Machine learning of large deviations

Hugo Touchette

Department of Mathematical Sciences
Stellenbosch University, South Africa

Rare Events: Analysis, Numerics, and Applications
Brin Mathematics Research Center
University of Maryland, USA

- Jiawei Yan and Grant Rotskoff
  PRE **105**, 024115, 2022
  arxiv:2107.03348

Stellenbosch UNIVERSITY IYUNIVESITHI UNIVERSITEIT | forward together sonke siya phambili saam vorentoe

Dynamical large deviations

- Markov process: $(X_t)_{t=0}^T$
- Observable: $A_T = A_T[x]$

Rare event probability

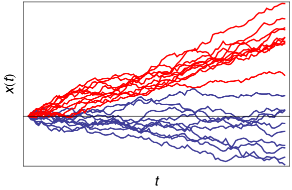
$$P(A_T = a) \approx e^{-T I(a)}$$

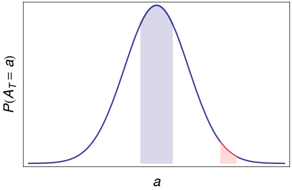
Generating function

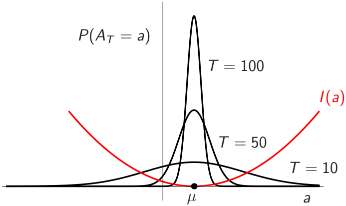
$$E[e^{T\lambda A_T}] \approx e^{T\psi(\lambda)}$$

Prediction problem

- How are fluctuations created?
- Conditioning: $X_t | A_T = a$
- Fluctuation / effective process







Different large deviations

Transition event

$$P(X_\tau^\varepsilon \in B | X_0^\varepsilon \in A) \approx e^{-I/\varepsilon}$$

- Low-noise limit
- Transition path

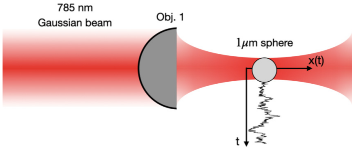
Extensive event

$$P\left(\frac{1}{T}\int_0^T f(X_t)dt \in C\right) \approx e^{-TI}$$

- Long-time limit
- Family of paths (process)

Physics

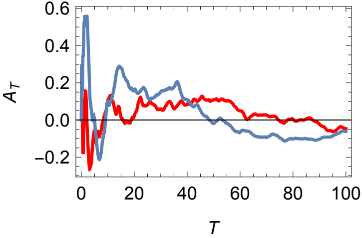
- Work done W_T on a system
- Heat Q_T exchanged
- Fluctuations related to dissipation



Simulations

- Time-averaged estimators
- Convergence determined by LDs
- Non-reversible acceleration

  [Rey-Bellet + Spiliopoulos 2015]



Large deviation theory

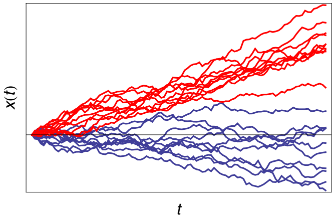
LD functions

- Rate function:

$$I(a) = \max_{\lambda \in \mathbb{R}} \{\lambda a - \psi(\lambda)\}$$

- SCGF:

$$\psi(\lambda) = \text{dom eigenval}(\mathcal{L}_\lambda)$$

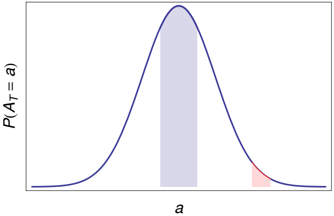


Fluctuation process

$$d\tilde{X}_t = \tilde{F}(\tilde{X}_t)dt + \sigma dW_t$$

- Modified drift:

$$\tilde{F} = F + D\nabla \ln r_\lambda, \quad I'(a) = \lambda$$

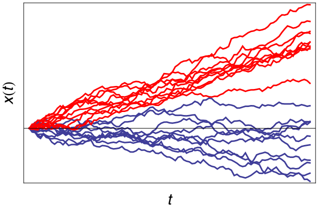


- Effective process creating fluctuation
- Efficient process for importance sampling

