \}. \tag{2.9}$$

The Bellman optimality operator $L^* : \mathbb{R}^{d_s} \to \mathbb{R}^{d_s}$ on a vector V is given by

$$(L^* V)(s) = \sup_{a \in \mathscr{A}} \{ R(s,a) + \gamma \sum_{s' \in \mathscr{S}} V(s') P(s' \mid s, a) \}.$$

As a result of L^*, Eqn 2.9 can be written in compact form.

$$L^*V^* = V^*$$

As compared to the linear operator L, L^* is a non-linear operator, because of the sup function. Hence V^* cannot be solved using linear methods. Exploiting the fact that V^* is a fixed point of L^* and making use of dynamic programming with the recurrence relation

$$V_\pi^{k+1}(s) = R(s, \pi(s)) + \gamma \sum_{s' \in \mathscr{S}} V_\pi^k(s') P(s' \mid s, \pi(s)), \quad k \geq 0$$

a sequence of functions is generated,

$$V_{k+1} = L^* V_k,$$

and the optimal value function can be obtained in polynomial time. L and L^* are contractions [13], and due to Banach's fixed point theorem, $\lim_{k\to\infty} V_k = V^*$.

Theorem 2. *(Banach Fixed Point) Suppose U is a Banach space and $T : U \to U$ is a contraction mapping. Then*

- *there exists a unique $\boldsymbol{v}*$ in U such that $T\boldsymbol{v}* = \boldsymbol{v}*$.*
- *for an arbitrary $\boldsymbol{v}^0 \in U$, the sequence $\boldsymbol{v}^k$ converges to $\boldsymbol{v}*$*

Proof. The proof of this theorem can be found in [14] □

In this chapter, we reviewed the mathematical background of reinforcement learning. An RL problem is formulated as an MDP. The solution of an MDP is a policy that maximizes Eqn (2.3). The state value function measures the performance of a policy at a state, while the action-value function measures the performance of a policy for each state-action pair. One can infer the optimal policy from the optimal value function $\pi_* = \arg\max_\pi V_\pi(s), \pi_* = \arg\max_\pi Q_\pi(s, a)$. This optimal value function can be computed efficiently via the Bellman equations (2.5) and (2.6)). Thanks to Banach's fixed point theorem we are sure to find the optimal value function via dynamic programming in polynomial time. Algorithms that infer the optimal policy from the value functions are called value based methods.

For simplicity, we considered a finite MDP. This model has limitations, as the state and action spaces in the real world are often continuous. We look at algorithms that find the optimal policy by directly parameterizing the policy function and searching through the policy space.

2.4 Policy Optimization

In the real world where the action and state spaces are continuous and have large dimensions, it is not possible to enumerate all these spaces in order to get an exact solution. Function approximation solves the problem in large spaces. In the case of policy search methods, the policy is represented by a parametric function $\pi_\theta(a \mid s)$, where θ is a vector of parameters. In this setting, searching for the optimal parameter is equivalent to searching for the optimal policy. The optimal parameter can be found via the gradient ascent algorithm, which is based on the classic gradient descent algorithm. The difference is at the level of the update rule. While in gradient descent we

minimize the gradient by taking steps opposite to the direction of the gradient, in gradient ascent, we maximize the objective function by moving in the direction of the gradient. The objective function is the value function averaged over the state distributions.

$$J_{\pi_\theta} = \int_{s \in \mathscr{S}} \mu_0(s) V_{\pi_\theta}(s) ds,$$

where μ_0 is the initial state distribution.
Computing the gradient $\nabla_\theta J_{\pi_\theta}$ is problematic since it depends on the state distribution, which in turn depends depends on the target policy. As a result, the distribution changes after each policy update [10]. The policy gradient theorem provides an estimate for the gradient of the objective function J_θ.

Theorem 3. *(Policy Gradient Theorem) [10] For any MDP,*

$$\nabla_\theta J_\theta = \int_{\mathscr{S}} \int_{\mathscr{A}} \mu_{\pi_\theta}(s) \nabla_\theta \pi_\theta(a|s) Q_{\pi_\theta}(s,a) da ds.$$

Proof. We first derive the gradient of the state value function.

$$\begin{aligned}
\nabla_\theta V_{\pi_\theta}(s) &= \frac{\partial V_{\pi_\theta}(s)}{\partial \theta} \\
&= \frac{\partial}{\partial \theta} \int_{a \in \mathscr{A}} \pi(a \mid s) Q_{\pi_\theta}(a) da \qquad (2.7) \\
&= \int_{a \in \mathscr{A}} \left[\frac{\partial \pi_\theta(a \mid s)}{\partial \theta} Q_{\pi_\theta}(s,a) + \pi_\theta(a \mid s) \frac{\partial}{\partial \theta} Q_{\pi_\theta}(s,a) \right] da \\
&= \int_{a \in \mathscr{A}} \left[\frac{\partial \pi_\theta(a \mid s)}{\partial \theta} Q_{\pi_\theta}(s,a) + \pi_\theta(a \mid s) \frac{\partial}{\partial \theta} \left(R(s,a) + \gamma \int_{s' \in \mathscr{S}} V_{\pi_\theta}(s') P(s' \mid s,a) ds' \right) \right] da \qquad (2.8) \\
&= \int_{a \in \mathscr{A}} \left[\frac{\partial \pi_\theta(a \mid s)}{\partial \theta} Q_{\pi_\theta}(s,a) + \pi_\theta(a \mid s) \left(\gamma \int_{s' \in \mathscr{S}} P(s' \mid s,a) \frac{\partial}{\partial \theta} V_{\pi_\theta}(s') ds' \right) \right] da.
\end{aligned}$$

It follows that

$$\nabla_\theta V_{\pi_\theta}(s) = \int_{a \in \mathscr{A}} \left[\nabla_\theta \pi_\theta(a \mid s) Q_{\pi_\theta}(s,a) + \pi_\theta(a \mid s) \left(\gamma \int_{s' \in \mathscr{S}} P(s' \mid s', a) \nabla_\theta V_{\pi_\theta}(s') ds' \right) \right] da \qquad (2.10)$$

Eqn 2.10 has a nice recursive property since we can also express $\nabla_\theta V_{\pi_\theta}(s')$ as a function of $\nabla_\theta V_{\pi_\theta}(s'')$. That is the gradient of the value function of the current state is expressed as the gradient of the value function of the next state.
Let $\rho_{\pi_\theta}(s \to x, k)$ denotes the probability for transitioning from state s to state x after k steps under policy π_θ. When $k = 1$, $\rho_{\pi_\theta}(s \to s', 1) = \int_{a \in \mathscr{A}} \pi_\theta(a \mid s) P(s' \mid s, a) da$.

For simplification let $\varphi(s) = \int_{a\in\mathscr{A}} \nabla_\theta \pi_\theta(a \mid s) Q_{\pi_\theta}(s,a)da$. We obtain

$$\begin{aligned}
\nabla_\theta V_{\pi_\theta}(s) &= \varphi(s) + \int_{a\in\mathscr{A}} \pi_\theta(a \mid s)\gamma \int_{s'\in\mathscr{S}} P(s' \mid s', a)\nabla_\theta V_{\pi_\theta}(s')ds'da \\
&= \varphi(s) + \int_{s'\in\mathscr{S}} \int_{a\in\mathscr{A}} \pi_\theta(a \mid s)\gamma P(s' \mid s', a)\nabla_\theta V_{\pi_\theta}(s')dads' \\
&= \varphi(s) + \gamma \int_{s'\in\mathscr{S}} \rho_{\pi_\theta}(s \to s', 1)\nabla_\theta V_{\pi_\theta}(s')ds' \\
&= \varphi(s) + \gamma \int_{s'\in\mathscr{S}} \rho_{\pi_\theta}(s \to s', 1)\left[\varphi(s') + \gamma \int_{s''\in\mathscr{S}} \rho_{\pi_\theta}(s' \to s'', 1)\nabla_\theta V_{\pi_\theta}(s'')ds''\right] ds' \\
&= \varphi(s) + \gamma \int_{s'\in\mathscr{S}} \rho_{\pi_\theta}(s \to s', 1)\varphi(s')ds' + \gamma^2 \int_{s''\in\mathscr{S}} \rho_{\pi_\theta}(s \to s'', 2)\varphi(s'))\nabla_\theta V_{\pi_\theta}(s'')ds'' \\
&\vdots \quad \text{after expanding } \nabla_\theta V_{\pi_\theta} \text{ for several steps} \\
&= \sum_{k=0}^{\infty} \gamma^k \int_{x\in\mathscr{S}} \rho_{\pi_\theta}(s \to x, k)\varphi(x)dx
\end{aligned}$$

with the initial state denoted as s_0, it then follows that

$$\begin{aligned}
\nabla_\theta J_{\pi_\theta} &= \nabla_\theta V_{\pi_\theta}(s_0) \\
&= \sum_{k=0}^{\infty} \gamma^k \int_{s\in\mathscr{S}} \rho_{\pi_\theta}(s_0 \to s, k)\varphi(s)ds \\
&= \int_{s\in\mathscr{S}} \eta(s)\varphi(s)ds \qquad \text{where } \eta(s) = \sum_{k=0}^{\infty} \gamma^k \rho_{\pi_\theta}(s_0 \to s, k) \\
&= \int_{s\in\mathscr{S}} \eta(s) \int_{s\in\mathscr{S}} \frac{\eta(s)}{\int_{s\in\mathscr{S}} \eta(s)\, ds} ds\varphi(s)ds \text{ normalizing } \eta(s) \text{ results in a distribution} \\
&= \int_{s\in\mathscr{S}} \frac{\eta(s)}{\int_{s\in\mathscr{S}} \eta(s)\, ds}\varphi(s)ds \\
&\propto \int_{s\in\mathscr{S}} \mu_{\pi_\theta}(s) \int_{a\in\mathscr{A}} \nabla_\theta \pi_\theta(a|s) Q_{\pi_\theta}(s,a)dads \\
\text{where } \mu_{\pi_\theta}(s) &= \frac{\eta(s)}{\int_{s\in\mathscr{S}} \eta(s)\, ds} \text{is a stationary distribution.}
\end{aligned}$$

□

Here $Q_\pi(s,a)$ can be estimated via Monte-Carlo sampling[15]. However when the policy is updated, the state distribution changes, in order to apply again this policy update, we need to interact with the environment. This continual interaction with the environment causes a high sample complexity. To solve this problem, we make use of an off-policy setting, where the agent learns a target policy, while using samples collected with a behavioral policy. Off-policy gradient estimation can be further divided in two main classes: the semi-gradient and the importance sampling correction. which we present in Chapter 3.

On-Policy and Off-Policy Methods

Sample efficiency plays an important role in RL algorithms. Off-policy algorithms are generally more sample efficient than off-policy algorithms. In an off-policy setting, the target policy π

(policy used to train the algorithm) is non-identical to the behaviour policy β (policy generating the data.). Examples include Q-learning [16], DDPG [7], NOPG [17]. Conversely, in on-policy methods like SARSA [18], REINFORCE [15] the behaviour policy is the same as the target policy.
In policy search, on-policy gradient methods perform only one gradient step per environment sample, while off-policy gradients use the same samples more than once to improve a policy. Since off-policy methods enable data reuse, they are more sample efficient.

2.5 Gaussian Mixture Regression (GMR)

In this section we introduce the GMR [19, 20] which is a mixture of linear model (i.e. partitions the input space into sub-regions and fits a linear model in each such region). The basic assumption made is that the data can be represented by a mixture of a finite number of Gaussian distribution. Given the input variable X and the output variable Y, GMR first estimates the joint density function p(x, y) of the independent variable and the dependent variable and then infers the probability of the output variable conditioned to the input.

Preliminaries

Here we review some basic definitions and theorems on multivariate Gaussian distribution.

Definition 4. A random variable $\mathbf{X}$ has a multivariate Normal distribution $\mathbf{X} \sim \mathcal{N}(\mu, \mathbf{\Sigma})$, if its density is given by

$$\mathbf{p}(x; \mu, \mathbf{\Sigma}) = \frac{1}{(2\pi)^{k/2}|(\mathbf{\Sigma})|^{1/2}} \exp\left\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right\}$$

where

$$\mu \in \mathbb{R}^{d_x}, \quad \mathbf{\Sigma} \in \mathbb{R}^{d_x \times d_x} \text{ is a positive symmetric, positive definite matrix.}$$

The covariance matrix defines the spread and the orientation of the distribution. The number of parameters of a multivariate gaussian depends on the type of covariance matrix used. Three different kinds of covariance matrices are frequently used in the literature:

- A model with full a covariance matrix has $d_x(d_x+1)\cdot 2^{-1} + d_x$ parameters
- A model with a diagonal covariance matrix has $2 \cdot d_x$ parameters and 0s in the off diagonal.
- A model with a spherical covariance matrix $\Sigma = \sigma^2\mathbf{I}$ has $1 + d_x$ parameters.

For simplicity, let us consider three Gaussians where $d_x = 2$. Each has mean $\mu = \begin{bmatrix}0 & 0\end{bmatrix}^\intercal$ and covariance matrices

$$\mathbf{\Sigma_1} = \begin{bmatrix}1 & 0.8\\ 0.8 & 1\end{bmatrix}, \qquad \mathbf{\Sigma_2} = \begin{bmatrix}0.6 & 0\\ 0 & 1\end{bmatrix}, \quad \text{and} \quad \mathbf{\Sigma_3} = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}.$$

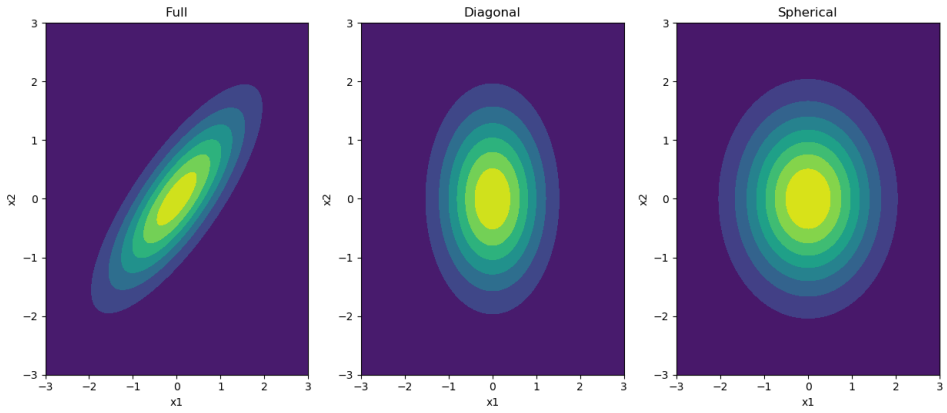


Figure 2.3.: Type of covariance matrix.

