Neural bridge sampling for evaluating safety-critical autonomous systems

NeurIPS 2020

Aman Sinha*, Matthew O’Kelly*, Russ Tedrake, & John Duchi




Testing safety-critical systems

- As ML moves into safety-critical systems, we need to rigorously evaluate safety

- We need to verify systems are 99.99999999% reliable

Current methods are dangerous, slow, and/or expensive

Governing problem

- Given: Base distribution of behavior $X \sim P_0$ (with density ρ_0) and a simulator
- Given: Objective function (i.e. safety metric) $f : \mathcal{X} \to \mathbb{R}$
- Goal: probability of dangerous event $p_\gamma := \mathbb{P}_0(f(X) < \gamma)$
- Naive methods are too slow for small probabilities

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

Monte Carlo estimate

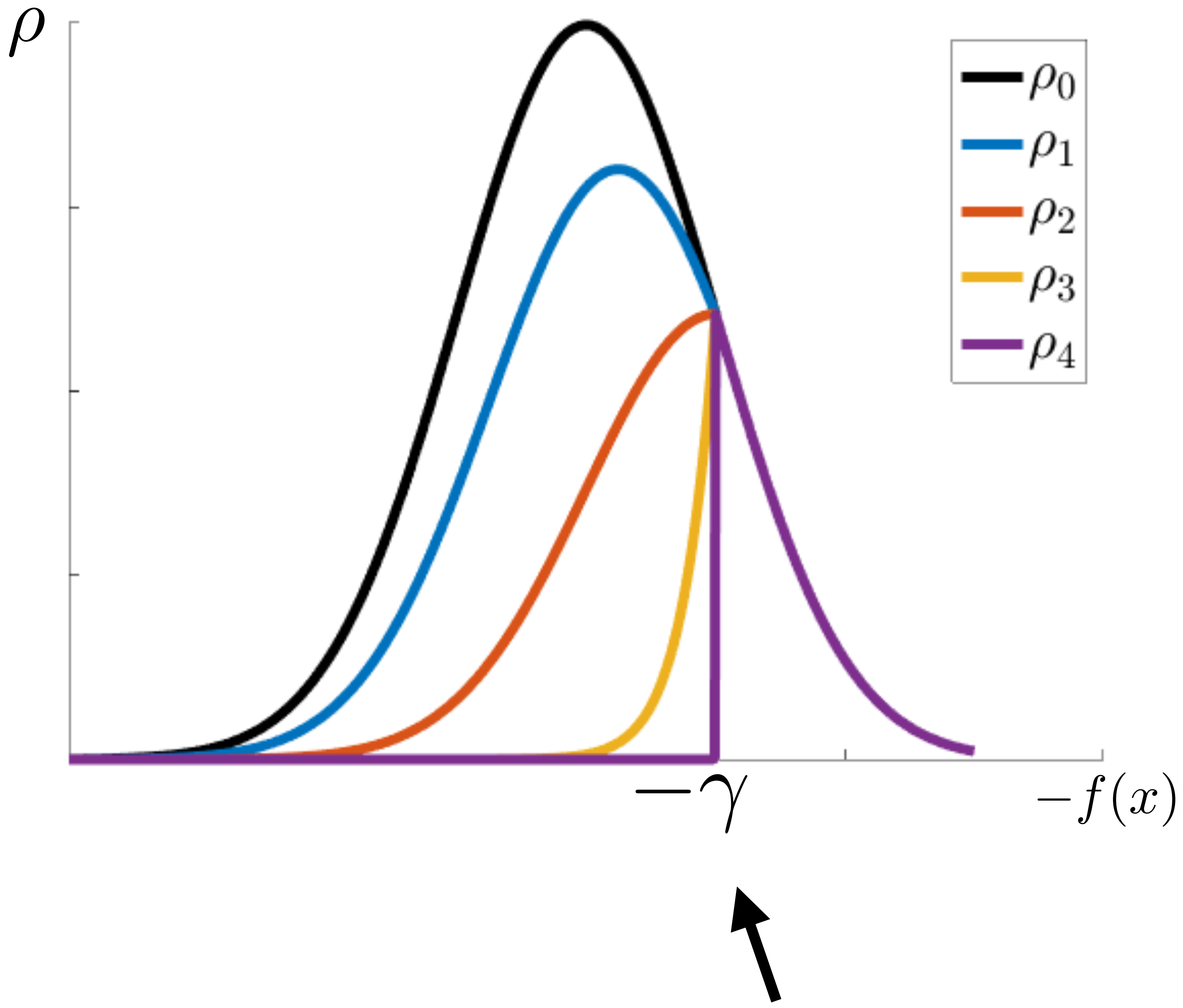
$$\mathbb{E}[(\hat{p}_\gamma / p_\gamma - 1)^2] = \frac{1 - p_\gamma}{N p_\gamma}$$

Error of Monte Carlo estimate

Our approach

- A ladder towards failure

$$\rho_k(x) := \rho_0(x) \underbrace{\exp\left(-\beta_k \left[f(x) - \gamma\right]_+\right)}_{\text{exponential barrier}}$$

