EE 290-005 Integrated Perception, Learning, and Control **Spring 2021**

Lecture 9: Safety and Performance Guarantee

Scribes: Kshitij Kulkarni, Yibin Li, Harry Zhang, Jason Zhou

Presented by Michael Lim, Yiling You, Yaodong Yu

9.1 Neural Bridge Sampling for Evaluating Safety-Critical Autonomous Systems

9.1.1 Motivation: Operating Safety-Critical Control Systems

In generic closed-loop settings, it is difficult to *certify* or make guarantees regarding the performance and quality of control systems; for data-driven and learning-based control approaches, this evaluation becomes even more challenging. This barrier greatly limits the ability for these methods to be deployed in safety-critical settings such as transportation and robot-assisted surgery, *despite the development of autonomous agents that may outperform proficient humans at those same tasks.* Due to the necessity of such safety guarantees, the set of available tools are restricted to those limited in depth and expressivity, such as simple verifiable protocols, PID control, or convex optimization - non-ideal solutions for unstructured, real-world tasks, but solutions that nevertheless can provide provable properties regarding stability, consensus, recursive feasibility, and such.

Thus, to move beyond the barriers that come with these stylized methods, better evaluation systems are required; specifically, more *efficient* evaluation schemes are desired as current methods are often prohibitively expensive (with regards to the required number of queries to the system simulator in the process of evaluation) [5].

9.1.2 Governing Goal: Estimation of Adverse Event Probability

The task of evaluating safety-critical systems can be encapsulated as estimating the probability of a catastrophic event occurring, given that a particular agent is operating in the environment. Thus, the given parameters of the estimation problem are as follows:

- Simulator of the particular environment/system is assumed to be available.
- Distribution $X \sim P_0$: simulation parameters that describe the typical state of the environment; note that P_0 is assumed to be known through standard system-identification and generative modeling techniques outlined in relevant literature
- $f : \mathcal{X} \to \mathbb{R}$: safety score function of a realization x of the particular agent and environment, where by convention higher outputs denote safer characteristics
- $\gamma \in \mathbb{R}$: safety score threshold defining an adverse event (occurring at safety scores below γ)

The goal can now be denoted succinctly as estimating the following quantity:

$$p_\gamma := \mathbb{P}_0(f(X) < \gamma) \tag{9.1}$$

Point of Discussion: What if some of the above availability assumptions do not hold?

9.1.3 Related Works

9.1.3.1 Naive Monte Carlo Approaches

Note that if an agent is highly proficient at its task (i.e. p_γ is a very small quantity), then intuitively, we are less confident in our estimate of p_γ by nature of observing these adverse events (i.e. $f(X) < \gamma$) much less frequently. Concretely, we can examine these effects through naive Monte Carlo estimation ($\hat{p}_\gamma$) by querying the safety function f and the simulator; we are in essence exploring the search space through direct random samples from P_0 to produce $\hat{p}_\gamma$:

$$\hat{p}_\gamma = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}\{f(x_i) < \gamma\} \tag{9.2}$$

While this method is unbiased, and straightforward to implement and parallelize, it exhibits poor sample complexity. The error for the Monte Carlo estimation is:

$$\mathbb{E}[(\frac{\hat{p}_\gamma}{p_\gamma} - 1)^2] = \frac{1 - p_\gamma}{N p_\gamma} \tag{9.3}$$

Observe that the error is inversely proportional to p_γ; the reduction of this error (to a logarithmic relationship with regards to true p_γ) is another area upon which the paper will improve current evaluation methods.

By adding an adaptive *exploitative* component to the above naive scheme (which is purely *exploratory* as it stands), the resulting approximations can be improved. This is achieved by examining the most informative random samples from a sequence of such distributions P_k that approximate P_0. However, no first-order information about the safety function f itself is leveraged in these standard adaptive Monte Carlo methods.

Additionally, the proposed method avoids the computation of the Hessian $\nabla^2 f(x)$, which is generally exorbitantly expensive to calculate and commonly employed in naive first-order optimization methods for estimation, by relying only on using the gradient $\nabla f(x)$.

9.1.3.2 Safety Evaluation: Verification & Falsification

The verification community has developed many tools in similar attempts to evaluate closed-loop systems, their performance, and safety. In general, verification methods *certify whether or not failure is possible* by investigating dangerous executions of the system. However, these tools require both that the model is written in a formal language, and that whitebox access to this formal model is available.

In falsification, attempts are made at *finding any failure modes of the system*; in essence then, the goal in falsification is to minimize $f(x)$ itself. This is in contrast to estimating p_γ, the probability of catastrophic events occurring. Falsification is closely related to adversarial machine learning, although in adversarial machine learning the domain from which the adversarial examples come from are restricted (generally to a $p-$norm ball around a point of the dataset), while falsification enforces no such constraints on the domain.

Note that in real-world settings it is generally trivial to demonstrate failure cases (e.g. performed by verification and falsification), and it may be significantly more beneficial to consider the probability and overall frequency of failures as well as the relative frequency amongst different failure modes instead. After all, verification and falsification methods provide only a coarse binary label on whether or not the system is safe or unsafe.

9.1.3.3 Sampling Techniques and Density Estimation

For sampling rare (potentially adverse) events, we can review two dominant branches of work:

1. Parametric Adaptive Importance Sampling (AIS). For example, the cross-entropy method falls under this category. These methods postulate a family of distributions for the optimal importance-sampling distribution, and then iteratively update the sampling distribution through heuristic optimization procedures.
2. Nonparametric Sequential Monte Carlo (SMC). For example, particle filters, smoothing filters, etc. fall under this category. These techniques sample from a sequence of probability distributions defined directly by the samples (i.e. nonparametrically).