Optimal control representation

[Fleming 70s-80s; Chetrite & HT 2013-15; Jack & Sollich 2015]

$$X_t \sim P[x] \qquad \longrightarrow \qquad \tilde{X}_t \sim \tilde{P}[x]$$



- Cost function:

$$C_T = \frac{1}{T} \ln \frac{\tilde{P}[x]}{P[x]}, \qquad E_{\tilde{X}}[C_T] = \frac{1}{T} D(\tilde{P}||P)$$

SCGF

$$\psi(\lambda) = \lim_{T\to\infty} \min_{\tilde{X}} E_{\tilde{X}}[\lambda A_T - C_T]$$

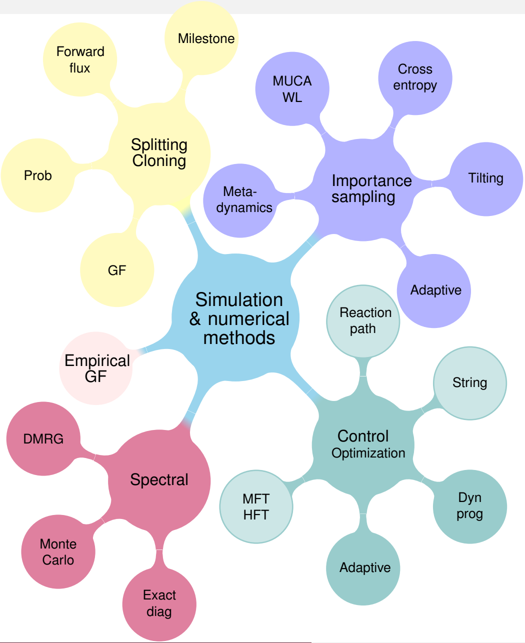
Rate function

$$I(a) = \lim_{T\to\infty} \min_{\substack{\tilde{X}_t \\ E_{\tilde{X}}[A_T]=a}} E_{\tilde{X}}[C_T]$$

- Dual optimization problems
- Minimizer: Effective process
- Cost estimator:

$$\hat{C}_T = \frac{1}{2\sigma^2} \int_0^T [F(\tilde{X}_t) - \tilde{F}(\tilde{X}_t)]^2 dt$$

Different methods



Numerical

- Spectral problem
- Control problem

Simulation

- Importance sampling
- Splitting / cloning

Problems

- High dim functions
- Find optimal sampler
- Simulate many traj

Machine learning approaches

- Solve spectral or control problem
- Representation: $r_\lambda(x) \approx u(x; \lambda, \underbrace{\theta}_{\text{params}})$

- Basis functions: David Limmer's group, Berkeley
  [Ray et al. PRL 2017, JCP 2020]
  [Das et al. JCP 2019]
- MPS and tensor nets: Juan Garrahan's group, Nottingham
  [Bañuls & Garrahan PRL 2019]
  [Causer et al. PRE 2021]
- Neural networks: [Oakes et al. ML Sci. & Tech. 2020]
- Reinforcement learning: [Rose et al. NJP 2021], [Das et al. JCP 2021]

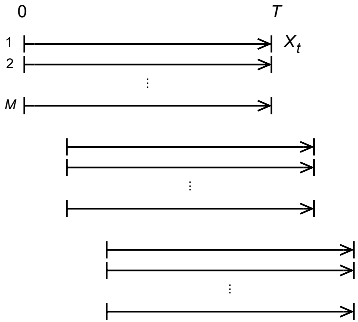
Our approach

- Trajectory gradient minimization of control cost
- NN representation of control force

