Learning Committor Functions and Invariant Distributions for Randomly Perturbed Dynamical Systems

Weiqing Ren

Department of Mathematics
National University of Singapore

Joint work with Qianxiao Li and Bo Lin (NUS)

Dynamical systems perturbed by small noise

Consider a dynamical system modelled by the stochastic differential equation (SDE):

$$dX_t = f(X_t)dt + \sqrt{\varepsilon}dW_t, \quad X_t \in R^d$$

where f: deterministic force field ;
W_t: Wiener process; ε: strength of the noise.

In this talk, we discuss machine learning methods for the computation of

- The committor function;
- The invariant distribution.

The committor function

- We consider the over-damped Langevin dynamics, where $f = -\nabla V$.
- Assume the system has two metastable states (sets), A and B
- When the noise is small, the dynamics is characterized by long waiting period in metastable states followed by transitions from one state to the other.

The committor function (cont'd)

The committor function $q : \Omega \to [0, 1]$ for the transition from A to B is defined as

$$q(x) = \text{Prob}\Big\{\tau_B(x) < \tau_A(x)\Big\}.$$

where $\tau_{A,B}(x)$ is the first hitting time of A (B) initiated at x.

- q characterizes the progress of transitions from A to B.
- Much information regarding the transition, e.g. pathways, transition states and rates, can be obtained from q.

Example of the committor function

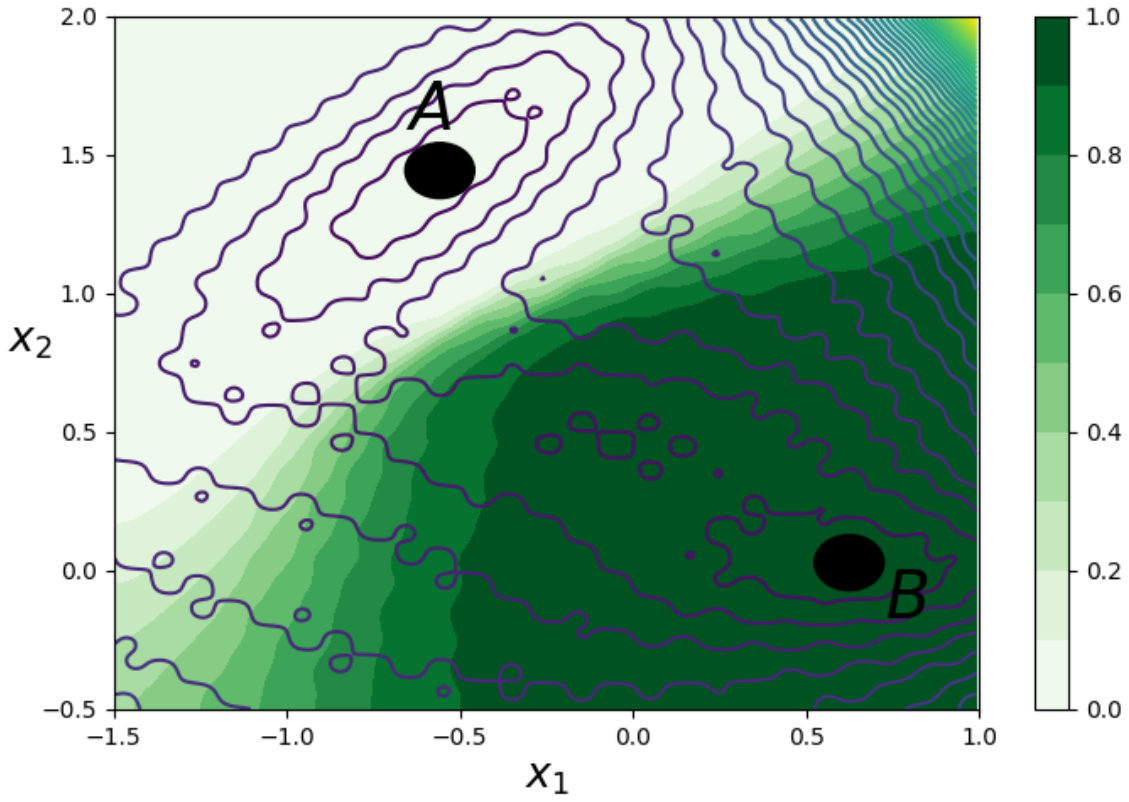


Figure: Contour lines of the potential V and the committor function q at $\varepsilon = 10$ (the energy barrier ≈ 100).