The proposed neural bridge method employs bridge sampling, which uses elements of both AIS and SMC; namely, parametric warping distributions are used within the SMC setting. Other closely related methods include umbrella sampling (which is more brittle than the bridge sampling distribution), multilevel splitting (which uses hard barriers/indicator functions, while neural bridge uses smooth exponential barriers), and path sampling.

9.1.4 Markov Chain Monte Carlo Methods

We now build towards the proposed approach, beginning with the naive Monte Carlo sampling formulation. As mentioned above, Monte Carlo sampling is commonly employed to estimate the expected value of a function with a sum by generating samples from a given distribution. Namely, we can approximate $\int f(x)p(x)dx$ for which $x^{(i)} \sim p(x)$, in the following manner empirically:

$$\int f(x)p(x)dx \approx \frac{1}{N}\sum_{n=1}^{N} f(x^{(i)}) \tag{9.4}$$

The limitations of naive Monte Carlo sampling were covered previously. Additionally, in situations where independent samples from the distribution upon which we are performing inference cannot be drawn or drawn easily, Monte Carlo sampling may not be used. In this case, we turn to Markov Chain Monte Carlo (MCMC), which are a broad class of methods that, at a high level, employ Markov Chains to adaptively explore near the target distribution in order to converge onto the equilibrium (or posterior) distribution using simulated samples. One such standard MCMC method is Metropolis Hastings.

9.1.5 Metropolis Hastings

The Metropolis Hastings algorithm attempts to solve the sampling problem by generating a Markov Chain through rejection. The algorithm proceeds as follows, where we want to sample from a probability distribution $P(x)$; we generate a Markov chain whose stationary distribution $\pi(x) = P(x)$.

1. Generate a random state x' from a proposed distribution $g(x'|x_t)$
2. Accept the sample with probability:
$$A(x', x) = \min\left(1, \frac{P(x')g(x|x')}{P(x)g(x'|x)}\right) \tag{9.5}$$
3. Choose a random number uniformly from $[0, 1]$, and accept if $u \leq A(x', x)$ and reject otherwise.

A note about Metropolis Hastings: the algorithm is a subclass of Markov Chain Monte Carlo methods.

9.1.6 Metropolis-adjusted Langevin Algorithm

We first introduce Langevin dynamics, both the first order and second order versions, which are idealized versions of molecular dynamics via stochastic differential equations. First, we propose the first-order version:

$$\dot{X} = -\nabla U(X) + K\dot{W} \tag{9.6}$$

where W is a Brownian motion. There is also a (second-order) damped version that was introduced in the lecture, but we note that standard MALA algorithms do not make use of this Langevin diffusion:

$$\ddot{X} = -\nabla U(X) - \gamma M\dot{X} + \sqrt{2M\gamma k_B T}R(t) \tag{9.7}$$

where once again, R(t) is a Brownian motion.

Now, we can propose the MALA algorithm, which uses a discretized version of the first order Langevin diffusion to propose a state::

$$\tilde{X}_{k+1} = X_k + \tau\nabla \log \pi(X_k) + \sqrt{2\tau}\xi_k \tag{9.8}$$

such that ξ_k an i.i.d. draw from a multivariate Gaussian with zero mean and covariance $I_{d\times d}$ and $\pi(\cdot)$ is the distribution we are interested in drawing from.

We then accept or reject this state according to the Metropolis-Hastings algorithm:

$$\alpha = \min\{1, \frac{\pi(\tilde{X}_{k+1}q(X_k|\tilde{X}_{k+1}}{\pi(X_k)q(\tilde{X}_{k+1}|X_k)}\} \tag{9.9}$$

where $q(x'|x)$ is distribution proportional to an exponential distribution in $x' - x$. The core idea here is the stationary distribution exists through the detailed balance condition, and that compared to standard Metropolis-Hastings, the algorithm reaches regions of high π probability faster.

9.1.7 Hamiltonian Monte Carlo

Hamiltonian Monte Carlo is used to compute integrals with respect to a target distribution $P(x)$. These integrals take the following form:

$$\int f(x)P(x)dx \tag{9.10}$$

which are approximated as:

$$\int f(x)P(x,v)dxdv \tag{9.11}$$

$$\approx \frac{1}{N}\sum_{n=1}^{N} f(x^i), x, v \sim P(x,v) \tag{9.12}$$

Further, we see that we can write the Hamiltonian of a system with states (p, q) as:

$$H(p,q) = K(p) + U(q) \tag{9.13}$$

where $K(p) = \frac{p^T M^{-1} p}{2}$ is a kinetic energy term, $U(q)$ is a potential term, and the dynamics are:

$$\dot{q} = M^{-1} p \tag{9.14}$$

$$\dot{p} = -\frac{\partial U}{\partial q} \tag{9.15}$$

The idea in HMC is to simulate a forward trajectory of the Hamiltonian dynamics in (p, q)-space, and use this as a proposal for the forward trajectory. The following is the forward simulation of the dynamics:

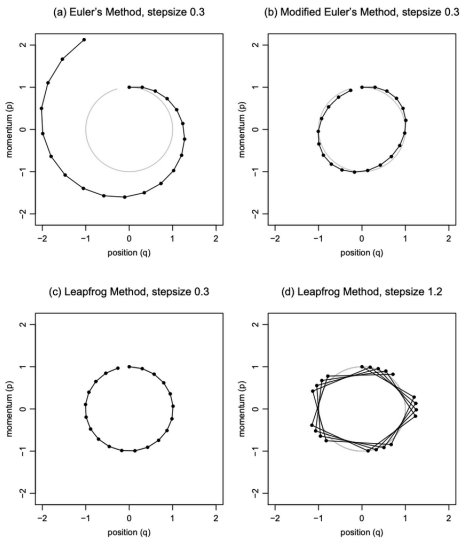


Figure 9.1: Forward simulation of the dynamics

For an example implementation of HMC, see: https://github.com/chi-feng/mcmc-demo

Further, here is an example of the comparison of HMC and MALA:

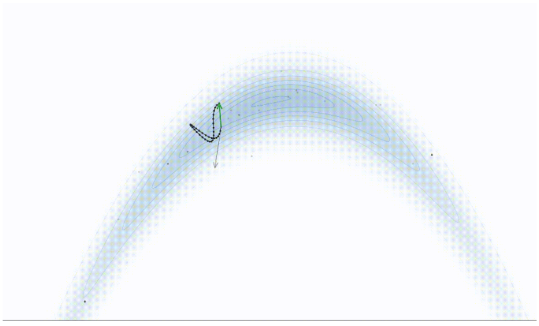


Figure 9.2: Comparisons of HMC and MALA

9.1.8 Overview of MCMC Methods

Till now, we have surveyed the following (gradient-based) Markov Chain Monte Carlo Methods:

1. Metropolis Adjusted Langevin Algorithm (MALA)
2. Hamiltonian Monte Carlo

There are also so-called temperature-based MCMC methods, such as:

1. Parallel Tempering
2. Wang-Landau

However, we do not go deep into these here.

MCMC methods also have relations to reinforcement learning and control. For example, there are state-space methods in papers such as:

1. "Neural bridge sampling", Sinha et. al.
2. "A Markov chain Monte Carlo algorithm for Bayesian policy search", Aghaei et. al.
3. "Trans-dimensional MCMC for Bayesian Policy Learning", Hoffman et. al.

Finally, there may be a link to energy-space MCMC through so-called deep energy models.

9.1.9 Autoencoding

An auto encoder is a generative model that feeds the input data into an encoder network, encoding the data into a latent space and then feeds the encoded latent representation of the data into the decoder in order to reconstruct the data.

In comparison, a variation also auto encoder has the smart idea of reconstructing the input data, but instead of trying to directly learn the data distribution, a VAE approximates the distribution with a more tractable one and samples from that distribution using the reparameterization trick to reconstruct the input data.

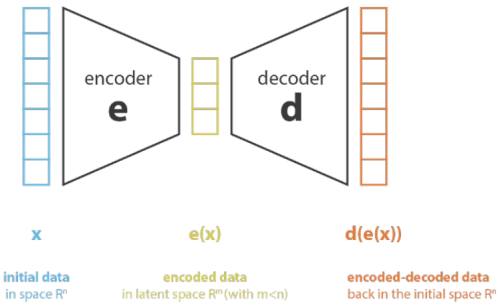


Figure 9.3: Structure of Autoencoder

In the autoencoder formulation, we have $x \in \mathbb{R}^n$ and an encoder $e(x) : \mathbb{R}^n \to \mathbb{R}^n \to \mathbb{R}^m$ where $m < n$, as well as a decoder $d : \mathbb{R}^m \to \mathbb{R}^n$ such that $d(e(x)) = x$. In the case when $d(e(x)) = x$ for all x, we have lossless encoding and when $d(e(x)) \neq x$ we have lossy encoding. That is, information is loss when reducing the dimension and thus the data x cannot be recovered.

9.1.10 Variational Autoencoders

Variational autoencoders use the encoding decoding process combined with a sampler to add randomization to the autoencoding process. That is, we have $x \to e(x) + \epsilon \to d(e(x) + \epsilon)$ where ϵ is a sampled vector from the latent space. We further minimize the decoding error to recompute x.

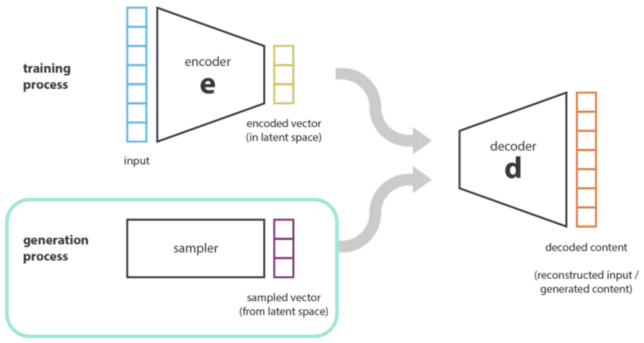


Figure 9.4: Structure of Varaitional autoencoders

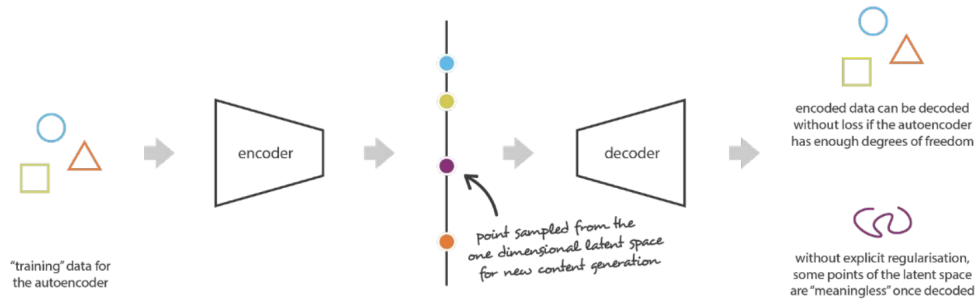


Figure 9.5: Visual Depiction of the Autoencoding Process

The core idea is that the training data can be embedded and then points can be sampled from the latent space and decoded. The full autoencoding process as compared to simple autoencoders can be found below.