Stochastic minimization

Algorithm

0. Initialize NN: $\tilde{F}(x) = u(x; \lambda, \theta)$
1. Simulate M trajectories (batches)
2. Estimate cost $\hat{C}_{M,T}(\lambda, \theta)$
3. Compute gradient $\nabla_\theta \hat{C}_{M,T}(\lambda, \theta)$
   - Autodiff
   - Adjoint method
4. Update NN: $\theta' = \theta - \gamma \nabla_\theta \hat{C}_{M,T}$
5. Repeat 1-4 (training)
6. Repeat for different λ



Extras

- Transfer learning: $\lambda : 0 \to \Delta\lambda \to 2\Delta\lambda \to \cdots$
- Replica exchange: $\lambda \leftrightarrow \lambda'$

Application 1: Simple diffusion

[Nemoto et al. PRE 2016]

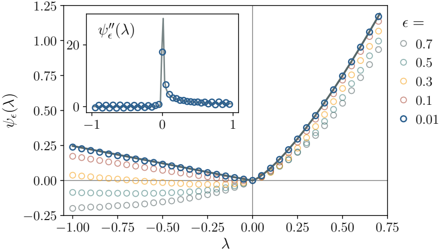
- Dynamics:

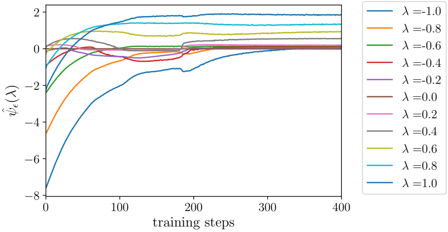
$$dX_t = -X_t^3 + \sqrt{2\varepsilon}dW_t$$

- Observable:

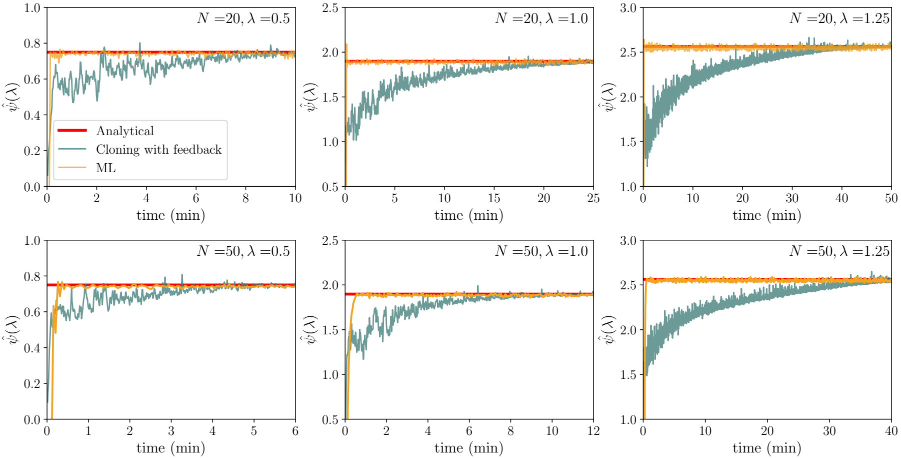
$$A_T = \frac{1}{T}\int_0^T X_t(X_t+1)dt$$

- NN: 2 layers, hidden dim = 50
- Training step: 220 traj, $T = 10$
- Autodiff (PyTorch)
- DPT at $\lambda = 0$
- No slowing down $\varepsilon \to 0$





Comparison with cloning

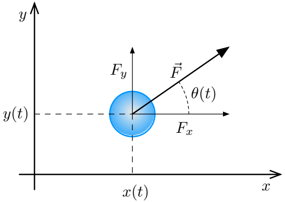


- Cloning with feedback [Nemoto et al. PRE 2016]
- Time, single workstation, mins
- N = batch size = no. trajectories

Application 2: Active Brownian particles

[Cagneta et al. PRL 2017; Chiarantoni et al. JPA 2020; GrandPre et al. PRE 2018, PRE 2021]

$$\dot{\mathbf{X}}_t^{(i)} = \underbrace{-\mu\frac{\partial U(\mathbf{X}_t)}{\partial \mathbf{x}^{(i)}}}_{\text{repulsive pair potential}} + \underbrace{v\mathbf{b}_t^{(i)}}_{\text{active drive}} + \underbrace{\sqrt{2D}\boldsymbol{\xi}_t^{(i)}}_{\text{noise}}$$