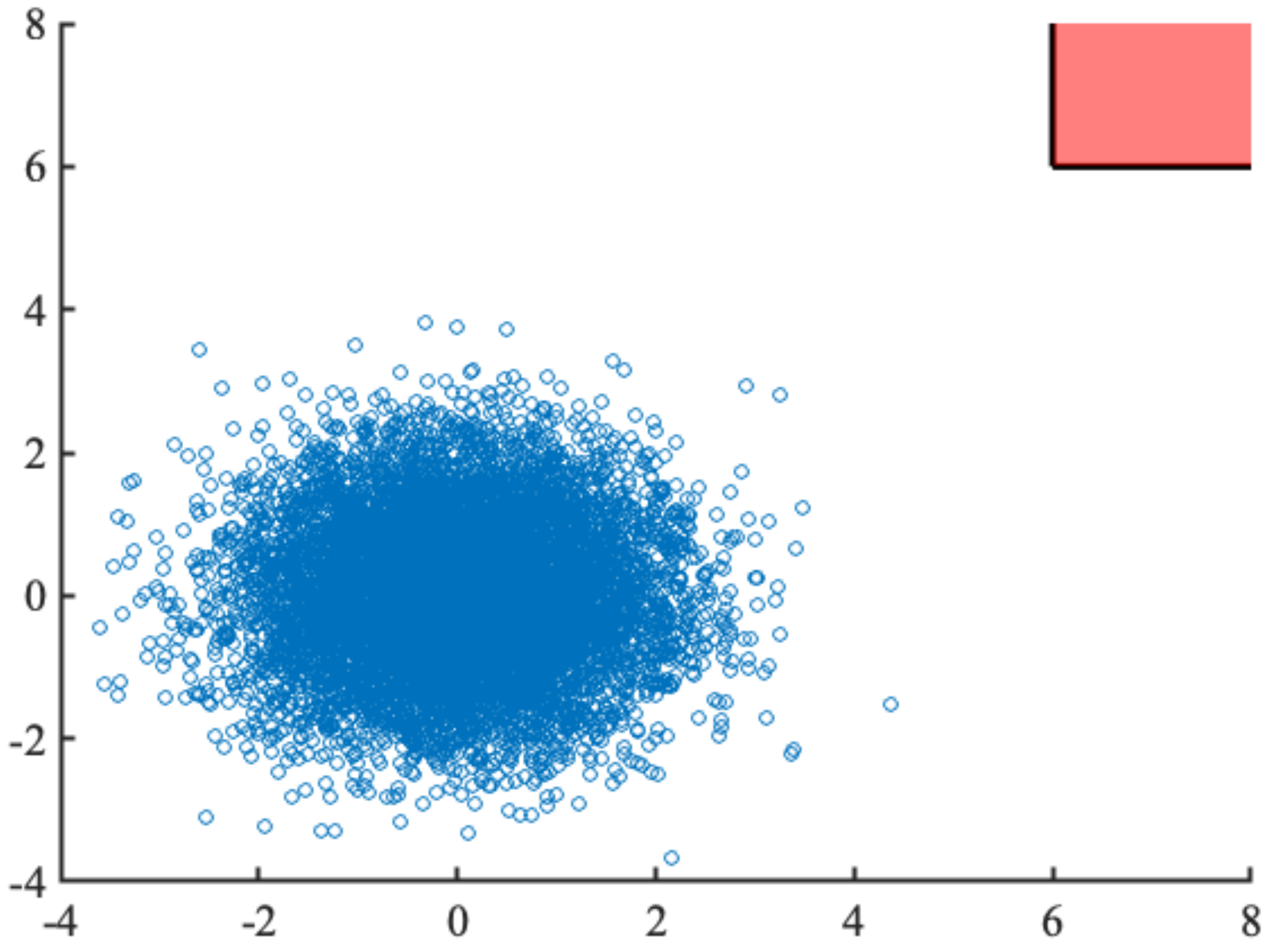


Threshold for dangerous events

Our approach

- A ladder towards failure

$$P_0 = \mathcal{N}(0, I)$$



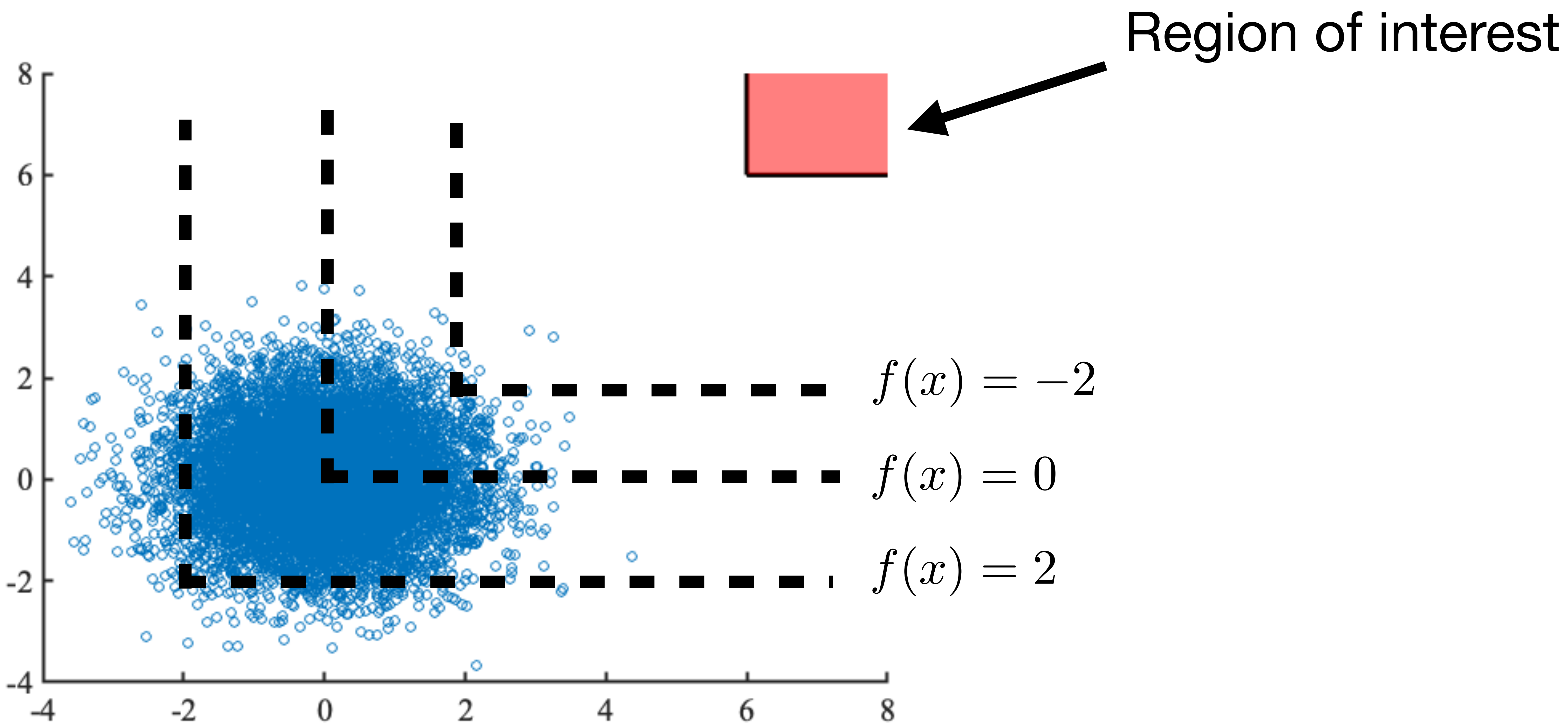
Our approach

- A ladder towards failure

$$P_0 = \mathcal{N}(0, I)$$

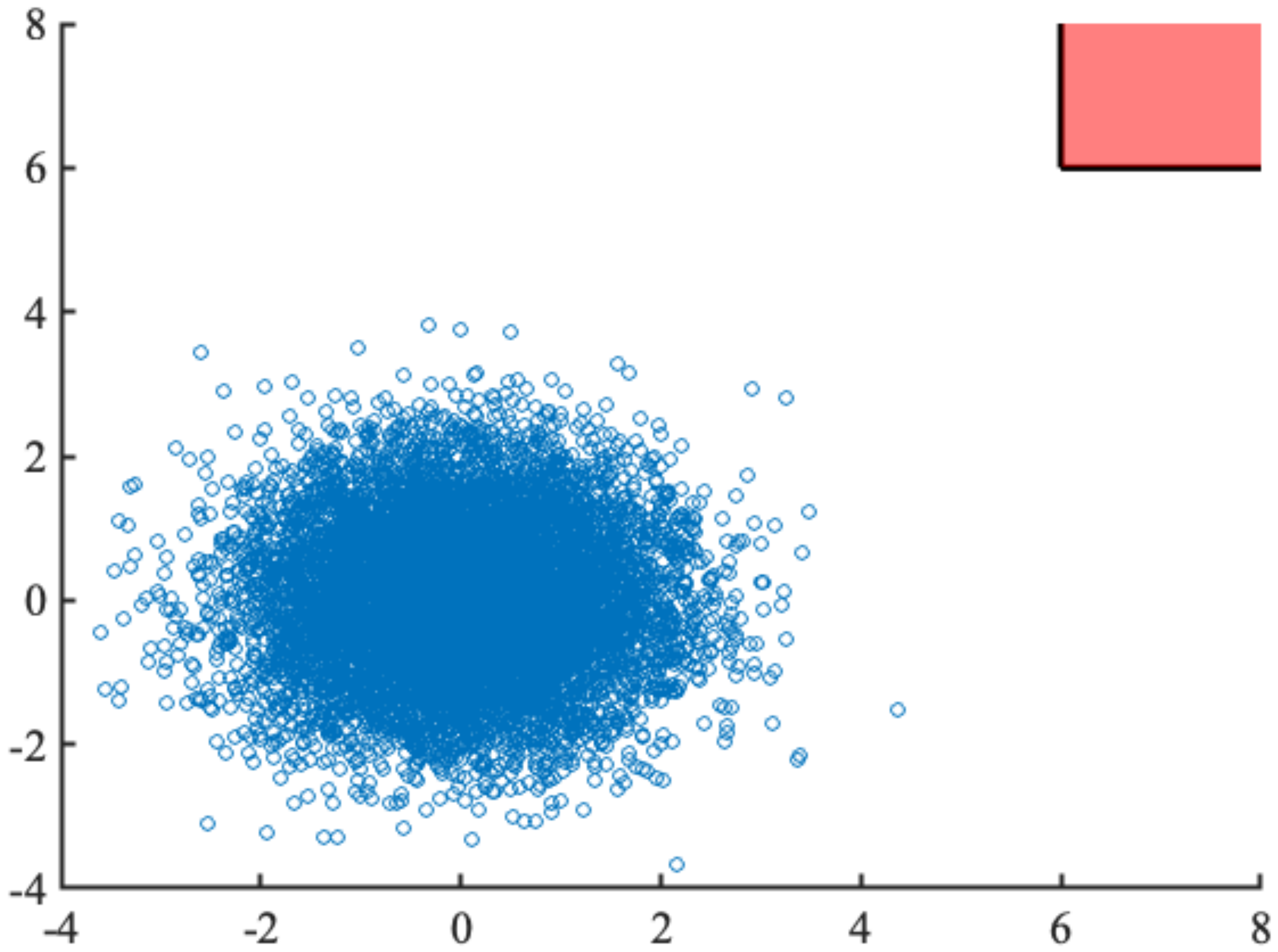
$$f(x) = -\min(x_{[i]})$$

$$\gamma = -6$$



Our approach

- A ladder towards failure
- Exploit: determine the next β using current samples (k[th] distribution)
- Explore + optimize: utilize gradient-based MCMC to sample from (k+1)[st] distribution
- Estimate: compute Z_{k+1}/Z_k via bridge sampling