We mention the application of the various covariance matrices further in this section under density estimation using Gaussian mixture models.

Suppose $\mathbf{Z} \sim \mathcal{N}(\mu, \mathbf{\Sigma})$, if we partition $\mathbf{Z}$ as $\mathbf{Z} = (\mathbf{X}, \mathbf{Y})$ then $\mu = (\mu_{\mathbf{x}}, \mu_{\mathbf{y}})$ and

$$\mathbf{\Sigma} = \begin{pmatrix} \mathbf{\Sigma}_{xx} & \mathbf{\Sigma}_{xy} \\ \mathbf{\Sigma}_{yx} & \mathbf{\Sigma}_{yy} \end{pmatrix}.$$

Theorem 4. *For* $\mathbf{Z} \sim \mathcal{N}(\mu, \mathbf{\Sigma})$,

- *The marginal distribution of* $\mathbf{X}$ *is* $\mathbf{X} \sim \mathcal{N}(\mu_{\mathbf{x}}, \mathbf{\Sigma_{xx}})$.
- *The conditional distribution of* $\mathbf{Y}$ *given* $\mathbf{X} = \mathbf{x}$ *is* $\mathbf{x}$

$$\mathbf{Y} \mid \mathbf{X} = \mathbf{x} \sim \mathcal{N}\left(\mu_y + \mathbf{\Sigma}_{yx}\mathbf{\Sigma}_{xx}^{-1}\left(\mathbf{x} - \mu_x\right),\ \mathbf{\Sigma}_{yy} - \mathbf{\Sigma}_{yx}\mathbf{\Sigma}_{xx}^{-1}\mathbf{\Sigma_{xy}}\right).$$

- *X and Y are independent if and only if they are uncorrelated.*

Density Estimation with Gaussian Mixture Model(GMM)

Definition 5. Given a random variable X, the density function using k Gaussian mixtures is a convex combination of normal densities defined as

$$p(\mathbf{x}) = \sum_{i=1}^{k} z_i p(\mathbf{x} \mid i),$$

$$\text{where} \quad p(\mathbf{x}|i) = \mathcal{N}(\mathbf{x} \mid \mu^i, \mathbf{\Sigma}^i), \qquad p(i) = z_i, \qquad \sum_{i=1}^{k} z_i = 1, 0 \leq z_i \leq 1.$$

z_i is the mixing coefficient. It captures the probability of choosing component i. The greater its value, the more frequent component i will be selected during sampling.
The model is defined by its parameters. Associated to the model is the posterior probability $p(i \mid \mathbf{x})$ known as the responsibilities. Given a point $\mathbf{x}$, $p(i \mid \mathbf{x})$ is the probability that point $\mathbf{x}$ is from component i (i.e., a distribution on which component generated vector x). Following Bayes rule (BR), the responsibility is given by

$$p(i \mid \mathbf{x}) = \frac{p(i)p(\mathbf{x} \mid i)}{\sum_j p(j)\mathbf{p}(\mathbf{x} \mid j)}. \tag{2.11}$$

The density model is completely defined by the parameters $\mu = \left\{\mu^1, \cdots, \mu^k\right\}$, $\mathbf{\Sigma} = \left\{\mathbf{\Sigma}^1, \cdots, \mathbf{\Sigma}^k\right\}$, and $z = \{p(i), \cdots, p(k)\}$. Hence estimating the density is equivalent to finding the set of parameters that maximizes the log of the likelihood function given by

$$\ln \mathbf{p}\left(\mathbf{X} \mid z, \mu, \mathbf{\Sigma}\right) = \sum_{n=1}^{N} \ln \left\{ \sum_{i=1}^{K} z_i \mathcal{N}\left(\mathbf{x}_n | \boldsymbol{\mu}^i, \mathbf{\Sigma}^i\right) \right\}, \tag{2.12}$$

where $\mathbf{X}$ is a dataset of N samples.
Eqn 2.12 is maximized via the expectation maximization (EM) algorithm [21]. The intuition behind is straightforward. Given a dataset from a mixture of Gaussians, the parameters of each Gaussian can efficiently be computed if one knows from which Gaussian each observation was generated. Conversely if the data point's source is unknown (i.e. the only information available is that the points are from k different Gaussians. However, we do not know which point belongs to a specific component) and the parameters of the Gaussians are known, a guess can be made on which component generated a fixed data vector.

1 Initialize parameters $z, \mu, \mathbf{\Sigma}$;
2 **repeat**
3 | **Expectation step**: Given fix parameters, compute the responsibilities $\mathbf{p}(i \mid \mathbf{x})$;
4 | **Maximization Step**: Update the parameters using the current responsibilities
5 **until** *Untill convergence*;

Algorithm 1: Expectation maximization for Gaussian mixtures.

- **Expectation:** compute

$$p(i \mid \mathbf{x_n}) = \frac{z_i \mathcal{N}(\mathbf{x}_n \mid \mu^i, \mathbf{\Sigma}^i)}{\sum_j z_j \mathcal{N}(\mathbf{x}_n \mid \mu^j, \mathbf{\Sigma}^j)}.$$

- **Maximization:** Update the parameters

$$\begin{aligned}
\boldsymbol{\mu}^i_{\text{new}} &= \frac{1}{N_i} \sum_{n=1}^{N} \mathbf{x}_n, \\
\mathbf{\Sigma}^i_{\text{new}} &= \frac{1}{N_i} \sum_{n=1}^{N} p(i \mid \mathbf{x}_n) \left(\mathbf{x}_n - \boldsymbol{\mu}^i_{\text{new}}\right) \left(\mathbf{x}_n - \boldsymbol{\mu}^i_{\text{new}}\right)^{\mathrm{T}}, \\
z_i^{\text{new}} &= \frac{N_i}{N}, \\
N_i &= \sum_{n=1}^{N} p(i \mid \mathbf{x}_n).
\end{aligned}$$

EM converges to a local optimum and is numerically stable since the likelihood increases in each iteration .

The GMM can estimate any density function if given enough components. However, choosing the number of components manually can be difficult. The number of Gaussian components k can be chosen efficiently by setting k that maximizes the Bayesian Information Criteria.

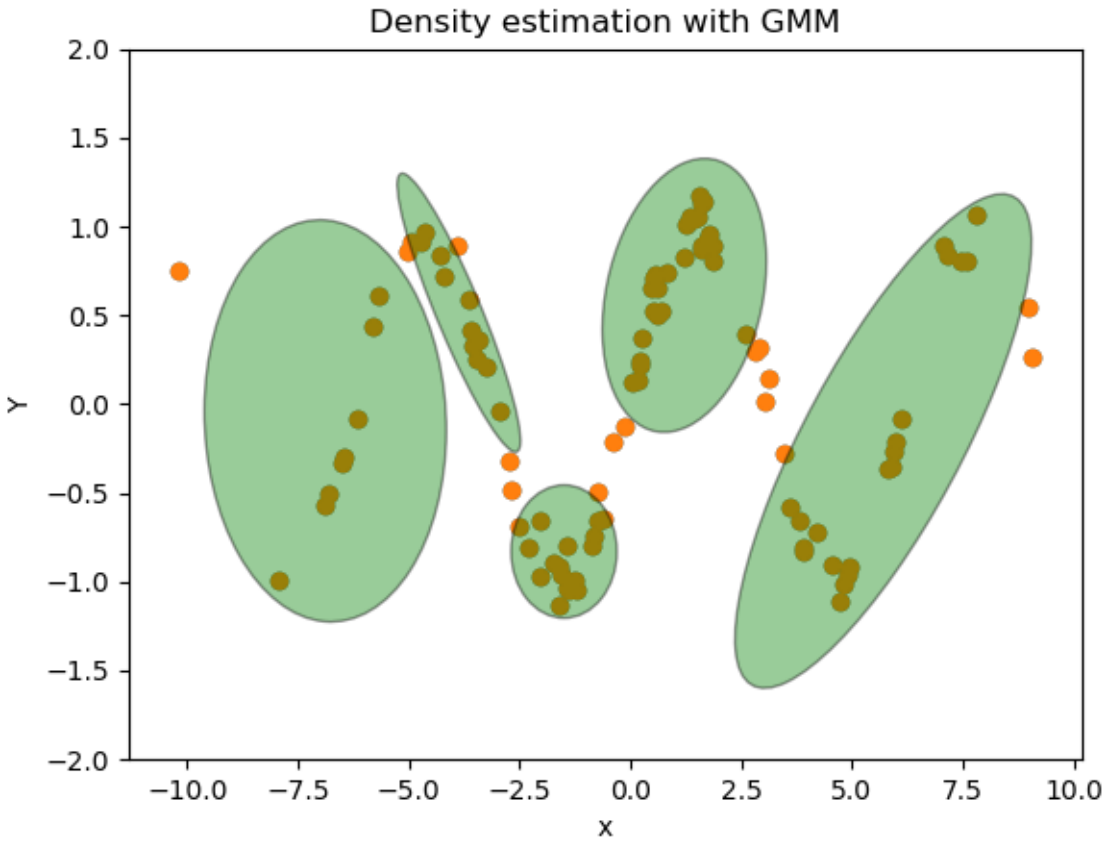
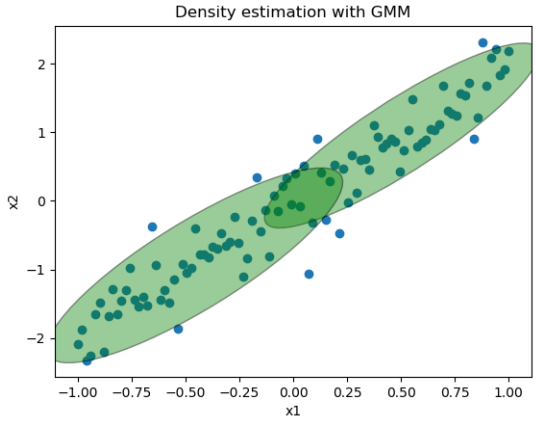


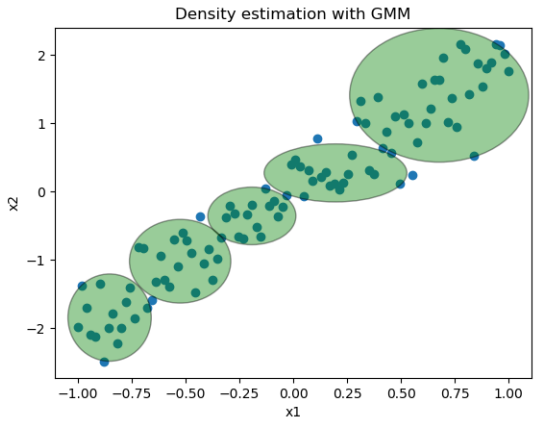
Figure 2.4.: Density estimation via GMM. Using 5 components, the model is partitioned into different clusters.

The time and computational efficiency of this model depends on the dimension (d_x) of the data vector. Which determines the number of parameters of the model. Hence the choice of the covariance matrix is an important issue. For large number of parameters, more samples are needed to train the model. This results in longer training time. The full covariance matrix allows for correlation ($\sigma_{x_1 x_2}$) between the features. For this reason, we can avoid the computation cost relative to the correlation by enforcing the correlation between the random variables to be zero (e.g. using Diagonal and Spherical covariance matrices) hence less parameters.

It is important to note that setting the correlation parameters of the model to zero, do not influence the nature of the correlation in the data set. The model will only set the correlation parameters to zero, and find the remaining optimal parameters. In this setting the model's learning ability will be unchanged by increasing the number of Gaussians as illustrated in Figure 2.5b and Figure 2.5a.



(a) Full covariance matrix and 2 Gaussians.



(b) Diagonal covariance matrix and 5 Gaussians.

