ADAPTIVE ALGORITHMS FOR EFFICIENT RISK ESTIMATION OF BLACK-BOX SYSTEMS

A THESIS
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE
WITH DISTINCTION IN RESEARCH

Kyu-Young Kim
November 2022

I certify that I have read this thesis and that, in my opinion, it is fully adequate in scope and quality as a thesis for the degree of Master of Science.

Mykel J. Kochenderfer, Principal Adviser

I certify that I have read this thesis and that, in my opinion, it is fully adequate in scope and quality as a thesis for the degree of Master of Science.

Dorsa Sadigh

Abstract

Rigorous safety evaluation of learning-based autonomous systems, especially those in safety-critical domains, is important to be done prior to wider deployment. Such autonomous systems are often highly complex, involving several interaction layers of learned models. These systems are therefore treated as a black-box in practice and are evaluated using simulations of random processes, with the goal of estimating relevant risk measures. This thesis presents a novel framework called curriculum sampling that aims to efficiently evaluate such black-box systems using random sampling.

One of the main challenges in designing an estimator based on random sampling is ensuring that a desired level of variance can be achieved with a reasonable sample size. This is particularly important in domains such as the safety evaluation task considered in this work, where simulation of the given system is computationally expensive. Importance sampling is a common variance reduction method that uses an alternative sampling distribution to improve sample efficiency. If not used carefully, however, the method could lead to an estimate with infinite variance and hence be worse than simple Monte Carlo. Using the idea of adapting the sampling distribution from adaptive importance sampling, curriculum sampling aims to achieve sample efficiency while maintaining stability of the procedure by decomposing the given problem into a series of subtasks solved in sequence.

In curriculum sampling, the distribution is adapted depending on the subtask being solved and the representation of uncertainty of the current estimate used. A given subtask is considered solved, if the uncertainty on the current estimate reaches some threshold, and the procedure continues until the last one is solved. The full set of samples collected is then used to estimate one or more measures of interest. We apply the approach to both sequential and non-sequential cases, demonstrating its use in estimating the tail risk measures called value-at-risk and conditional value-at-risk. The algorithm that we propose for the sequential case, in particular, can be adopted for many risk estimation settings by appropriately designing the input space. Specifically, we apply the algorithm to evaluating a sequential system in presence of state-dependent disturbances in a given environment. We also demonstrate the approach in evaluating an autonomous vehicle policy in simulation, with the goal of estimating the risk measures across a set of scenarios.

Acknowledgments

First of all, I would like to thank my primary advisor Prof. Mykel Kochenderfer for the opportunity to work on this research and the guidance he has provided me along the way. I am especially grateful for his recommending me for the graduate program at Stanford University, which was crucial in my endeavor to start pursuing my intellectual aspirations once again. I own a great debt for his support. I would like to also acknowledge the support from Prof. Dorsa Sadigh who severed as my secondary advisor on this research.

I am grateful for the opportunity to collaborate closely with several graduate students and researchers on this work. In particular, I would like to thank Robert Moss and Anthony Corso of the Stanford Intelligent Systems Lab and Shubh Gupta of the Navigation and Autonomous Vehicles Lab for the constructive feedback that they have given me with patience and respect. Their previous research work and the tools that they built were tremendously useful for my own work as well.

Everyone I interacted with at Google shaped me into a more mature engineer, researcher, and individual overall. Particular appreciation is due to Scott Roy, my manager at the company for the longest time. He has not only helped me develop deeper technical skills, but also showed me the importance of treating others with kindness and respect. I would also like to thank Cliff Brunk, who as my mentor has taught me to approach problems with meticulousness and develop solutions with rigor. The full list of my coworkers would be too long to be included here, but I would like to show my appreciation also to Michael Schwarz, David Tao, Minh Truong, and Justin Zhao.

Lastly, I must thank my family for the lifelong guidance, support, and love that they have graciously provided to me. My parents have taught me many important values in life that I still strive to live by such as diligence, perseverance, humility, integrity, appreciation, and selflessness, and, most importantly, faith in God and Christ. I would like to thank my sibling, Kyuwon, who has been a great friend and often a mentor as I navigate through various challenges in life. I am forever indebted for everything that they have done for and with me. I wish to also thank my spiritual family at the Shincheonji Church of Jesus, particularly the pastor Man-Hee Lee, who has taught me the true words of God so that I can hope for the things that endure to eternal life in Christ.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Recent advances in the field of artificial intelligence allowed developing autonomous systems for an increasing number of real-world applications. More of such learning-based systems are applied to safety-critical domains [16] such as autonomous driving, and rigorous safety evaluation of the systems is hence becoming increasingly important. These systems, however, are often highly complex, involving multiple layers of interactions between learned models. As such, in practice, such systems are treated as a black-box [18], where we have access only to the output from the system for the input that we provide. In this work, we explore adaptive algorithms [4] for evaluating such black-box systems using random sampling with the goal of efficiently estimating relevant risk measures.

In evaluating black-box systems, we are typically interested in estimating such measures as the probability of failure or tail quantiles. These measures by definition are concerned with rare events, and simple Monte Carlo is often inefficient in terms of the number of samples needed to achieve a desired level of accuracy or variance. Importance sampling is a common variance reduction method that is natural to employ for such estimation problems involving rare events. It uses an alternative distribution that is better suited to the given estimation problem to draw samples, and, in a more advanced method such as adaptive importance sampling, the sampling distribution is adaptively updated in order to further improve efficiency. We extend this idea of adaptive sampling distribution to propose a novel framework for estimating a set of dependent tail statistics in a stable and sample efficient manner.

After an introduction of the general framework, we present a detailed implementation of an importance sampler for estimating tail risks in non-sequential settings. We then discuss an extension of the framework to sequential settings, presenting a tree search-based algorithm for evaluating sequential systems. Lastly, we apply the algorithm to evaluation of autonomous vehicle policy in simulation, demonstrating that the algorithm can be used to estimate risks and also to discover relevant failure events.

1.1 Contributions

The following is a brief summary of the contributions of this work. A more detailed summary can be found in Section 6.1.

- A novel framework called *curriculum sampling* designed for stable and sample efficient estimation of a set of dependent tail statistics is proposed.
- A concrete implementation of the framework for estimating value-at-risk and conditional value-at-risk is given with experimental results demonstrating the algorithm.
- An application of curriculum sampling to risk estimation in sequential settings with a tree search algorithm is presented.
- An application of the tree search algorithm to evaluating autonomous vehicle systems across a space of scenarios is presented.

1.2 Overview

This chapter has briefly introduced the core problems to be solved and the approaches explored in this work. The rest of the thesis is organized as follows.

Chapter 2 provides an overview of the core concepts discussed throughout the thesis. These include the basics of importance sampling, the risk measures to estimate using importance sampling, and the formulation of reinforcement learning tasks.

Chapter 3 motivates the idea of adaptive importance sampling and introduces the curriculum sampling framework where a representation of uncertainty in estimates is used to design a stable sampling procedure to estimate a dependent set of statistics. This chapter provides a detailed implementation of the framework for estimating the risk measures introduced in Chapter 2 and demonstrates the algorithm on a quantile estimation problem.

Chapter 4 extends the fundamental idea of curriculum sampling to the problem of evaluating sequential systems. This chapter discusses a formulation of the task as an instance of Markov decision process and presents a tree search-based algorithm for estimating risks in presence of state-dependent disturbances to the system under test. This chapter concludes with experimental results demonstrating the algorithm on a test environment.

Chapter 5 applies the tree sampling algorithm presented in Chapter 4 to evaluating an autonomous vehicle policy in simulation. Applying the algorithm to the space of scenarios of interest, the chapter demonstrates that the algorithm can be adopted to both evaluate risks and discover relevant failure scenarios for autonomous vehicle systems in simulation.

Chapter 6 summarizes the thesis and concludes with ideas for further research.

Chapter 2

Background

This chapter discusses the core concepts explored throughout this work, reviewing the basics of importance sampling, risk estimation, and reinforcement learning. Following a brief introduction in the first section, the second section explains the basic theory on importance sampling. The third section presents the definitions of the tail risk measures that we use to evaluate black-box systems and motivates their use. The fourth section discusses the basic formulation of reinforcement learning tasks also introducing relevant notation. We close with a short discussion summarizing the chapter at the end.

2.1 Introduction

Monte Carlo methods are concerned with learning about a system by estimating the relevant statistics using simulations of random processes [25]. It is conceptually simple and typically easy to implement, and, depending on the problem, it is often the only feasible approach [28].

In analyzing a random sampling-based estimator, we are interested in understanding its bias and variance. Roughly speaking, the bias of an estimator is a measure of how much the expected estimate deviates from the true value, whereas the variance is a measure of how much the estimates are expected to spread out from the mean. We generally want an estimator that is unbiased and of low variance, but, in practice, one often has to make trade-offs between the two when choosing an estimator. Among a set of unbiased estimators, the one with the lowest variance is preferred. There are several common variance reduction methods, and the one called importance sampling is explored in depth in this thesis.

2.2 Importance Sampling

Consider the problem of estimating $\mathbb{E}_p[f(X)]$, that is, the mean of the function $f(X)$ with X drawn according to the distribution p. Suppose that $f(X)$ is such that it is non-zero only in some region T for which $p(X \in T)$ is small. In such a case, the vast majority of the samples collected using simple Monte Carlo would be outside of the region and not contribute to the estimate. Intuitively, we want to draw samples from this region of importance that would contribute more to the estimation. This is the fundamental idea of *importance sampling*, where we use an alternative distribution q that puts higher weights over such regions of importance to draw samples. To account for the bias from using a different distribution, the weights of the samples are adjusted appropriately so that we have an unbiased estimate of the quantity.

Importance sampling is a popular variance reduction method that can make problems that are otherwise too hard amenable to Monte Carlo methods. However, depending on the choice of the distribution or the weighting scheme, it could lead to issues such as infinite variance. It is hence a method that is useful but difficult to use well [32].

We now review the basic theory on bias and variance of a simple importance sampling estimator.

2.2.1 Basics

Let p be the probability density on $\mathbb{R}^d$ under which we want to estimate the mean of the function f, that is, $\mu = \mathbb{E}_p[f(X)]$. Suppose that q is another probability density such that $q(X) > 0$ whenever $f(X)p(X) \neq 0$, then

$$\mu = \int f(x)p(x)\,dx = \int \frac{f(x)p(x)}{q(x)}q(x)\,dx = \mathbb{E}_q\left[\frac{f(X)p(X)}{q(X)}\right], \tag{2.1}$$

where $\mathbb{E}_q(\cdot)$ denotes that the expectation is over X drawn from q. The condition of $q(x) > 0$ whenever $f(x)p(x) \neq 0$ roughly means that we want to be able to draw all samples under q that could contribute to the mean (i.e., x for which $f(x)p(x) \neq 0$). Note that this is not the same as requiring q to be positive whenever p is positive, which is a stronger condition that is often used in other importance sampling settings.

Eq. (2.1) tells us that computing the original μ, which is the expected value of $f(X)$ for $X \sim p$, is equivalent to computing the expected value of $\frac{f(X)p(X)}{q(X)}$ for $X \sim q$. This equivalence is established by making an adjustment using the correction factor $w(x) = p(x)/q(x)$, called the likelihood or importance ratio, to $f(x)$. The probability density p is called the nominal or target distribution and q the proposal or importance distribution. Note that we would not encounter those samples for which $q(x) = 0$ when sampling from q, and hence not need to worry about division-by-zero when computing the expectation. However, samples with $q(x)$ close to 0 could lead to numerical issues [32].

Given a proposal distribution q as described above, the importance sampling estimator of $\mu = \mathbb{E}_p[f(X)]$ is given by

$$\hat{\mu}_q = \frac{1}{n}\sum_{i=1}^{n} \frac{f(X_i)p(X_i)}{q(X_i)} = \frac{1}{n}\sum_{i=1}^{n} f(X_i)w(X_i), \tag{2.2}$$

where each X_i is drawn according to the distribution q.

2.2.2 Analysis

We now show that the importance sampling estimator as defined in Eq. (2.2) is an unbiased estimator of μ and analyze its variance.

Theorem 2.2.1. *The importance sampling estimator $\hat{\mu}_q$ of μ is unbiased and has variance σ_q^2/n where*

$$\sigma_q^2 = \int \frac{f(x)^2 p(x)^2}{q(x)}\,dx - \mu^2 = \int \frac{(f(x)p(x) - \mu q(x))^2}{q(x)}\,dx. \tag{2.3}$$

Proof. We first show that $\hat{\mu}_q$ is an unbiased estimator, that is, $\mathbb{E}_q[\hat{\mu}] = \mu$. Indeed,

$$\begin{aligned}
\mathbb{E}_q[\hat{\mu}] &= \mathbb{E}_q\left[\frac{1}{n}\sum_{i=1}^{n}\frac{f(X_i)p(X_i)}{q(X_i)}\right] = \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}_q\left[\frac{f(X)p(X)}{q(X)}\right] \\
&= \frac{1}{n}\sum_{i=1}^{n}\int \frac{f(x)p(x)}{q(x)}q(x)\,dx = \frac{1}{n}\sum_{i=1}^{n}\int f(x)p(x)\,dx \\
&= \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}_p[f(X)] = \mu.
\end{aligned}$$

Note that the second-to-last equality holds, because we assumed that $q(x) > 0$ whenever $f(x)p(x)$. Using what we just derived, we compute the variance of the estimator as

$$\begin{aligned}
\mathrm{Var}_q[\hat{\mu}] &= \frac{1}{n}\left(\int\left(\frac{f(x)p(x)}{q(x)} - \mu\right)^2 q(x)\,dx\right) \\
&= \frac{1}{n}\left(\int \frac{(f(x)p(x) - \mu q(x))^2}{q(x)}\,dx\right) \\
&= \frac{1}{n}\left(\int \frac{(f(x)p(x))^2 - 2(f(x)p(x))(\mu q(x)) + \mu^2 q(x)^2}{q(x)}\,dx\right) \\
&= \frac{1}{n}\left(\int \frac{(f(x)p(x))^2}{q(x)}\,dx - 2\mu\cdot\int f(x)p(x)\,dx + \mu^2\cdot\int q(x)\,dx\right) \\
&= \frac{1}{n}\left(\int \frac{(f(x)p(x))^2}{q(x)}\,dx - 2\mu^2 + \mu^2\right) = \frac{1}{n}\left(\int \frac{(f(x)p(x))^2}{q(x)}\,dx - \mu^2\right).
\end{aligned}$$

□

The variance analysis above gives us a sense of what might be a good choice of proposal distribution, that is, the one that minimizes the variance. Specifically, the first equality in Eq. (2.3) tells us that we want q such that $\int \frac{f(x)^2 p(x)^2}{q(x)}\,dx$ is as small as possible. Since μ^2 is fixed, the integral term gives an upper bound on σ_q^2. Moreover, the second equality in the equation suggests that the one that makes the numerator $f(x)p(x) - \mu q(x)$ as small as possible would be desirable. However, we should also be careful about the denominator $q(x)$, as it can amplify the squared difference in the numerator.