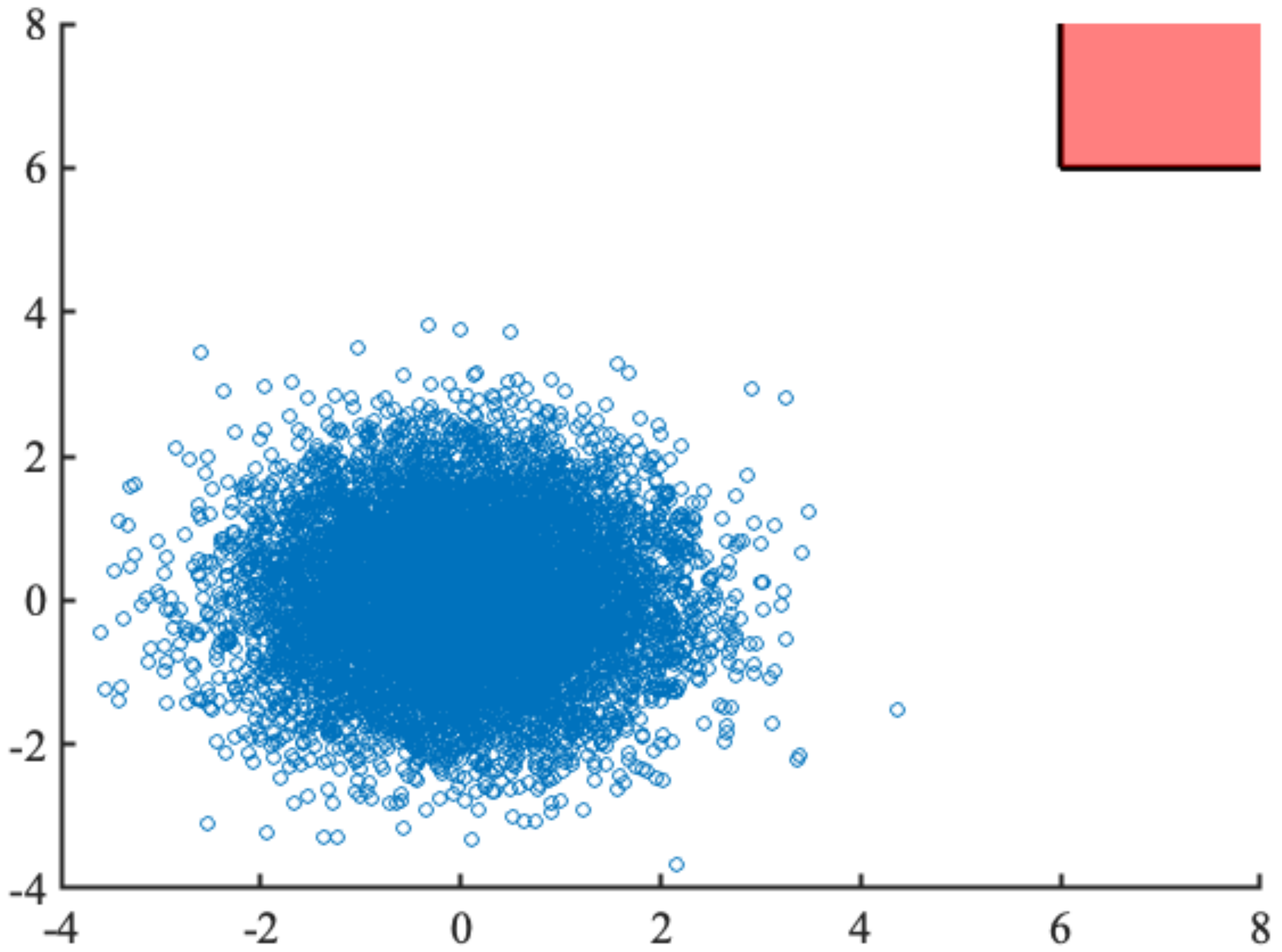
Our approach

- A ladder towards failure
- Exploit: determine the next β using current samples (k[th] distribution)
- Explore + optimize: utilize gradient-based MCMC to sample from (k+1)[st] distribution
- Estimate: compute Z_{k+1}/Z_k via bridge sampling

Choose β_{k+1} such that

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

(binary search)

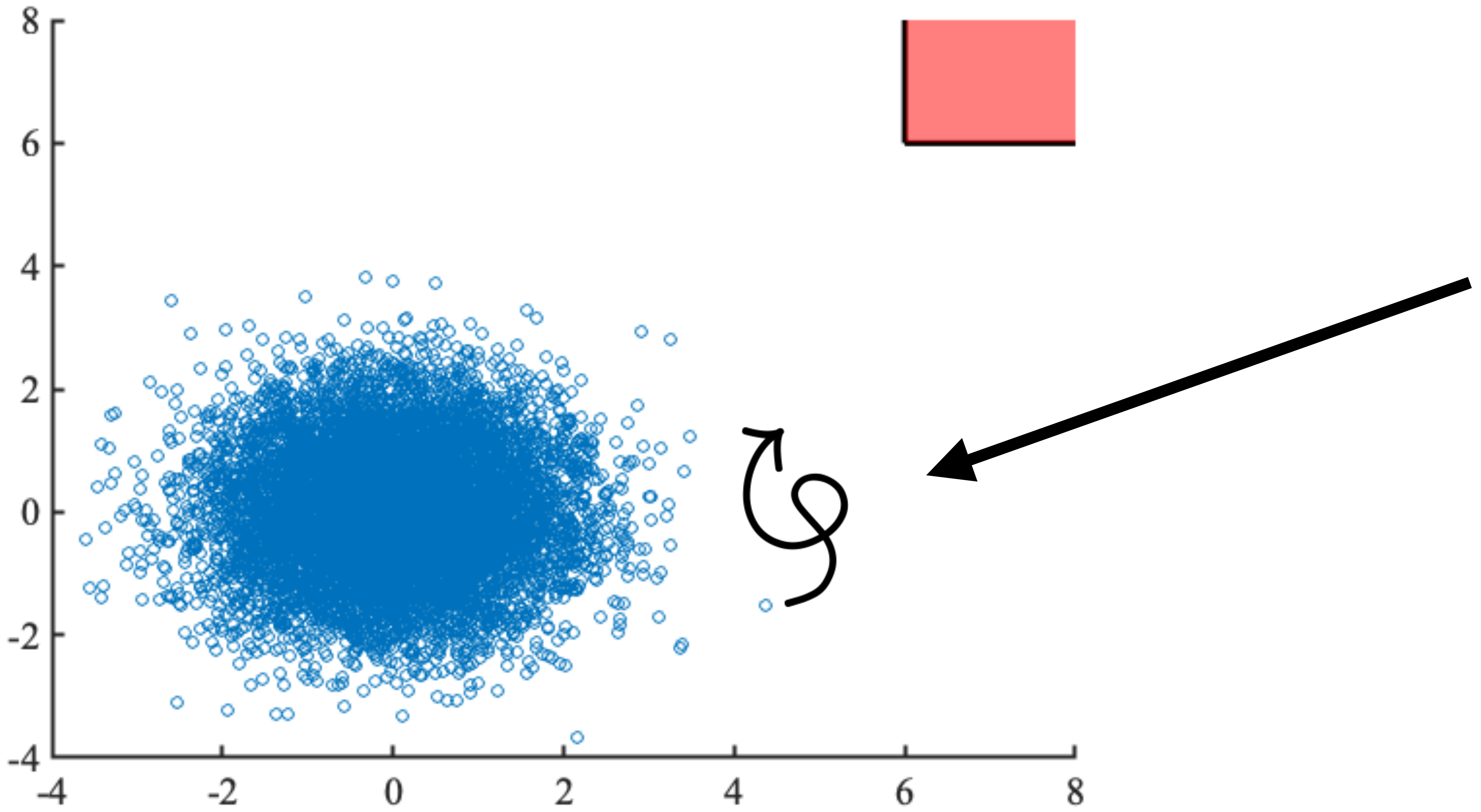


Our approach

- A ladder towards failure
- Exploit: determine the next β using current samples (k[th] distribution)
- Explore + optimize: utilize gradient-based MCMC to sample from (k+1)[st] distribution
- Estimate: compute Z_{k+1}/Z_k via bridge sampling

Automatic tradeoff between exploration and optimization

$$\nabla \log \rho_k(x) = \underbrace{\nabla \log \rho_0(x)}_{\text{exploration}} - \underbrace{\beta_k \nabla f(x) I\{f(x) > \gamma\}}_{\text{optimization}}$$