The backward Kolmogorov equation

A “simple” mathematical characterization:

$$\begin{cases} \nabla V \cdot \nabla q - \beta^{-1} \Delta q = 0, \\ q(x) = 0, \ x \in \partial A; \quad q(x) = 1, \ x \in \partial B \end{cases}$$

- Difficult to solve numerically due to high dimensionality.
- We introduce a numerical method which combines deep learning and data sampling to efficiently compute the committor function for *high-dimensional systems* and *at low temperatures*.
- Related work: Khoo, Lu & Ying; Evans, Cameron & Tiwary; Vanden-Eijnden et al; ...

A variational formulation

An equivalent variational formulation:

$$\min_q \frac{1}{Z} \int_{\Omega\setminus(A\cup B)} |\nabla q(x)|^2 e^{-\beta V(x)} dx,$$

subject to the condition

$$q(x) = 0, \ x \in \partial A,$$
$$q(x) = 1, \ x \in \partial B.$$

Here $Z = \int_{\Omega\setminus(A\cup B)} e^{-\beta V(x)} dx$.

Parameterization of q using a neural network

- To impose the BCs, we minimize the objective functional over functions of the form

$$q(x) = (1 - \chi_A(x)) \left[(1 - \chi_B(x)) \tilde{q}(x) + \chi_B(x) \right],$$

where $\chi_{A,B}(x)$ are indicator functions on A and B, resp.

- Paramterize $\tilde{q}$ using a feed-forward neural network:

$$q_\theta(x) = (1 - \chi_A(x)) \left[(1 - \chi_B(x)) \tilde{q}_\theta(x) + \chi_B(x) \right].$$

Hyperbolic tangent function and sigmoid function are used as the activation function in the hidden layers and output layer, resp.

Unsupervised learning problem

- The unconstrained learning problem:

$$\arg\min_{\theta} \frac{1}{Z} \int_{\Omega \setminus (A \cup B)} |\nabla_x q_\theta(x)|^2 e^{-\beta V(x)} dx.$$

or in the form of an expectation

$$\arg\min_{\theta} \mathrm{E}_{X \sim \rho} \left[|\nabla_x q_\theta(X)|^2 p_\beta(X) / \rho(X) \right],$$

where $p_\beta(x) = Z^{-1} e^{-\beta V(x)}$ and the expectation is taken w.r.t. the distribution ρ.

Unsupervised learning problem (cont'd)

Approximate the expectation by sample average:

$$\mathrm{E}_{X\sim\rho}\left[|\nabla_x q_\theta(X)|^2 \frac{p_\beta(X)}{\rho(X)}\right] \approx \frac{1}{N}\sum_{i=1}^{N} |\nabla_x q_\theta(X^{(i)})|^2 \frac{p_\beta(X^{(i)})}{\rho(X^{(i)})},$$

where $X^{(i)}$ are i.i.d. samples from ρ.

Choice of ρ ?

ρ should be chosen such that the transition state region, where q changes sharply from 0 to 1, is adequately sampled.

Sampling distribution

In the numerical examples, we use

- Equilibrium distribution at an artificial (higher) temperature

$$\rho(x) = \frac{1}{Z'} e^{-\beta' V(x)}$$

with $\beta' = 1/k_B T'$ and $T' > T$.

- Metadynamics

$$\rho(x) = \frac{1}{Z_G} e^{-\beta \left[V(x) + V_G(x) \right]}.$$

where V_G is superposition of localized Gaussian functions.