In general, we would like to choose a density that is close to being proportional to $|f(x)|p(x)$ and the one that does not have light tails. Using such a density q means that $X \sim q$ with higher $|f(x)|p(x)$ is more likely to be drawn. This intuitively makes sense, as $X \sim q$ with higher $|f(x)|p(x)$ potentially contributes more to the mean that we want to estimate, and hence is of greater importance. The next theorem formalizes this observation.

Theorem 2.2.2. *Let $q^*(x) = |f(x)|p(x)/c$, where c is the normalization constant, that is, $c = \int |f(x)|p(x)\,dx$. If q is an arbitrary density such that $q(x) > 0$ whenever $f(x)p(x)$, then $\sigma_{q^*} \leq \sigma_q$.*

Proof. Using Eq. (2.3),

$$\begin{aligned}
\mu^2 + \sigma_{q^*}^2 &= \int \frac{f(x)^2 p(x)^2}{q^*(x)}\,dx = \int \frac{f(x)^2 p(x)^2}{|f(x)|p(x)/c}\,dx \\
&= c \cdot \int |f(x)|p(x)\,dx = \left(\int |f(x)|p(x)\,dx\right)^2 \\
&= \left(\int \frac{|f(x)|p(x)}{q(x)} q(x)\,dx\right)^2 \\
&\leq \int \frac{f(x)^2 p(x)^2}{q(x)^2} q(x)\,dx \\
&= \int \frac{f(x)^2 p(x)^2}{q(x)}\,dx = \mu^2 + \sigma_q^2,
\end{aligned}$$

where the inequality is a consequence of the Cauchy-Schwarz inequality [32]. □

2.3 Risk Measures

Let $X \in \mathbb{R}^d$ be a random vector representing the input to a system under test or the environment in which such a system is evaluated. Also, let $c : \mathbb{R}^d \to \mathbb{R}$ be a deterministic, positive, real-valued function that represents the cost associated with the system given input X. The cost $c(X)$ then is a random variable with the distribution induced by that of the input X. The random vector X can represent, for instance, a set of sensor noises drawn from some fixed distributions applied to an autonomous vehicle system under test. And the cost function c can be the potential loss in property values that such disturbances applied to the autonomous system could lead to.

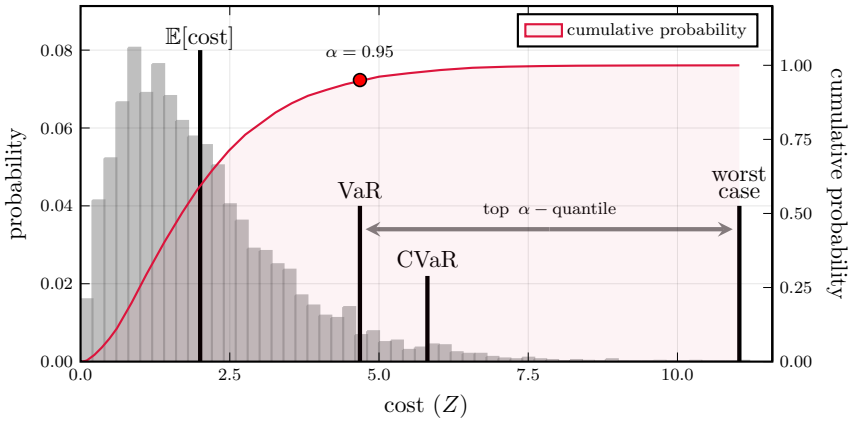


Figure 2.1: An illustartion of VaR_α and CVaR_α with $\alpha = 0.95$ of a cost distribution [30].

Among various risk measures, value-at-risk (VaR) and conditional value-at-risk (CVaR) [35] are commonly used, especially in risk management, portfolio optimization [26], etc. The risk measures are defined as

$$\text{VaR}_\alpha(X) = \inf\{c \mid F_{c(X)}(c) \geq \alpha\} \tag{2.4}$$

$$\text{CVaR}_\alpha(X) = \mathbb{E}[c(X) \mid c(X) \geq \text{VaR}_\alpha(X)], \tag{2.5}$$

where $F(\cdot)$ is the cumulative distribution function (c.d.f.) of the cost $c(X)$. That is,

$$F_{c(X)}(c) = \int_{c(x) \leq c} p(x)\, dx \tag{2.6}$$

with $p(\cdot)$ denoting the probability density function (p.d.f.) of X. In words, VaR_α is the smallest c such that the cost would not exceed this value with probability α, and CVaR_α is the conditional expectation of the cost above VaR_α. Mathematically, VaR_α is the α-quantile of the cost distribution. The value of α is generally chosen to be $\alpha \approx 1$ with typical values 0.90, 0.95, and 0.99 [34].

Note that the worst-case cost is another common risk measure that is often analyzed, but, in some domains, it is considered less informative than the more subtle tail risks such as VaR_α and CVaR_α. This is especially the case if the probability of the event is so low that incurring such a cost is extremely unlikely.

2.4 Reinforcement Learning

Later in this thesis, we will consider the problem of evaluating sequential systems and formulate the task as an instance of reinforcement learning (RL) problem. In this section, we briefly discuss some of the basics of RL and introduce relevant concepts and notations.

RL is concerned with constructing an autonomous *agent* that is able to act in an environment in order to maximize the expected return. In contrast to the standard supervised learning, no direct supervision as to what actions should be taken in different situations is provided to the agent. The goal is for the agent to learn a good policy based on the series of reward signals that it receives from the environment through interaction. Common challenges in RL include training the agent to generalize to unseen environment states, designing a suitable strategy to explore promising state space without incurring too much opportunity cost, and what is often referred to as the credit assignment problem, where the agent needs to appropriately attribute the reward signals it receives to the previous actions taken possibly from a distant past.

This setup is typically formalized using a Markov decision process (MDP) which is defined by a set of states S, a set of actions A, a transition function $T(s' \mid s, a)$ representing the probability distribution over the next states following action a from state s, and a reward function $R(s, a)$ representing the scalar reward given to the agent for executing a from s. In each discrete time step t, the agent in state s_t chooses an action a_t based on its policy and transitions to a new state s_{t+1} according to the dynamics model of the environment unknown to the agent. Following each transition, the agent receives a scalar reward r_{t+1} which the agent uses to update its policy. The discounted sum of rewards defined as $R = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$ with $\gamma \in [0, 1]$ is what the agent wants to maximize. Hence, the objective of the agent is to learn a good policy so that the expected discounted return is maximized [1]. That is,

$$\pi^* = \arg\max_{\pi} \mathbb{E}[R \mid \pi]. \tag{2.7}$$

An agent can either learn a policy directly or construct one based on some other learned quantities. In value-based RL methods, an agent learns an estimate of the expected discounted return, represented as the value function $V^\pi(s)$ or the state-action value function $Q^\pi(s, a)$. Given such an estimate, a policy can be constructed by selecting actions that result in the highest estimated expected returns. Q-learning is one such value-based method that using the recursive relation

$$Q(s_t, a_t) = \mathbb{E}_{s_{t+1}}[r_t + \gamma \cdot Q^\pi(s_{t+1}, \pi(s_{t+1}))]$$

iteratively improves the estimate with the bootstrap target as

$$Q(s_t, a_t) \leftarrow (1 - \alpha)Q(s_t, a_t) + \alpha(r_t + \gamma \cdot \max_a Q(s_{t+1}, a)),$$

where α is the learning rate. The algorithm converges to an optimal Q-function in the tabular case, that is, $Q_i \to Q^*$ as $i \to \infty$ [40].

Estimating the Q-function for each state and action pair becomes infeasible when the state or

action space is too large or even continuous. For such problems, a function approximator parameterized by θ is used to estimate the Q-function: $Q(s, a; \theta) \approx Q^*(s, a)$. A simple linear function or a more complex non-linear function such as a deep neural network can be used as the function approximator. Using a deep neural network to approximate the Q-function led to the Deep Q-network (DQN) algorithm that achieved human-level performance on Atari games [20].

2.5 Discussion

In this chapter, we reviewed some of the fundamental concepts discussed throughout this work including basic importance sampling, risk measures to estimate, and formulation of reinforcement learning problems.

We discussed that, in designing a Monte Carlo estimator, the one that is unbiased and of low variance is more desirable, and that importance sampling is a common variance reduction method that is natural to employ in certain situations. Specifically, for problems in which estimating the desired quantity using samples drawn from the nominal distribution is inefficient, one can use an alternative sampling distribution to design a better estimator. It is worth emphasizing that importance sampling makes more sense in such cases where the samples that would contribute most to estimating the target quantity have low chance of being drawn from the nominal distribution. In other cases, simple Monte Carlo estimation may well be sufficient.

In risk evaluation, we are typically interested in estimating quantities that are related to rare events such as the tail risk measures that we discussed. The chance of drawing those samples that are more relevant to such measures is by definition low under the nominal distribution. Hence, importance sampling is conceptually sound for such estimation problems, if an appropriate proposal distribution can be devised.

In the rest of the thesis, we explore several importance sampling ideas in the context of risk estimation, proposing a general sampling framework as well as concrete implementations applicable to different problem settings considered.

Chapter 3

Curriculum Sampling

This chapter introduces an importance sampling method that is explored in detail throughout the thesis called adaptive importance sampling, presenting the fundamental concepts and the general framework. The first section motivates the core idea of adapting proposal distributions during sampling. The second section discusses the fundamental concepts such as using multiple proposal distributions, sample weighting strategies, and the basic framework of adaptive importance sampling. The third section motivates and presents a novel framework called curriculum sampling that incorporates a representation of uncertainty into the sampling procedure such that it can, for instance, be adopted for stable estimation of tail statistics. The fourth section demonstrates applying curriculum sampling to a quantile estimation task presenting the experimental results and analysis.

3.1 Introduction

In order to use importance sampling for estimation well, we need to carefully choose a proposal distribution that is appropriate for the given problem. In general, we would like a distribution that lets us draw more samples from parts of the space that are most relevant to estimating the quantity of interest. Designing such a proposal, however, is often difficult to do a priori. This leads to the idea of *adaptive* importance sampling in which, in an iterative process, we draw samples from one or more proposal distributions that are adapted based on the previous samples to improve the quality of the future samples and hence the estimation accuracy [7].

The idea of using an adaptive procedure is especially relevant to risk estimation of black-box systems in which we have limited understanding of the systems a priori to design an appropriate algorithm. For instance, the cost distribution $c(X)$ induced by the input random vector X as described in Section 2.3 is often unknown a priori and needs to be estimated during the sampling procedure. Hence, it makes intuitive sense for us to adapt our sampling strategy, that is, the proposal distribution, as our estimate of the cost distribution gradually improves with additional samples that

we iteratively collect.

3.2 Adaptive Importance Sampling

3.2.1 Multiple Importance Sampling

Adaptive importance sampling is closely related to the importance sampling method called multiple importance sampling [31, 12] in which a set of proposal distributions $\{q_i(x)\}_{i=1}^n$ is used to draw samples. Given a set of distributions, we can draw samples from each of the proposals, from a mixture of the distributions, etc. in each step of the sampling procedure. Using multiple distributions is useful, for instance, if f and the nominal p are such that finding a single proposal distribution that is approximately optimal, that is, close to being proportional to fp and has heavier tails than p is difficult.

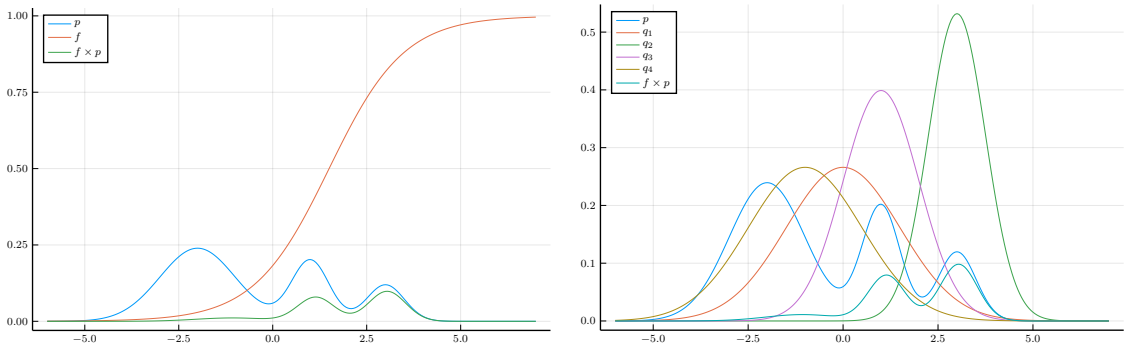
With multiple distributions to sample from, there are also several ways to assign weights to the samples drawn. Recall that, in case of a single proposal distribution, each sample is assigned the weight of $w(X) = p(X)/q(X)$. For multiple importance sampling, common weighting strategies used include the following:

1. Standard weighting [9]:
$$w(X) = \frac{p(X)}{q_i(X)}. \tag{3.1}$$

2. Deterministic mixture (DM) weighting [31]:
$$w(X) = \frac{p(X)}{n^{-1}\sum_{i=1}^n q_i(X)}. \tag{3.2}$$

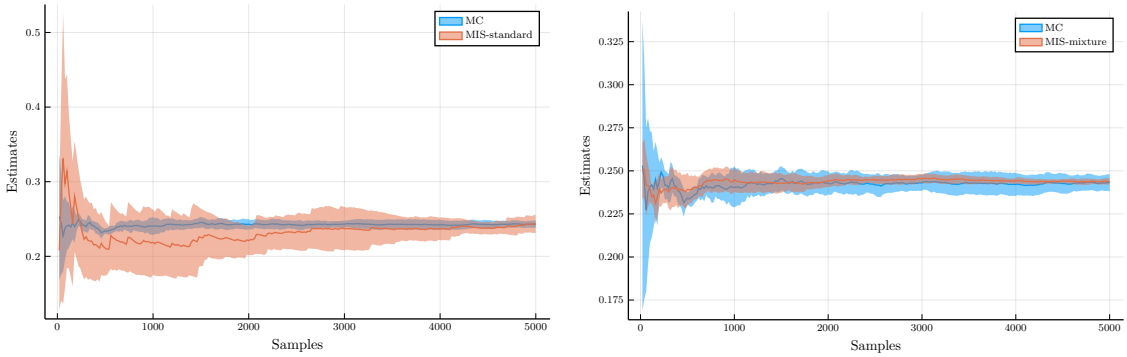
Note that the standard weighting is equivalent to the weighting strategy used for the case of a single proposal. In other words, the weight of each sample is computed based only on the proposal distribution from which the sample was drawn. On the other hand, in deterministic mixture weighting, the weight is computed based on the average of all n proposals for the given sample. Hence, this weighting strategy is more computationally expensive as it involves $n+1$ evaluations of the proposals (as opposed to 2 for the standard weighting), but it generally leads to lower variance than standard weighting. This intuitively makes sense, because using the average in the denominator is likely to result in a more stable weight. Note that we still have an unbiased estimator using this weighting strategy. Besides the two discussed, more sophisticated weighting strategies such as applying non-linear weight transformations have also been explored [22].

Consider the example illustrated in the first plot of the Figure 3.1 in which the nominal p is a mixture of three Gaussian distributions such that fp is multi-modal, or has multiple peaks. Suppose also that we want to use multiple Gaussians as our proposal distributions. Given the multi-modal



(a) Multi-modal nominal distribution. (b) Multiple Gaussian proposals.