2.5.1 Gaussian Mixture Regression

The joint probability density of two random variables X and Y can be estimated via GMM. Since the EM algorithm is unsupervised, no distinction is made between the input observation $\mathbf{x}_n$ and the output observation $\mathbf{y}_n$. Any link between the two random variables can then be estimated by using the learned density. We are interested in the relation ship between X and Y. The regression function

$$\begin{aligned} m : \mathbb{R}^{d_x} &\to \mathbb{R}, \\ x &\mapsto m(x) = \mathbb{E}(Y \mid X = \mathbf{x}), \end{aligned}$$

summarizes this relationship. Our goal is to estimate the regression function from the set of observations $\{\mathbf{x}_n, \mathbf{y}_n\}_{n=1}^N$ via GMM. This function is derived as follows

$$\begin{aligned}
\mathbb{E}(Y|X=\mathbf{x}) &= \int y \cdot p(y|\mathbf{x})dy \\
&= \int y \cdot \frac{p(y,\mathbf{x})}{p(x)}dy \\
&= \frac{\int y \cdot p(y,\mathbf{x})dy}{p(\mathbf{x})} \\
&\overset{\text{GMM}}{=} \frac{\int y \cdot \sum_i^k p(y,\mathbf{x} \mid i)p(i)dy}{p(\mathbf{x})} \\
&= \frac{\int y \cdot \sum_i^k p(y \mid \mathbf{x}, i)p(\mathbf{x} \mid i)p(i)dy}{p(\mathbf{x})} \\
&= \frac{\sum_i^k \int y \cdot p(y \mid \mathbf{x}, i)dy \cdot p(\mathbf{x} \mid i)p(i)}{p(\mathbf{x})} \\
&\overset{\text{BR}}{=} \sum_i^k \int y \cdot p(y \mid \mathbf{x}, i)dy \cdot p(i \mid \mathbf{x}) \\
&\overset{\text{Thm.4}}{=} \sum_i^k \left(\mu_y^i + \mathbf{\Sigma}_{yx}^i {\mathbf{\Sigma}_{xx}^i}^{-1}(\mathbf{x} - \mu_{\mathbf{x}}^i)\right) \cdot p(i|\mathbf{x})
\end{aligned} \tag{2.13}$$

$p(i \mid \mathbf{x})$ is the responsibility (Eqn 2.11). Using the notation from [17], and making the assumption that $d_y = 1$ the above product is vectorized

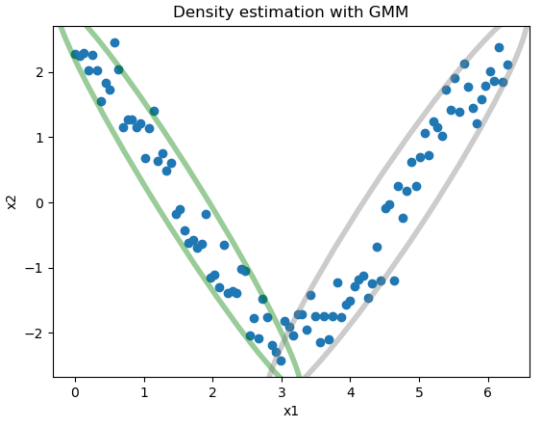
$$\mathbb{E}(Y|X=\mathbf{x}) = \varepsilon^\intercal(\mathbf{x})\chi \tag{2.14}$$

where

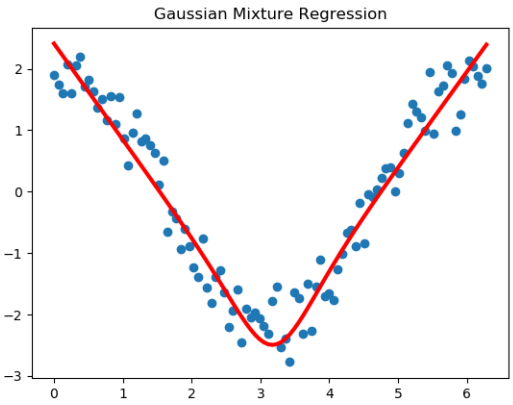
$$\varepsilon^\intercal : \mathbb{R}^{d_x} \to \mathbb{R}^{1\times((1+d_x)\cdot k)},$$
$$x \mapsto \varepsilon^\intercal(x) = \left[p(1|\mathbf{x}) \quad \cdots \quad p(k|\mathbf{x}) \quad p(1|\mathbf{x})(\mathbf{x} - \mu_{\mathbf{x}}^1)^\intercal {\mathbf{\Sigma}_{xx}^1}^{-\intercal} \cdots p(k|x)(x - \mu_x^k)^\intercal {\mathbf{\Sigma}_{xx}^k}^{-\intercal}\right],$$

$$\chi = \begin{bmatrix} \mu_y^1 \\ \vdots \\ \mu_y^k \\ {\mathbf{\Sigma}_{yx}^1}^T \\ \vdots \\ {\mathbf{\Sigma}_{yx}^k}^T \end{bmatrix} \in \mathbb{R}^{((1+d_x)\cdot k)\times 1},$$

where ε is a feature matrix , and χ a vector of parameters. GMR perform the regression by conditioning the unsupervised density estimate of the GMM. The GMM makes predictions on new input features by deriving the conditional distribution from the joint density.



(a) GMM with 2 components.



(b) Line fitted by GMR.

3 Related Works

Off-Policy Semi-Gradient

An important advance in the off-policy gradient estimation was made by the introduction of the off-policy gradient Theorem [5]. However, the theorem, in order to deliver a tractable estimate, introduces two approximations. Firstly, it considers a modified discounted infinite horizon return objective $\tilde{J}_\pi = \int \rho_\beta(s) V_\pi(s) ds$ where ρ_β is the stationary state distribution under the behavioral policy π_β. Secondly, in the derivation of the gradient,

$$\begin{aligned} \nabla_\theta J_\pi &= \nabla_\theta \int_S \rho_\beta(s) \int_A \pi_\theta(a|s) Q_\pi(s,a) da ds \\ &= \int_S \rho_\beta(s) \int_A \nabla_\theta \pi_\theta(a|s) Q_\pi(s,a) + \pi_\theta(a|s) \nabla_\theta Q_\pi(s,a) da ds \\ &\approx \int_S \rho_\beta(s) \int_A \nabla_\theta \pi_\theta(a|s) Q_\pi(s,a) da ds, \end{aligned} \tag{3.1}$$

the term $\pi_\theta(a|s) \nabla_\theta Q_\pi(s,a)$ is omitted (Equation 3.1). The authors provide a proof that the semi-gradient converges to the optimal policy in a discrete MDP setting. However, in more complex scenarios, where also other sources of error are introduced (such as functional approximation of the critic), the stability is not guaranteed [8].

Path-Wise Importance Sampling

Another technique to deliver an off-policy estimation is by employing importance sampling. More in detail, one can use a behavioural policy to collect the samples and then correct the probability of each trajectory using importance sampling [22, 23, 24]. An example of the gradient estimation with importance sampling is given by

$$\nabla_\theta J_\pi = \mathbb{E}\left[\sum_{t=0}^{T-1} \rho_t Q_\pi(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t|s_t) \right] \tag{3.2}$$

where $\rho_t = \prod_{z=0}^{t} \pi_\theta(a_z|s_z) / \pi_\beta(a_z|s_z)$. This technique is restricted only to stochastic policies and requires the knowledge of the behavioural policy π_β. Additionally importance sampling suffers from high variance, which grows exponentially in the number of steps.

Non-Parametric Off Policy Policy gradient (NOPG)

In [17], a full gradient estimate that does not suffer from the downside of importance sampling and semi-gradient methods was introduced. Based on a non-parametric Bellman equation, the closed form solution of the gradient for both deterministic and stochastic policy was computed. A

non-parametric bellman equation was previously computed in [25] where they used kernel density estimation to represent the system. The closed form solution of the value function was computed as well. But in contrast to Non-Parametric Dynamic Programming (NPDP) [25] that does not depend on the policy parameter, NOPG expresses the critic as a function of the policy parameter θ so that the analytic expression of the gradient is derived.

The problem with NOPG lies in the fact that non-parametric methods are not scalable (as there are as many parameters as the number of samples). Non-parametric regression requires a sample size that grows exponentially with the dimension of the sample space, thus reducing the performance of the algorithm. To circumvent this problem, we use Gaussian mixture models which is more flexible.

Conclusion

In the discrete state space, it is possible to solve for the value functions of a fixed policy directly using linear solvers. Furthermore optimal value functions can be obtained in polynomial time using dynamic programming. Unfortunately, exact methods fail to compute the optimal value functions in continuous spaces. Policy optimization directly maximize the objective function via gradient ascent by parameterizing the policy directly. The policy gradient theorem expresses the gradient of the objective with respect to the policy parameter. Two cases of policy gradient exist: on-policy gradient and off-policy gradient. The former is less sample efficient than the latter. Designing off-policy gradient algorithms is challenging. Semi-gradient methods suffer from high bias while importance sampling experiences high variance. Building a non-parametric Bellman equation and computing the full gradient estimate offers a better solution. However, the performance of non-parametric methods decreases as the number of samples increases. In the next chapter, we present an alternative solution that is scalable.

4 Policy Gradient Estimation via Gaussian Mixture Regression

In this chapter we present an off-policy method, with a full gradient estimate using GMM. Solving this optimization problem requires the integral of a non-linear function that is generally intractable. Two approaches are presented to handle this complexity. We follow the notation introduced in [17] and [26].

4.1 Problem Statement

RL algorithms are not yet able to deliver satisfying results in the real world. One of the reasons is due to their sample inefficiency (i.e they require a large number of samples to learn). Off-policy methods are of great importance in the design of RL algorithms. Sample efficient algorithms are designed via off-policy methods because these methods are able to learn about an optimal policy while following an exploratory policy, and learn under human guidance [27].

The goal of the learning agent is to maximize the return (Eqn. 2.3). In the off-policy setting the performance objective is adjusted to be the value function of the target policy averaged over the state distribution

$$J_{\pi_\theta} = \int_{\mathscr{S}} \mu_0(\mathbf{s}) V_{\pi_\theta}(\mathbf{s}) \mathbf{ds},$$

where $\mu_0(s)$ is the initial state distribution. The policy π_θ is a differentiable function of a weight vector $\theta \in \mathbb{R}^{d_\theta}$. The value function V_{π_θ} depends on the nature of the target policy. If the target policy is stochastic, integration is over the state and action spaces and the value function is obtained by solving the Bellman equation:

$$V_{\pi_\theta}(\mathbf{s}) = \int_{\mathscr{A}} \pi_\theta(\mathbf{a}|\mathbf{s}) \left(R(\mathbf{s},\mathbf{a}) + \gamma \int_{\mathscr{S}} V_{\pi_\theta}(\mathbf{s}') \, P(\mathbf{s}'|\mathbf{s},\mathbf{a}) \, \mathbf{ds}' \right) \mathbf{da}.$$

If the target policy is deterministic, we avoid the integral over the action space:

$$V_{\pi_\theta}(\mathbf{s}) = R(\mathbf{s}, \pi_\theta(\mathbf{s})) + \gamma \int_{\mathscr{S}} V_{\pi_\theta}(\mathbf{s}') \, P(\mathbf{s}' \mid \mathbf{s}, \pi_\theta(\mathbf{s})) \, \mathbf{ds}' \qquad \forall \mathbf{s} \in \mathscr{S}.$$

The aim of the optimization problem is to maximize J_{π_θ} subject to the Bellman equation.

$$\begin{aligned} \max_\theta \quad & J_{\pi_\theta} = \int \mu_0(\mathbf{s}) V_{\pi_\theta}(\mathbf{s}) \mathbf{ds} \\ \text{s.t} \quad & V_{\pi_\theta}(\mathbf{s}) = \int_{\mathscr{A}} \pi_\theta(\mathbf{a}|\mathbf{s}) \left(R(\mathbf{s},\mathbf{a}) + \gamma \int_{\mathscr{S}} V_{\pi_\theta}(\mathbf{s}') \, P(\mathbf{s}'|\mathbf{s},\mathbf{a}) \, \mathbf{ds}' \right) \mathbf{da} \quad \forall \mathbf{s} \in \mathscr{S}. \end{aligned} \tag{4.1}$$

For a deterministic policy, the constraint (Eqn. 4.1) becomes

$$V_{\pi_\theta}(\mathbf{s}) = R(\mathbf{s}, \pi_\theta(\mathbf{s})) + \gamma \int_{\mathscr{S}} V_{\pi_\theta}(\mathbf{s}') \, P(\mathbf{s}' \mid \mathbf{s}, \pi_\theta(\mathbf{s})) \, \mathbf{ds}' \qquad \forall \mathbf{s} \in \mathscr{S}. \tag{4.2}$$

Gradient ascent is used in the literature to solve the optimization problem. At each iteration, the update rule is given by

$$\theta \leftarrow \theta + \alpha \nabla_\theta J_{\pi_\theta}.$$

Computing the gradient ($\nabla_\theta J_{\pi_\theta}$) of the objective function analytically is unfeasible, except under special conditions, e.g when the objective function is quadratic and subject to linear constraints (section 5.3). The set of constraints in the continuous setting is generally neither guaranteed to be convex, nor linear. As a result, obtaining an expression of the gradient of the performance objective function with respect to the policy' s parameter θ is complex [17]. To circumvent this problem, a closed form solution of the value function via GMR (Eqn. 2.14) that is dependent on the policy's parameter is used.

4.2 Gaussian Mixture Model Bellman Equation

In this section we derive a closed form solution of the value function using GMR and use this solution to estimate the gradient of J_{π_θ}. Let us assume to have a dataset of n samples $D \equiv \{\mathbf{x}_i\}_{i=1}^n$ where $\mathbf{x}_i = (s_i, a_i, r_i, s_i', Q(s_i, a_i))$, $s_i, s_i' \in \mathbb{R}^{d_s}$, $a_i \in \mathbb{R}^{d_a}$, $r_i \in \mathbb{R}$, $Q(s_i, a_i) \in \mathbb{R}$. s_i, a_i, r_i, s_i' are sampled from the environment. Several methods can be used to estimate $Q(s_i, a_i)$ (e.g temporal difference, Monte Carlo, approximate dynamic programming).
In Section 2.2 we saw that given an MDP $< \mathscr{S}, \mathscr{A}, R, P, \gamma, \mu_0 >$ and a fixed policy π_θ we have an MRP $< \mathscr{S}, R_{\pi_\theta}, P_{\pi_\theta}, \gamma, \mu_0 >$ with

$$R_{\pi_\theta}(s) = \begin{cases} R(s, \pi_\theta(s)) & \text{if } \pi_\theta \text{ is deterministic} \\ \int_{a \in \mathscr{A}} R(s,a)\pi_\theta(a \mid s) da & \text{if } \pi_\theta \text{ is stochastic,} \end{cases}$$

$$P_{\pi_\theta}(s' \mid s) = \begin{cases} P(s' \mid s, \pi_\theta(s)) & \text{deterministic policy} \\ \int_{a \in \mathscr{A}} P(s' \mid s, a)\pi_\theta(a \mid s) da & \text{stochastic policy.} \end{cases}$$