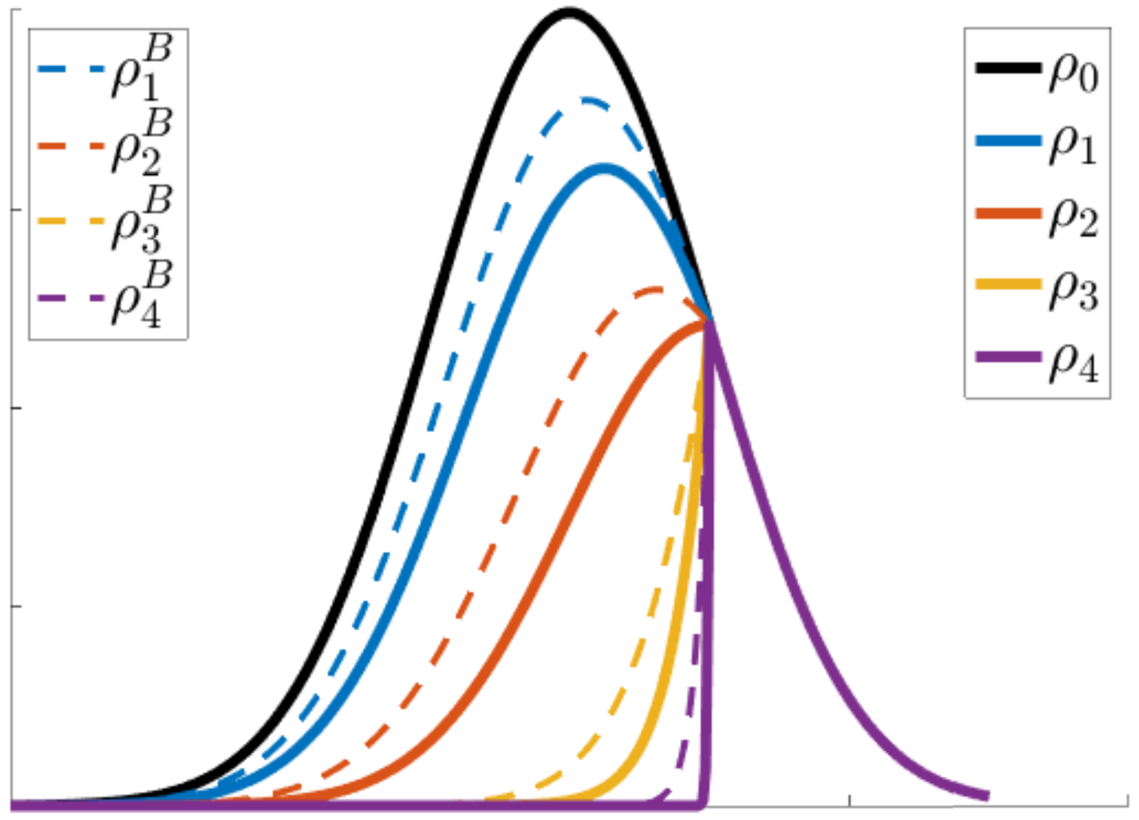


Hamiltonian Monte Carlo (HMC)

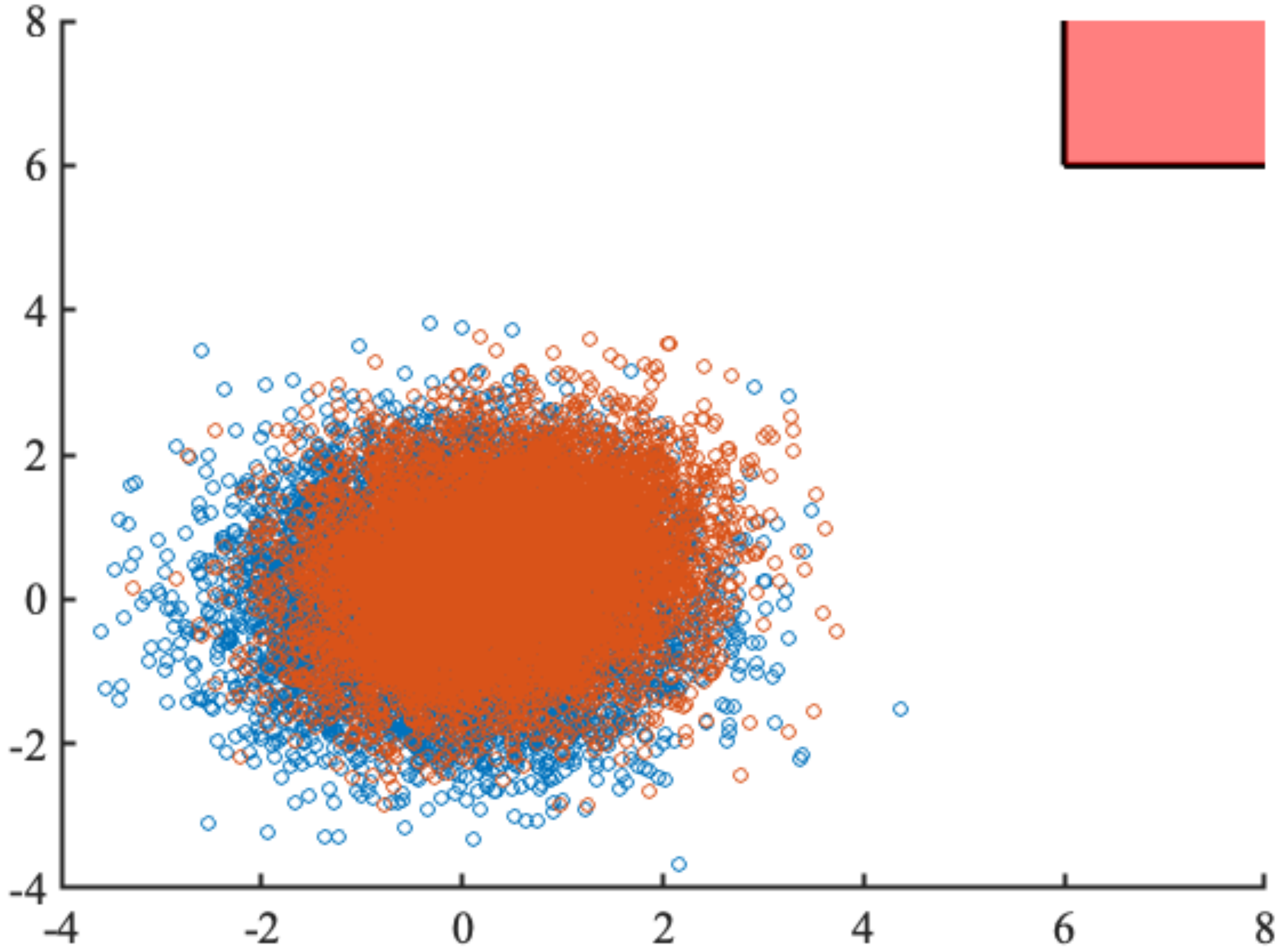
Our approach

- A ladder towards failure
- Exploit: determine the next β using current samples (k[th] distribution)
- Explore + optimize: utilize gradient-based MCMC to sample from (k+1)[st] distribution
- Estimate: compute Z_{k+1}/Z_k via bridge sampling

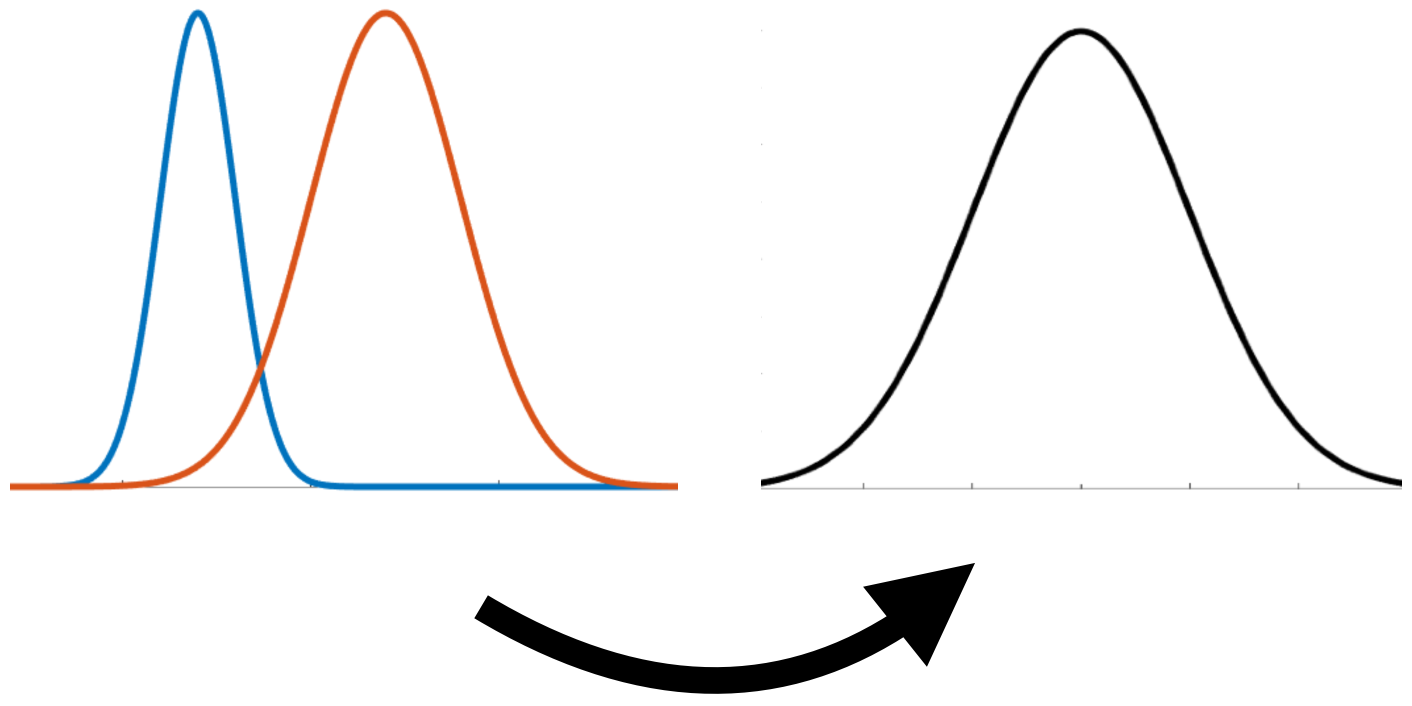
Use both sets of samples to compute an accurate estimate of Z_{k+1}/Z_k