Figure 3.1: Multiple importance sampling for a multi-modal nominal.



(a) Standard weighting. (b) Deterministic mixture weighting.

Figure 3.2: Convergence of multiple importance sampling with different weighting schemes.

shape of p (as well as that of fp), it is natural to choose Gaussians each of which is located near one of the peaks of fp. Note that it would be more challenging for a single Gaussian proposal distribution to adequately cover the regions of interest given such a shape.

Figure 3.2 shows convergence graphs of both the baseline of using the nominal distribution and multiple importance sampling with three Gaussians as the proposals. The plots demonstrate that, depending on the weighting strategy used, the performance of multiple importance sampling in terms of variance reduction can vary quite dramatically. The first plot shows the convergence behavior when the standard weighting is used, and, in this case, multiple importance sampling had noticeably larger variance compared to the baseline of simple Monte Carlo. This is due to those samples drawn from the tails of the Gaussians, which have low $q_i(x)$ values that magnify the weights of the samples when the standard weighting is used. In comparison, the second plot shows the convergence behavior when the deterministic mixture weighting is used, and multiple importance sampling, in this case, had lower variance compared to the baseline. Even if samples are drawn from tails of one of the

Algorithm 3.1 Adaptive importance sampling framework.

Input: $\mathcal{D} \leftarrow \{\mathcal{D}_i\}_{i=1}^{n}$ ▷ Initial proposal distributions
Output: $\mathcal{S}$ ▷ Samples collected

1: **for** $i \leftarrow 1$ **to** N **do**
2: $\quad \mathcal{S}' \leftarrow \{\}$
3: $\quad$ **for** $j \leftarrow 1$ **to** n **do**
4: $\quad\quad \{s_1, \ldots, s_k\} \sim \mathcal{D}_i$ ▷ Draw k samples
5: $\quad\quad \{w_1, \ldots, w_k\} \leftarrow \texttt{weight}(\{s_1, \ldots, s_k\})$ ▷ Weight k samples
6: $\quad\quad \mathcal{S}' \leftarrow \mathcal{S}' \cup \{(s_1, w_1), \ldots, (s_k, w_k)\}$
7: $\quad \mathcal{D} \leftarrow \texttt{adapt}(\mathcal{S}', \mathcal{D})$ ▷ Adapt proposal distributions
8: $\quad \mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{S}'$
9: **return** $\mathcal{S}$

proposals, the weights of the samples can still be comparatively moderate, as the average of all proposals is used to compute the weights.

3.2.2 General Framework

In adaptive importance sampling, one or more proposal distributions are iteratively adapted based on previous samples such that subsequent samples to be drawn are more useful for the estimation task at hand. Hence, adaptive importance sampling is closely related to multiple importance sampling discussed above, and several of the ideas such as sample weighting strategies can naturally be adopted.

The general framework of the algorithm, as outlined in algorithm 3.1, can be described as an iterative process that repeats the following three steps: drawing samples from the set of chosen proposal distributions, computing the weights of the samples (optionally update weights of previous samples), and adapting the proposal distributions for the subsequent iteration [7]. Depending on the choices made for the different aspects of the framework, one can have a distinct implementation of the algorithm. For instance, one can decide what types of parameterized distributions to use as the proposals, how many samples to draw in each step, how sample weights are computed, etc.

To illustrate the idea with an example, suppose that we want to estimate $\mathbb{E}_p[f(X)]$ using adaptive importance sampling. Based on our prior knowledge about f and p, we choose a sufficiently flexible function approximator that we use to approximate fp. This strategy is utilizing the general result that using a proposal q that is approximately proportional to $|f|p$ leads to lower variance when estimating the mean as discussed in Section 2.2. In each step of the algorithm, we draw some number of samples according to the current estimation of fp, compute the weights, and adapt the approximation based on the new set of samples. As fp may not necessarily be a valid density, we can use the self-normalized estimator in which the mean of the weights is used as an unbiased estimation of the normalization constant. As this example illustrates, the idea of adapting the proposal distribution based on previous samples is a powerful one that opens the door to many

advanced and effective methods. Bugallo et al. [7] discusses several of such advanced algorithms in greater detail.

3.3 Curriculum Sampling

Curriculum sampling is a new framework motivated by the idea of proposal adaptation in adaptive importance sampling that is especially suitable for estimation problems that can logically be broken into series of steps. The core idea of curriculum sampling is to design a *curriculum* of intermediate target statistics and the corresponding proposals in order to estimate the desired measure in a stable manner. This idea can naturally be applied to a broad set of estimation problems, but we present the framework in the context of estimating the risk measures VaR and CVaR described in Section 2.3.

3.3.1 Estimation of Risk Measures

Given a deterministic cost function $c : \mathbb{R}^d \to \mathbb{R}$, we estimate VaR_α and CVaR_α using simple Monte Carlo by drawing n independent samples $\{X_1, X_2, \ldots, X_n\}$ from the nominal p and using the following estimators

$$\tilde{\text{VaR}}_\alpha = \inf\{c \mid \tilde{F}_n(c) \geq \alpha\} \tag{3.3}$$

$$\tilde{\text{CVaR}}_\alpha = \tilde{\text{VaR}}_\alpha + \frac{1}{n(1-\alpha)} \sum_{i=1}^{n} \max\{c(X_i) - \tilde{\text{VaR}}_\alpha, 0\}, \tag{3.4}$$

where $\tilde{F}_n(c)$ is the empirical c.d.f. computed as

$$\tilde{F}_n(c) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}\{c(X_i) \leq c\}. \tag{3.5}$$

Given that the measures are potentially concerned with extreme tail regions of the distribution, the number of samples n may need to be very large before reaching our desired level of accuracy, especially if the cost distribution has a long, thin tail. Hence, importance sampling is a natural variance reduction strategy to use to estimate these measures.

Let q be an alternative distribution such that $q(x) \neq 0$ whenever $p(x) \neq 0$. For risk estimation, we might choose q that we believe would more likely to lead to regions of high costs. Given such a proposal q, we draw n independent samples $\{X_1, X_2, \ldots, X_n\}$ from q and use the following well-known importance sampling estimators

$$\hat{\text{VaR}}_\alpha = \inf\{c \mid \hat{F}_n(c) \geq \alpha\} \tag{3.6}$$

$$\hat{\text{CVaR}}_\alpha = \text{VaR}_\alpha + \frac{1}{n(1-\alpha)} \sum_{i=1}^{n} \max\{c(X_i) - \text{VaR}_\alpha, 0\} w(X_i), \tag{3.7}$$

where $\hat{F}_n(c)$ is the empirical c.d.f.

$$\hat{F}_n(c) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\{c(X_i) \leq c\} w(X_i) \tag{3.8}$$

and $w(X_i) = p(X_i)/q(X_i)$ the likelihood ratio. Under mild assumptions, we can establish the following asymptotic properties of the estimators

$$\sqrt{n}(\hat{\text{VaR}}_\alpha - \text{VaR}_\alpha) \Rightarrow \frac{\sqrt{\text{Var}_q[\mathbb{1}\{c(X) \geq \text{VaR}_\alpha\} w(X)]}}{p(\text{VaR}_\alpha)} N(0,1) \tag{3.9}$$

$$\sqrt{n}(\hat{\text{CVaR}}_\alpha - \text{CVaR}_\alpha) \Rightarrow \frac{\sqrt{\text{Var}_q[\max\{c(X) - \text{VaR}_\alpha, 0\} w(X)]}}{1-\alpha} N(0,1) \tag{3.10}$$

as $n \to \infty$. Sun and Hong [38] gives greater detail on the asymptotic variance analysis. Intuitively, the above analysis suggests that the samples whose costs are in the region near or above the α-quantile are considered most useful, and the proposal distributions that put higher weights over such samples can lead to lower variance compared to the simple Monte Carlo estimators.

We consider the following such distributions

$$q_{\text{VaR}_\alpha}(x) \propto p(x) \cdot \mathbb{1}\{c(x) \geq \text{VaR}_\alpha\} \tag{3.11}$$

for the VaR_α estimator and

$$q_{\text{CVaR}_\alpha}(x) \propto p(x) \cdot (c(x) - \text{VaR}_\alpha) \cdot \mathbb{1}\{c(x) \geq \text{VaR}_\alpha\} \tag{3.12}$$

for the CVaR_α estimator. One subtlety to note is that among the samples in the tail region, the proposal for the VaR_α estimator prefer those samples with higher densities under the nominal distribution p, whereas that for the CVaR_α estimator prefer those samples with higher expected costs under the nominal.

Since VaR_α and CVaR_α provide complementary information about potential risks, both are often estimated simultaneously. And, given the definition of CVaR_α, an accurate estimation of VaR_α is required before it itself can be accurately estimated. Hence, the challenge is in designing the sampling procedure in such a way that would allow us to accurately estimate such a dependent set of statistics in a sample efficient manner.

3.3.2 Approximate Proposals

The proposal distributions described above cannot be directly used in practice, as they depend on the unknown VaR_α, which is one of the very quantities that we want to estimate. One natural relaxation to apply is to use the running estimate $\tilde{\text{VaR}}_\alpha$ in place of VaR_α. In practice, however, this can lead to biased estimates if not used carefully.

Consider the relaxed proposal distribution

$$q_{\mathrm{VaR}_\alpha}(x) \propto p(x) \cdot \mathbb{1}\{c(x) \geq \tilde{\mathrm{VaR}}_\alpha\}, \tag{3.13}$$

which draws samples in the *estimated* α-quantile region according to the nominal p. Suppose that at some point during the sampling procedure, our running estimate $\tilde{\mathrm{VaR}}_\alpha$ exceeds the true VaR_α. In subsequent iterations, we would sample from the region beyond the true α-quantile, leading our running estimate $\tilde{\mathrm{VaR}}_\alpha$ to further overestimate the true VaR_α. This can ultimately lead to highly biased estimates of the quantile. Hence, in order to compute an unbiased estimate in a stable manner, we need to be conservative about how we use the running estimate $\tilde{\mathrm{VaR}}_\alpha$ even if this requires more samples to be drawn to achieve the desired level of variance. Otherwise, once the estimate exceeds the true quantile, the bias issue can become progressively worse for the rest of the sampling procedure.

To address this overestimation issue, we can apply another relaxation to the proposal as

$$q_{\mathrm{VaR}_\alpha}(x) \propto p(x) \cdot P(c(x) \geq \mathrm{VaR}_\alpha), \tag{3.14}$$

where we replace the indicator function with an estimated probability function. The basic idea is to assign non-zero probabilities even to those samples that are believed to be below the α-quantile. This could give us a chance to correct our running estimate when it exceeds the true VaR_α by allowing us to draw samples in the neighborhood of $\tilde{\mathrm{VaR}}_\alpha$. Independent of the choice of the probability function, it is important to incorporate some representation of uncertainty in our running estimate into the function so that we can adjust the probabilities as our confidence in the estimate changes. This is to allow us to maintain the added stability in the procedure without sacrificing sample efficiency too much. We discuss possible candidates for the probability function in the following section.

For settings in which the value of α is extremely close to 1 or our estimate of the uncertainty in $\tilde{\mathrm{VaR}}_\alpha$ is too noisy, we can apply yet another relaxation as

$$q_{\mathrm{VaR}_{\alpha'}}(x) \propto p(x) \cdot P(c(x) \geq \mathrm{VaR}_{\alpha'}), \tag{3.15}$$

where $\alpha' < \alpha$. Sampling from the more modest α'-quantile region instead of the original α is a reasonable strategy especially in the early part of the sampling procedure when the estimate $\tilde{\mathrm{VaR}}_\alpha$ is likely to be noisy.

Note that there is an inherent tension between stability of the procedure and sample efficiency. That is, a modest value of α' can lead to a more stable procedure but correspondingly require more samples until convergence. As a compromise, it is also natural to use a type of a schedule for the value of α', where we start out with a more modest value and progressively increase it towards the original target α. This is similar to the idea of using a learning rate schedule in training machine

Algorithm 3.2 Curriculum sampling framework.

Input: $\mathcal{D} \leftarrow \{\mathcal{D}_i\}_{i=1}^{d}$, $C \leftarrow \{c_i\}_{i=1}^{d}$ ▷ Distributions and required confidence levels
Output: $\mathcal{S}$ ▷ Samples collected

```
for i ← 1 to d do
    D ← D_i
    c ← −∞
    while c < c_i do
        D ← adapt(S, D, c)          ▷ Adapt distribution
        s ∼ D                       ▷ Draw samples
        S ← S ∪ {s}
        c ← confidence(S)           ▷ Update confidence score
return S
```

learning models, where we typically start with a relatively high learning rate at the beginning and gradually reduce it towards the end for smoother convergence [41].

Once we have an accurate estimate of VaR_α, we can start drawing samples from a proposal distribution for estimating CVaR_α. By applying a similar relaxation as above, we have a proposal

$$q_{\text{CVaR}_\alpha}(x) \propto p(x) \cdot (c(x) - \tilde{\text{VaR}}_\alpha) \cdot P(c(x) \geq \text{VaR}_\alpha), \tag{3.16}$$

where we use the running estimate $\tilde{\text{VaR}}_\alpha$. If the probability function P is designed in such a way that it assigns higher probabilities to samples of large values, then the above proposal can also be simplified as

$$q_{\text{CVaR}_\alpha}(x) \propto p(x) \cdot P(c(x) \geq \text{CVaR}_\alpha), \tag{3.17}$$

where the estimated probability that $c(x) \geq \text{VaR}_\alpha$ is replaced with $c(x) \geq \text{CVaR}_\alpha$.

Note that if we reverse the series of relaxations applied above, we end up with a reasonable sequence of steps that we can take to simultaneously estimate both VaR_α and CVaR_α in a stable manner. That is, we use the following proposal distributions in order

1. Estimate $\text{VaR}_{\alpha'}$: $q_{\text{VaR}_{\alpha'}}(x) \propto p(x) \cdot P(c(x) \geq \text{VaR}_{\alpha'})$ for $\alpha' < \alpha$.
2. Estimate VaR_α: $q_{\text{VaR}_\alpha}(x) \propto p(x) \cdot P(c(x) \geq \text{VaR}_\alpha)$.
3. Estimate CVaR_α: $q_{\text{CVaR}_\alpha}(x) \propto p(x) \cdot (c(x) - \tilde{\text{VaR}}_\alpha) \cdot P(c(x) \geq \text{VaR}_\alpha)$.

We transition to the subsequent step when we are reasonably confident about our estimate of the current target statistic.