From the definition of the reward function, the transition probabilities (Eqn. 2.2 and 2.1), and the definition of the conditional expectation of a continuous random variable (Eqn 2.13) , it follows that

$$\begin{aligned} R(s,a) &= \mathbb{E}[R \mid S = s, A = a] \\ &= \int_{r \in R} \frac{p(r,s,a)}{p(s,a)} r dr, \end{aligned}$$

and

$$P(s' \mid s, a) = \frac{p(s', s, a)}{p(s, a)}.$$

To estimate the reward function using GMM the mean and the covariance matrix for the i^{th} Gaussian component decomposes to

$$\mu^i = \begin{pmatrix} \mu^i_{s,a} \\ \mu^i_r \end{pmatrix} \quad , \qquad \mathbf{\Sigma}^i = \begin{pmatrix} \mathbf{\Sigma}^i_{(s,a),(s,a)} & \mathbf{\Sigma}^i_{(s,a),r} \\ \mathbf{\Sigma}^i_{r,(s,a)} & \mathbf{\Sigma}^i_{r,r} \end{pmatrix}.$$

Assume for simplicity that the policy π_θ is deterministic. We directly estimate the value and the reward function at each state with GMR (see Eqn 2.14)

$$\begin{aligned} R_{\pi_\theta}(s) &= R(s, \pi_\theta(s)) \\ &\approx \int_{r \in R} \frac{p(r, s, \pi_\theta(s))}{p(s, \pi_\theta(s))} r dr \\ &= \sum_i^k \int r p(r \mid s, \pi_\theta(s), i) dr \; p(i \mid \pi_\theta(s)) \\ &= \sum_i^k \left(\mu^i_r + \mathbf{\Sigma}^i_{r,(s,a)} {\mathbf{\Sigma}^i_{(s,a)(s,a)}}^{-1} \left(\begin{bmatrix} \mathbf{s} \\ \pi_\theta(s) \end{bmatrix} - \begin{bmatrix} \mu^i_{\mathbf{s}} \\ \mu^i_a \end{bmatrix} \right) \right) p(i|\mathbf{s}, \pi_\theta(\mathbf{s})) \\ &= \varepsilon^{\intercal}_{\pi_\theta}(s) \mathbf{r}. \end{aligned} \tag{4.3}$$

Similarly,

$$\begin{aligned} V_{\pi_\theta}(s) &= Q(s, \pi_\theta(s)) \\ &= \mathbb{E}\left[V \mid S = s, A = \pi_\theta(s) \right] \\ &\approx \varepsilon^{\intercal}_{\pi_\theta}(s) \mathbf{v}. \end{aligned} \tag{4.4}$$

where

$$\mathbf{r} = \begin{bmatrix} \mu^1_r & \cdots & \mu^k_r & {\mathbf{\Sigma}^1_{r,(s,a)}}^{\intercal} & \cdots & {\mathbf{\Sigma}^k_{r,(s,a)}}^{\intercal} \end{bmatrix}^{\intercal} \in \mathbb{R}^{((1+d_s+d_a)\cdot k)\times 1},$$
$$\mathbf{v} = \begin{bmatrix} \mu^1_v & \cdots & \mu^k_v & {\mathbf{\Sigma}^1_{v,(s,a)}}^{\intercal} & \cdots & {\mathbf{\Sigma}^k_{v,(s,a)}}^{\intercal} \end{bmatrix}^{\intercal} \in \mathbb{R}^{((1+d_s+da)\cdot k)\times 1},$$

$$\varepsilon^{\intercal}_{\pi_\theta} : \mathbb{R}^{d_s} \to \mathbb{R}^{1\times((1+d_s+d_a)\cdot k)}$$

$$\begin{aligned} \mathbf{s} \mapsto \varepsilon^{\intercal}_{\pi_\theta}(s) = \Big[& p(1|\mathbf{s}, \pi_\theta(\mathbf{s})) \quad \cdots \quad p(k|\mathbf{s}, \pi_\theta(\mathbf{s})) \quad p(1|\mathbf{s}, \pi_\theta(\mathbf{s})) \left(\mathbf{x}_{\mathbf{s},\pi_\theta(\mathbf{s})} - \mu^1_{s,a} \right)^{\intercal} {\mathbf{\Sigma}^1_{(s,a)(s,a)}}^{-\intercal} \cdots \\ & p(k|\mathbf{s}, \pi_\theta(\mathbf{s})) (\mathbf{x}_{\mathbf{s},\pi_\theta(\mathbf{s})} - \mu^k_{s,a})^{\intercal} {\mathbf{\Sigma}^k_{(s,a)(s,a)}}^{-\intercal} \Big], \end{aligned} \tag{4.5}$$

with

$$\mathbf{x}_{s,\pi_\theta(\mathbf{s})} = \begin{bmatrix} \mathbf{s} \\ \pi_\theta(s) \end{bmatrix}, \; \mu^i_{s,a} = \begin{bmatrix} \mu^i_{\mathbf{s}} \\ \mu^i_a \end{bmatrix}.$$

We denote the estimate of the reward and the value function as follows:

$$\begin{aligned}\hat{R}_{\pi_\theta}(s) &= \varepsilon_{\pi_\theta}^{\intercal}(s)\mathbf{r},\\ \hat{V}_{\pi_\theta}(s) &= \varepsilon_{\pi_\theta}^{\intercal}(s)\mathbf{v},\end{aligned} \tag{4.6}$$

where ε_{π_θ} is a vector valued function that depends of the parameter θ. Consequently its value changes after each policy update. The vectors $\mathbf{r}$ and $\mathbf{v}$ on the other hand are independent of the parameter θ.
We built a closed form solution of the reward function and the value function using the GMR. Recall that our goal is to express the set of constraints of our optimization problem as a system of linear equations. To achieve this goal, it remains to find a linear expression for the term $\int_{\mathscr{S}} V_{\pi_\theta}(\mathbf{s}')\, P(\mathbf{s}' \mid \mathbf{s}, \pi_\theta(\mathbf{s}))\, \mathbf{ds}'$. Solving this integral is intractable since $V_{\pi_\theta}(s') \approx \varepsilon^{\intercal}(s')\mathbf{v}$ and $\varepsilon^{\intercal}$ is a non-linear function. We present two methods to compute this integral.

Taylor series Expansion

$$\begin{aligned}\int_{s\in\mathscr{S}} V_{\pi_\theta}(\mathbf{s}')\, P(\mathbf{s}' \mid \mathbf{s}, \pi_\theta(\mathbf{s}))\, \mathbf{ds}' &= \frac{\int p(s', s, \pi_\theta(s)) V_{\pi_\theta}(s') ds'}{p(s, \pi_\theta(s))}\\ &\overset{\text{GMM}}{=} \frac{\int \sum_{i=1}^{k} p(s', s, \pi_\theta(s)|i) p(i) V_{\pi_\theta}(s') ds'}{p(s, \pi_\theta(s))}\\ &= \frac{\int \sum_{i=1}^{k} p(s'|s, \pi_\theta(s), i) p(s, \pi_\theta(s)|i) p(i) V_{\pi_\theta}(s') ds'}{p(s, \pi_\theta(s))}\\ &= \sum_{i=1}^{k} \int p(s'|s, \pi_\theta(s), i) V_{\pi_\theta}(s') ds' p(s, \pi_\theta(s)|i) \frac{p(i)}{p(s, \pi_\theta(s))}\\ &\overset{\text{BR}}{=} \sum_{i=1}^{k} p(i|s, \pi_\theta(s)) \underbrace{\int p(s'|s, \pi_\theta(s), i) V_{\pi_\theta}(s') ds'}_{\boldsymbol{B}}\end{aligned} \tag{4.7}$$

The term $\boldsymbol{B}$ is an integral of a non-linear function (V_{π_θ}), which is hard to solve. Function V_{π_θ} can be linearized using the *first-order Taylor expansion* at a specific point y. The mean of the next state seems to be a suitable choice, given that it is the point at which the probability density function is maximum

$$V_{\pi_\theta}(s') \approx V_{\pi_\theta}(y) + (s' - y)^{\intercal} \nabla_{s'} V_{\pi_\theta}(s')\big|_{s'=y},$$

with $\nabla_{s'} V_{\pi_\theta}(s')|_{s'=y} = \nabla V_{\pi_\theta}(y)$.
By linearizing in $\mu_{s'}^i$ and substituting the value function in Equation(4.7) we obtain

$$\int_{s\in\mathscr{S}} V_{\pi_\theta}(\mathbf{s}')\, P(\mathbf{s}' \mid \mathbf{s}, \pi_\theta(\mathbf{s}))\, \mathbf{ds}' \approx \sum_{i=1}^{k} p(i|s, \pi_\theta(s)) \int_{s\in\mathscr{S}'} p(s'|s, \pi_\theta(s), i)(V_{\pi_\theta}(\mu_s'^i) + (s' - \mu_{s'}^i)^{\intercal} \nabla_z V_{\pi_\theta}(z))|_{\mu_{s'}^i} ds'$$

where

$$\begin{aligned} p(s'|s, \pi_\theta(s), i) &= \mathcal{N}(s'|\mu, \boldsymbol{\Sigma}),\\ \text{with} \qquad \mu &= \boldsymbol{\mu}_{s'}^i + \boldsymbol{\Sigma}_{s',(s,a)}^i {\boldsymbol{\Sigma}_{(s,a)(s,a)}^i}^{-1} (\mathbf{x}_{s,\pi_\theta(s)} - \mu_{s,a}^i),\\ \Sigma &= \Sigma_{s's'} - \Sigma_{s'(s,a)} {\Sigma_{(s,a)(s,a)}}^{-1} \Sigma_{(s,a)s'}. \end{aligned}$$

It follows that

$$
\begin{aligned}
&\int_{s\in\mathscr{S}} V_{\pi_\theta}(\mathbf{s}')P(\mathbf{s}' \mid \mathbf{s}, \pi_\theta(\mathbf{s}))\,\mathrm{d}\mathbf{s}' \approx \\
&\qquad \sum_{i=1}^{k} p(i|s,\pi_\theta(s)) \left(V_{\pi_\theta}(\mu^i_{s'}) + (\boldsymbol{\mu}^i_{s'} + \boldsymbol{\Sigma}^i_{s'a}{\boldsymbol{\Sigma}^i_{(s,a)(s,a)}}^{-1}(\mathbf{x}_{s,\pi_\theta(s)} - \boldsymbol{\mu}^i_{s,a}) - \mu^i_{s'})^\intercal \nabla_z V_{\pi_\theta}(z)|_{z=\boldsymbol{\mu}^i_{s'}} \right) \\
&\qquad = \sum_{i=1}^{k} p(i|s,\pi_\theta(s)) \left(V_{\pi_\theta}(\boldsymbol{\mu}^i_{s'}) + \boldsymbol{\Sigma}^i_{s',(s,a)}{\boldsymbol{\Sigma}^i_{(s,a)(s,a)}}^{-1}(x_{s,\pi_\theta(s)} - \boldsymbol{\mu}^i_{s,a}) \nabla_z V_{\pi_\theta}(z)|_{z=\boldsymbol{\mu}^i_{s'}} \right) \\
&\qquad \approx \sum_{i=1}^{k} p(i|s,\pi_\theta(s)) \left(\varepsilon^\intercal_{\pi_\theta}(\mu^i_{s'})\mathbf{v} + \boldsymbol{\Sigma}^i_{s'(s,a)}{\boldsymbol{\Sigma}^i_{(s,a)(s,a)}}^{-1}(\mathbf{x}_{s,\pi_\theta(s)} - \boldsymbol{\mu}^i_{s,a}) \nabla_z \varepsilon^\intercal_{\pi_\theta}(z)|_{z=\boldsymbol{\mu}^i_{s'}} \mathbf{v} \right) \\
&\qquad = \sum_{i=1}^{k} p(i|s,\pi_\theta(s)) \left(\varepsilon^\intercal_{\pi_\theta}(\mu^i_{s'}) + \boldsymbol{\Sigma}^i_{s'(s,a)}{\boldsymbol{\Sigma}^i_{(s,a)(s,a)}}^{-1}(\mathbf{x}_{s,\pi_\theta(s)} - \boldsymbol{\mu}^i_{s,a}) \nabla_z \varepsilon^\intercal_{\pi_\theta}(z)|_{z=\boldsymbol{\mu}^i_{s'}} \right) \mathbf{v} \\
&\qquad = \varepsilon^\intercal_{\pi_\theta}(s)\mathbf{P}_{\pi_\theta}\mathbf{v},
\end{aligned}
$$

where

$$
\mathbf{P}_{\pi_\theta} = \begin{bmatrix} \varepsilon^\intercal_{\pi_\theta}(\mu^1_{s'}) \\ \vdots \\ \varepsilon^\intercal_{\pi_\theta}(\mu^k_{s'}) \\ {\Sigma^1_{s',(s,a)}}^\intercal \nabla_z \epsilon^\intercal_{\pi_\theta}(z)|_{z=\mu^1_{s'}} \\ \vdots \\ {\Sigma^k_{s',(s,a)}}^\intercal \nabla_z \varepsilon^\intercal_{\pi_\theta}(z)|_{z=\mu^k_{s'}} \end{bmatrix} \in \mathbb{R}^{(k+k\cdot(d_s+d_a))\times(1+d_s+d_a)k},
$$

$$
\varepsilon^\intercal_{\pi_\theta}(\mu^i_{s'}) \in \mathbb{R}^{1\times((1+d_s+d_a)\cdot k)},
$$

$$
\nabla_z \varepsilon^\intercal_{\pi_\theta}(z)|_{z=\mu^i_{s'}} = \begin{bmatrix} \frac{\partial \epsilon_1}{\partial z_1} & \cdots & \frac{\partial \epsilon_{(1+d_{s'})\cdot k}}{\partial z_1} \\ \vdots & & \vdots \\ \frac{\partial \epsilon_1}{\partial z_{d_{s'}}} & \cdots & \frac{\partial \epsilon_{(1+d_{s'})\cdot k}}{\partial z_{d_{s'}}} \end{bmatrix} \in \mathbb{R}^{(d_s+d_a)\times((1+d_s+d_a)k)}.
$$

Linearizing using Taylor's first order approximation results to a simple linear equation with a tractable integral. However this method introduces an error that grows rapidly away from the operating point ($\mu_{s'}$). To avoid this error, we use Monte Carlo method to estimate the integral. Taylor's first order approximation is indeed computationally efficient, but provides a high biased estimation, while Monte Carlo estimation is unbiased but requires many samples to reduce the variance, hence more computationally demanding.