9.1.11 Normalizing Flows

Normalizing flows are a class of algorithms that learn transforms that normalize. The idea is to learn an embedding into a Gaussian-like space and use an invertible smooth mapping (a bijection) such that:

$$p(y) = q(z) \left| det \frac{\partial f}{\partial z} \right|^{-1} \tag{9.16}$$

We have an example here of a normalizing flow: In the context of MCMC, we know that HMC is inefficient.

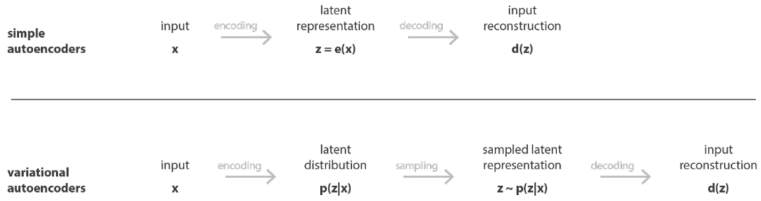


Figure 9.6: The variational autoencoding process compared to simple autoencoders

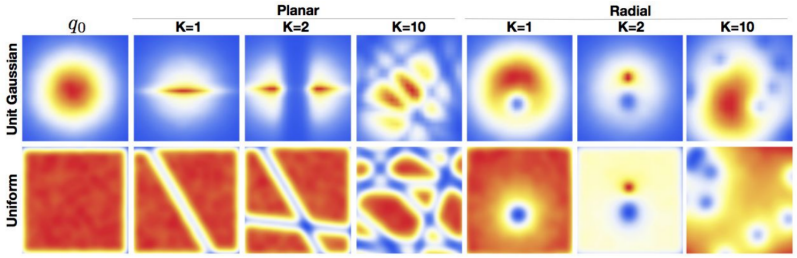


Figure 9.7: Example of a normalizing flows

Specifically, HMC mixes (or reaches a stationary point) poorly in spaces with ill-conditioned geometries. The idea is to transform a space into something close to an isotropic Gaussian.

Therefore, normalizing flows for transformations are such that we learn an invertible transformation of P_k to a Gaussian.

9.1.12 Proposed Approach

9.1.12.1 A Ladder Towards Failure

In order to sample from a rate probability, we decompose it to a sequence of network probability (a ladder towards failure). We let this β_k to be increasing and finally go to plus infinity; that's how we approach our target density of the distribution.

$$\rho_x(x) := \rho_0(x) exp(\beta_k[\gamma - f(x)])$$

where

P_k: the k^{th} distribution

ρ_k: the k^{th} un-normalized density

$Z_k := \int_X \rho_k(x)dx$: the normalizing constants

$exp(\beta_k[\gamma - f(x)])$: exponential barrier function

to iteratively compute the probability of γ

$$p_\gamma := \mathbb{P}_0(f(X) \leq \gamma) = \mathbb{E}_{P_k}\left[\frac{Z_K}{Z_0}\frac{\rho_\infty(X)}{\rho_K(X)}\right]$$

$$\frac{Z_K}{Z_0} = \prod_{k=1}^{K}\frac{Z_k}{Z_{k-1}}$$

Figure 9.1 shows one example of the sequence of the ρ case.

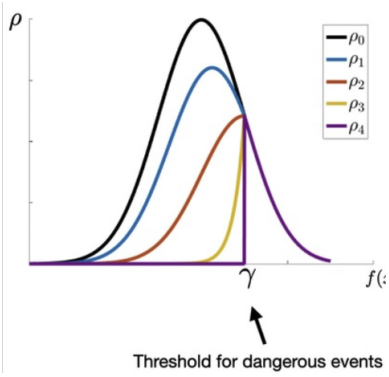


Figure 9.8: Threshold for dangerous events

Figure 9.2 is an example of choosing space function as the standard Gaussian function and we're interested in the choice of safety measure. By picking the value of γ function the pink area on the top right is our region of interest.

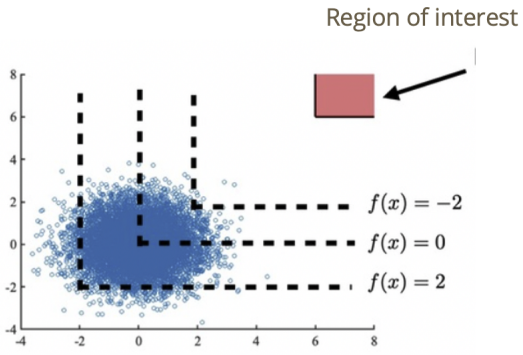


Figure 9.9: Finding the Right Space Function

The proposed algorithm consists of three steps:

- Exploit: determine the next β using current samples (k^{th} distribution)
- Explore + optimize: utlize gradient-based MCMC to sample from $(k+1)^{st}$ distribution
- Estimate: compute Z_{k+1}/Z_k via bridge sampling

9.1.12.2 Exploit

In the exploitation step we explore the existing samples those points in the blue cluster to estimate the next intermediate distribution by doing a **binary search** to solve an optimization problem.