We call this type of sampling strategy that consists of a series of intermediate proposal distributions in order to estimate the target statistics in a stable manner *curriculum sampling*. This is similar to the idea of curriculum learning in which a sequence of training tasks is carefully designed in order to speed up convergence of training deep neural networks [3]. The algorithm can also be

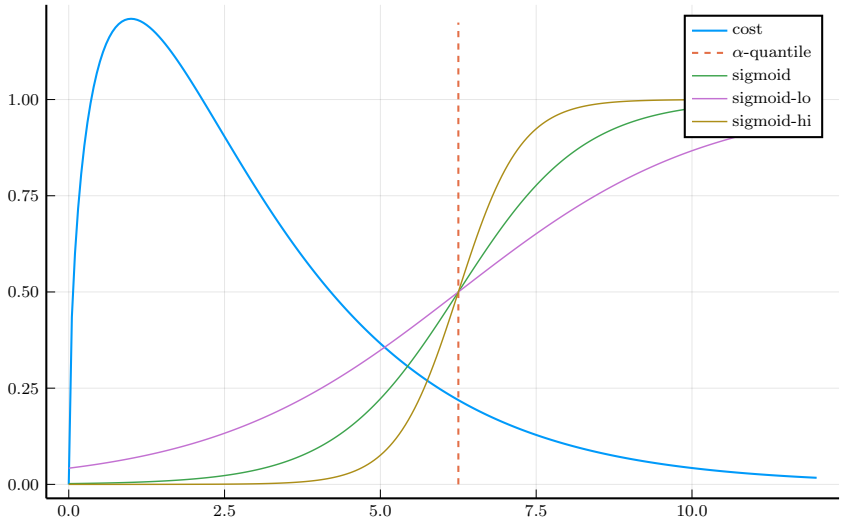


Figure 3.3: Sigmoid probability functions.

viewed as a type of adaptive importance sampling that is suitable when a dependent set of statistics need to be estimated simultaneously.

3.3.3 Implementation

We now describe possible implementations of the algorithm in the context of estimating VaR_α and CVaR_α, and propose suitable choices for such components of the algorithm as the probability function and representation of uncertainty. Note that the implementation detail depends on the estimation problem at hand, but the core idea of the framework is one that is more broadly applicable.

Choice of Probability Function

In choosing the probability function P, as used in Eq. (3.13), we want a function that is fast to compute and one where we can easily incorporate our uncertainty estimate. One natural choice of the function P is a type of sigmoid function, which is also commonly used in such models as logistic regression [19], that is centered around the target estimate, that is,

$$P(c(x) \geq \text{VaR}_\alpha) = \frac{1}{1 + \exp\left(-(c(x) - \tilde{\text{VaR}}_\alpha)/s\right)}, \tag{3.18}$$

where s is a quantity computed based on our representation of uncertainty in the estimate $\tilde{\text{VaR}}_\alpha$.

The probability function is adapted in such a way that the value of s is decreased (or increased) if our uncertainty in the estimate decreases (or increases). Because the coefficient s determines the steepness of the sigmoid curve, the function approaches the step function $\mathbb{1}\{c(x) \geq \tilde{\text{VaR}}_\alpha\}$ as our

uncertainty decreases, leading to higher probabilities of drawing samples above the estimate $\tilde{\text{VaR}}_\alpha$. If we are uncertain about the estimate, however, we would correspondingly increase the slope s such that samples are drawn more broadly in the neighborhood of the current estimate. See Figure 3.3 for an illustration.

Representation of Uncertainty

One simple approach to quantifying uncertainty of our estimate is to use empirical variance computed based on intermediate estimates. For instance, suppose that we want to estimate VaR_α for a given α. During the sampling procedure, we compute intermediate estimates of VaR_α as we collect additional samples to have a set of estimates $\{\tilde{\text{VaR}}_\alpha^1 \dots \tilde{\text{VaR}}_\alpha^m\}$. Then, we can compute the empirical variance as

$$S^2 = \frac{1}{m-1}\sum_{i=1}^{m}(\tilde{\text{VaR}}_\alpha^i - \bar{\text{VaR}}_\alpha),$$

where $\bar{\text{VaR}}_\alpha = \frac{1}{m}\sum_{i=1}^{m}\tilde{\text{VaR}}_\alpha^i$ is the empirical mean.

Given such an empirical variance, a few approaches are natural for translating it into the slope s in the sigmoid probability function as a way of adapting the proposal distribution. One approach is to set s such that the probability value of $\tilde{\text{VaR}}_\alpha + S$, which is the cost that is one empirical standard deviation above the mean $\tilde{\text{VaR}}_\alpha$, to be some chosen $p_s > 0.5$:

$$p_s = \frac{1}{1+\exp\left(-((\tilde{\text{VaR}}_\alpha + S) - \tilde{\text{VaR}}_\alpha)/s\right)} = \frac{1}{1+\exp\left(-S/s\right)}.$$

Simple algebraic manipulation yields the slope

$$s = -S/\log\left((1-p_s)/p_s\right).$$

Depending on the choice of p_s or the desired value for which we want the probability to be p_s, different slope calculations are possible.

Alternatively, in case such empirical variance is too noisy for such use, simple heuristics such as starting with some initial slope s_{init} and annealing it towards some minimum slope s_{min} are possible. Regardless of the approach used, we experimentally found that having a minimum slope could be beneficial for numerical stability, which we discuss more in the experiments section below.

Variations

Beyond the general framework discussed above for estimating VaR_α and CVaR_α, we present several extensions that are natural to implement.

Mixture Distribution. By definition, estimating CVaR_α requires that we have an accurate estimation of VaR_α. To estimate both measures simultaneously, it could make sense to combine the two steps into one and use a mixture distribution that is a convex combination of the two proposals. That is,

$$q_{\text{mixture}_\alpha}(x) \propto \lambda \cdot p(x) \cdot P(c(x) \geq \text{VaR}_\alpha) + (1-\lambda) \cdot p(x) \cdot (c(x) - \tilde{\text{VaR}}_\alpha) \cdot P(c(x) \geq \text{VaR}_\alpha), \quad (3.19)$$

where $\lambda \in [0, 1]$ is the mixture parameter. For such a distribution, we can anneal the parameter λ to gradually put more weight on the CVaR_α component as our confidence in the estimate of VaR_α increases. Note that a proper normalization is desirable when mixing the two distributions, as the proposal for the CVaR_α estimator has the cost component that can otherwise dominate the sum.

Burn-in Period. The estimate $\tilde{\text{VaR}}_\alpha$ is likely to be noisy at the start of the sampling procedure, especially for values of $\alpha \approx 1$. One approach we discussed for addressing this issue was to start with $\alpha' < \alpha$ before the target α. We can also complement this approach with an initial period of sampling from the nominal distribution until we start sampling from the proposal for estimating $\text{VaR}_{\alpha'}$. We call this the *burn-in period* given its resemblance to a similar idea in Markov chain Monte Carlo (MCMC) [33]. However, the samples collected during the burn-in period are not discarded but used in estimation.

Alpha Schedule. In designing a curriculum that consists of multiple of the steps of using $\alpha' < \alpha$, we can use a few simple scheduling algorithms to set α'. Given the number of such steps N, we can set α' such that it approaches the target α either linearly or exponentially. That is,

$$\alpha' + k(\alpha - \alpha')/N \quad \text{or} \quad \alpha' r^k,$$

where $k \in [0, N)$ is the step count and $r = \exp(\log(\alpha/\alpha')/N)$ the multiplicative factor. Similar approaches are also commonly used in designing learning rate schedules for training machine learning models as previously discussed.

3.4 Experiments

In this section, we discuss experimental results of applying curriculum sampling to the problem of quantile estimation and evaluate some of the core ideas discussed in this chapter. We first give an overview of the setup including the cost distribution used, then present and analyze the results.

3.4.1 Setup

For quantile estimation experiments, we constructed a one-dimensional discrete distribution that the random variable X is sampled from and considered the identity function as the cost function c, that is, $c(X) = X$. We created the cost distribution in such a way that it has most of its mass concentrated around a low cost region with a long, thin tail with a few peaks in regions of high costs. In particular, we first created a continuous distribution of the desired shape, computed the densities for a chosen discrete support, and normalized the values to construct a discrete distribution over the support. The support of the discretized distribution used was the set of numbers between 0.0 and 25.0, inclusive, that are 0.05 apart, that is, $\{0.0, 0.05, \ldots, 24.95, 25.0\}$.

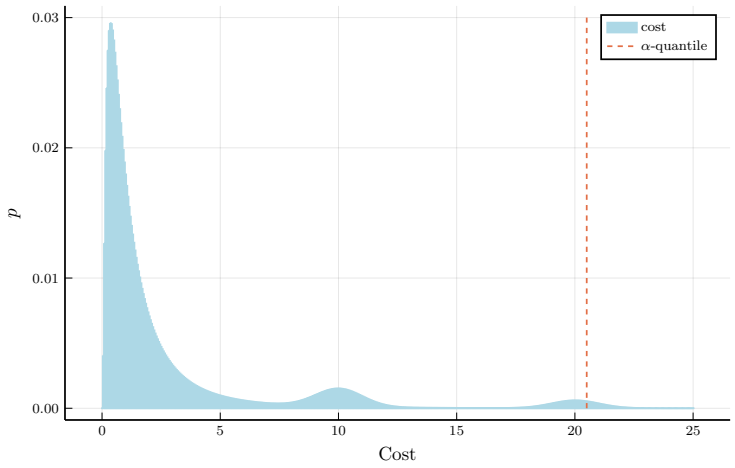


Figure 3.4: A discrete cost distribution with 0.99-quantile.

The continuous distribution used is a mixture distribution of a log normal distribution with log-mean 0.0 and scale 1.0, a Gaussian with mean 10.0 and variance 1.0, and a Gaussian with mean 20.0 and variance 1.0 with mixture weights of 0.9, 0.07, and 0.03, respectively. Specifically,

$$f_{\text{mix}}(x) = \frac{0.9}{x\sqrt{2\pi}} \exp\left(-\frac{(\log x)^2}{2}\right) + \frac{0.07}{\sqrt{2\pi}} \exp\left(-\frac{(x-10)^2}{2}\right) + \frac{0.03}{\sqrt{2\pi}} \exp\left(-\frac{(x-20)^2}{2}\right). \quad (3.20)$$

For discretizing this mixture distribution, we chose the support to be the set of numbers between 0.0 and 25.0, inclusive, evenly distanced by 0.05, and normalized the corresponding densities to create a valid probability mass function. Intuitively, the log normal component describes the mass concentrated in a low cost region and the Gaussians the two peaks in high cost regions. See Figures 3.4 for an illustration.

Given this distribution, quantiles of which we want to estimate, we evaluated the curriculum sampling algorithm in terms of variance reduction and estimation accuracy for the following cases:

- Using fixed sigmoid probability functions.

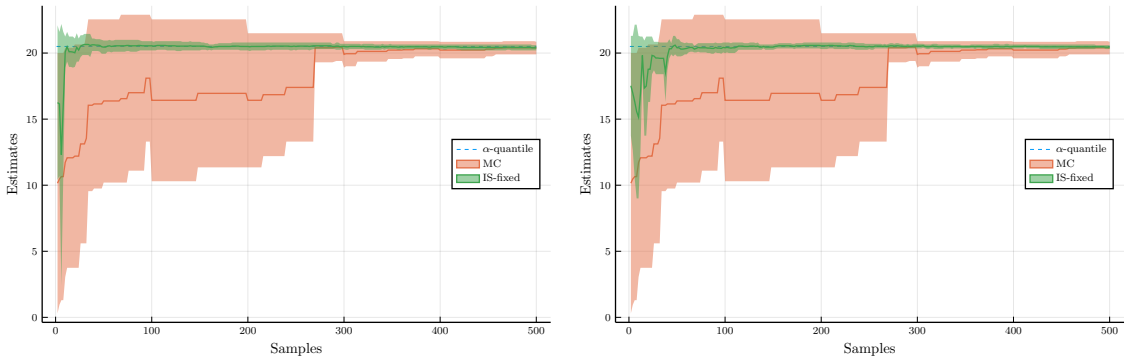
- Using sigmoid probability functions with fixed slope and adaptive mean at $\tilde{\text{VaR}}_\alpha$.
- Using sigmoid probability functions with adaptive slope and adaptive mean at $\tilde{\text{VaR}}_\alpha$.

For the cases of adaptive slope and/or mean, we also examined the effect of the frequency of updates on the convergence behavior. To estimate the variance of the estimators, we ran 10 trials of each experiment and report the results.

3.4.2 Results and Analysis

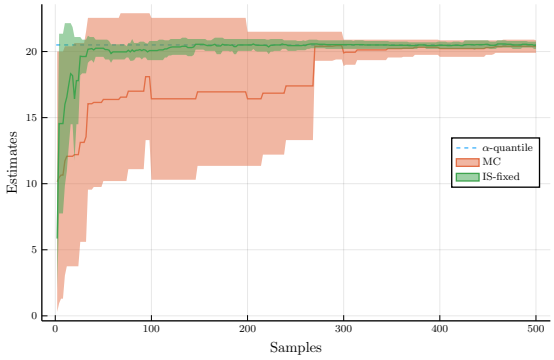
Fixed Proposals

Figure 3.5 compares the simple Monte Carlo and the importance sampler with fixed sigmoid probability functions for estimating the 0.99-quantile of the cost distribution, which can be directly computed from the distribution and is 20.5.



(a) Sigmoid with $\mu = 20.0$ and $s = 4.0$ (1A).

(b) Sigmoid with $\mu = 25.0$ and $s = 4.0$ (1B).



(c) Sigmoid with $\mu = 10.0$ and $s = 4.0$ (1C).

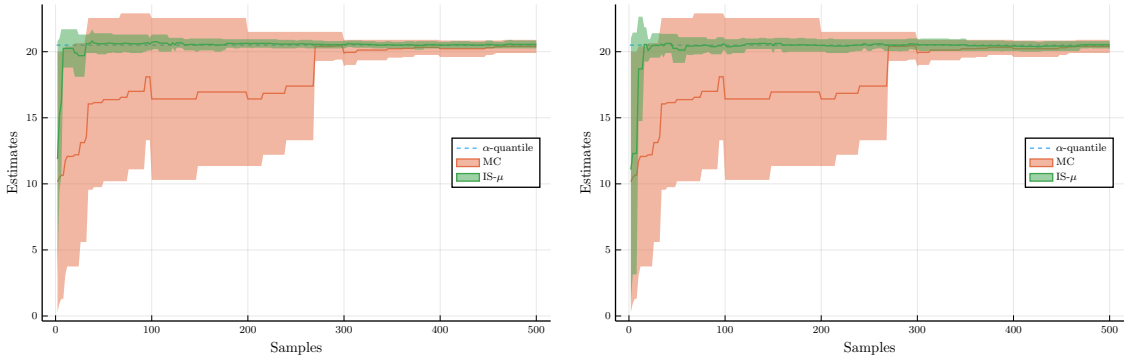
Figure 3.5: Convergence of fixed mean and slope.

Note that when the sigmoid function is located near the true quantile, the importance sampler converges quickly even with a few tens of samples and exhibits a significantly smaller variance than

the Monte Carlo estimator. In comparison, when the sigmoid function is located beyond the true quantile so that we favor samples in a more extreme tail region than desired, the importance sampler exhibits a relatively larger variance and requires drawing more samples before convergence. This demonstrates the potential issue discussed in Section 3.3.2 in which, if the running quantile estimate is not carefully used in a proposal such as the one in Eq. (3.13), the importance sampler could result in a larger variance. Finally, if the sigmoid function is located far below the true quantile, we see a larger variance as expected.