We want to express the term $\int_{s\in\mathscr{S}} V_{\pi_\theta}(\mathbf{s}')\, P(\mathbf{s}' \mid \mathbf{s}, \pi_\theta(\mathbf{s}))\, \mathbf{ds}'$ in closed form.

$$\begin{aligned}
\int_{s\in\mathscr{S}} V_{\pi_\theta}(\mathbf{s}')\, P(\mathbf{s}' \mid \mathbf{s}, \pi_\theta(\mathbf{s}))\, \mathbf{ds}' &\overset{\text{Eqn.4.7}}{=} \sum_i^k p(i|s,\pi_\theta(s)) \int p(s'|s,\pi_\theta(s),i) V_{\pi_\theta}(s') ds' \\
&\approx \sum_i^k p(i|s,\pi_\theta(s)) \int p(s'|s,\pi_\theta(s),i) \varepsilon_{\pi_\theta}^{\intercal}(s') \mathbf{v} ds' \\
&= \sum_i^k \int \varepsilon_{\pi_\theta}^{\intercal}(s') p(s'|s,\pi_\theta(s),i) ds' p(i|s,\pi_\theta(s)) \mathbf{v} \\
&= \sum_i^k \int \varepsilon_{\pi_\theta}^{\intercal}(s') p(s'|s,\pi_\theta(s),i) ds' p(i|s,\pi_\theta(s)) \mathbf{v}
\end{aligned}$$

Notice that, the correlation parameter of the model between the pair $\{s', s\}$ and the pair $\{s', a\}$ can be set to zero. It follows from the third point of Thm. (4) that s' and s are independent. Moreover, s' and a are independent. As a result $p(s' \mid s, \pi_\theta(s), i) = p(s', i)$. Hence,

$$\begin{aligned}
\int_{s\in\mathscr{S}} V_{\pi_\theta}(\mathbf{s}')\, P(\mathbf{s}' \mid \mathbf{s}, \pi_\theta(\mathbf{s}))\, \mathbf{ds}' &\approx \sum_i^k \int_{\mathscr{S}} \varepsilon_{\pi_\theta}^{\intercal}(s') p(s'|i) ds' p(i|s,\pi_\theta(s)) \mathbf{v} \\
&= \varepsilon_{\pi_\theta}^{\intercal}(s) \mathbf{P}_{\pi_\theta} \mathbf{v} \qquad (4.8)
\end{aligned}$$

where

$$P_{\pi_\theta} = \begin{bmatrix} \int \varepsilon_{\pi_\theta}^{\intercal}(s') p(s' \mid 1) ds' \\ \vdots \\ \int \varepsilon_{\pi_\theta}^{\intercal}(s') p(s' \mid k) ds' \\ \mathbf{O} \end{bmatrix} \in \mathbb{R}^{(k+(ds+da)k)\times(1+ds+da)k}, \qquad (4.9)$$

$$\mathbf{O} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} \in \mathbb{R}^{(ds+da)k\times((1+ds+da)k)}$$

The product entry in the matrix P_θ is computed using Monte Carlo

$$\int \varepsilon^{\intercal}(s') p(s' \mid i) ds' \approx \frac{1}{N^p} \sum_{j=1}^{N^p} \varepsilon^{\intercal}(s'_j),$$
$$\text{with } s'_j \sim \mathcal{N}(\mu^i_{s'}, \mathbf{\Sigma}^i_{s's'}).$$

With the above estimates, the Bellman equation (Eqn. 4.2) becomes

$$\hat{V}_{\pi_\theta}(s) = \hat{R}_{\pi_\theta}(s) + \gamma \varepsilon^{\intercal}(s) \mathbf{P}_{\pi_\theta} \mathbf{v}$$

We name the above equation to GMR Bellman equation.

Theorem 5. *The GMM Bellman equation has a fixed point solution given by*

$$\hat{V}^*_{\pi_\theta}(s) = \varepsilon^{\intercal}_{\pi_\theta}(s)\mathbf{\Lambda}^{-1}_{\pi_\theta}\mathbf{r}$$

with $\mathbf{\Lambda}_{\pi_\theta} = \mathbf{I} - \gamma\mathbf{P}_{\pi_\theta}$

.

Proof. The results follows directly from the derivation above. We show that $\mathbf{v} = {\mathbf{\Lambda}_{\pi_\theta}}^{-1}\mathbf{r}$.

$$\begin{aligned}
\hat{V}_{\pi_\theta}(s) &= \hat{R}_{\pi_\theta}(s) + \gamma\varepsilon^{\intercal}(s)\mathbf{P}_{\pi_\theta}\mathbf{v} \\
\varepsilon^{\intercal}(s)\mathbf{v} &= \varepsilon^{\intercal}(s)\mathbf{r} + \gamma\varepsilon^{\intercal}(s)\mathbf{P}_{\pi_\theta}\mathbf{v} \\
\mathbf{v} &= \mathbf{r} + \gamma\mathbf{P}_{\pi_\theta}\mathbf{v} \\
\mathbf{v} &= {\mathbf{\Lambda}_{\pi_\theta}}^{-1}\mathbf{r}
\end{aligned}$$

It follows directly from Eqn. 4.6 that

$$\hat{V}^*_{\pi_\theta}(s) = \varepsilon^{\intercal}_{\pi_\theta}(s)\mathbf{\Lambda}^{-1}_{\pi_\theta}\mathbf{r}$$

□

See Appendix A.3 for the proof about the invertibility of $\mathbf{\Lambda}_{\pi_\theta}$.
We can now substitute the value function in the objective function by its estimate in closed form.

$$\begin{aligned}
J_{\pi_\theta} &= \int_{\mathscr{S}} \mu_0(\mathbf{s})V_{\pi_\theta}(\mathbf{s})ds \\
&\approx \int_{\mathscr{S}} \mu_0(\mathbf{s})\hat{V}_{\pi_\theta}(\mathbf{s})ds \\
&= \int_{\mathscr{S}} \mu_0(\mathbf{s})\varepsilon^{\intercal}_{\pi_\theta}(s)\mathbf{v}^*_{\pi_\theta}ds, \quad \text{with } \mathbf{v}^*_{\pi_\theta} = \mathbf{\Lambda}^{-1}_{\pi_\theta}\mathbf{r} \\
&= \int_{\mathscr{S}} \mu_0(\mathbf{s})\varepsilon^{\intercal}_{\pi_\theta}(s)ds(\mathbf{v}^*{}_{\pi_\theta})
\end{aligned}$$

Following the notation in [17] we define $\epsilon^{\intercal}_{\pi_\theta,0} = \int \mu_0(s)\epsilon^{\intercal}_{\pi_\theta}(s)ds$. This quantity is approximated using Monte Carlo simulation.

$$\int_{\mathscr{S}} \mu_0(s)\epsilon^{\intercal}_{\pi_\theta}(s)ds \approx \frac{1}{N^I}\sum_{j=1}^{N^I} \varepsilon^{\intercal}_{\pi_\theta}(s_j) \quad \text{with } s_j \sim \mu_0(s) \tag{4.10}$$

4.2.1 Policy Gradient Computation

The gradient of the objective function can now be computed analytically using the closed form solution of $\hat{V}^*_{\pi_\theta}$,

$$\begin{aligned}
\nabla_\theta\hat{V}^*_{\pi_\theta}(s) &= \frac{\partial}{\partial\theta}\left(\varepsilon^{\intercal}_{\pi_\theta}(s)\Lambda^{-1}_{\pi_\theta}\mathbf{r}\right) \\
&= \frac{\partial}{\partial\theta}\left(\varepsilon^{\intercal}_{\pi_\theta}(s)\Lambda^{-1}_{\pi_\theta}\right)\mathbf{r} \\
&= \frac{\partial}{\partial\theta}\varepsilon^{\intercal}_{\pi_\theta}(s)\mathbf{\Lambda}^{-1}_{\pi_\theta}\mathbf{r} + \varepsilon^{\intercal}_{\pi_\theta}(s)\frac{\partial}{\partial\theta}\mathbf{\Lambda}^{-1}_{\pi_\theta}\mathbf{r} \\
&= \frac{\partial}{\partial\theta}\varepsilon^{\intercal}_{\pi_\theta}(s)\mathbf{\Lambda}^{-1}_{\pi_\theta}\mathbf{r} + \varepsilon^{\intercal}_{\pi_\theta}(s)\frac{\partial}{\partial\theta}(\mathbf{I} - \gamma\mathbf{P}_{\pi_\theta})^{-1}\mathbf{r} \\
&= \frac{\partial}{\partial\theta}\varepsilon^{\intercal}_{\pi_\theta}(s)\Lambda^{-1}_{\pi_\theta}\mathbf{r} + \gamma\varepsilon^{\intercal}_{\pi_\theta}(s)\mathbf{\Lambda}^{-1}_{\pi_\theta}\left(\frac{\partial}{\partial\theta}\mathbf{P}_{\pi_\theta}\right)\Lambda^{-1}_{\pi_\theta}\mathbf{r}
\end{aligned}$$

With the derivative of the value function we compute the gradient of the objective function. In the derivation we follow the notation in [17],

$$\begin{aligned}
\nabla_\theta J_{\pi_\theta} &\approx \nabla_\theta \hat{J}_{\pi_\theta} \\
&= \nabla_\theta \int_{\mathscr{S}} \mu_0(s) \hat{V}_{\pi_\theta}(s) ds \\
&= \int_{\mathscr{S}} \mu_0(s) \nabla_\theta \hat{V}_{\pi_\theta}(s) ds \\
&= \int_{\mathscr{S}} \mu_0(s) \left(\frac{\partial}{\partial \theta} \varepsilon_{\pi_\theta}^{\intercal}(s) \Lambda_{\pi_\theta}^{-1} \mathbf{r} + \gamma \varepsilon_{\pi_\theta}^{\intercal}(s) \mathbf{\Lambda}_{\pi_\theta}^{-1} \left(\frac{\partial}{\partial \theta} \mathbf{P}_{\pi_\theta} \right) \Lambda_{\pi_\theta}^{-1} \mathbf{r} \right) ds \\
&= \left(\int_{\mathscr{S}} \mu_0(s) \frac{\partial}{\partial \theta} \varepsilon_{\pi_\theta}^{\intercal}(s) ds \right) \Lambda_{\pi_\theta}^{-1} \mathbf{r} + \gamma \left(\int_{\mathscr{S}} \mu_0(s) \varepsilon_{\pi_\theta}^{\intercal}(s) ds \right) \mathbf{\Lambda}_{\pi_\theta}^{-1} \left(\frac{\partial}{\partial \theta} \mathbf{P}_{\pi_\theta} \right) \Lambda_{\pi_\theta}^{-1} \mathbf{r} \\
&= \left(\frac{\partial}{\partial \theta} \int_{\mathscr{S}} \mu_0(s) \varepsilon_{\pi_\theta}^{\intercal}(s) ds \right) \Lambda_{\pi_\theta}^{-1} \mathbf{r} + \gamma \left(\int_{\mathscr{S}} \mu_0(s) \varepsilon_{\pi_\theta}^{\intercal}(s) ds \right) \mathbf{\Lambda}_{\pi_\theta}^{-1} \left(\frac{\partial}{\partial \theta} \mathbf{P}_{\pi_\theta} \right) \Lambda_{\pi_\theta}^{-1} \mathbf{r} \\
&= \frac{\partial}{\partial \theta} \varepsilon_{\pi_\theta,0}^{\intercal} \mathbf{v}_{\pi_\theta}^{*} + \gamma \mu_{\pi_\theta}^{\intercal} \left(\frac{\partial}{\partial \theta} \mathbf{P}_{\pi_\theta} \right) \mathbf{v}_{\pi_\theta}^{*},
\end{aligned}$$

where $\mu_{\pi_\theta} = \mathbf{\Lambda}_{\pi_\theta}^{-\intercal} \epsilon_{\pi_\theta,0}$.

In [17], it has been shown empirically that the term μ_{π_θ} is an estimate of the state distribution. For a discrete state space, this reduces to the initial state distribution. While the value function quantifies the expected return obtained from starting at a fixed state s and following the policy π_θ, μ_{π_θ} quantifies the performance resulting from starting at the initial state distribution, following policy π_θ up to the present state.
The policy gradient obtained by modelling the Bellman equation via GMR is given by

$$\nabla_\theta \hat{J}_{\pi_\theta} = \frac{\partial}{\partial \theta} \varepsilon_{\pi_\theta,0}^{\intercal} \mathbf{v}_{\pi_\theta}^{*} + \gamma \mu_{\pi_\theta}^{\intercal} \left(\frac{\partial}{\partial \theta} \mathbf{P}_{\pi_\theta} \right) \mathbf{v}_{\pi_\theta}^{*}. \tag{4.11}$$

From the estimate of the policy gradient given in the above equation, the optimal policy can be found in a batch and off-policy setting, provided enough data is given to estimate the reward function and the critic. The process is formally described in Algorithm 2.
In contrast to the matrix $\mathbf{P}_{\pi_\theta}$ in [17] that is stochastic and an estimate of the system's dynamics in an MDP, the matrix $\mathbf{P}_{\pi_\theta}$ in our model (Eqn. 4.11) is not stochastic, but results from the estimation of the integral of a non-linear function. Furthermore the matrix $\mathbf{P}_{\pi_\theta}$ in [17] is of dimension $n \times n$ where n is the number of samples. While the dimension of the matrix P_{π_θ} in Eqn. (4.11) is only $(1 + d_s + d_a)k \times (1 + d_s + d_a)k$ where k is the number of Gaussian components. As a result the complexity of our model is bounded, even if the amount of data is unbounded.

The pseudocode for the algorithm is provided below. We name it Policy Optimization via Experts' Mixture (POEM). This is because a GMM is a particular case of a machine learning framework, the mixture of experts [28]. For the GMM, each expert is encoded by a Gaussian.

Input:

Dataset D $\equiv \{\boldsymbol{s}_i, \boldsymbol{a}_i, r_i, \boldsymbol{s}'_i, \}_{i=1}^n$;
Parameterized policy π_θ;
Learning rate α;
Discount factor γ;
Number of components k.