Use an auxiliary "bridge" distribution



Error grows with distance between distributions, so we "warp" them

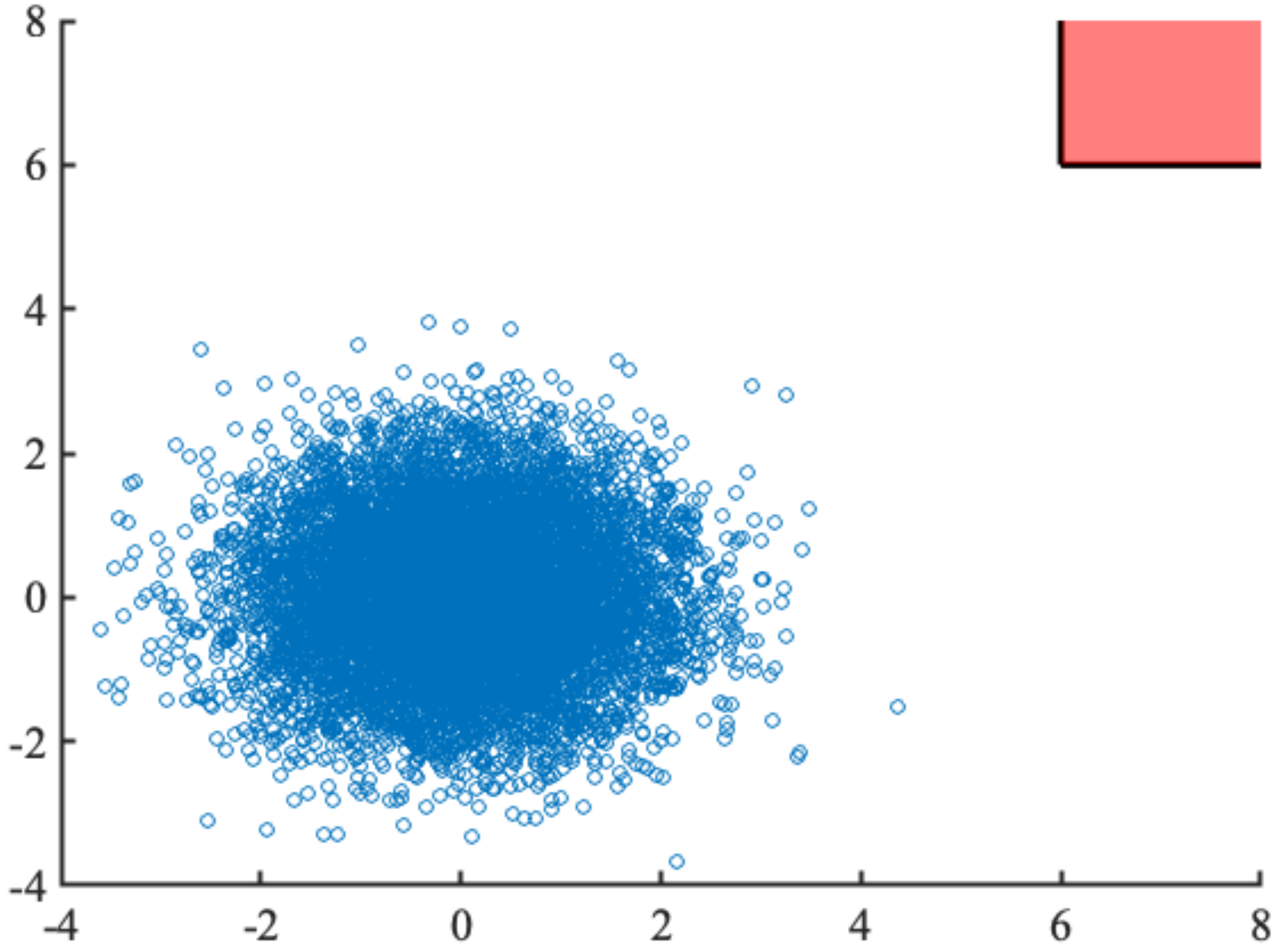


normalizing flow

[Hoffman et al. 2019, Papamakarios et al. 2019]

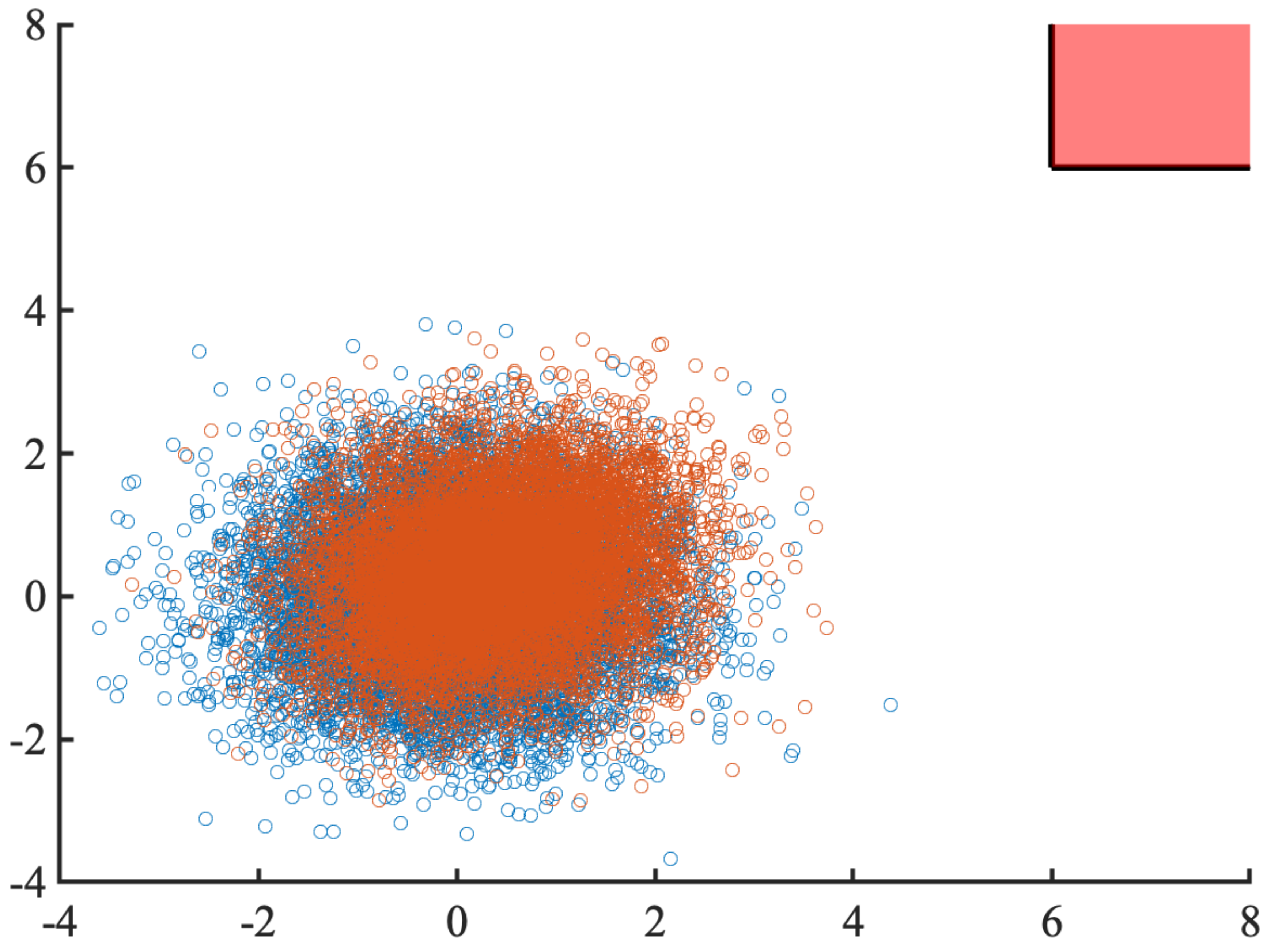
Our approach

- A smoother ladder towards failure
- Exploit: determine the next β using current samples (kth distribution)
- Explore + optimize: utilize gradient-based MCMC to sample from (k+1)st distribution
- Estimate: compute Z_{k+1}/Z_k via bridge sampling