Adaptive Mean

Instead of using a fixed probability function, we can use the running estimate $\tilde{\text{VaR}}_\alpha$ to adaptively change the mean of the sigmoid function and hence the proposal distribution, as we collect more samples. Figure 3.6 shows convergence graphs of the importance sampler using a sigmoid probability function initialized with $\mu = 10.0$ and $s = 4.0$ and whose mean is updated every some fixed number of steps to the running estimate $\tilde{\text{VaR}}_\alpha$. It is more computationally expensive to update the function more frequently, but it could more quickly adapt the probability function to be near the desired location, though higher noise can be observed in the beginning.



(a) Sigmoid with updates every 10 steps (2A). (b) Sigmoid with updates every 2 steps (2B).

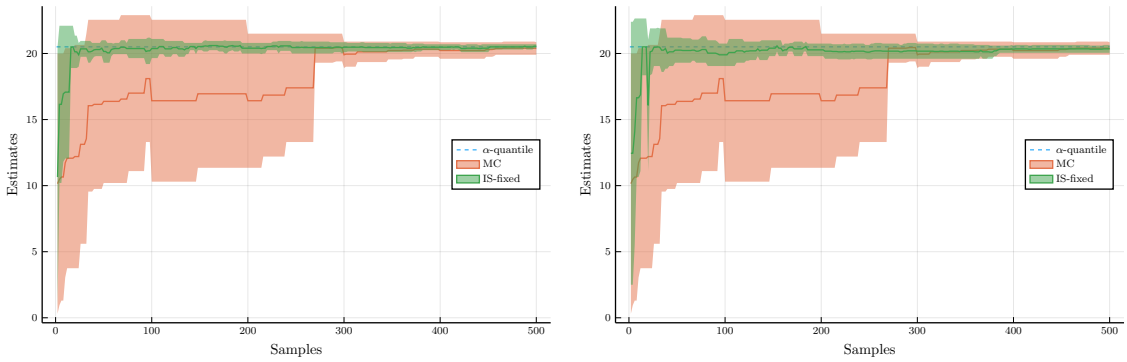
Figure 3.6: Convergence of adaptive mean with initial $\mu = 10.0$ and $s = 4.0$.

Note that compared to Figure 3.5, using adaptive mean exhibits a lower variance even when the initial location is equally suboptimal in terms of its proximity to the true quantile with more frequent updates possibly leading to even a lower variance.

Adaptive Mean and Slope

To better estimate the desired proposal, we can adapt the slope in addition to adapting the mean based on the estimate $\tilde{\text{VaR}}_\alpha$. In this experiment, we used a heuristic-based slope adaptation where the slope is set such that the probability of the value one empirical standard deviation above the

current mean is at 0.95. We also used a minimum slope of 2.0 for numerical stability to guard against the curve from becoming overly steep due to a noisy variance estimate. Figure 3.7 shows convergence graphs of the importance sampler with a sigmoid function initialized the same as before and whose mean and slope are updated every some fixed number of steps.



(a) Sigmoid with updates every 10 steps (3A). (b) Sigmoid with updates every 2 steps (3B).

Figure 3.7: Convergence of adaptive mean and slope with initial $\mu = 10.0$ and $s = 4.0$.

Using adaptive slope exhibits a similar level of variance reduction as when only the mean is adapted, though arguably slightly more noise can be observed at the beginning of the procedure. This is presumably due to that we update the two parameters independently of one another, which can be less desirable when the estimate $\tilde{\text{VaR}}_\alpha$ based on which we apply the adaptation is noisy. A more sophisticated adaptation scheme in which we decide how to update multiple parameters as a whole is conceivable.

Table 3.1 gives a summary of the experiments with the experiment identifiers given in the figures.

Table 3.1: Estimation of the 0.99-quantile using curriculum sampling.

Exp.	$n = 10$	$n = 30$	$n = 50$	$n = 100$	$n = 300$
MC	11.725 ± 8.675	13.125 ± 7.525	16.375 ± 6.175	16.425 ± 6.125	19.900 ± 1.000
1A	20.075 ± 1.175	20.650 ± 0.800	20.450 ± 0.500	20.550 ± 0.350	20.500 ± 0.200
1B	15.150 ± 6.150	19.625 ± 1.225	20.400 ± 0.650	20.400 ± 0.400	20.525 ± 0.175
1C	16.000 ± 6.150	19.625 ± 1.275	20.250 ± 0.650	20.075 ± 0.725	20.525 ± 0.325
2A	20.250 ± 1.450	19.700 ± 1.600	20.600 ± 0.800	20.700 ± 0.600	20.550 ± 0.300
2B	18.700 ± 3.950	20.475 ± 0.925	20.250 ± 1.100	20.525 ± 0.425	20.575 ± 0.425
3A	17.075 ± 5.025	20.350 ± 0.600	20.150 ± 0.850	20.375 ± 0.725	20.475 ± 0.375
3B	16.650 ± 6.000	20.500 ± 1.450	20.250 ± 0.950	19.900 ± 0.850	20.175 ± 0.575

3.5 Discussion

As discussed in this chapter, the idea of adapting the proposal distribution in importance sampling based on previous samples is particularly desirable in case constructing an appropriate proposal a priori is difficult. This may be the case, for instance, for estimating potential risks of black-box systems, where we have limited understanding a priori to design a suitable importance sampler.

Extending this idea of proposal adaptation, we proposed a novel sampling framework called curriculum sampling, where we approach a given problem using a series of proposal distributions suitable for estimating the corresponding intermediate statistics in order to estimate the original target statistics in a stable manner. For instance, this framework may be adopted for estimating extreme tail statistics or a dependent set of such statistics. We presented an implementation of this framework for estimating the risk measures VaR and CVaR, and demonstrated how a representation of uncertainty can be incorporated into the sampling procedure to adapt the proposal distribution.

One note worth mentioning on the importance sampler presented for estimating Var and CVaR is that, in case the random vector X comes from a continuous distribution, a more sophisticated proposal than the one defined in Eq. (3.14) should be considered. This is because it is potentially inefficient to sample directly according to Eq. (3.14) in the continuous case. It is still possible, for instance, by using a type of rejection sampling, where we draw samples according to the nominal p first then decide whether to accept or reject the samples based on the probability function. However, this requires sampling from the nominal distribution, which essentially defeats the purpose of using importance sampling for the estimation. To this end, approaches including using Gaussian kernel function to design approximate proposals have been explored for quantile estimation as in Morio [29].

In the following chapter, we apply the importance sampling ideas discussed in this chapter to estimate risks of sequential black-box systems. In particular, we consider methods from reinforcement learning, which deals with the problem of sequential decision making under uncertainty, to design an appropriate importance sampler.

Chapter 4

Risk Estimation of Sequential Systems

This chapter introduces a search-based sampling algorithm that is suitable for evaluating sequential decision making systems. In the first section, we briefly discuss the motivation behind assessing risks of sequential systems. In the second section, we formalize the problem of estimating risks of such systems and discuss how the importance sampling ideas from the previous chapter can be adopted. The third section introduces an importance sampler based on a tree search algorithm that can be applied to this estimation problem. We then present experimental results to demonstrate the algorithm and close with a discussion.

4.1 Introduction

Sequential decision making is concerned with developing a system (or an agent) that is capable of making decisions typically under some source of uncertainty over a series of discrete time steps so as to maximize gains. Many problems of practical interest are naturally formulated as sequential decision tasks [2], and an extensive body of work exists on the methods for solving such problems including a range of reinforcement learning (RL) algorithms [23].

Recent advances including use of deep neural networks for function approximation and representation learning [27] allowed scaling RL to problems previously considered intractable [1], and increasingly more of such learning-based autonomous systems are being developed for real-world applications such as autonomous driving [21]. Rigorous safety evaluation of these systems is therefore becoming increasingly important. In this chapter, we present methods for efficiently evaluating risks of such sequential systems using a tree search-based algorithm, using the importance sampling approaches discussed in the previous chapter.

4.2 Sequential Systems

In a sequential task, a decision making system is trained to act in a given environment through a series of interactions in order to maximize the rewards received. Such a sequential task is generally formulated as a Markov decision process (MDP) with the goal of learning a policy $\pi : S \to A$, that is, a mapping from the state space to action space, so as to maximize the expected return (see Section 2.4). Given a sequential system that has been trained, our task is to estimate risks of the system in presence of disturbances to the states that the system observes. We formulate this task also as an MDP and present a tree search algorithm for solving it.

4.2.1 Adversarial MDP

Consider a system that has been trained to solve a sequential task $\mathcal{M}$ with state and action spaces S and A, respectively. Suppose that the environment to which the system is deployed has sources of state-dependent disturbances such that a random disturbance X drawn according to probability density $p(x \mid s)$ is applied to the state $s \in S$ that the system observes. Given this noisy state s, the system chooses an action $a \in A$ based on its learned policy π, and transitions to the subsequent state according to the transition model T of the environment. Hence, at each time step t,

$$x_t \sim p(x \mid s_t), \quad a_t = \pi(s_t + x_t), \quad s_{t+1} \sim T(s' \mid s_t, a_t). \tag{4.1}$$

The process continues until the system reaches a terminal state $s_n \in S_{\text{term}}$ producing a trajectory $\tau = \{s_1, x_1, \ldots, x_{n-1}, s_n\}$ and incurring a cost $c(\tau)$. In case of autonomous driving, for instance, the disturbance could be noise added to system sensors and the cost the expected cost of a collision due to a series of such sensor disturbances applied to the system. See [13] for a similar but more formal formulation of the safety validation problem.

Given a model of state-dependent disturbances, our goal is to understand the distribution of the cost induced by the disturbance distributions, and to efficiently estimate relevant risk measures. This estimation task can itself be treated as a sequential task and be formulated as what we call an adversarial MDP $\mathcal{M}_{\text{adv}}$ defined by the following components:

- A set of states S_{adv} equivalent to S.
- A set of actions A_{adv} of the state-dependent disturbance distributions.
- A transition function $T_{\text{adv}}(s' \mid s, a) = T(s' \mid s, \pi(s + a))$, where $s \in S_{\text{adv}}$ and $a \in A_{\text{adv}}$.
- A reward function designed based on the definition of cost for the given application.

Note that solving for the task $\mathcal{M}_{\text{adv}}$ in the canonical sense would be developing a system that has learned to make sequential decisions (disturbances to apply) in order to maximize the cost. While

common RL methods can be applied to this MDP to estimate the worst-case cost, we are interested in estimating more subtle tail measures such as VaR_α and CVaR_α. Moreover, for efficient estimation, the importance sampling methods discussed previously are natural to adopt for the sequential setting considered.

4.2.2 Importance Sampler

Given a state-dependent nominal distribution $p(x \mid s)$ over disturbances, we have the following distribution over trajectories (the corresponding sequences of disturbances)

$$p(\tau) \propto \prod_{t=1}^{n-1} p(x_t \mid s_t), \tag{4.2}$$

which induces a distribution over the cost associated with the trajectories. The goal is then to construct a proposal distribution $q(x \mid s)$, and hence $q(\tau)$, in order to efficiently compute VaR_α and CVaR_α of the cost distribution.

The results from Section 3.3.1 lead to the following proposal distributions that we can use to design importance sampling estimators

$$q_{\mathrm{VaR}_\alpha}(\tau) \propto p(\tau) \cdot \mathbb{1}\{c(\tau) \geq \mathrm{VaR}_\alpha\} \tag{4.3}$$

$$q_{\mathrm{CVaR}_\alpha}(\tau) \propto p(\tau) \cdot (c(\tau) - \mathrm{VaR}_\alpha) \cdot \mathbb{1}\{c(\tau) \geq \mathrm{VaR}_\alpha\}. \tag{4.4}$$

However, the above proposals depend on the unknown VaR_α, and the indicator functions as used above could in practice lead to biased estimates. We instead use the approximate proposals

$$q_{\mathrm{VaR}_\alpha}(\tau) \propto p(\tau) \cdot P(c(\tau) \geq \mathrm{VaR}_\alpha) \tag{4.5}$$

$$q_{\mathrm{CVaR}_\alpha}(\tau) \propto p(\tau) \cdot (c(\tau) - \tilde{\mathrm{VaR}}_\alpha) \cdot P(c(\tau) \geq \mathrm{VaR}_\alpha), \tag{4.6}$$

where $\tilde{\mathrm{VaR}}_\alpha$ is the running estimate of VaR_α and the probability function defined as in Eq. (3.18). See Section 3.3.2 for more discussion on why such approximate proposals are used.

Finally, the state-dependent proposal $q(x \mid s)$ that would result in the distributions over trajectories above is given by

$$q(x_t \mid s_t) = \frac{p(x_t \mid s_t) \cdot Q(s_t, x_t)}{\mathbb{E}_p[Q(s_t, X)]}, \tag{4.7}$$

where

$$Q(x_t, s_t) = \begin{cases} P(c(\tau) \geq \text{VaR}_\alpha) & \text{if } s_t \in S_{\text{term}} \text{ and for VaR} \\ (c(\tau) - \tilde{\text{VaR}}_\alpha) \cdot P(c(\tau) \geq \text{VaR}_\alpha) & \text{if } s_t \in S_{\text{term}} \text{ and for CVaR} \\ \mathbb{E}_p[Q(s_{t+1}, X)] & \text{otherwise} \end{cases} \tag{4.8}$$

It is straightforward to verify this by computing a product similar to the one in Eq. 4.2. Given a trajectory τ, the corresponding importance weight is then

$$w(\tau) = \frac{p(\tau)}{q(\tau)} = \prod_{t=1}^{n-1} \frac{p(x_t \mid s_t)}{q(x_t \mid s_t)}. \tag{4.9}$$

Having derived the target state-dependent proposals to use, we next discuss a search-based sampling method for approximating the distributions.

4.3 Tree Importance Sampling

Monte Carlo tree search (MCTS) [15] is a heuristic algorithm that builds a search tree based on random sampling in the decision space that is used to approximate optimal decisions in a given domain [6]. It has had a profound impact on a range of applications, especially on sequential decision tasks such as AI games. In this section, we discuss some of the core ideas of the algorithm and discuss how they can be adopted to design an importance sampler used to assess risks of sequential systems.

4.3.1 Monte Carlo Tree Search

In MCTS, the true value of an action in a given state is approximated using random simulations. Based on these values, the strategy used to search over the decision space is adapted to efficiently find optimal actions. At a conceptual level, this is similar to the basic idea behind adaptive importance sampling in which the proposal distribution is adjusted based on previous samples for efficient estimation. The results of the random simulations are stored in a tree, and the stored estimates become increasingly accurate as the algorithm progresses. The tree is built in an iterative manner until some computational budget such as a maximum time, or number of iterations is exhausted. Each iteration consists of the following four steps:

1. Selection: Traverse the tree to find the most promising node to expand according to some chosen selection policy starting from an initial state node.