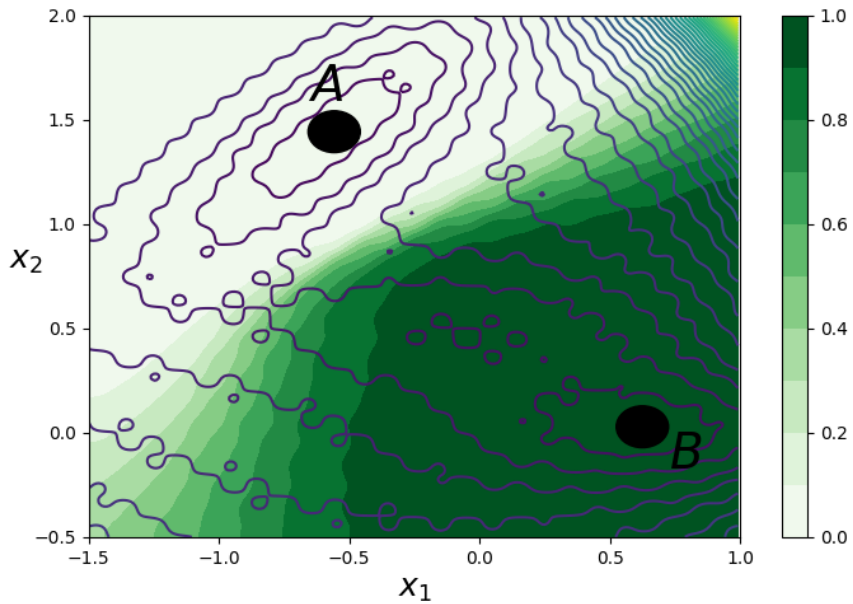
Example: Extended Mueller potential

- Consider the Mueller potential embedded in the 10-dimensional space

$$V(x) = V_m(x_1, x_2) + \frac{1}{2\sigma^2} \sum_{i=3}^{10} x_i^2, \quad x \in \mathbb{R}^{10}$$

where $V_m(x_1, x_2)$ is the rugged Mueller potential in two dimensions.

- "Exact" solution $q_m(x_1, x_2)$ at $k_B T = 10$ obtained by solving the backward equation using the finite element method:



Example: Extended Mueller potential

- Data are sampled at the temperature $k_B T' = 20$.

  Of these data, 70% and 30% are used as the training and validation dataset respectively.

- The neural network is fully connected, and the hyperbolic tangent function is used as the activation function in the hidden layers. **Sigmoid function** is used in the output layer.

- Use the package *Tensorflow* to train the network at the physical temperature $k_B T = 10$.

Example: Extended Mueller potential

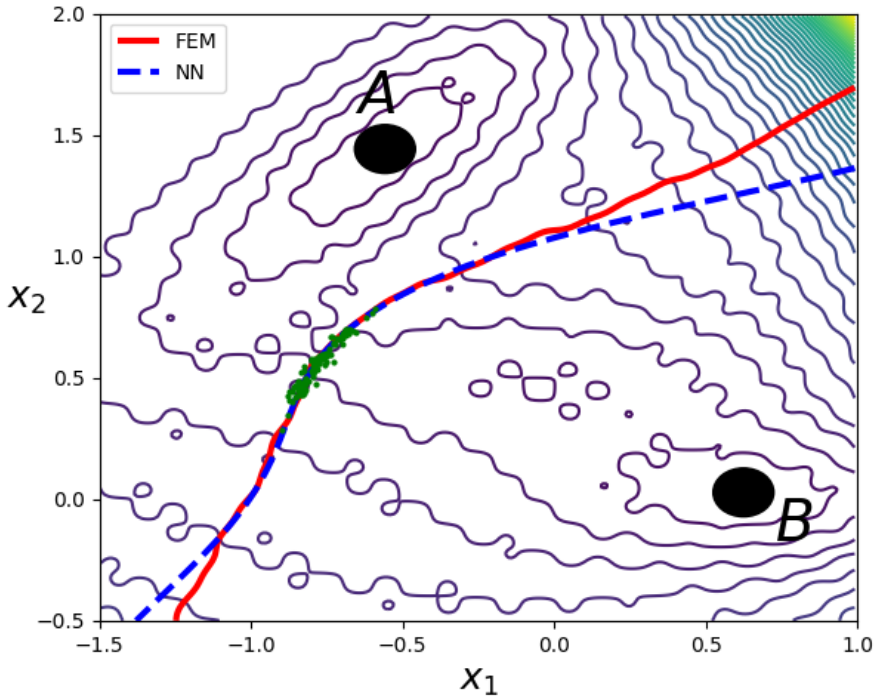
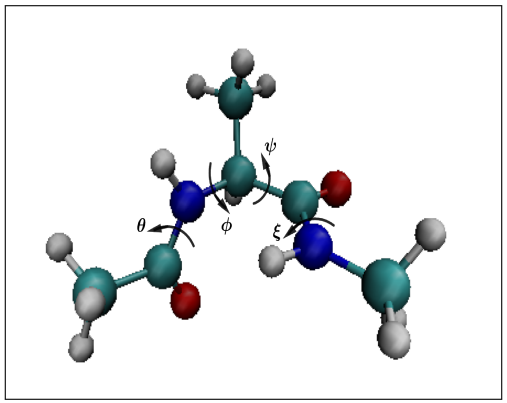