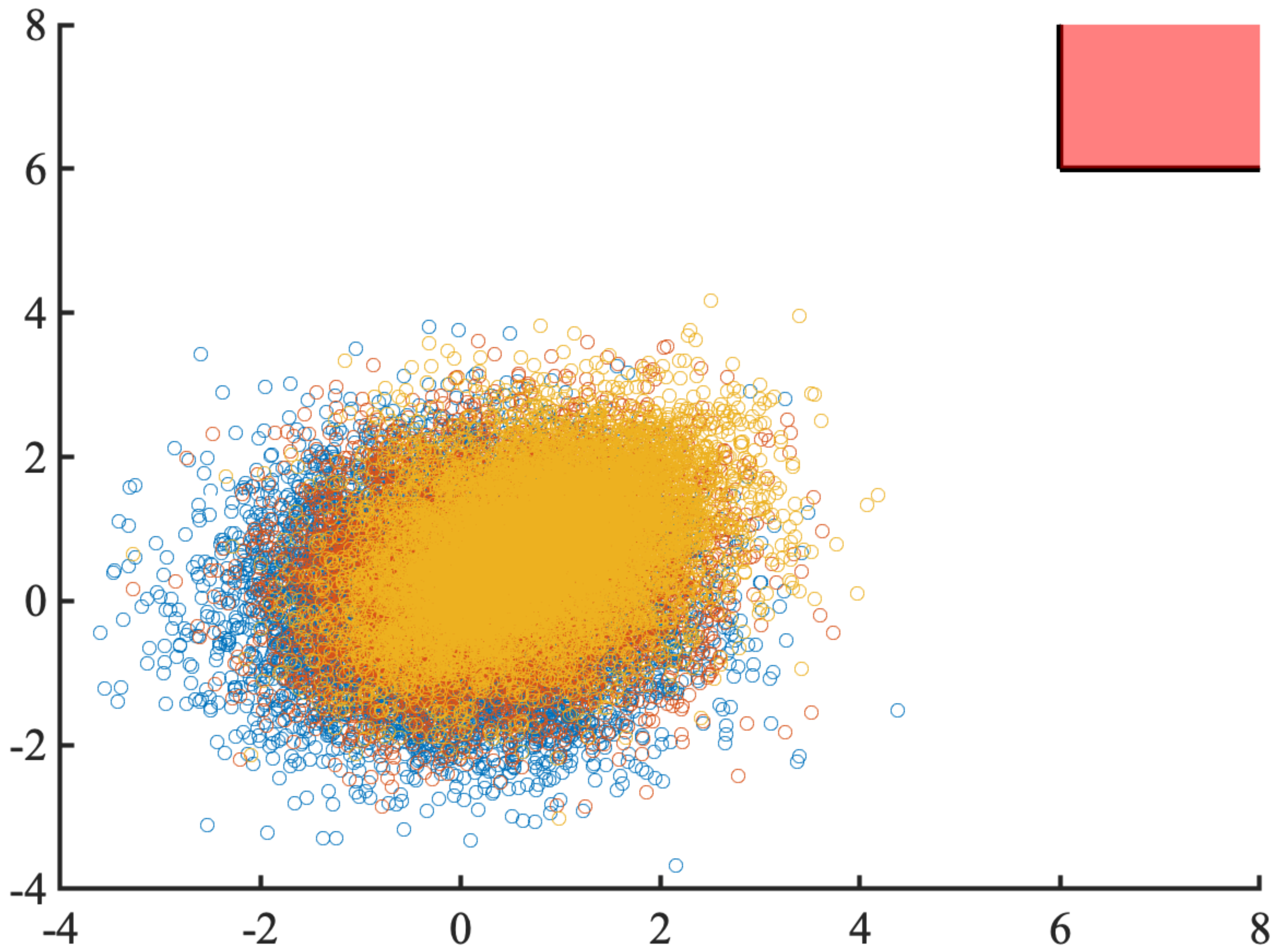
Our approach

- A smoother ladder towards failure
- Exploit: determine the next β using current samples (kth distribution)
- Explore + optimize: utilize gradient-based MCMC to sample from (k+1)st distribution
- Estimate: compute Z_{k+1}/Z_k via bridge sampling



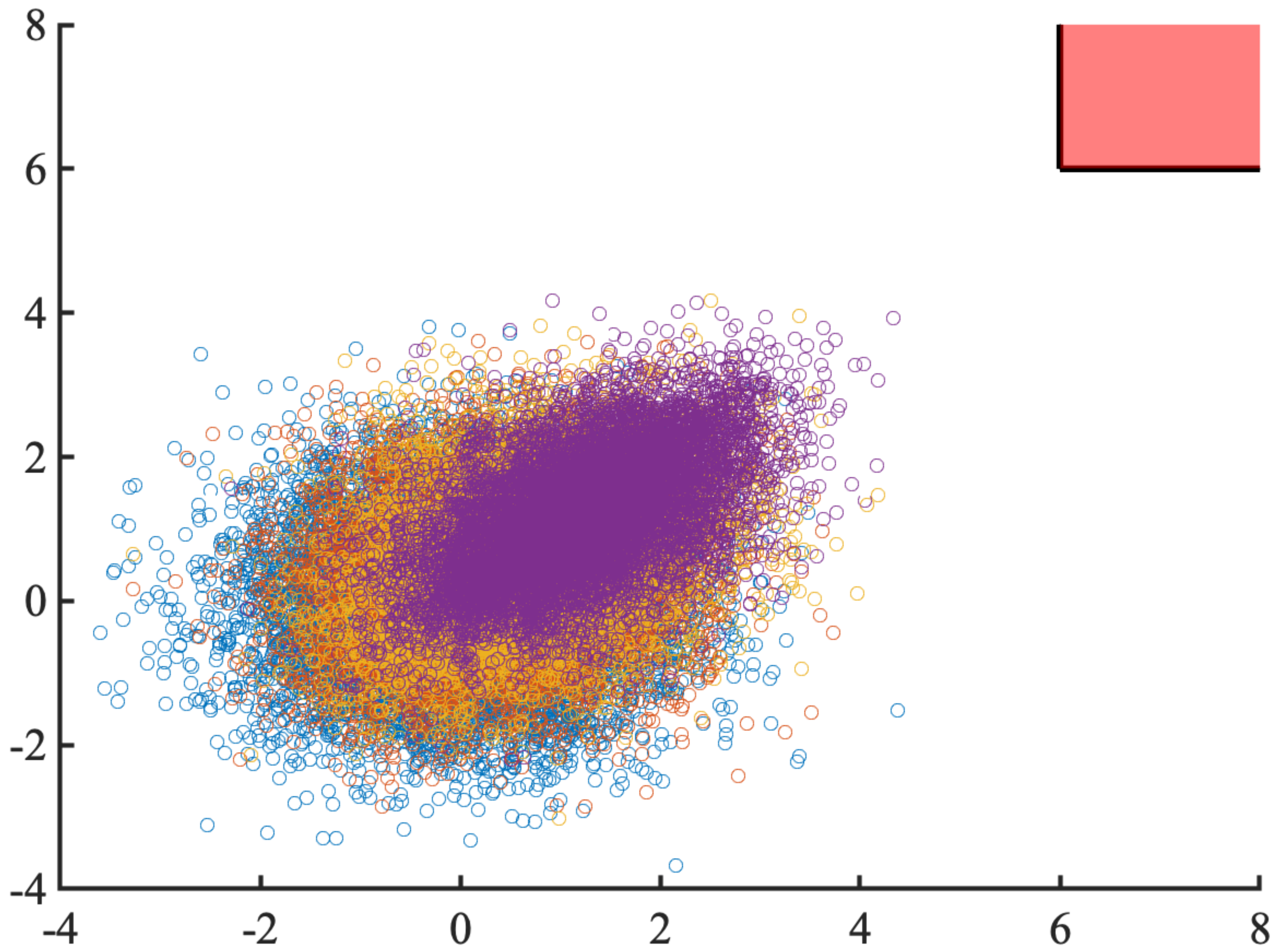
Our approach

- A smoother ladder towards failure
- Exploit: determine the next β using current samples (kth distribution)
- Explore + optimize: utilize gradient-based MCMC to sample from (k+1)st distribution
- Estimate: compute Z_{k+1}/Z_k via bridge sampling