2. Expansion: Add one or more child nodes to the tree based on the possible actions from the selected node.

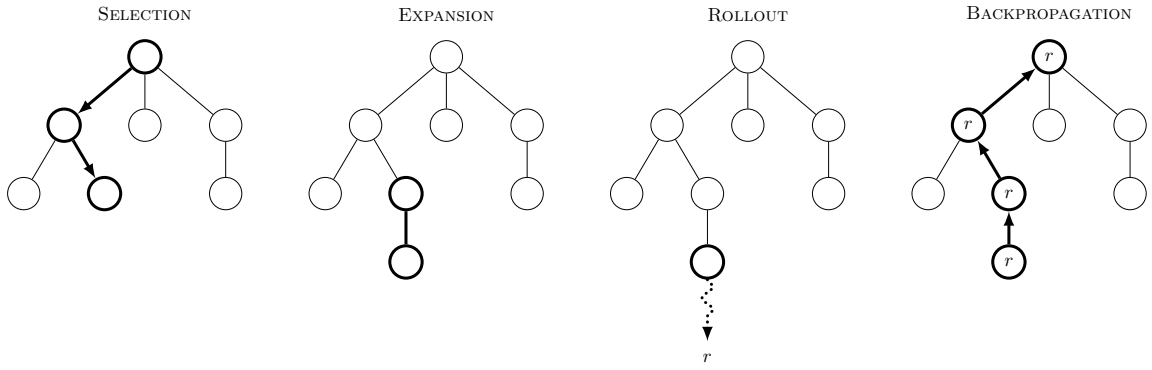


Figure 4.1: One iteration of Monte Carlo tree search.

3. Rollout: Run a simulation from the new node(s) according to some chosen rollout policy and evaluate the outcome.

4. Backpropagation: The result from the rollout is backpropagated through the sequence of selected nodes to update the statistics stored in the nodes.

Note that there are two distinct policies in play: selection policy and rollout policy. The selection policy defines how a leaf node is selected from the existing nodes in the tree for expansion. Since we want to explore more promising part of the decision space, the selection policy is typically designed such that it incorporates some measure of utility suitable for the application at hand. In case of the upper confidence bound for trees (UCT) [24], which is one of the most popular MCTS algorithms, the selection of child of node v is treated as an independent multi-armed bandit problem, and a child node v' that maximizes the following quantity is chosen

$$\mathrm{UCT}_{v'} = \frac{Q(v')}{N(v')} + c\sqrt{\frac{2\ln N(v)}{N(v')}}, \tag{4.10}$$

where $Q(v')/N(v')$ is an empirical mean of the rewards received with v' selected, c the exploration constant, $N(v)$ the total number of times the parent node has been selected, and $N(v')$ the total number of times v' has been selected. This formulation based on multi-armed bandit allows us to address the problem of exploration-exploitation with the exploration constant used to configure how the algorithm balances between the two [24]. In case of estimating risk measures, we can design a selection policy that biases the search towards sampling from the desired tail region in order to reduce the variance of the estimator. The rollout policy, on the other hand, defines how a simulation is run from the chosen non-terminal node to estimate its value. A simple rollout policy would be to choose an action uniformly at random from each state during a simulation.

Beyond the basic MCTS algorithm, an extensive set of variations have been studied, for instance, to apply the algorithm to problems with continuous decision space [14], or to parallelize simulations

to take advantage of multi-core hardwares [11].

4.3.2 Tree Search as Sampling

While the canonical use of MCTS is for finding optimal actions in a given state, it can also be adapted as a sampling method. Canevet et al. [8], for instance, proposes a tree-based algorithm inspired by MCTS that is used to sample training examples based on some importance weights to more efficiently train machine learning models. We now discuss an MCTS-based algorithm that implements the importance sampler considered in Section 4.2.

Given an instance of adversarial MDP $\mathcal{M}_{\text{adv}}$, random simulations are used to approximate the true cost associated with each state-dependent disturbance. The search tree built consists of nodes each representing a state that the system under test has reached following some number of transitions starting from an initial state. For a state node s, each branch represents a possible disturbance from that state, that is, $x \in \text{supp}(p(x \mid s))$. Based on simulation results, we compute an empirical estimate of $Q(s, x)$ in Eq. (4.7) and use it to approximate the state-dependent proposal $q(x \mid s)$. In case of discrete, finite disturbances, the proposal is approximated as

$$q(x \mid s) \approx \frac{p(x \mid s) \cdot \tilde{Q}(s, x)}{\sum_{x' \in \mathcal{X}} p(x' \mid s) \tilde{Q}(s, x')}. \tag{4.11}$$

This approximate proposal defines our selection policy, and for rollout we use a simple random rollout.

In short, in one iteration of simulation, we start from some initial state and traverse the tree according to the selection policy defined by the approximate proposals. Once we reach a terminal state, we compute the cost associated with the trajectory followed and backpropagate the result through the tree. Based on the result, we update the state-dependent proposals to increase the chance of sampling from the desired tail region of the cost distribution. As the approximate proposals depend on the running estimate $\tilde{\text{VaR}}_\alpha$ that could be noisy at the beginning, several of the extensions discussed in Section 3.3.3 such as having an initial burn-in period can be additionally employed to improve stability of the procedure. Algorithm 4.1 outlines a high-level summary of the approach.

4.3.3 Parallelization

Monte Carlo methods are often easily parallelizable so that complex problems that require a large number of samples can be made amenable to the sampling-based methods through parallel computing. There are several approaches to adding parallelism to MCTS which can accelerate the tree importance sampling algorithm discussed above. Depending on the part of the algorithm that is parallelized, there are three common methods: leaf parallelization, root parallelization, and tree parallelization [10, 11].

Algorithm 4.1 Tree importance sampling.

function TREESAMPLING(s_0, N) ▷ Initial state, no. of samples
  $\mathcal{S} \leftarrow \{\}$ ▷ Collection of samples
  for $i \leftarrow 1$ **to** N **do**
    $s', c', w' \leftarrow$ SELECT(s_0)
    $c, w \leftarrow$ ROLLOUT(s', c', w')
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{(c, w)\}$
    BACKPROP($s', c, \mathcal{S}$)
  return $\mathcal{S}$

function SELECT(s) ▷ Initial state
  $c, w \leftarrow 0, 0$
  while `!isterminal(`s`)` **do**
    $s', x', c' \leftarrow$ SAMPLEDISTRUBANCE($q(x \mid s), s$) ▷ Use approximate proposal
    $c, w \leftarrow c + c',$ `weight(`w, x'`)`
    $s \leftarrow s'$
  return s, c, w

function ROLLOUT(s, c, w) ▷ State, cost, weight
  while `!isterminal(`s`)` **do**
    $\mathcal{D}_{\text{rollout}} \leftarrow p(x \mid s)$ or $\mathcal{U}$ ▷ Use nominal or uniform random
    $s', x', c' \leftarrow$ SAMPLEDISTRUBANCE($\mathcal{D}_{\text{rollout}}, s$)
    $c, w \leftarrow c + c',$ `weight(`w, x'`)`
  return c, w

function SAMPLEDISTURBANCE($\mathcal{D}, s$) ▷ State-dependent distribution, state
  $x' \sim \mathcal{D}$ ▷ Draw a random disturbance
  $s', c' \sim T_{\text{adv}}(s' \mid s, x')$ ▷ Transition to next state
  return s', x', c'

function BACKPROP($s, c, \mathcal{S}$) ▷ State, cost, collection of samples
  $\tilde{\text{VaR}}_\alpha \leftarrow$ `var_alpha(`$\mathcal{S}$`)` ▷ Re-approximate VaR_α
  while `!isnull(s)` **do**
    $N(s) \leftarrow N(s) + 1$
    $Q(s) \leftarrow Q(s) + c$
    Re-approximate $q(x \mid s)$ with $\tilde{\text{VaR}}_\alpha$ using Eq. (4.11)
    $s \leftarrow$ parent of s
  return s

Parallel MCTS

In leaf parallelization, one thread is used in the selection phase to choose a leaf node to expand, and multiple independent simulations are run in parallel using the available threads starting from the chosen node. Once all simulations finish, a single thread is used to backpropagate the results through the tree. Despite the simplicity, having to wait for all simulations to finish before backpropagating

the results can be a significant bottleneck, and running simulations starting from a single leaf node can lead to exploring only a limited part of the search space.

Root parallelization is another method that is relatively simple to implement. In this method, multiple search trees are built independently by the available threads, and when the computational budget is exhausted, the trees are merged together in order to combine the statistics. This method addresses some of the shortcomings of leaf parallelization, but the threads still do not share their intermediate simulation results, and hence, it is possible for the threads to duplicate efforts.

Lastly, in tree parallelization, one search tree is simultaneously accessed by the threads, each independently selecting a node to expand, running a simulation, and backing up the results in the tree. To prevent potential data corruption due to multiple threads accessing the same part of the tree, a locking mechanism is typically implemented at the node level. Moreover, what is called virtual loss is often used to adjust the scores of those nodes that have been selected by other threads in order to encourage the threads to explore more broadly across the search space. This has been reported to often significantly improve scalability [11]. In some domains, updates applied by multiple threads are performed lock-free to maximize scalability, accepting that occasional race conditions are possible. This approach is conceivable if such race conditions are expected to be rare, or the potential data corruptions are not expected to significantly change the outcome of the search (e.g., the action considered to be optimal). For instance, this lock-free implementation was used in AlphaGo [37].

Parallel Sampling

For the tree importance sampler, we use a type of tree parallelization with a node-level locking mechanism to prevent potential data corruption of the node statistics. The lock-free approach may make more sense for the canonical use of MCTS of estimating optimal actions, especially when such collisions are expected to be infrequent. In case of the importance sampler, however, the node statistics are used to define the proposal distribution, and, hence, it is important to ensure data consistency for the sampling use case. Each thread runs an independent iteration of the sampling procedure with the following steps:

1. Using the samples collected thus far, compute the needed estimates such as VaR_α.
2. Traverse the tree according to the selection and rollout policies with the computed estimates until a terminal state is reached.
3. Compute the associated cost and store the sample with the corresponding weight.

This type of parallelization is especially useful for problems for which estimating costs at terminal states is computationally expensive but can be run in parallel.

4.4 Experiments

In this section, we discuss a set of experiments in which we apply the tree importance sampling algorithm to the problem of risk estimation in a sequential setting. We start with an overview of the setup, then present and analyze the results.

4.4.1 Setup

Gridworld is a classic RL problem in which the environment is a grid consisting of contiguous cells each with an associated reward or cost (that maybe zero) [39]. An agent starts from some initial cell and navigates through the environment collecting rewards until reaching one of the terminal states. The goal of the agent is to maximize the discounted sum of rewards collected, and a range of RL methods can be applied to this sequential task. Given an agent that has been trained, we want to estimate potential risks associated with deploying the agent to a similar environment but with state-dependent disturbances.

Gridworld MDP

We used a gridworld MDP of size 20×20 with the states and associated rewards as given in Table 4.1. For terminal states, we used a mix of positive and negative rewards, and, for non-terminal states, we used negation of random rewards drawn from a uniform distribution over $[0, 1)$. The probability of a successful transition according to the selected action (called transition probability) was set to 0.7. Lastly, a discount factor of 0.95 was used.

Table 4.1: Gridworld MDP states.

State	Reward	Terminal
$(4, 3)$	-10.0	Y
$(4, 6)$	-5.0	Y
$(9, 3)$	10.0	Y
$(8, 8)$	3.0	Y
$(16, 18)$	-30.0	Y
Others	$-\mathcal{U}_{[0,1)}$	N

To develop an agent to evaluate, we applied the Deep Q-network (DQN) algorithm where a neural network is used as a function approximator of the Q-function to the gridworld MDP (see Section 2.4). The input, which is the state represented as a pair of numbers, is mapped to a latent vector of size 32 that, following a ReLU activation, is then mapped to an output vector of size 4. The output vector represents the learned Q-values that correspond to the four possible actions (right, left, up, down). The agent was trained for 5M iterations using an ϵ-greedy policy as the exploration strategy with the value of ϵ linearly decayed from 1.0 to 0.1 over the first half of the training. Standard approaches such as using a replay buffer to store and sample transitions were

used. Once the algorithm converged, we constructed a policy based on the learned Q-function that maps each state to the action with the highest Q-value. Instead of the DQN algorithm, a simpler RL method such as value iteration could also have been used to train an agent for this relatively simple environment.

Adversarial MDP

For an adversarial MDP, we need to define the set of actions, which are the state-dependent disturbance distributions, and a reward function that is appropriate for the evaluation task. The set of states and the transition function are defined based on the original MDP as discussed in Section 4.2.1.

For state-dependent disturbances, we used the following distribution for every state

$$\forall s \in S_{\text{adv}},\, p(x \mid s) = \begin{cases} 0.90 & \text{for } x = (0,0) \\ 0.025 & \text{for } x \in \{(3,0),(0,3),(-3,0),(0,-3)\} \end{cases}$$

In words, with 90.0% chance, the agent observes the true state it is in, and with 2.5% chance it observes one of the four states that is of distance 3 away from the true state. For the reward function (or the cost), we took each state with a negative reward in the original MDP and used the absolute value of the reward for the adversarial MDP. For states with positive rewards, we used reward of 0 for the adversarial MDP. Intuitively, in this setup, we are concerned more about the agent's reaching cells with negative rewards in the original MDP but indifferent to its reaching cells with positive rewards.

4.4.2 Results and Analysis

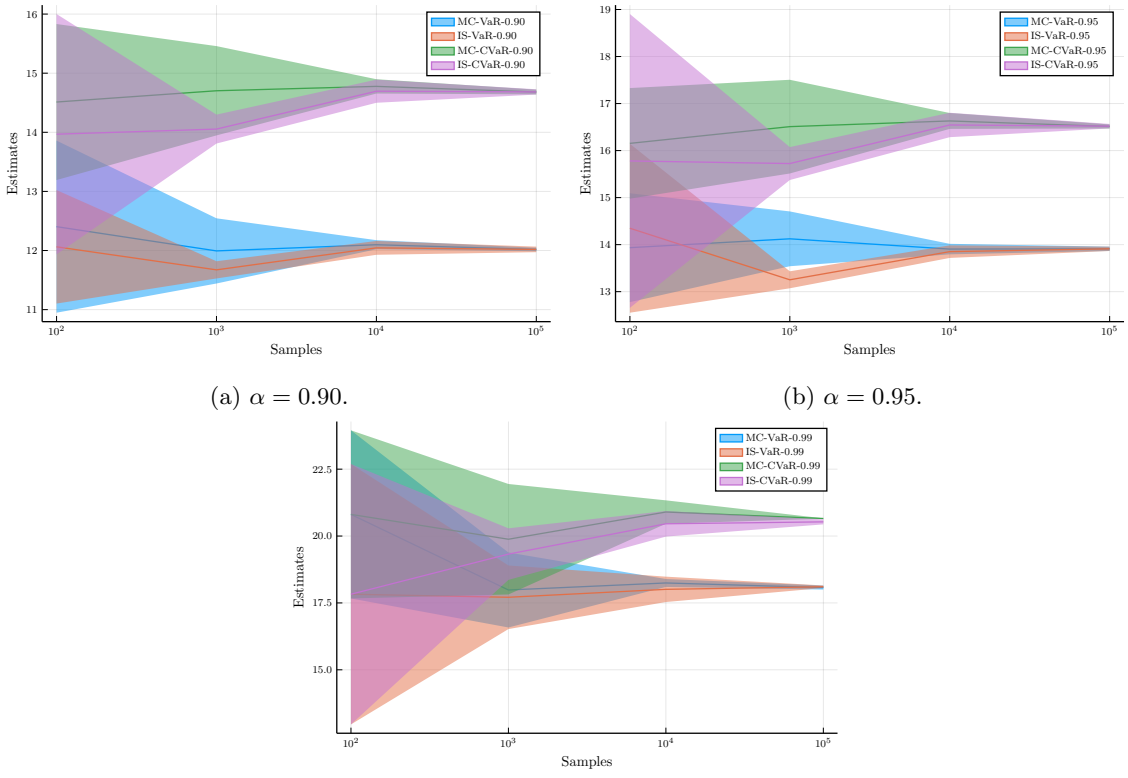
For both the Monte Carlo baseline and importance sampling experiments, we used the cell $(10, 10)$ as the initial state and report the estimates of VaR_α and CVaR_α for $\alpha \in \{0.90, 0.95, 0.99\}$ computed over 3 runs. Note that in this sequential setting, neither the distribution over trajectories nor the cost function are known explicitly. Hence, the true values are estimated based on samples collected from Monte Carlo simulations.