Binary search: choose β_{k+1} such that

$$\frac{Z_{k+1}}{Z_k} \approx \alpha$$

9.1.12.3 Explore + optimize

In this step we will use the gradient based Hamiltonian Monte-Carlo sampling method to sample from the updated distribution. The gradient to the intermediate density as the figure shows how automatically trades up between the exploration with the ρ_0 and optimization with the gradient out.

Trade-off between exploration and optimization

$$\nabla \log \rho_k(x) = \nabla \log \rho_0 - \beta_k \nabla f(x) I\{f(x) > \gamma\}$$

$\nabla \log \rho_0$: exploration

$\beta_k \nabla f(x) I\{f(x) > \gamma\}$: optimization

9.1.12.4 Estimate

After two steps now we have the new sample from the updated distribution which is shown as the red cluster. Our final goal is to estimate the ratio of the normalizing constant $\frac{Z_{k+1}}{Z_k}$ via bridge sampling.

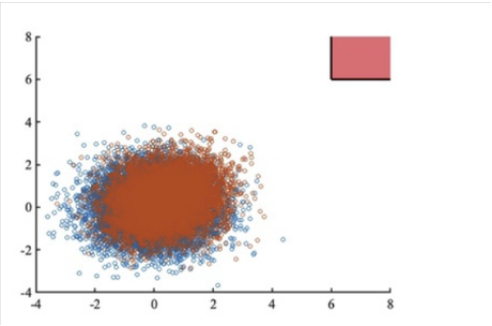


Figure 9.10: Estimation Update

9.1.12.5 Bridge Sampling

Putting all those three steps together we now have all the variables that translate from the blue cluster into the pink region of interest.

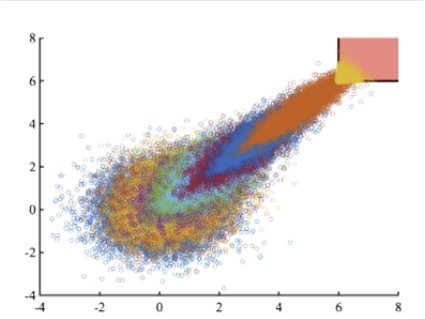


Figure 9.11: Bridge Sampling Reaches the Region of Interest

9.1.13 Summary of the Approach

We summarized the overall approach presented in the paper here.

Algorithm 1 Neural bridge sampling

Input: N samples $x_i^0 \overset{\text{i.i.d.}}{\sim} P_0$, MCMC steps T, step size $\alpha \in (0,1)$, stop condition $s \in (0,1)$
Initialize $k \leftarrow 0$, $\beta_0 \leftarrow 0$, $\log(\hat{p}_\gamma) \leftarrow 0$
while $\frac{1}{N}\sum_i I\{f(x_i^k) \le \gamma\} < s$ **do**
  $\beta_{k+1} \leftarrow$ solve problem (8) ◇—exploitation
  for $i = 1$ to N, in parallel
    $x_i^{k+1} \overset{\text{i.i.d.}}{\sim} \text{Mult}(\{\rho_{k+1}(x_i^k)/\rho_k(x_i^k)\})$ // multinomial resampling ◇—resampling
  for $t = 1$ to T
    for $i = 1$ to N, in parallel
      $x_i^{k+1} \leftarrow \text{WarpedHMC}(x_i^k, \theta_k)$ // Appendix A ◇—Exploration + optimization + warping
  $\theta_{k+1} \leftarrow$ argmin problem (6) // train normalizing flow on $\{x_i^{k+1}\}$ via SGD ◇—Embedding training
  $\log(\hat{p}_\gamma) \leftarrow \log(\hat{p}_\gamma) + \log(Z_{k+1}/Z_k)$ // warped bridge estimate (5) ◇—Estimation
  $k \leftarrow k+1$
$\log(\hat{p}_\gamma) \leftarrow \log(\hat{p}_\gamma) + \log(\frac{1}{N}\sum_i I\{f(x_i^k) \le \gamma\})$

Figure 9.12: Pseudocode for the algorithm

9.1.14 MCMC with an Exponential Barrier

We summarize the exploration and optimization process via HMC. First, we realize that HMC is volume preserving, and therefore, there is no need to adjust the acceptance probability, and therefore a Hessian calculator is unnecessary.

Exponential barriers combine exploration and optimization with the following function:

$$\log \rho_k(x) = \log \rho_0(x) + \beta_k[\gamma - f(x)] \tag{9.17}$$

β_k is a modulation factor that measures how much ∇f (the optimization component) and $\nabla \log \rho_0(x)$ (exploration component) contribute.

9.1.15 Bridge Sampling Estimation

We introduce bridge sampling. It estimates the ratio of normalizing constants Z_k/Z_{k+1} in our case. We define the geometric bridge:

$$\rho_k^B(x) = \sqrt{\rho_{k-1}\rho(x)} \tag{9.18}$$

The ratio of $\rho_{k-1}(x)$ and $\rho(x)$ cancel out the $\rho_0(x)$ dependence.

Therefore, in bridge sampling estimation, we have $E_k = Z_k/Z_{k-1}$ and therefore E_k can be estimated without calculating $\rho_0(x)$.

9.1.16 Neural Warping

For neural warping, we improve HMC mixing by transforming a space into an isotropic Gaussian. This also improves the bridge sampling efficiency of P_k. We look for invertible transformations of the form $y_i^k = W_k(x_i^k)$.