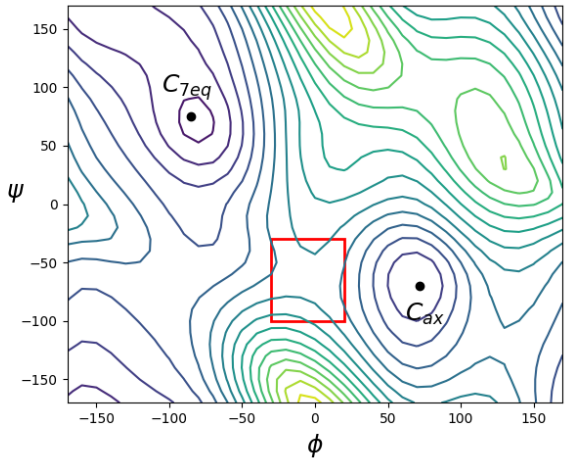
Figure: The $1/2$-isosurface of q_θ projected onto the (x_1, x_2) plane and 100 transition states sampled on the isosurface.

Example: Isomerization of Alanine dipeptide

- The molecule has two main metastable conformations C_{7eq} and C_{7ax}. The metastable sets A and B are chosen as

$$A = \{x : |(\phi(x), \psi(x)) - C_{7eq}| < 10^\circ\},$$
$$B = \{x : |(\phi(x), \psi(x)) - C_{7ax}| < 10^\circ\}.$$





- We study the isomerization process of the alanine dipeptide in vacuum at $T = 300\,\mathrm{K}$.

Example: Isomerization of Alanine dipeptide

Feature engineering:

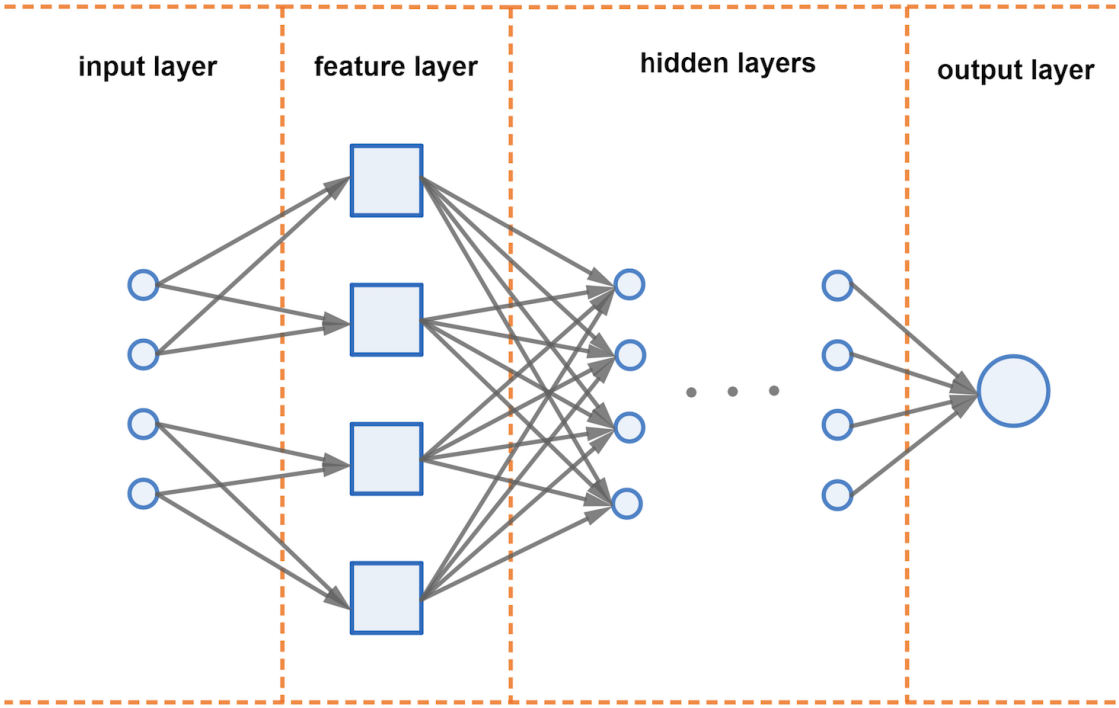
- For conformational changes of bio-molecules, very often only a few collective variables (e.g. torsion angles, bond angles), play a major role in the transition event.
- The committor function is well-approximated by a function of these collective variables

$$q(x) = f(z_1(x), ..., z_m(x)).$$

- We make use of collective variables in the design of the first input transformation layer of the network.

Example: Isomerization of Alanine dipeptide

Architecture of the neural network:



The feature layer consists of the sine and cosine of selected torsion angles.

Example: Isomerization of Alanine dipeptide

- Data sampling: 10^5 data points are sampled from metadynamics (biasing ϕ and ψ).

  70% and 30% of the data are used as the training and validation set, resp.

- Train the network at $T = 300\,\mathrm{K}$ using *Tensorflow*. The computation is terminated when the validation error no longer decreases.

Example: Isomerization of Alanine dipeptide

To check the accuracy of the numerical results,

- Sample 100 states on the $1/2$-isocommittor surface using the constrained Langevin dynamics at $T = 300\,\mathrm{K}$.
- Compute the "true" committor value using 200 trajectories initiated from each state with randomly chosen initial velocities.