Table 4.2: Hyperparameters used for the gridworld importance sampler.

Hyperparameter	**Value**
maximum depth	100
exploration constant	3.0
rollout strategy	nominal
minimum slope	5.0
weight schedule	linear from 1.0 to 0.95 over 500k steps

For the importance sampler, we used adaptive mean and slope, where the location of the sigmoid

probability function is adapted based on the running estimate of VaR_α and the slope based on the empirical variance of the estimates. For numerical stability, a minimum slope was used to prevent the curve from becoming overly steep due to a potentially noisy empirical variance. See Section 3.4 for a similar set of experiments for quantile estimation. To estimate both VaR_α and CVaR_α simultaneously, we used the mixture distribution as defined in Eq. (3.19). The mixture weight λ was adjusted based a schedule in which it is initially set to 1.0 and linearly decayed to 0.95 over the first 500k iterations. Such a schedule was used in order to first have an accurate estimate of VaR_α before sampling more narrowly from the estimated tail region. Lastly, we used the nominal distribution as the rollout strategy, which had somewhat less variance than the uniform rollout in the beginning. When converged, however, both strategies resulted in similar estimates. Table 4.2 summarizes the set of hyperparameters used for the experiments.



(a) $\alpha = 0.90$.

(b) $\alpha = 0.95$.

(c) $\alpha = 0.99$.

Figure 4.2: Estimation of VaR_α and CVaR_α for gridworld.

Figure 4.2 shows convergence graphs of the Monte Carlo baseline and the importance sampler for estimating the risks for $\alpha \in \{0.90, 0.95, 0.99\}$. Overall, the importance sampler had a larger variance than the baseline in the beginning (i.e., 10^2 samples), when there is a higher degree of uncertainty

about the estimates of the expected costs, or the Q-values, associated with the disturbances. The Q-values become more accurate as more samples are collected, and the variance of the estimator quickly reduces even at 10^3 samples. For the experiments, we assumed no prior knowledge on the potential costs associated with the disturbances and initialized the Q-values to zeros. In case some domain knowledge is accessible, the Q-values can be better initialized in order to speed up convergence and so to more quickly reduce the variance of the estimator. This is indeed a general strategy commonly used to configure MCTS for faster convergence. Beyond 10^3 samples, the Monte Carlo baseline also converged with a similar level of variance as that of the importance sampler. Table 4.3 and Table 4.4 summarize the results of the experiments for VaR_α and CVaR_α, respectively.

Table 4.3: Estimates of VaR_α with $\alpha \in \{0.90, 0.95, 0.99\}$ for gridworld.

Exp.	$\boldsymbol{\alpha}$	$\boldsymbol{n = 10^2}$	$\boldsymbol{n = 10^3}$	$\boldsymbol{n = 10^4}$	$\boldsymbol{n = 10^5}$
MC	0.90	12.404 ± 1.456	11.993 ± 0.552	12.098 ± 0.077	12.017 ± 0.029
	0.95	13.935 ± 1.158	14.123 ± 0.583	13.906 ± 0.112	13.911 ± 0.046
	0.99	20.809 ± 3.142	17.984 ± 1.401	18.246 ± 0.148	18.074 ± 0.075
IS	0.90	12.063 ± 0.960	11.673 ± 0.146	12.045 ± 0.117	12.019 ± 0.047
	0.95	14.349 ± 1.796	13.252 ± 0.181	13.851 ± 0.134	13.911 ± 0.046
	0.99	17.826 ± 4.867	17.712 ± 1.189	18.008 ± 0.474	18.106 ± 0.048

Table 4.4: Estimates of CVaR_α with $\alpha \in \{0.90, 0.95, 0.99\}$ for gridworld.

Exp.	$\boldsymbol{\alpha}$	$\boldsymbol{n = 10^2}$	$\boldsymbol{n = 10^3}$	$\boldsymbol{n = 10^4}$	$\boldsymbol{n = 10^5}$
MC	0.90	14.512 ± 1.320	14.703 ± 0.755	14.777 ± 0.121	14.682 ± 0.040
	0.95	16.153 ± 1.176	16.509 ± 0.996	16.630 ± 0.169	16.513 ± 0.043
	0.99	20.809 ± 3.142	19.881 ± 2.065	20.898 ± 0.437	20.656 ± 0.007
IS	0.90	13.968 ± 2.033	14.054 ± 0.245	14.695 ± 0.196	14.683 ± 0.045
	0.95	15.780 ± 3.123	15.724 ± 0.350	16.545 ± 0.259	16.519 ± 0.048
	0.99	17.826 ± 4.867	19.325 ± 0.964	20.458 ± 0.478	20.530 ± 0.094

4.5 Discussion

In this chapter, we extended the importance sampling ideas from the previous chapter to the sequential setting in which the goal is to estimate potential risks of a sequential system in presence of state-dependent disturbances. This, for instance, can be applied to evaluating an autonomous vehicle system given random noise to the sensors.

We formulated this estimation problem as an instance of MDP where the set of actions correspond to the disturbance distributions and the rewards to the relevant costs. We then presented an MCTS-based importance sampling algorithm that iteratively estimates the expected costs associated with the disturbances, based on which samples are drawn to estimate the risk measures of interest. Despite

that the sequential setting considered involves the extra challenge of simultaneously learning the expected costs of actions and adapting the proposal distribution, we experimentally demonstrated that the algorithm can be employed as a variance reduction method for the estimation problem. Lastly, we briefly discussed possible approaches to adding parallelism in order to scale the algorithm to problems in which measuring the costs is computationally expensive.

Although in the experiments we considered a fixed initial state from which the algorithm was run, initial state itself can be naturally included as part of the input vector such that we additionally consider a distribution over initial states in estimating risks. For instance, if we have some domain knowledge regarding which initial states are more or less likely than others for the given sequential system under test, this distribution could be incorporated in the design of the adversarial MDP to estimate more informative risk measures.

In designing the tree importance sampling algorithm discussed in this chapter, we used the proposal distributions that we have shown could lead to lower variance estimates of VaR_α and CVaR_α. The distributions, in particular, were used to design the action selection strategy of the sampling algorithm. For other similar estimation problems, we could use alternative distributions to design action selection strategies more appropriate for the given problems. It is worth noting, however, that while having the algorithm favor a more relevant part of the search space is desirable, it is also important to ensure a sufficient level of exploration by not allowing the probability of reaching a different part of the space converge to zero. Otherwise, it could lead to biased estimates of the quantities of interest. This also justifies using some representation of uncertainty about the estimates in designing the proposal distributions, as we have done for the importance sampler considered in this work.

Chapter 5

Applications

This chapter discusses the search-based importance sampling algorithm previously introduced as applied in risk estimation and validation of autonomous vehicle (AV) policies in simulation. The first section gives a brief introduction to the type of application considered in this chapter. The second section discusses in detail the experiment setup, the open-source software tools used, and the results. The third section concludes the chapter with a discussion on possible extensions of the application to other policy types and environments.

5.1 Introduction

Given a system under test, we generally want to evaluate the system's performance on the types of environments to which the system is expected to be deployed. In case of an AV policy, such an environment (or scenario) typically consists of a set of components such as the road condition, the weather condition, and the type of other actors involved in the scene. Each of these components, in turn, can be defined by a set of parameters that take on different values with different probabilities. For instance, given a particular road condition, we might expect to see certain types of vehicles to be present more than others.

Given the sequential dependencies among such components, we formulate the selection of parameter values as a sequential decision-making task and apply the importance sampling algorithm to efficiently evaluate the given AV policy across the scenarios considered. The goal is to estimate the potential risks associated with an AV policy across a space of scenarios that we care about. Using the importance sampling algorithm, we explore the space of scenarios and simulate the AV policy in each of the scenarios selected. As the algorithm progresses, it gradually biases the search towards the part of the space deemed most relevant to estimating VaR and CVaR of the cost distribution.

5.2 Experiments

In this section, we discuss the details of the experiments run including the space of scenarios considered, the AV policy evaluated, and the software packages used.

5.2.1 Setup

For each experiment, we effectively repeat the following two steps until convergence or the maximum number of iterations is reached:

1. Selection of the scenario parameter values.
2. Simulation of the AV policy in the selected scenario to compute the associated cost.

Our goal is to estimate such metrics as VaR and CVaR of the cost distribution that is induced by the distribution over the scenario parameters as efficiently as possible. We demonstrate that the importance sampling algorithm can achieve better sample efficiency than simple Monte Carlo in this estimation task.

Note that the overall framework is modular in the sense that it does not depend on specific choice of the cost metric or the driving simulator. Instead, as long as the chosen simulator allows simulating the given AV policy in the selected scenario to compute the chosen cost metric, the proposed framework can be naturally adopted. For the experiments introduced in this chapter, we used an open-source software called CARLA [17] for simulating autonomous driving and impact force as the cost metric. See [30] for a presentation of a similar but more extensive risk assessment framework.

The main framework is implemented primarily in Julia language [5] with the client code that connects to the CARLA backend written in Python. The AutonomousRiskFramework.jl[1] package implements the client that communicates with CARLA to simulate the given AV policy in different scenarios. The ParallelTreeSampling.jl[2] package implements the core tree importance sampling algorithm used to search over the space of scenarios.

Scenarios

Choosing the space of scenarios to search over is a design choice to be made depending on the environments in which the AV policy is expected to be used. In our experiments, the search space consisted of the type of driving scene with an actor (e.g., car or pedestrian), the maximum speed of the ego vehicle, and the maximum brake it can apply. For driving scenes, we considered the following four scenes.

[1] `https://github.com/sisl/AutonomousRiskFramework.jl`
[2] `https://github.com/kykim0/ParallelTreeSampling.jl`

Figure 5.1: CARLA driving simulator.

- `FollowLeadingVehicle`: The ego vehicle follows a leading actor (a car or a motorcycle) that gradually slows down and comes to a stop.
- `DynamicObjectCrossing`: An actor (a bicycle or a pedestrian) suddenly moves into the way of the ego vehicle, which has to stop at that moment.
- `VehicleTurningRight`: The ego vehicle takes a right turn and an actor (a bicycle or a pedestrian) suddenly moves into its way, and it needs to stop accordingly.
- `OtherLeadingVehicle`: The ego vehicle follows a leading actor (a car or a motorcycle) that decelerates so that it has to change lane to avoid a collision.

More details on the scene types can be found in the official CARLA documentation. For the maximum speed of the ego vehicle, we considered the values $\{10, 20, 30, 40, 50\}$. The maximum brake is a scalar in the range $[0.0, 1.0]$ that controls the vehicle brake, and we considered the values $\{0.1, 0.2, 0.3, 0.4, 0.5\}$. In selecting the scenario parameter values, we first choose the scene type, then the maximum speed of the ego vehicle, and then finally the maximum brake. The nominal distribution over the possible values for each component is uniform. That is, simple Monte Carlo would sample a value uniformly at random for each of the scene type, the maximum speed, and the maximum brake.

Once we finish selecting all parameter values, we simulate the given AV policy in the selected scenario using CARLA. The simulation is run without additional disturbances, for instance, to the sensors of the ego vehicle. In case of a collision, the associated impact force is returned, which is used as our cost metric.

Autonomous Vehicle Policy

A driving agent in CARLA manages the ego vehicle by controlling such parameters as the vehicle throttle, brake, and gear based on it policy. The AV policy used in our experiments in built on top

of a basic agent implementation provided by the CARLA API and has the additional logic to detect imminent collisions based on the GNSS data and to apply brake accordingly.

Situated in a scenario, the agent based on the AV policy devises a high level plan of route consisting of a series of waypoints to follow to reach the given destination. This global planning is done using the A* search algorithm with a distance heuristic [36], and a parameter that controls the sampling resolution determines the granularity of the waypoints. In following the trajectory of waypoints, two PID controllers (for the lateral and longitudinal controls) are used to control the low level motion of the ego vehicle, with additional local waypoints dynamically generated as needed. The implementation of this agent that we call the GNSS agent is available in the AutonomousRiskFramework.jl package.

5.2.2 Results and Analysis

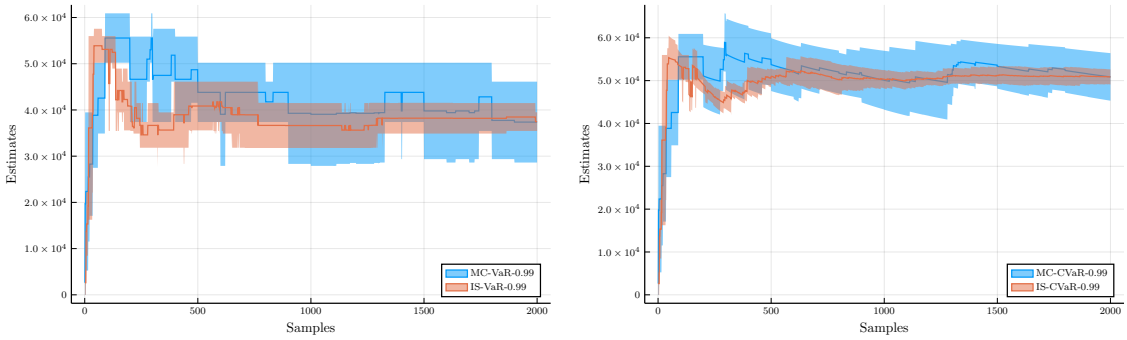
For both the baseline and the importance sampler, we estimated VaR_α and CVaR_α for $\alpha = 0.99$ over 3 runs each using a different random seed in order to evaluate convergence as well as variance.

Table 5.1: Hyperparameters used for the scenario importance sampler.