Output:

Optimized policy π_θ^*

Estimate the action value function $q(s_i, a_i)$ (e.g. via approximate dynamic programming);
Build new dataset $D' \equiv \{\boldsymbol{s}_i, \boldsymbol{a}_i, r_i, \boldsymbol{s}'_i, q(\boldsymbol{s}_i, \boldsymbol{a}_i)\}_{i=1}^n$;
Train GMM on the dataset D' via EM such that the matrices $\boldsymbol{\Sigma}_{s',s}$, $\boldsymbol{\Sigma}_{s',a}$ are diagonal matrices.;
while *not converged* **do**
    Compute $\varepsilon_{\pi_\theta,0}$ as in Eqn 4.10 and ε_{π_θ} as in Eqn 4.5. ;
    Compute $\mathbf{P}_{\pi_\theta}$;
    Built matrix $\boldsymbol{\Lambda}_{\pi_\theta} = \mathbf{I} - \gamma \mathbf{P}_{\pi_\theta}$;
    Solve $\mathbf{r} = \boldsymbol{\Lambda}_{\pi_\theta} \mathbf{v}^*_{\pi_\theta}$, and $\varepsilon_{\pi,0} = \boldsymbol{\Lambda}^\top_{\pi_\theta} \mu_{\pi_\theta}$ for μ_{π_θ} and $\mathbf{v}^*_{\pi_\theta}$ via conjugate gradient;
    Estimate the critic $\hat{V}(s) = \varepsilon^\intercal_{\pi_\theta}(s) \mathbf{v}^*_{\pi_\theta}$;
    Estimate the objective $\hat{J}_{\pi_\theta} = \varepsilon^\intercal_{\pi_\theta,0} \mathbf{v}^*_{\pi_\theta}$;
    Update θ using the gradient ascent update $\theta \leftarrow \theta + \alpha \nabla \hat{J}_{\pi_\theta}$.
end

Algorithm 2: Policy Optimization via Experts' Mixture (POEM).

5 Experiment

In this chapter we present the experimental settings and the results. The chapter begins with the description of the main technologies used in the implementation of the algorithm. Then we provide a thorough description of the environments on which the experiments were performed. We further describe and analyse the results obtained. Our analysis is based on three main aspects: Predicting the value function V_{π_θ}, the learning behaviour of the algorithm, and lastly the direction of the gradient computed by the algorithm.

5.1 Software Details

POEM is implemented in python (3) [29]. For efficient Mathematical computation, we use several libraries. The library Numpy (1.18.3) [30] is used for matrix manipulation. Automatic differentiation and the implementation of neural networks (actor) are provided by Pytorch (1.5.0) [31]. Training an GMM is done with Scikit-learn (0.22.2) [32]. The library "Reinforcement Learning Helper" (HeRL) is used for RL tasks. In addition to the already mentioned libraries, HeRL is also based on OpenAI Gym [33], which is a framework for RL tasks. Furthermore, the HeRL libary provides a framework to analyse RL algorithms.

5.2 The Swing-up Pendulum

The swing-up pendulum is a classic control environment. Under the OpenAI Gym library it is known as the Pendulum-v0 environment. Following the setting provided by this environment, the starting position of the pendulum is random, and the goal is to swing the pendulum up so that it is upright and remains in that position.

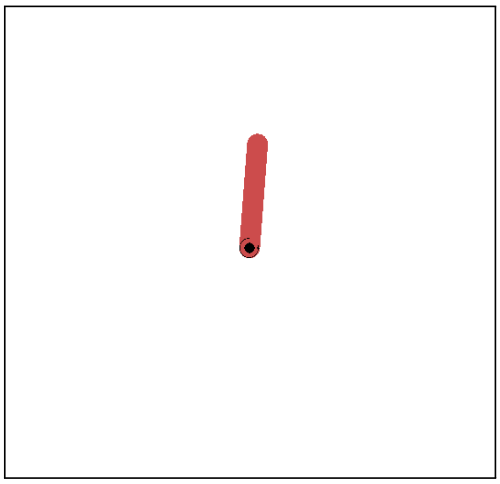


Figure 5.1.: A stable pendulum.

The state space is defined by the angle $\alpha \in [-\pi,\ \pi]$ and the velocity $\dot{\alpha} \in [-8.0,\ 8.0]$. Precisely, it is given by the 3-tuple $(\cos\alpha,\ \sin\alpha,\ \dot{\alpha})$. This can be converted to a 2-tuple $(\alpha,\ \dot{\alpha})$, with

$\alpha = \arctan 2(\sin\alpha, \cos\alpha)$. The action consist of a continuous torque $u \in [-2, 2]$. The reward equation is given by

$$R(s, u) = -\left(\alpha^2 + 0.1\dot{\alpha}^2 + 0.001u^2\right).$$

The new state is given by

$$s' = (\alpha_{new},\ \dot{\alpha}_{new}),$$

where

$$\dot{\alpha}_{new} = \dot{\alpha} + \left(\frac{-3g}{2l}\sin(\alpha + \pi) + \frac{3}{ml^2}u\right)dt,$$
$$\alpha_{new} = \alpha + \dot{\alpha}_{new}dt,$$

with $dt = 0.05,\ g = 10.0,\ m = 1.0,\ l = 1.0$.
The default maximum length of an episode is set up to 200. The GMM estimates the probability in the joint space $(\alpha,\ \dot{\alpha},\ u,\ r,\ \alpha_{new},\ \dot{\alpha}_{new},\ q)$. The Gaussians of the mixture model are 7-dimensional.

5.3 Linear Quadratic Regulator (LQR)

Another problem on which our algorithm was tested is the LQR problem. The LQR problem is a special case of an MDP that has an exact solution, regardless of the continuous nature of the state and action spaces [34]. As a result, several problems in robotics are reduced to this framework. For this problem we introduce the notations x_t and u_t, that refer respectively to the state and action at time t. The problem is defined by quadratic rewards

$$R(x_t, u_t) = -\mathbf{x}_t^\top \mathbf{Q}\mathbf{x}_t - \mathbf{u}_t^\top \mathbf{R}\mathbf{u}_t,$$

and linear state transitions

$$\boldsymbol{x}_{t+1} = \mathbf{A}\boldsymbol{x}_t + \mathbf{B}\boldsymbol{u}_t.$$

The optimization problem is formulated as

$$\max_{\boldsymbol{u}_t} J = -\sum_{t=0}^{\infty}\gamma^t\left(\boldsymbol{x}_t^\top \mathbf{Q}\mathbf{x}_t + \boldsymbol{u}_t^\top \mathbf{R}\boldsymbol{u}_t\right)dt$$
$$\text{s.t. } \boldsymbol{x}_{t+1} = \mathbf{A}\boldsymbol{x}_t + \mathbf{B}\boldsymbol{u}_t \quad \forall t$$

where $\boldsymbol{x}_t \in \mathbb{R}^{d_x}, \boldsymbol{u}_t \in \mathbb{R}^{d_u}, \mathbf{Q} \in \mathbb{R}^{d_x \times d_x}, \mathbf{R} \in \mathbb{R}^{d_u \times d_u}, \mathbf{A} \in \mathbb{R}^{d_x \times d_x}, \mathbf{B} \in \mathbb{R}^{d_x \times d_u}, \gamma \in [0, 1)$, and $\mathbf{x}_0 \in \mathbb{R}^{d_x}$. We consider a 2-dimensional state space $\mathscr{S}$ and a 2-dimensional action space $\mathscr{A}$ with $\mathbf{x}_t = (x_1,\ x_2) \in \mathbb{R}^2$ and $\mathbf{u}_t = (u_1, u_2) \in \mathbb{R}^2$. The GMM estimates the probability in the joint space $(x_1,\ x_2,\ u_1,\ u_2,\ r,\ x_1^{new},\ x_2^{new},\ q)$. As a result, the Gaussians of the mixture model are 8-dimensional.

5.4 Value Function Prediction

In this section we illustrate the performance of our algorithm to predict the value function for the Pendulum problem and the LQR problem.

Value Function Estimation for the Pendulum Problem

In the Pendulum-v0 problem, we investigated the ability of the algorithm to predict the value function under a uniformly generated dataset, because learning in this setting results to a less biased estimate. However, since most real-world events are random, we further learn the value function under a randomly sampled dataset.

In the case of the uniformly sampled data, we discretize the state-action space. This setting has only three distinct actions $u \in \{-2, 0, 2\}$. Figure 5.2 depicts the obtained state distribution. By

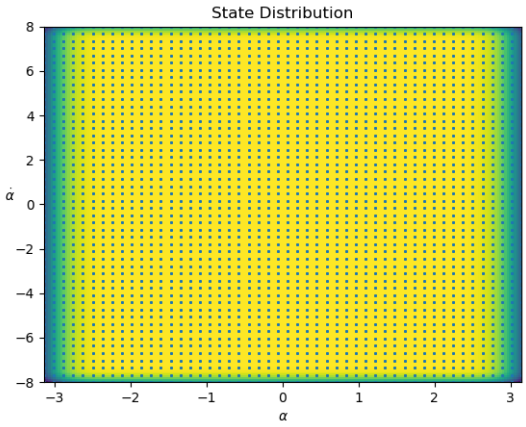


Figure 5.2.: Data distribution and density in a grid.

sampling from a grid with 50 angles, 50 velocities and 3 actions (resulting to a dataset size of 7500), we obtained a satisfying estimate of the value function as presented in Figure 5.3.

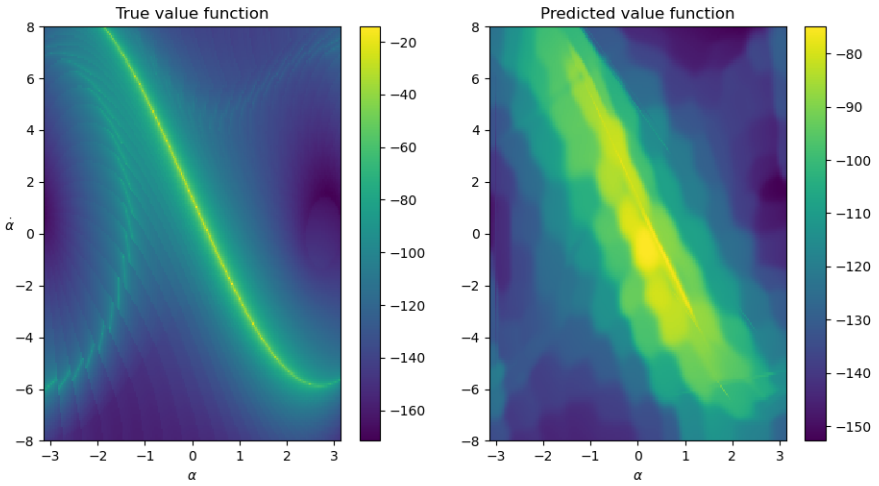


Figure 5.3.: Value function prediction with uniform grid data and 200 Gaussians.

Likewise to the uniform grid experiment, we show the ability of POEM to learn the value function using a dataset sampled from a random policy, as is the case in many real world problems. To generate the random episodes in this environment, the initial position is chosen randomly, then the

actions are selected by sampling from a random policy which in this case, is a uniform distribution with a low value of -2 and a high value of 2 (Figure 5.4). The maximum episode length is set to 200.

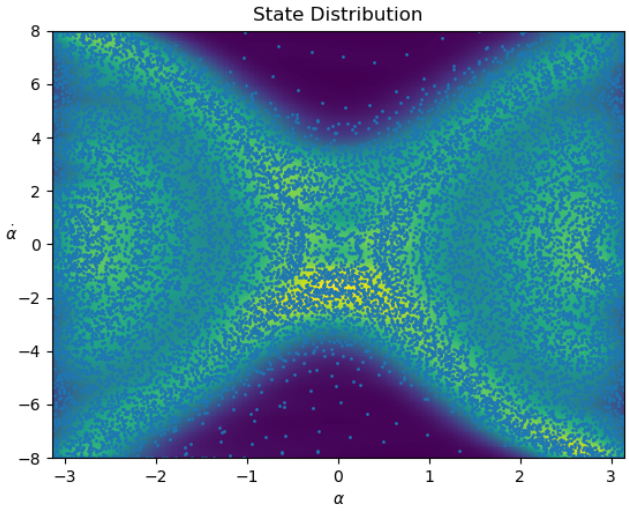


Figure 5.4.: Data distribution and density of samples generated from a random policy.

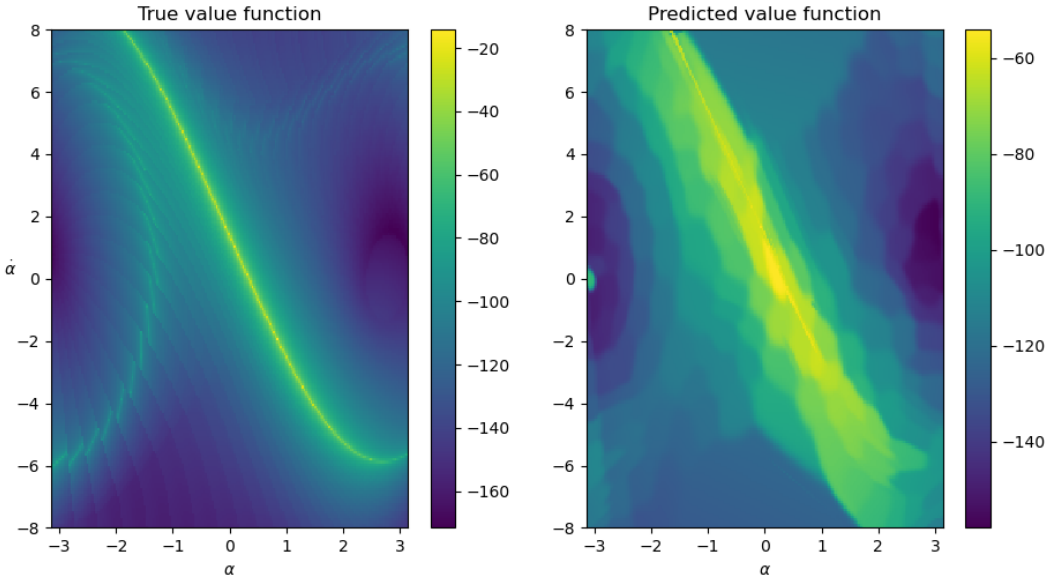


Figure 5.5.: Policy evaluation of randomly generated data (16000 samples) and 200 components.