Minimizing the KL divergence between the transformed samples and the standard Gaussian results in the following objective:

$$\min_\theta \sum_{i=1}^{N} |detJ_{W_k}(x_i^k;\theta) + \frac{1}{2}\|W_k(x_i^k;\theta)\|_2^2 \tag{9.19}$$

9.1.17 Adaptive Intermediate Distributions

In order to choose β_k, we exploit the previous progress to choose β_k online.

We choose β_k as the solution to the optimization problem (with binary search). Optimization therefore balances the naive Monte Carlo estimate of p_γ with neural warping estimate of p_γ to determine the severity of exploration.

9.1.18 Performance Analysis

We can bound the asymptotic relative mean squared error:

$$\mathbb{E}[(\hat{\rho}_\gamma/\rho_g amma)^2] \tag{9.20}$$

is encapsulated in the following figure:

$$\underbrace{\frac{2}{N}\sum_{k=1}^{K}\left(\frac{1}{G(P_{k-1},P_k)^2}-1\right)}_{\text{Variance of } \widehat{E}_k} - \underbrace{\frac{2}{N}\sum_{k=1}^{K-1}\left(\frac{G(P_{k-1},P_{k+1})}{G(P_{k-1},P_k)G(P_k,P_{k+1})}-1\right)}_{\text{Covariance terms}} + \underbrace{\frac{1-s}{sN}}_{\text{Relative variance of the Monte Carlo estimate } \frac{1}{N}\sum_{i=1}^{N}\mathbf{1}\left\{f(x_i^K)<\gamma\right\}} + o\left(\frac{1}{N}\right)$$

Figure 9.13: Bound on mean-squared error

The above upper bound is empirically estimable.

There is an overall efficiency gain of $\mathcal{O}(\frac{1}{p_\gamma}\log(p_\gamma)^2)$ over Monte Carlo. These are the relative advantages:

	Cost	Error
Neural bridge sampling	$N\log(1/p_\gamma)$	$\frac{\log(1/p_\gamma)}{N}$
Monte Carlo	N	$\frac{1}{p_\gamma N}$

Figure 9.14: Relative Efficiency Gains over Monte Carlo

9.1.19 Experiments

The main task of this paper is to evaluate the safety of autonomous systems **efficiently**.

The experiments are divided into parts:

1. Synthetic problem
2. Formally verified neural network controller on the OpenAI Gym example continuous MountainCar under a domain perturbation.
3. There are two examples of engineering design in high dimensional settings:
   (a) Comparing thruster sizes to safely land a rocket in the presence of wind
   (b) Comparing two algorithms on the OpenAI Gym CarRacing environment.

The baseline methods are:

1. Naive Monte Carlo
2. Ablation studies for the effects of neural warping (denoted as NB with warping and B without).
3. Adaptive multilevel splitting

All methods are given the same computational budget as measured by evaluations of the simulator.

The ground-truth values p_γ for the non-synthetic problems using a fixed very large number of MC queries.

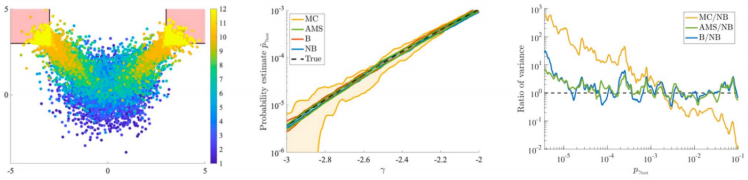
9.1.19.1 Experiments: Synthetic Problem

As an example, we consider the synthetic problem:

$$f(x) = -\min(|x_1|, x_2) \tag{9.21}$$

and $\gamma = -3$ and $P_0 = \mathcal{N}(0, I)$.

Experiments lead to:



(a) Samples colored by iteration (b) $\hat{p}_{\gamma_{\text{test}}}$ vs. γ_{test} (c) Ratio of variance vs. $p_{\gamma_{\text{test}}}$

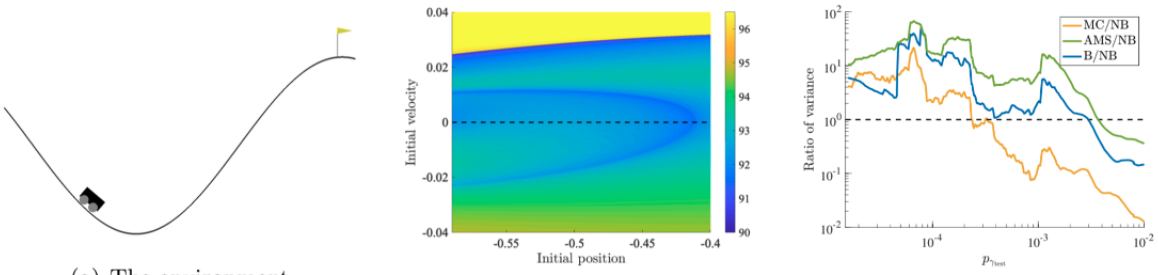
Figure 1. Experiments on a synthetic problem. 10 trials are used to calculate the 99% confidence intervals in (b) and variance ratios in (c). All adaptive methods perform similarly in this well-conditioned search space except at very small γ, where NB performs the best.

Figure 9.15: Caption

9.1.19.2 Experiments: Formally-Verified Controller under Domain Shift

Although this controller is verified, but it is kind of brittle or not robust to the distribution shift. So it assume the velocity to be zero at the initial point but in this case we add some perturbation to the initial-like condition.