Hyperparameter	**Value**
exploration constant	10,000.0
rollout strategy	nominal
minimum slope	100.0
weight schedule	linear from 1.0 to 0.90 over 1k steps

We used a similar importance sampler as that used in the previous chapter. In particular, we used adaptive mean and slope, with the location of the sigmoid probability function adapted based on the running estimate of VaR_α and the slope based on the empirical variance. Similarly as before, a minimum slope was used to achieve better numerical stability. To estimate both quantities more efficiently, the mixture distribution as defined in Eq. (3.19) was used, with the weight linearly decayed to gradually favor sampling from the estimated tail. The set of hyperparameters used is summarized in Table 5.1

Figure 5.2 shows convergence of the simple Monte Carlo baseline and the importance sampler for estimating VaR_α and CVaR_α of the cost associated with the GNSS agent. Overall, the importance sampler achieved smaller variance than the baseline, particularly in CVaR_α estimation. In the early part of the procedure when the importance sampler had more incentives to explore the parameter space, its VaR_α estimate was somewhat higher than what it eventually converged to. It was able to, however, adjust its estimate fairly quickly with additional samples to achieve a lower variance estimate, particularly of CVaR_α, than the baseline. Table 5.2 summarizes the VaR_α and CVaR_α estimates computed by the two approaches.



(a) VaR_α with $\alpha = 0.99$. (b) CVaR_α with $\alpha = 0.99$.

Figure 5.2: Estimation of VaR_α and CVaR_α with $\alpha = 0.99$ for the GNSS agent.

Table 5.2: Estimates of VaR_α and CVaR_α with $\alpha = 0.99$ for the GNSS agent.

Exp.	$n = 100$	$n = 300$	$n = 500$	$n = 1,000$	$n = 2,000$
MC-VaR_α	$55,564 \pm 5,343$	$46,631 \pm 9,241$	$43,805 \pm 6,415$	$39,060 \pm 11,161$	$37,371 \pm 8,750$
MC-CVaR_α	$55,564 \pm 5,343$	$55,897 \pm 8,035$	$53,742 \pm 6,175$	$50,087 \pm 6,878$	$50,865 \pm 5,552$
IS-VaR_α	$53,103 \pm 2,882$	$36,656 \pm 4,832$	$40,851 \pm 5,267$	$36,631 \pm 4,861$	$37,477 \pm 2,019$
IS-CVaR_α	$53,320 \pm 3,099$	$46,036 \pm 2,706$	$50,957 \pm 1,802$	$50,324 \pm 2,148$	$50,855 \pm 1,756$

5.3 Discussion

In this chapter, we demonstrated an application of the tree importance sampling algorithm introduced in Chapter 4 to risk estimation of an AV policy treated as black-box in simulation. In case the goal is to evaluate an AV policy across scenarios defined by a set of parameters, values of which are natural to be chosen in sequence, we can formulate the parameter selection as an instance of MDP and apply the tree importance sampling algorithm to estimate VaR and CVaR of the cost distribution induced by that over the scenarios. In particular, we simulated the GNSS agent across a space of scenarios defined by the driving scene, the maximum speed of the ego vehicle, and the maximum brake in order to collect cost samples and to compute the associated risk metrics in a sample efficient manner.

Note that the nature of the application explored in this chapter is somewhat distinct from that of the experiments considered in Chapter 4. Previously, we applied the importance sampling algorithm to an MDP where the action is a set of possible disturbances to the state observed by the system under test. Using the algorithm on such an MDP, we would be estimating the risk metrics of the cost distribution induced by the stochastic disturbances applied to the system. In this chapter, on the other hand, we applied the algorithm to an MDP where the action is a set of possible values for the relevant scenario component. In this case, we would be estimating the risk metrics of the cost distribution induced by that over the scenario components considered. Given that the

algorithm is based on MCTS, which is a generic RL solver, it can be naturally adopted in similar risk estimation applications if the cost distribution of interest is one that is induced by stochastic sequential decisions.

For the experiments presented in this chapter, we considered a relatively simple AV policy that utilizes GNSS input to detect potential collisions. We can easily extend the application to more complex AV policies such as the ones that use image input. Also, we can consider additional scenario components such as weather condition that can plausibly be relevant to such image-based agents [30].

Chapter 6

Conclusions

The nature of intelligent systems developed for real-world applications has evolved from one that involves hand-designed heuristics based on domain knowledge to one that is learned from data. These learning-based systems often demonstrated significant improvement in performance over the heuristic-based counterparts but are typically highly complex and treated as a black-box in practice. In this thesis, we proposed several adaptive algorithms for efficiently estimating risks of such black-box systems. This chapter summarizes the approach explored in this work with a brief discussion on potential directions for further research.

6.1 Summary

Given a black-box system to evaluate, we need to carefully design the distribution from which to sample the input in order to develop an unbiased, sample efficient estimator of the risk measures of interest. Inspired by the idea of adapting the sampling distribution from importance sampling, we proposed a novel framework called curriculum sampling, which is designed to estimate a set of dependent measures in a stable and sample efficient manner. We adopted the framework to design an importance sampler to estimate the tail risk measures called VaR and CVaR, demonstrating the approach on both sequential and non-sequential problems.

The first part of the thesis introduces and motivates the main research problem of focus (Chapter 1) and discusses relevant preliminaries on importance sampling, the tail risk measures to estimate, and the reinforcement learning framework (Chapter 2). The remainder of the thesis makes the following contributions:

- **A novel framework for efficient estimation of a set of dependent statistics**. In Chapter 3, we introduced a more advanced importance sampling technique called adaptive importance sampling, where the sampling distribution is iteratively adapted based on previous

samples to improve sample efficiency. Adopting this idea in estimating a set of dependent measures, we proposed a novel framework called curriculum sampling that aims to achieve sample efficiency while maintaining stability of the sampling procedure by decomposing a given problem into a series of subtasks solved in order. Given some representation of uncertainty on the running estimates, we decide how the distribution is adapted and whether the current subtask can be considered solved. Using this framework, we implemented an importance sampler for estimating VaR and CVaR and demonstrated it on a quantile estimation task.

- **An adaptive algorithm based on MCTS for evaluating sequential systems.** In Chapter 4, we extended the adaptive importance sampler for estimating VaR and CVaR introduced in the context of non-sequential settings to evaluating sequential decision-making systems. In particular, we introduced a formulation of the problem of risk estimation of sequential systems in presence of state-dependent disturbances as an instance of MDP. We then proposed an adaptive algorithm based on MCTS called tree importance sampling for estimating VaR and CVaR of the cost distribution induced by that over the state-dependent disturbances. We demonstrated the algorithm on a gridworld environment to evaluate a DQN agent as a system under test.

- **An application in evaluating autonomous vehicle policies in simulation.** In Chapter 5, we studied the problem of evaluating autonomous vehicle policies across a space of scenarios in simulation. In case the scenario space of interest is defined by a set of parameters, possible values of which are discrete and chosen in sequence, we introduced a formulation of scenario selection as an instance of MDP. We then applied the tree importance sampling algorithm to the MDP to estimate VaR and CVaR of the cost distribution associated with the given AV policy across the space of scenarios.

6.2 Further Work

Curriculum sampling introduced in this thesis is a general adaptive algorithm framework that can be adopted for estimating an arbitrary set of dependent statistics with different uncertainty representations. In this work, we mainly focused on two particular tail statistics, VaR and CVaR, with the estimate uncertainty represented using sigmoid-based functions. Applying the framework to design adaptive sampling algorithms for estimating other risk measures and also with alternative uncertainty quantification methods would be one interesting line of further research.

For sequential decision-making settings, we proposed an MCTS-based importance sampling algorithm that can be used to evaluate sequential systems. In the formulation of the adversarial MDP, we implicitly assumed discrete action space and demonstrated the importance sampling algorithm on an MDP with discrete state-dependent disturbances. We can extend the algorithm to also support

continuous disturbances, for instance, by adopting the progressive widening method that has been used to allow MCTS to handle continuous action space [14].

Bibliography

[1] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. A brief survey of deep reinforcement learning. *CoRR*, abs/1708.05866, 2017. URL `http://arxiv.org/abs/1708.05866`.

[2] Andrew Barto, Richard Sutton, and Chris Watkins. Learning and sequential decision making. Technical report, University of Massachusetts, Amherst, 09 1989.

[3] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 41–48, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605585161. doi: 10.1145/1553374.1553380. URL `https://doi.org/10.1145/1553374.1553380`.

[4] Albert Benveniste, Michel Métivier, and Pierre Priouret. *Adaptive algorithms and stochastic approximations*, volume 22. Springer Science & Business Media, 2012.

[5] Jeff Bezanson, Stefan Karpinski, Viral B Shah, and Alan Edelman. Julia: A fast dynamic language for technical computing. *arXiv preprint arXiv:1209.5145*, 2012.

[6] Cameron B. Browne, Edward Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, 2012. doi: 10.1109/TCIAIG.2012.2186810.

[7] Monica F. Bugallo, Victor Elvira, Luca Martino, David Luengo, Joaquin Miguez, and Petar M. Djuric. Adaptive importance sampling: The past, the present, and the future. *IEEE Signal Processing Magazine*, 34(4):60–79, 2017. doi: 10.1109/MSP.2017.2699226.

[8] Olivier Canevet, Cijo Jose, and Francois Fleuret. Importance sampling tree for large-scale empirical expectation. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine*

Learning Research, pages 1454–1462, New York, New York, USA, 20–22 Jun 2016. PMLR. URL https://proceedings.mlr.press/v48/canevet16.html.

[9] O Cappé, A Guillin, J. M Marin, and C. P Robert. Population monte carlo. *Journal of Computational and Graphical Statistics*, 13(4):907–929, 2004. doi: 10.1198/106186004X12803. URL https://doi.org/10.1198/106186004X12803.

[10] Tristan Cazenave and Nicolas Jouandeau. On the parallelization of uct. In *Computer Games Workshop*, Amsterdam, Netherlands, Jun 2007. URL https://hal.archives-ouvertes.fr/hal-02310186.

[11] Guillaume M. J. B. Chaslot, Mark H. M. Winands, and H. Jaap van den Herik. Parallel monte-carlo tree search. In H. Jaap van den Herik, Xinhe Xu, Zongmin Ma, and Mark H. M. Winands, editors, *Computers and Games*, pages 60–71, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[12] JEAN-MARIE CORNUET, JEAN-MICHEL MARIN, Antonietta Mira, and Christian P Robert. Adaptive multiple importance sampling. *Scandinavian Journal of Statistics*, 39(4): 798–812, 2012.

[13] Anthony Corso, Robert J. Moss, Mark Koren, Ritchie Lee, and Mykel J. Kochenderfer. A survey of algorithms for black-box safety validation. *jair*, 72:377–428, 2021.

[14] Adrien Couëtoux, Jean-Baptiste Hoock, Nataliya Sokolovska, Olivier Teytaud, and Nicolas Bonnard. Continuous upper confidence trees. In Carlos A. Coello Coello, editor, *Learning and Intelligent Optimization*, pages 433–445, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[15] Rémi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *International conference on computers and games*, pages 72–83. Springer, 2006.

[16] Mary Cummings. Rethinking the maturity of artificial intelligence in safety-critical settings. *AI Magazine*, 42(1):6–15, Apr. 2021. URL https://ojs.aaai.org/index.php/aimagazine/article/view/7394.

[17] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[18] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM computing surveys (CSUR)*, 51(5):1–42, 2018.

[19] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning: with Applications in R*. Springer, 2013.

[20] Volodymyr Mnih Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013. URL `http://arxiv.org/abs/1312.5602`.

[21] Alex Kendall, Jeffrey Hawke, David Janz, Przemyslaw Mazur, Daniele Reda, John-Mark Allen, Vinh-Dieu Lam, Alex Bewley, and Amar Shah. Learning to drive in a day. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8248–8254, 2019. doi: 10.1109/ICRA.2019.8793742.

[22] Eugenia Koblents and Joaquín Míguez. A population monte carlo scheme with transformed weights and its application to stochastic kinetic models. *Statistics and Computing*, 25(2):407–425, Mar 2015. ISSN 1573-1375. doi: 10.1007/s11222-013-9440-2. URL `https://doi.org/10.1007/s11222-013-9440-2`.

[23] Mykel J Kochenderfer. *Decision making under uncertainty: theory and application*. MIT press, 2015.

[24] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou, editors, *Machine Learning: ECML 2006*, pages 282–293, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[25] Dirk P Kroese, Thomas Taimre, and Zdravko I Botev. *Handbook of monte carlo methods*. John Wiley & Sons, 2013.

[26] Pavlo Krokhmal, Jonas Palmquist, and Stanislav Uryasev. Portfolio optimization with conditional value-at-risk objective and constraints. *Journal of risk*, 4:43–68, 2002.

[27] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015. doi: 10.1038/nature14539. URL `https://doi.org/10.1038/nature14539`.

[28] D. J. C. Mackay. *Introduction to Monte Carlo Methods*, pages 175–204. Springer Netherlands, 1998. ISBN 978-94-011-5014-9. doi: 10.1007/978-94-011-5014-9_7. URL `https://doi.org/10.1007/978-94-011-5014-9_7`.

[29] Jérôme Morio. Extreme quantile estimation with nonparametric adaptive importance sampling. *Simulation Modelling Practice and Theory*, 27:76–89, 2012. ISSN 1569-190X. doi: https://doi.org/10.1016/j.simpat.2012.05.008. URL `https://www.sciencedirect.com/science/article/pii/S1569190X1200072X`.

[30] Robert J. Moss, Shubh Gupta, Marc R. Schlichting, Kyu-Young Kim, Anthony Corso, Grace X. Gao, and Mykel J. Kochenderfer. A modular framework for efficient autonomous vehicle risk assessment and validation. *IEEE Transactions on Intelligent Transportation Systems*, to be submitted.

[31] Art Owen and Yi Zhou. Safe and effective importance sampling. *Journal of the American Statistical Association*, 95(449):135–143, 2000. doi: 10.1080/01621459.2000.10473909. URL `https://www.tandfonline.com/doi/abs/10.1080/01621459.2000.10473909`.

[32] Art B. Owen. *Monte Carlo theory, methods and examples.* 2013.

[33] Adrian E Raftery and Steven M Lewis. Implementing mcmc. *Markov chain Monte Carlo in practice*, pages 115–130, 1996.

[34] R. Tyrrell Rockafellar and Stanislav Uryasev. Optimization of conditional value-at-risk. *Journal of Risk*, 2:21–41, 2000.

[35] R Tyrrell Rockafellar and Stanislav Uryasev. Conditional value-at-risk for general loss distributions. *Journal of banking & finance*, 26(7):1443–1471, 2002.

[36] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall Press, 3rd edition, 2009. ISBN 0136042597.

[37] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, Jan 2016. doi: 10.1038/nature16961.

[38] Lihua Sun and L. Jeff Hong. A general framework of importance sampling for value-at-risk and conditional value-at-risk. In *Proceedings of the 2009 Winter Simulation Conference (WSC)*, pages 415–422, 2009. doi: 10.1109/WSC.2009.5429348.

[39] Richard S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In Bruce Porter and Raymond Mooney, editors, *Machine Learning Proceedings 1990*, pages 216–224. Morgan Kaufmann, San Francisco (CA), 1990. ISBN 978-1-55860-141-3. doi: https://doi.org/10.1016/B978-1-55860-141-3.50030-4. URL `https://www.sciencedirect.com/science/article/pii/B9781558601413500304`.

[40] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction.* A Bradford Book, Cambridge, MA, USA, 2018. ISBN 0262039249.

[41] Matthew D. Zeiler. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701, 2012. URL `http://arxiv.org/abs/1212.5701`.