In contrast to the first setting where the data is uniformly distributed in a grid, more samples are needed in the second setting to achieve a valid prediction of the value function, since each action is not equally likely to be selected in the latter setting. Hence more data is needed to ensure that the agent visits a sufficient number of regions in the state space.

As in the pendulum problem, we demonstrate the ability of the algorithm to predict the value function of an LQR problem. To achieve this purpose we make use of a linear policy, characterized by the diagonal matrix

$$\mathbf{K} = \begin{bmatrix} \mathrm{k}_1 & 0 \\ 0 & \mathrm{k}_2 \end{bmatrix}.$$

Furthermore, the matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{Q}$ and $\mathbf{R}$ are also encoded by diagonal matrices. More details to the instance of the LQR problem are given in Appendix A.2. Figure 5.7 shows the prediction of the value function obtained by running an experiment with a deterministic experiment. A data set composed of 2000 trajectories, each of length 20, is used for training the algorithm. The initial state distribution is stochastic. The state distribution and the density of the training dataset is illustrated in Figure 5.6. We initialize the GMM model with 30 Gaussians, and select the number of Gaussians that yields the best value function estimate.

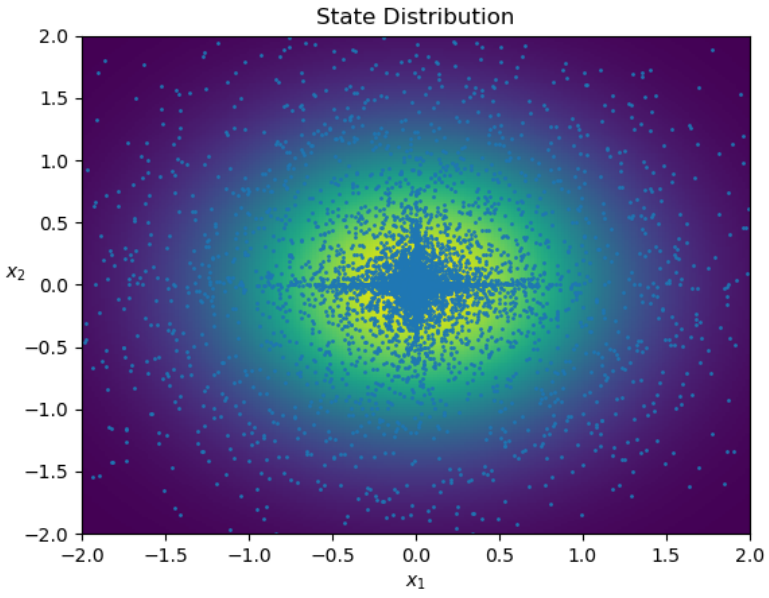


Figure 5.6.: Data distribution and density of samples generated from a deterministic linear policy.

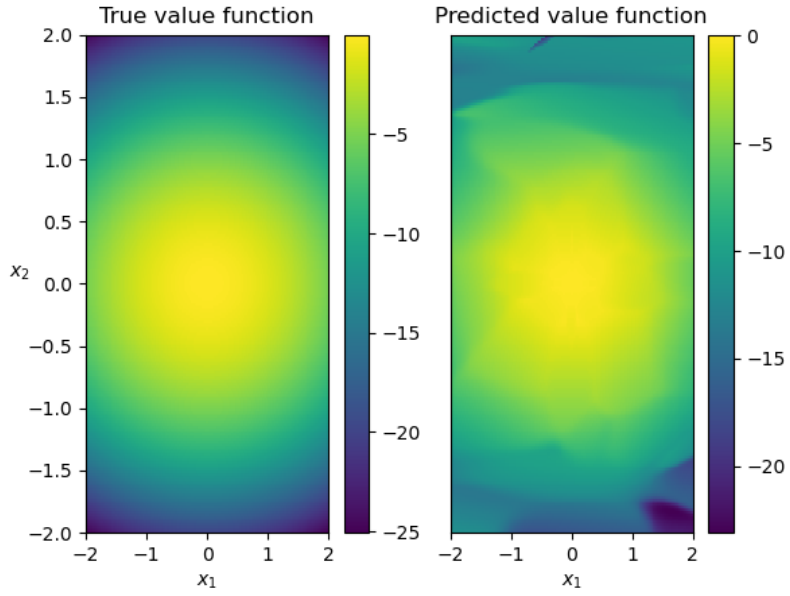


Figure 5.7.: Value function estimated in the LQR task with 40000 samples and 70 components.

5.5 Gradient Analysis

We illustrate the quality of the direction for the gradient estimate produced by the algorithm. Both the pendulum problem and the 2-dimensional LQR problem are considered. To achieve this we use the framework proposed by the *library HeRL*. In this method, n different parameterized policies are considered. For each policy, the ground truth gradient is computed via numerical gradient computation

$$\nabla_\theta \hat{J}(\theta) = \lim_{\varepsilon \to 0} \frac{\hat{J}(\theta + \varepsilon) - \hat{J}(\theta)}{\varepsilon}.$$

Then using the algorithm POEM, we also compute the gradient of $\hat{J}_{\pi_\theta}$ for each policies. Since the gradients are vectors, the angle δ between the ground truth gradient, and the gradient estimated by POEM is computed for each policy. For a single policy with true gradient $d\theta$ and estimated gradient via Poem $d\theta_{\text{estimate}}$, the angle is computed as

$$\cos(\delta) = \frac{d\theta \cdot d\theta_{\text{estimate}}}{\|d\theta\|_2 \, \|d\theta_{\text{estimate}}\|_2},$$

where $\|\|_2$ is the Euclidean norm. If the angle is less than $2^{-1}\pi$, then the direction is considered to be correct, else the direction is considered to be untrue.

Figures 5.8 and 5.9 show the gradient directions for the pendulum and the LQR problem. The blue line across the plot represents the density of the angles. It can be observed in Figure 5.8 that POEM estimates a good direction. Out of 100 policies, most computed angles are less than $2^{-1}\pi$. This fact is explicitly displayed by the mean estimate and the density, which is higher in the range $\left(2^{-1}\pi,\ 0\right]$. In Figure 5.9, even though the angle between the true gradient and the mean estimate is significant, the resulting estimate is still able to take steps in an acceptable direction.

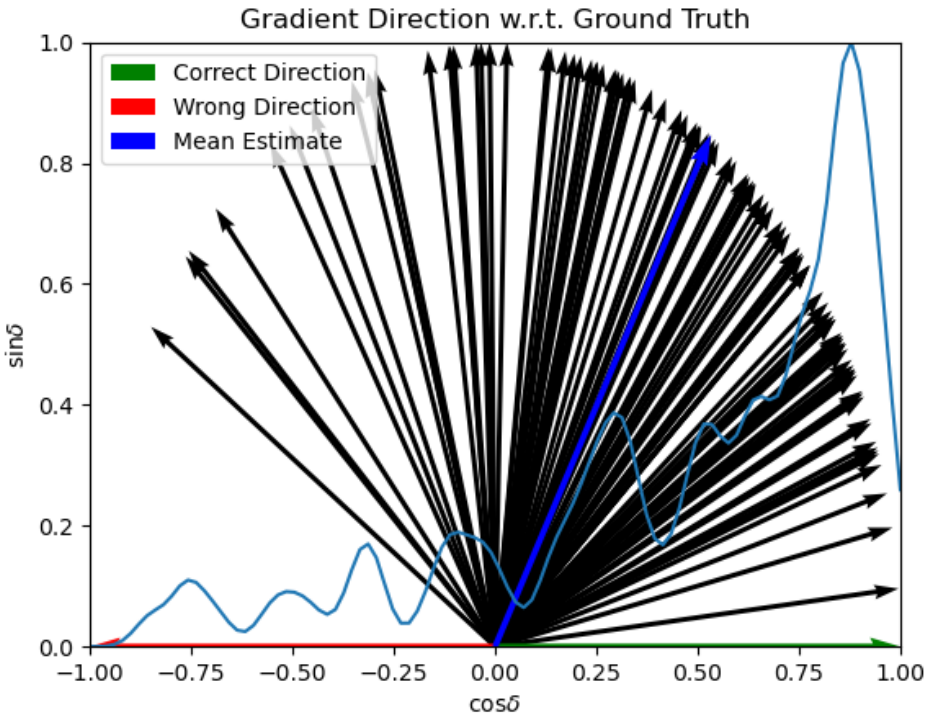


Figure 5.8.: Gradient direction with respect to ground truth for the pendulum environment. 100 policies parameters are used.

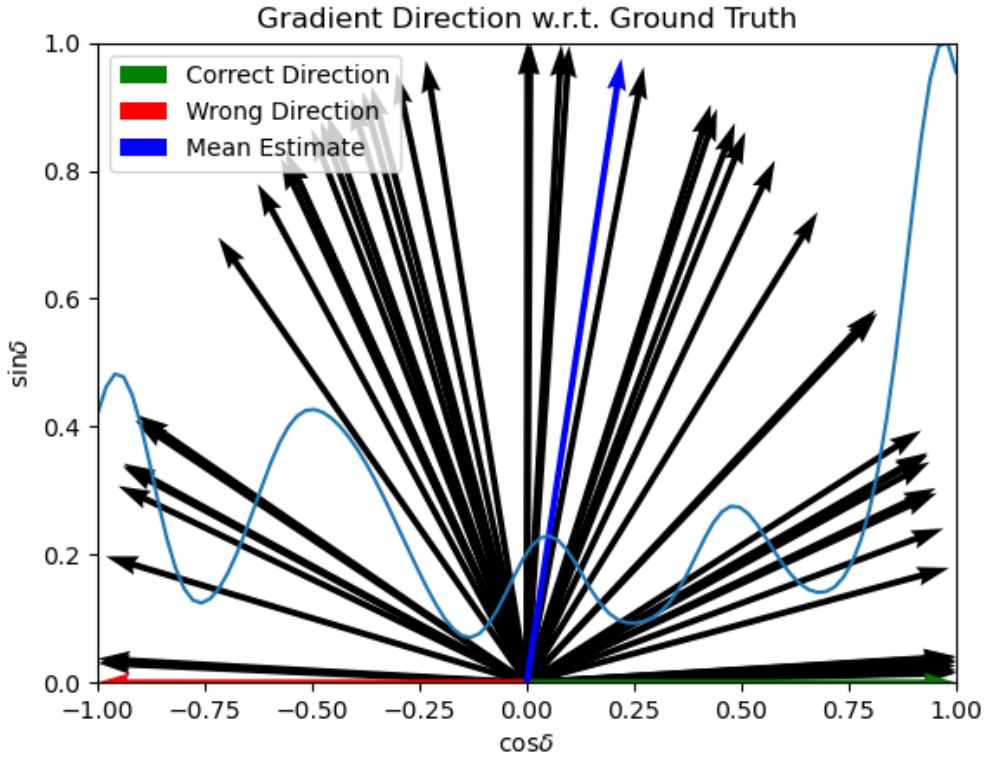


Figure 5.9.: Gradient direction with respect to ground truth for the LQR environment. 40 policies parameters are used.

Swing-up Pendulum

To analyze the performance of the algorithm in learning an optimal controller, we conduct an experiment under uniformly sampled datasets generated from a grid over state and action spaces of the Pendulum environment. We generate six datasets from a grid, with the same granularity. Moreover, a policy that is characterized by a neural network with one hidden layer of 50 neurons and ReLU activation functions is optimized for 10^5 iterations. After every 10^3 iterations, the learning control agent is evaluated on trajectories of 15×10^3 steps starting from the initial position of the pendulum (hanging down). This position is chosen because it is the worst case scenario of the environment. Figure 5.10 illustrates the learning curve of the objective function J_{π_θ} per iteration steps. It can be observed that the algorithm learns an acceptable controller, and the convergence is fast and stable.

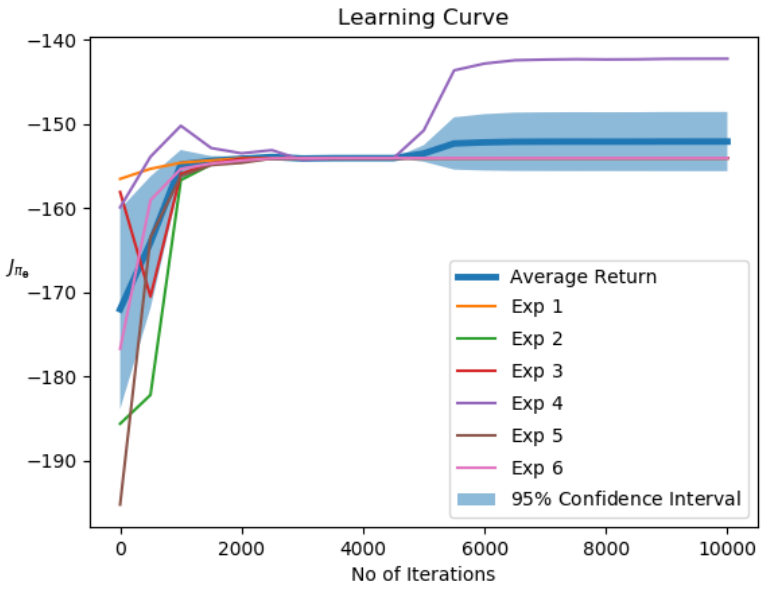


Figure 5.10.: Average return J_{π_θ} per iteration performed over 6 experiments with 95% confidence interval on data sampled from a uniform grid.