- IWA + 2019 created a formally-verified neural network controller to achieve reward > 90 over all initial positions in [0.59, 0.4] and 0 initial velocity.
- The guarantees of formal verification hold only with respect to the specified domain; even small domain perturbations can affect system performance.
- Here, adding a small perturbation to the initial velocity.



(a) The environment (b) Contours of $f(x)$ (c) Ratio of variance vs. $p_{\gamma_{\text{test}}}$

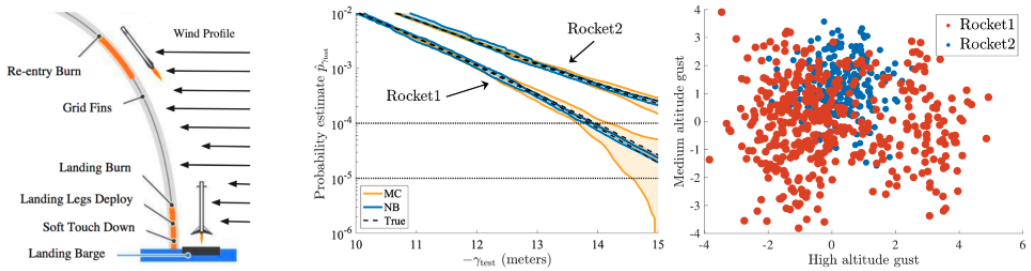
Figure 9.16: Experiments on the MountainCar Environment. The dashed horizontal line in (b) is the line along which the controller is formally verified. 10 trails are used for the variance ratios in (c). The irregular geometry degrades performance of AMS and B, but B benefits slightly from gradients over AMS. NB uses gradients and neural warping to outperform all other techinques.

9.1.19.3 Experiments: Rocket Design

A usable case for the estimation method.

- The amount of thrust which the rocket is capable of deploying to land safely must be **balanced against** the payload it is able to carry to space; stronger thrust increases safety but decreases payloads.

- Compare two rocket designs.
- Evaluate their respective probabilities of failure (*not landing safely on the landing pad*) for landing pad sizes up to 15 meters in radius.
- Help engineers quickly judge whether to increase the size of the landing pad or build a better rocket.



(a) Rocket landing (b) Failure rates (c) Failure modes

Figure 9.17: Rocket design experiments. NB's high-confidence estimates enable quick design iterations to either increase the landing pad radius or consider a third rocket that fails with probability around 10^{-5}. Lo-dimensional visualization shows that Rocket2's failure types are more concentrated than those of Rocket12, even though Rocket2 has a higher overall probability of failure.

9.1.19.4 Car Racing

In the deep reinforcement learning scenario for car racing, the paper compares two recent approaches:

- AttentionAgentRacer [TNH 2020]
- WorldModelRacer [HS 2018]

Both systems utilize one or more deep neural networks to plan in image-space, so neither has performance guarantees.

9.1.20 Experiments Comparison

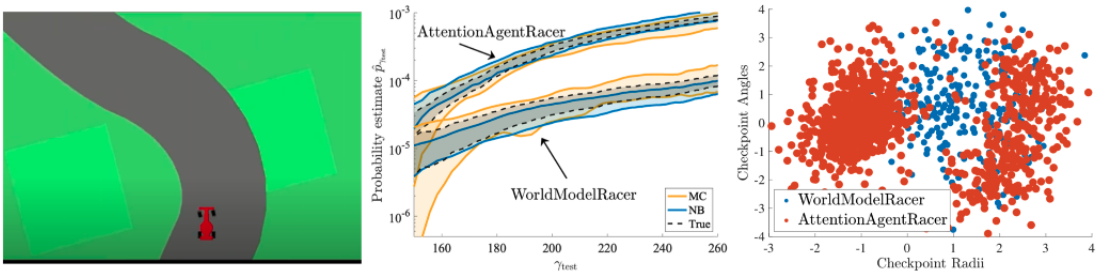
Table 1 shows that the method proposed in this paper can estimate the p_γ more accurately with other baseline approach.

9.1.21 Discussion

(Chat) Albert Qü: What does "guard set" mean in this context?

(Comment) Yiling You: Some criteria need to jump from one hybrid system to another.

(Chat) Claire Tomlin: Guard set is the set of states from which you can take a transition from one discrete state to another, as Yiling just said.



(a) The environment (b) Failure rates (c) Failure modes

Figure 9.18: CarRacing Experiments. MC cannot distinguish between the policies below $\gamma_{test} = 160$. NB's high-confidence estimates enable model comparisons at extreme limits of failure. Low-dimensional visualization of the failure modes hows that the algorithms fail in distinct ways.

Table 1: Relative mean-square error $\mathbb{E}[(\hat{p}_\gamma/p_\gamma - 1)^2]$ over 10 trials

	Synthetic	MountainCar	Rocket1	Rocket2	AttentionAgentRacer	WorldModelRacer
MC	1.1821	0.2410	1.1039	0.0865	1.0866	0.9508
AMS	0.0162	0.5424	0.0325	0.0151	1.0211	0.8177
B	0.0514	0.3856	0.0129	0.0323	0.9030	0.7837
NB	**0.0051**	**0.0945**	**0.0102**	**0.0078**	**0.2285**	**0.1218**
p_γ	$3.6 \cdot 10^{-6}$	$1.6 \cdot 10^{-5}$	$2.3 \cdot 10^{-5}$	$2.4 \cdot 10^{-4}$	$\approx 2.5 \cdot 10^{-5}$	$\approx 9.5 \cdot 10^{-6}$

Figure 9.19: Experiments Comparison

(Chat) Ayush Agrawal: I think this is a nice paper that talks about using tools from reachability where the model of the system is not explicitly known: Bridging Safety and RL Paper

(Chat) Chinmay Maheshwari: Are we supposed to solve the SDE in the MALA algorithm?