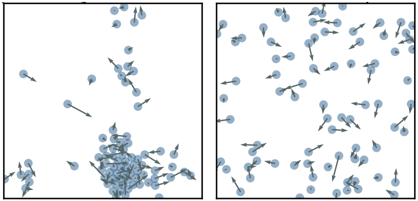


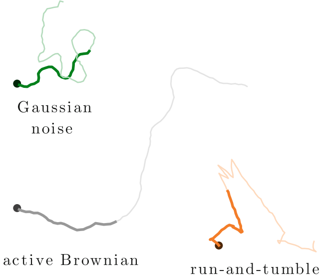
[Squarcini et al 2021]

- Active force:

$$\mathbf{b}_t^{(i)} = [\cos\phi_t^{(i)}, \sin\phi_t^{(i)}], \quad \phi_t^{(i)} = \sqrt{6D}\eta_t^{(i)}$$

- Directional persistence at short time scales
- Brownian run-and-tumble





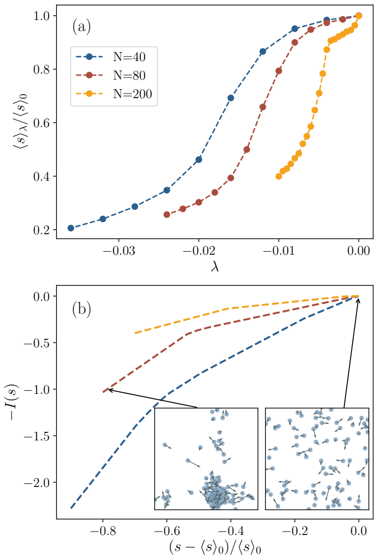
[Callegari and Volpe 2019]

Results

- Entropy production:

$$S_{N,T} = \frac{1}{NT} \sum_{i=1}^{N} \int_0^T D^{-1} v \mathbf{b}_t^{(i)} \circ d\mathbf{X}_t^{(i)}$$

- Different fluctuation phases
- $S_{N,T} = \langle S \rangle$:
  - Natural system
  - No clustering
- $S_{N,T} < \langle S \rangle$:
  - Directional force inhibited
  - Clustering
- Dynamical phase transition?
- 6 layers, hidden dim $= 1000$
- $M = 75$ or 20 for $N = 200$
- Adjoint gradient



Conclusions

- Scalable: Trajectories not stored
- Agnostic: No tuned representation
- Stable: Simple additive estimator
- Direct error estimates (batch means)
- Can be applied to Markov chains / jump processes

Future work

- Physics of modified force / interactions
- Trade-off density / flux
- Comparisons with other algorithms / benchmarks

J. Yan, H. Touchette, G. Rotskoff
Learning nonequilibrium control forces to characterize dynamical phase transitions
PRE **105**, 024115, 2022, arxiv:2107.03348

Source code: github.com/quark-strange/machine_learning_LDP

Gradient computation

Automatic differentiation

[Baydin et al. JMLR 2018]

```
def CostFunction(traj, params):
    ...
dC = grad(CostFunction, some_traj, some_params)
```

- PyTorch, TensorFlow, JAX
- Limited by size of computational graph

Adjoint sensitivity method

[Li et al. 2020]

$$\dot{x}(t) = u(x(t); \theta)$$

$$\frac{\partial}{\partial \theta} C = -\int_T^0 \underbrace{h(t)}_{\text{Lagrange param}} \underbrace{\frac{\partial u(x(t); \theta)}{\partial \theta}}_{\text{known}} dt$$

$$\dot{h}(t) = -h(t) \frac{\partial u(x(t); \theta)}{\partial x(t)}, \quad h(T) = \frac{\partial C(x(T))}{\partial x(T)}$$