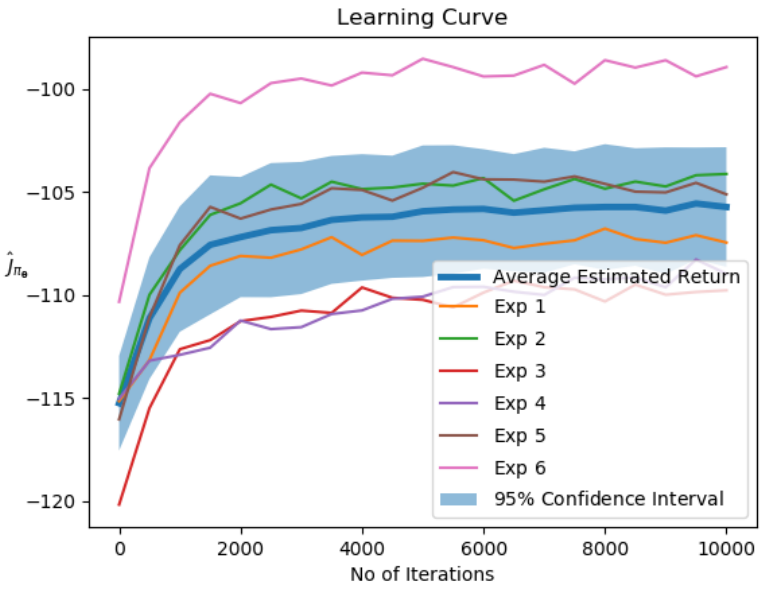


Figure 5.11.: Average estimated return $\hat{J}_{\pi_\theta}$ per iteration with 95% confidence interval on data from a uniform grid.

6 Conclusion

This thesis aimed to design and investigate an off-policy algorithm based on Gaussian mixture models. For Rl problems, where the state and action spaces are continuous, learning the probability density function in the joint space of the states, actions, rewards, next-states, and q-values is advantageous.
In discrete states and actions spaces, the Bellman equation can be solved in polynomial time via dynamic programming. However, in real-world applications, the state and action spaces are continuous. Value-based methods solve for the optimal policy indirectly by learning the value function and inferring the policy based on the learned values. Conversely, policy gradient methods learn the policy directly by encoding the policy by a parameterized function with respect to θ and maximizing the expected return via gradient ascend. However, computing the gradient of the objective function is hard because it involves computing the gradient of a stationary distribution over states. This gradient is not straight forward to estimate. Hopefully, the policy gradient theorem provides a simple expression for the gradient of the objective function. The inconvenience is that learning is on-policy. Hence sample inefficient.
Off-policy methods, on the other hand, are sample efficient. However, the state-of-art off-policy methods are limited (suffers from hi bias, high variance). Furthermore, the non-parametric off-policy method in [17] does not scale well with high dimensions.
Following the work in [17] we designed an off-policy algorithm that analytically expresses the full gradient estimate of the objective function. In contrast to [17], the reward function and the environment's dynamics can be estimated using a Gaussian mixture model. However, building the GMM-Bellman equation requires an integral over a non-linear function. Two approaches were presented to circumvent this problem. Firstly, we linearized the value function at an operator point. Secondly, by enforcing the Gaussians correlation parameters $\mathbf{\Sigma}_{s',s}$, and $\mathbf{\Sigma}_{s',a}$ to zero, we are able to obtain a better estimate of the integral term. This estimation, leads us to a GMM-Bellman equation that can be solved analytically. Our algorithm takes advantage of the scalability of Gaussian mixture models to solve an MDP with fewer parameters. We tested our algorithm on classic control tasks, obtaining promising results.

Outlook

Because of the limited amount of time and scarce computational resources, several different configurations and experiments have been left for future consideration. Further research could be done on the comparison of POEM with baselines. It could be interesting to test the algorithm on trajectories generated by humans. Additionally, some analysis could be made to study the correlation between the number of training samples, the performance of the algorithm and the number of Gaussians. Furthermore, the way the algorithm is implemented can be changed. Instead of using a batch method, an online version of the algorithm could be designed, as it would aloud for better exploration in the environment. Deciding how many components k to use is computationally expensive, since the algorithm is runned for different number of Gaussians, and

the k yielding the optimal value is selected. The Dirichlet Process Mixtures of Generalized Linear Models(DP-GLM) could be investigated, since it avoids the specification of the parameter k by automatically finding the number of components required by the data.

Bibliography

[1] J. Schulman, S. Levine, P. Moritz, M. Jordan, and P. Abbeel, "Trust Region Policy Optimization," in *Proceedings of the 32nd International Conference on Machine Learning*, pp. 1889–1897, 2015.

[2] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor," in *Proceeding of the 35th International Conference on Machine Learning*, pp. 1856–1865, 2018.

[3] P. Christiano, J. Leike, T. B. Brown, M. Martic, S. Legg, and D. Amodei, "Deep reinforcement learning from human preferences," 2017.

[4] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-Level Control Through Deep Reinforcement Learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[5] T. Degris, M. White, and R. S. Sutton, "Off-Policy Actor-Critic," in *Proceedings of the 29th International Coference on Machine Learning*, pp. 179–186, Omnipress, 2012.

[6] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic Policy Gradient Algorithms," in *Proceedings of the 31 st International Conference on Machine Learning*, 2014.

[7] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous Control with Deep Reinforcement Learning," in *International Conference on Learning Representations*, 2016. arXiv: 1509.02971.

[8] S. Fujimoto, D. Meger, and D. Precup, "Off-Policy Deep Reinforcement Learning without Exploration," in *Proceeding of the 36th International Conference on Machine Learning*, pp. 2052–2062, 2019.

[9] O. Sigaud and O. Buffet, *Markov Decision Processes in Artificial Intelligence*. Wiley-IEEE Press, 2010.

[10] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy Gradient Methods for Reinforcement Learning with Function Approximation," in *Advances in Neural Information Processing Systems*, pp. 1057–1063, 2000.

[11] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. The MIT Press, second ed., 2018.

[12] L. Wasserman, *All of Statistics: A Concise Course in Statistical Inference*. Springer Publishing Company, Incorporated, 2010.

[13] C. Szepesvári, *Algorithms for Reinforcement Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan Claypool Publishers, 2010.

[14] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. USA: John Wiley Sons, Inc., 1st ed., 1994.

[15] R. J. Williams, "Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[16] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, 1992.

[17] S. Tosatto, J. Carvalho, H. Abdulsamad, and J. Peters, "A nonparametric off-policy policy gradient," in *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020.

[18] G. A. Rummery and M. Niranjan, "On-line q-learning using connectionist systems," tech. rep., 1994.

[19] F. Stulp and O. Sigaud, "Many regression algorithms, one unified model — a review," *Neural Networks*, vol. 69, 06 2015.

[20] M. Hersch, F. Guenter, S. Calinon, and A. Billard, "Dynamical system modulation for robot learning via kinesthetic demonstrations," *IEEE Transactions on Robotics*, vol. 24, pp. 1463–1467, Dec 2008.

[21] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[22] N. Meuleau, L. Peshkin, and K.-E. Kim, "Exploration in Gradient-Based Reinforcement Learning," tech. rep., Massachusetts Institute of Technology, 2001.

[23] C. R. Shelton, "Policy Improvement for POMDPs Using Normalized Importance Sampling," in *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, UAI'01, pp. 496–503, Morgan Kaufmann Publishers Inc., 2001. event-place: Seattle, Washington.

[24] L. Peshkin and C. R. Shelton, "Learning from Scarce Experience," in *Proceedings of the Nineteenth International Conference on Machine Learning*, 2002. arXiv: cs/0204043.

[25] O. B. Kroemer and J. R. Peters, "A non-parametric approach to dynamic programming," in *Advances in Neural Information Processing Systems 24* (J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, eds.), pp. 1719–1727, Curran Associates, Inc., 2011.

[26] A. Agostini and E. Celaya, "Reinforcement learning with a gaussian mixture model," in *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2010.

[27] W. D. Smart and L. Pack Kaelbling, "Effective reinforcement learning for mobile robots," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, vol. 4, pp. 3404–3410 vol.4, 2002.

[28] Z. Ghahramani and M. I. Jordan, "Supervised learning from incomplete data via an em approach," in *Advances in Neural Information Processing Systems 6* (J. D. Cowan, G. Tesauro, and J. Alspector, eds.), pp. 120–127, Morgan-Kaufmann, 1994.

[29] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009.

[30] T. E. Oliphant, *A guide to NumPy*, vol. 1. Trelgol Publishing USA, 2006.

[31] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), pp. 8024–8035, Curran Associates, Inc., 2019.

[32] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[33] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," *arXiv:1606.01540*, 2016. arXiv: 1606.01540.

[34] F. Borrelli, A. Bemporad, and M. Morari, *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press, June 2017. Google-Books-ID: 7NUoDwAAQBAJ.

[35] R. A. Horn and C. R. Johnson, *Matrix Analysis*. USA: Cambridge University Press, 2nd ed., 2012.

A Appendix

We present the configurations for the different experiments carried out. Then we provide a proof for the invertibility of matrix $\mathbf{\Lambda}_{\pi_\theta}$.

A.1 Pendulum configurations

dataset size	7500
discount factors	0.95
number of Gaussians	100, 70
policy	neural network with parameter θ, one hidden layer of 50 units, Relu activation function
policy output function	2tanh
optimizer	adam
N^p	2000

Table A.1.: Parameters for policy evaluation under a uniform grid dataset.

dataset size	16000
# rollouts	80
max episode length	200
discount factors	0.95
number of Gaussians	200
policy	neural network with parameter θ, one hidden layer of 50 units, Relu activation function
policy output function	2tanh
optimizer	adam
N^p	2000

Table A.2.: Parameters for policy evaluation using data generated randomly.

A.2 LQR Configurations

dataset size	40000
# rollouts	2000
max episode length	20
discount factors	0.5
number of Gaussians	70
policy	Linear policy with parameter θ, encoded as diagonal matrix $\mathbf{K}$
N^p	2000

Table A.3.: Configurations for the LQR experiment.

The matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{Q}$, $\mathbf{R}$, $\mathbf{K}$ are set as:

$$\mathbf{A} = \begin{bmatrix} 1.2 & 0 \\ 0 & 1.1 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 1.0 & 0 \\ 0 & 1.0 \end{bmatrix}, \mathbf{Q} = \begin{bmatrix} 0.4 & 0 \\ 0 & 0.8 \end{bmatrix}, \mathbf{R} = \begin{bmatrix} 1.0 & 0 \\ 0 & 1.0 \end{bmatrix}, \mathbf{K} = \begin{bmatrix} \mathrm{k}_1 & 0 \\ 0 & \mathrm{k}_2 \end{bmatrix},$$

with $k_1 \sim \mathcal{N}(-1.2,\ 0.3)$ and $k_2 \sim \mathcal{N}(-1.1,\ 0.3)$.

A.3 Proof of the invertibility of Λ_{π_θ}

In this section we proof that the matrix $\mathbf{\Lambda}_{\pi_\theta} = \mathbf{I} - \gamma \mathbf{P}_{\pi_\theta}$ with $\mathbf{P}_{\pi_\theta}$ defined as in (4.9). We want to show that $\mathbf{\Lambda}_{\pi_\theta} = \mathbf{I} - \gamma \mathbf{P}_{\pi_\theta}$ is regular with $\gamma \in [0, 1)$.
By Neumann Series [35, Chapter 5], if a matix $\mathbf{A}$ has the property that $\lim_{i\to\infty}(\mathbf{I} - \mathbf{A})^i = 0$ then $\mathbf{A}$ is regular and $\mathbf{A}^{-1} = \sum_{i=0}^{\infty}(\mathbf{I} - \mathbf{A})^i$.

Thus we show that $\lim_{i\to\infty}(\mathbf{I} - \mathbf{\Lambda}_{\pi_\theta})^i = 0$

Proof.

$$\begin{aligned} \left(\mathbf{I} - \mathbf{\Lambda}_{\pi_\theta}\right)^i &= \left(\mathbf{I} - \left(\mathbf{I} - \gamma \mathbf{P}_{\pi_\theta}\right)\right)^i \\ &= (\gamma \mathbf{P}_{\pi_\theta})^i. \end{aligned}$$

We then show that $(\gamma \mathbf{P}_{\pi_\theta})^i \to 0$ as $i \to \infty$.
Notice the matrix $\mathbf{P}_{\pi_\theta}$ is of the form

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{0} & \mathbf{0} \end{bmatrix},$$

where $\mathbf{A}$ is a stochastic matrix, $\mathbf{B}$ a matrix, and $\mathbf{0}$ a matrix of zeros. Computing successive powers of $\mathbf{P}_{\pi\theta}$, we have $\mathbf{P}^2_{\pi\theta} = \begin{bmatrix} \mathbf{A^2} & \mathbf{AB} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$, $\mathrm{P}^3_{\pi\theta} = \begin{bmatrix} \mathbf{A}^3 & \mathbf{A}^2\mathbf{B} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$, $\mathrm{P}^4_{\pi\theta} = \begin{bmatrix} \mathbf{A^4} & \mathbf{A}^3\mathbf{B} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$. Hence in general

$$\mathbf{P}^i_{\pi_\theta} = \begin{bmatrix} \mathbf{A^i} & \mathbf{A}^{i-1}\mathbf{B} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}.$$

Because $\mathbf{A}$ is a stochastic matrix,

$$\left\|\mathbf{A}^i\right\|_\infty \leqslant 1, \quad \left\|\mathbf{A}^i\mathbf{B}\right\|_\infty \leqslant \|\mathbf{B}\|_\infty.$$

Thus

$$\left\|\mathbf{P}^i_{\pi_\theta}\right\|_\infty \leqslant \|\mathbf{B}\|_\infty.$$

It follows that

$$\lim_{i\to\infty} \left\|\gamma^i \mathbf{P}^i_{\pi_\theta}\right\|_\infty = \lim_{i\to\infty} \gamma^i \left\|\mathbf{P}^i_{\pi_\theta}\right\|_\infty \leq \lim_{i\to\infty} \gamma^i \|\mathbf{B}\|_\infty = 0$$

□