(Chat) Kshitij Kulkarni: In order to do the two-step process in MALA, you will have to discretize the diffusion, right?

(Comment) S.Shankar Sastry: For the Langevin equation, are you using the steady state distribution or are you using the time-varying solution?

(Comment) Michael Lim: Yeah so I think in order to use the Langevin equation what you are plugging in for the potential of the gradient is like some information about the target distribution or like at least the gradient of the target distribution at that point.

(Comment) S.Shankar Sastry: I presume the stationary distribution is exponential of minus u of x plus one half mx dot squared divided by the square of that variance of the white noise. That's the stationary distribution. So what do you do with that x dot squared term? So the diffusion on the $2n$-dimensional space whereas you are sampling x in R^n.

(Comment) Claire Tomlin: You are also sampling x dot, right? That's what you would have to do if you think about that as a vector differential equation.

(Comment) Yi Ma: In practice, you have to discretize.

(Comment) Claire Tomlin: The state space is $2n$-dimensional.

(Comment) Michael Lim: We add some damping term, estimate the finite different, and add some white noise.

(Comment) Yi Ma: We follow the gradient dynamics and use the Metropolis acceptance condition to sampling your state.

(Comment) Yi Ma: (summary for the overall approach) It depends on the F now, if f is multi model, the exploration can be high. How some one can certain if this will be converged?

(Comment) Yaodong Yu: You do need to sample all the region.

(Comment) Yi Ma: How do you establish the target distribution?

(Comment) Michael Lim: targeted distribution is the main part of bridge sampling. Use Barrier function to estimate, not sure about the kinetic energy though

(Chat) Albert Qü: So is the idea is that you sample both the second order and the first order? Or you sample second order and calculate first order X?

((Chat) Albert Qü: so I am assuming the second order sampling is to correct the bias with the first order sampling?

(Chat) Michael Lim: MCMC Demo Link

(Chat) Yi Ma: Here is a good website for MCMC samplers: Link

(Chat) Albert Qü: In perceptual problems, would framing f as the mis-classification / detection like accuracy scores be a good risk function to use and then use the sampling method to constrain the risk? (To some extent this seems very circular because the risk commonly were in the objective function explicitly?) I mean to calculate the derivative.

(Chat) Yaodong Yu: Also, for most examples studied in this paper, the f is kind of well defined. For example, the reward is used to act as the 'safety metric' for the DRL examples.

(Chat) Albert Qü: It looks like this is flipping the optimization problem of minimizing objective function and convert this to learning the distributions on the feasible set through a sequence of bridges?

(Chat) Yaodong Yu: Yeah, that sounds reasonable to me.

(Chat) Yi Ma: In general the task can seem ambitious and intractable. However, for proper class of distributions (as functions) with nice properties, this might not be daunting.

(Chat) Kshitij Kulkarni: @Prof Ma: The paper gives an example of the following function where gradient descent collapses x_i to a line. The function is $f(x) = \min(|x[1]|, x[2])$ for x in R^2. Their argument is that the non smoothness of this function doesn't allow one to track how volumes have been distorted.

(Chat) Yiling You: G is the so-called "Bhattacharyya coefficient". Intuitively it measures the distance between the distributions.

(Chat) Yi Ma: The variational auto-encoding part is a little strange: one essentially assume the distribution is unimodal... hence a smooth transformation to Gaussian... otherwise, if the distribution is multi-modal, you are approximating its convex envelop — sampling in between can be wasteful...

(Chat) Yi Ma: I always believe auto-encoding needs to address the multi-modal problem of possible mixed distributions — which is a more realistic assumption for real data. But that entails us to do clustering or unsupervised learning better. In the uni-modal case, the logdet objective of (6) is the "volume" or "coding rate" of the (Gaussian-like) distribution. Interested students can read our rate reduction paper: Link to paper This is a rather universal quantity. But one has to use it properly in case the data are multi-modal (and the distribution can be degenerate).

(Chat) Jay Monga: In this sense, does the work in the paper seem more useful for judging which actions will increase safety (changing aspects of rocket design for example), or to provide assurance beyond doubt of if a system is safe enough?

(Chat) Yaodong Yu: @Jay Yeah, I think the safety measurement could be helpful for judging which actions can increase safety.

(Chat) Yiling: This is a good point, I also wonder how this safety measurement can systematically auxiliate the design of safe controllers.

[2] [1] [4] [3]

References

[1] "Illustration of hamiltonian monte carlo." [Online]. Available: https://github.com/chi-feng/mcmc-demo

[2] "Illustration of markov chain monte carlo." [Online]. Available: https://www.researchgate.net/figure/Illustration-of-Markov-Chain-Monte-Carlo-method_fig1_334001505

[3] "Illustration of normalizing flow." [Online]. Available: https://arxiv.org/pdf/1505.05770.pdf

[4] "Illustration of variational autoencoder." [Online]. Available: https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

[5] A. Sinha, M. O'Kelly, R. Tedrake, and J. Duchi, "Neural bridge sampling for evaluating safety-critical autonomous systems," 2020.