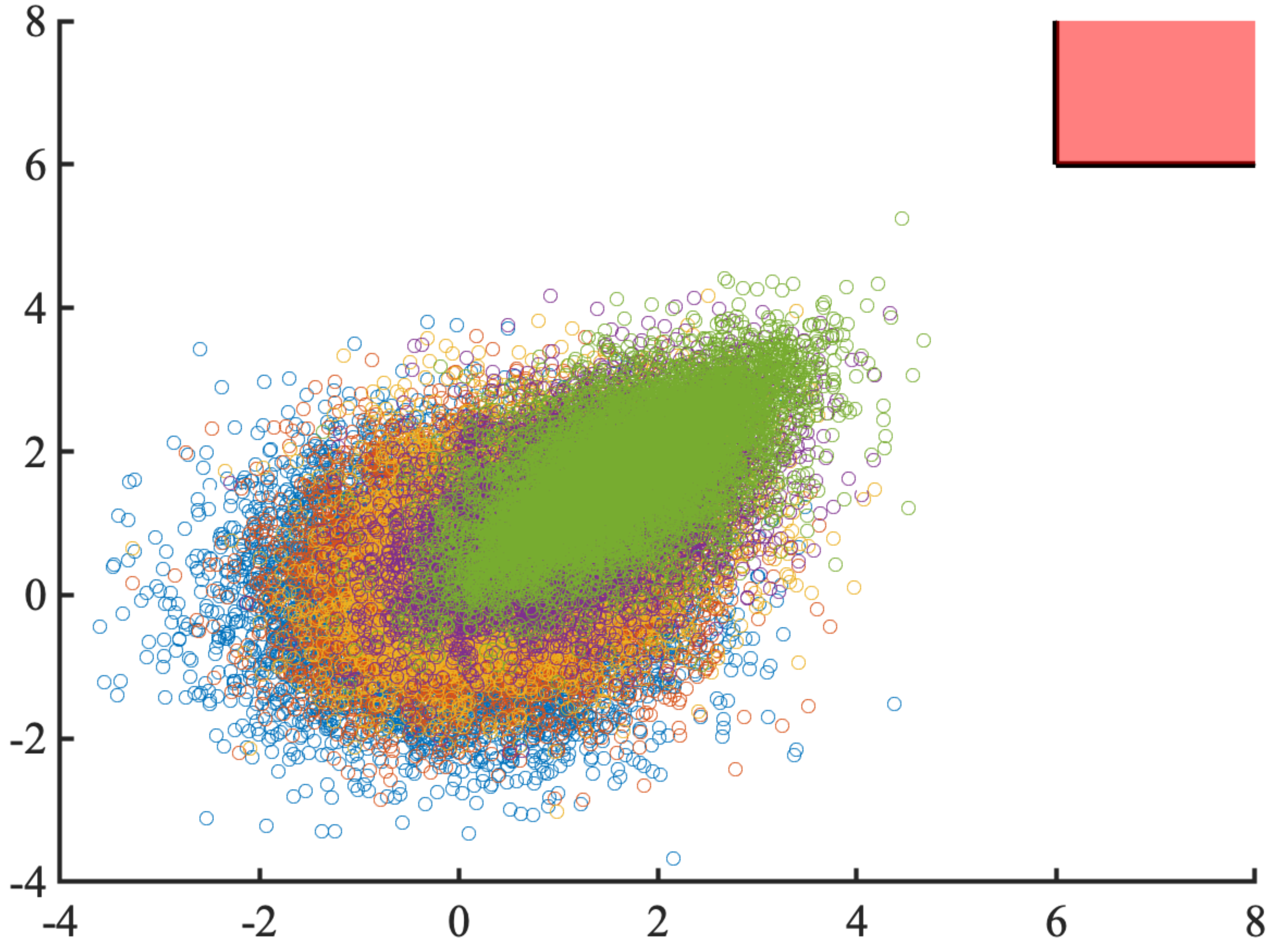
Our approach

- A smoother ladder towards failure
- Exploit: determine the next β using current samples (k[th] distribution)
- Explore + optimize: utilize gradient-based MCMC to sample from (k+1)[st] distribution
- Estimate: compute Z_{k+1}/Z_k via bridge sampling



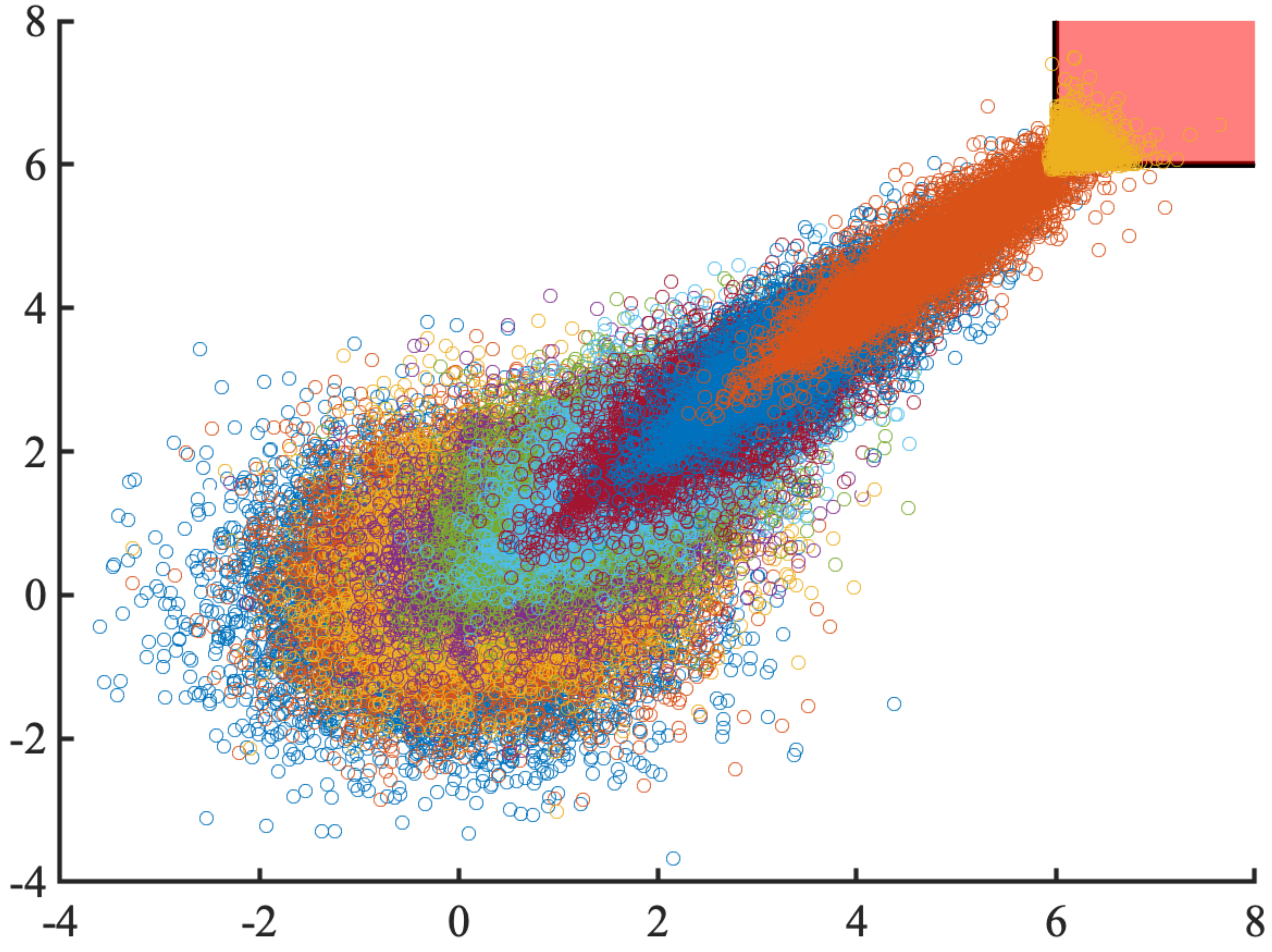
Our approach

- A smoother ladder towards failure
- Exploit: determine the next β using current samples (kth distribution)
- Explore + optimize: utilize gradient-based MCMC to sample from (k+1)st distribution
- Estimate: compute Z_{k+1}/Z_k via bridge sampling



Our approach

- A smoother ladder towards failure
- Exploit: determine the next β using current samples (kth distribution)
- Explore + optimize: utilize gradient-based MCMC to sample from (k+1)st distribution
- Estimate: compute Z_{k+1}/Z_k via bridge sampling



Performance guarantees

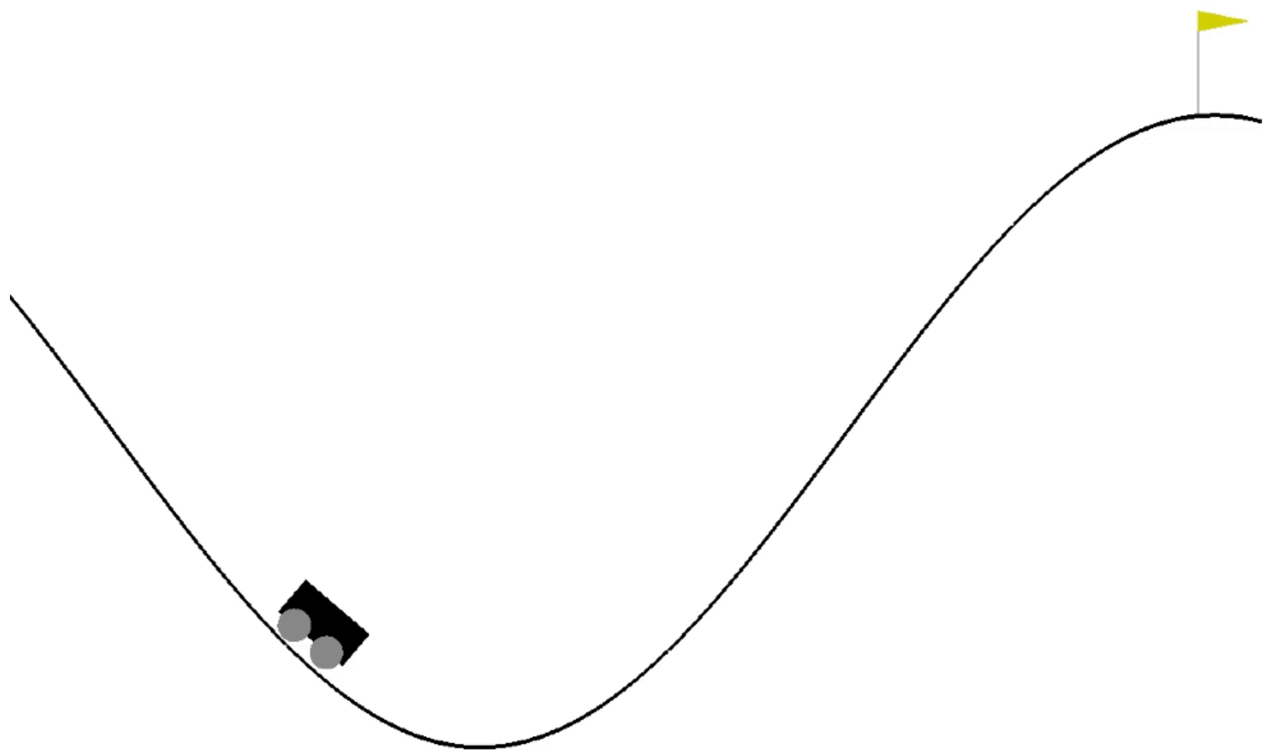
- Overall efficiency gain of $O\left(\frac{1}{p_\gamma \log(p_\gamma)^2}\right)$ over Monte Carlo

- Relative advantage scales with rarity

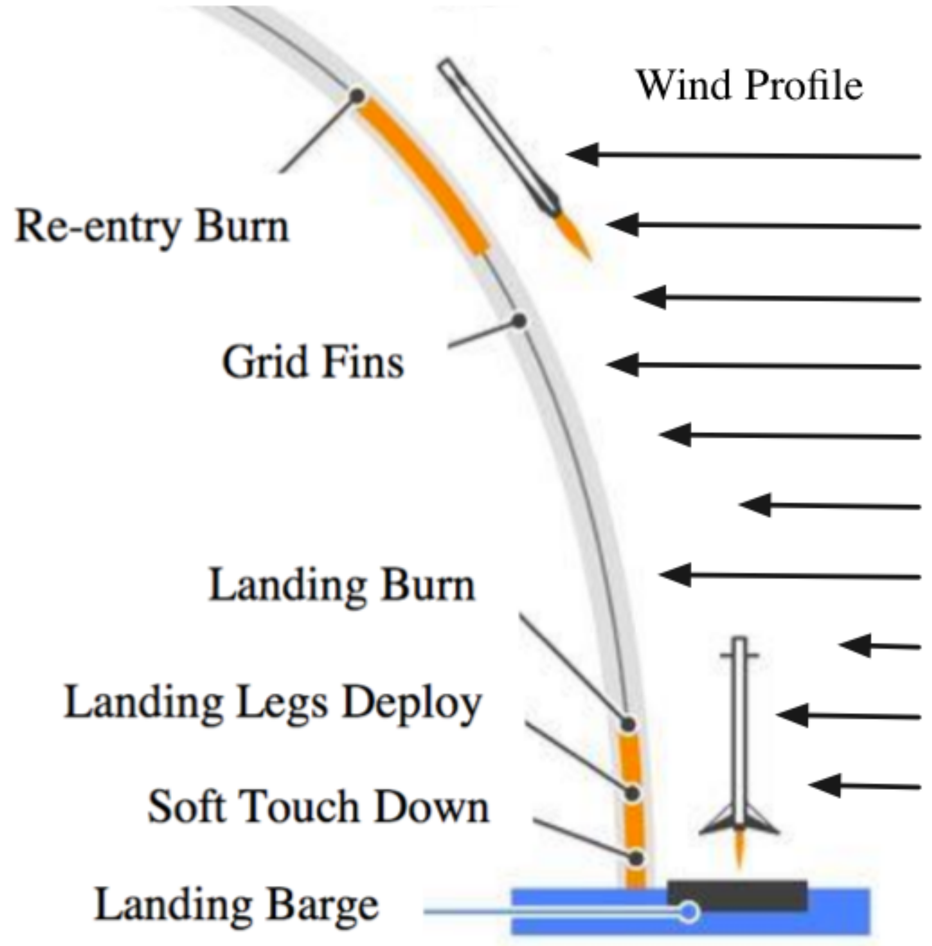
	Cost	**Error**
Neural bridge sampling	$N\log(1/p_\gamma)$	$\frac{\log(1/p_\gamma)}{N}$
Monte Carlo	N	$\frac{1}{p_\gamma N}$

Experiments

Formally-verified neural network for MountainCar



Compare two designs for vertical landing of an orbital-class rocket



Compare two SOTA methods for CarRacing

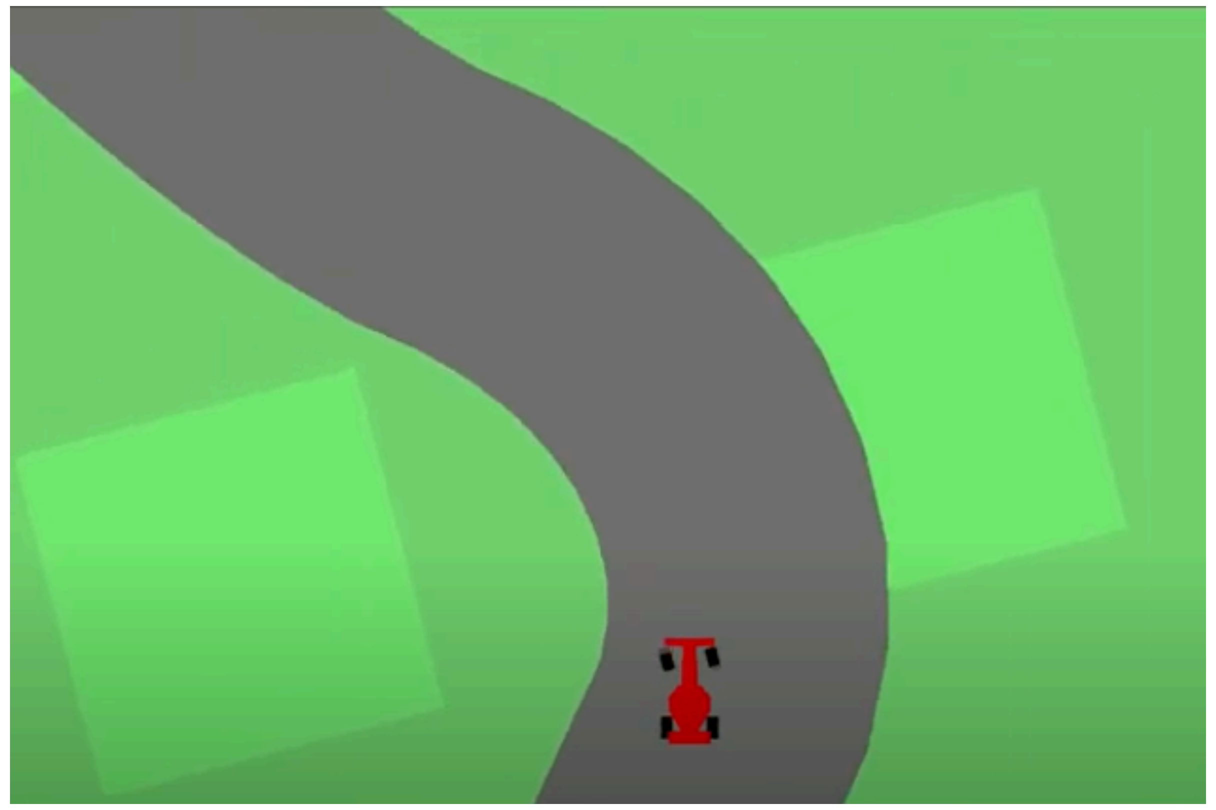


Table 1: Relative mean-square error $\mathbb{E}[(\hat{p}_\gamma/p_\gamma - 1)^2]$ over 10 trials

	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	$3.6 \cdot 10^{-6}$	$1.6 \cdot 10^{-5}$	$2.3 \cdot 10^{-5}$	$2.4 \cdot 10^{-4}$	$\approx 2.5 \cdot 10^{-5}$	$\approx 9.5 \cdot 10^{-6}$

Our approach (NB) outperforms other methods