Example: Isomerization of Alanine dipeptide

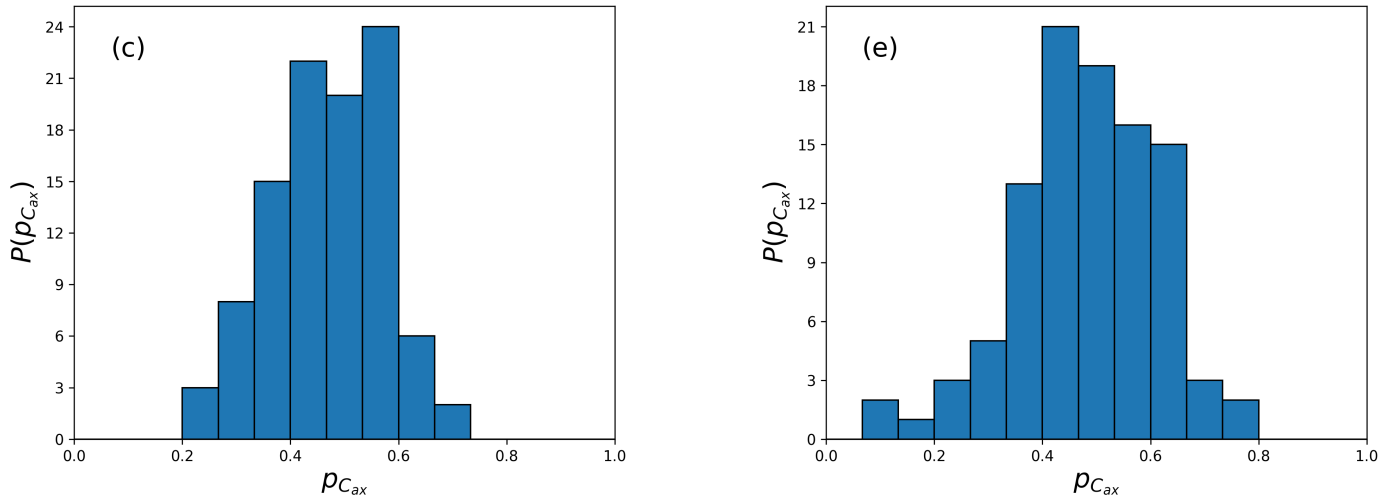


Figure: Distribution of the "true" committor values for the 100 states sampled on the $1/2$-isosurface of q_θ. The feature layer consists of the sines and cosines of 9 (left panel) and 41 (right panel) torsion angles.

Learning invariant distributions from noisy data

Consider a process $\mathbf{x}_t$ in the d-dimensional space modeled by the SDE:

$$d\mathbf{x}_t = \mathbf{f}(\mathbf{x}_t)dt + \sigma dW_t, \quad t > 0$$

where

- $f : R^d \to R^d$ is the force field
- W_t is a standard m dimensional Brownian motion; σ is a $d \times m$ matrix.

The invariant distribution

- The invariant distribution:

$$p(\mathbf{x}) = \lim_{T\to\infty} \frac{1}{T} \int_0^T \delta(\mathbf{x}_t - \mathbf{x}) dt.$$

- The Fokker-Planck equation:

$$-\nabla \cdot (\mathbf{f}(\mathbf{x})p(\mathbf{x})) + \epsilon \nabla \cdot (\bar{D}\nabla p(\mathbf{x})) = 0, \quad \mathbf{x} \in R^d$$

with the normalization condition

$$\int_{R^d} p(\mathbf{x}) d\mathbf{x} = 1.$$

Here $\epsilon\bar{D} = D = \sigma\sigma^T$, where $\|\bar{D}\| = 1$.

The generalized potential

- The generalized potential

$$V(\mathbf{x}) = -\epsilon \log p(\mathbf{x}).$$

- The governing equation for V:

$$\nabla V \cdot (\mathbf{f}(\mathbf{x}) + \bar{D}\nabla V) - \epsilon \nabla \cdot (\mathbf{f}(\mathbf{x}) + \bar{D}\nabla V) = 0$$

or equivalently

$$\mathbf{f}(\mathbf{x}) = -\bar{D}\nabla V(\mathbf{x}) + \mathbf{g}(\mathbf{x}),$$
$$\nabla V(\mathbf{x}) \cdot \mathbf{g}(\mathbf{x}) - \epsilon \nabla \cdot \mathbf{g}(\mathbf{x}) = 0.$$

Problem setup

- We only have partial knowledge of the dynamics:

  The dynamics is in the form of the SDE, but the force field f and the matrix σ are unknown.

- We have access to (possibly short) trajectories of the dynamics.
  - Given N trajectories: $x_i(t),\ i = 1, \ldots, N$.
  - Sample M pairs of states along each trajectory:

$$(X_i^{j,0}, X_i^{j,1}), \quad j = 1, \ldots, M$$

  where $X_i^{j,0} = x_i(t_j)$, $X_i^{j,1} = x_i(t_j + h)$, h is a small time step.

- **Goal**: Learn the force field f in the form of the decomposition and the diffusion matrix D, from the noisy data.

Parameterization of the force field

- Parameterized force field:

$$\mathbf{f}_\theta(\mathbf{x}) = -\bar{D}\nabla V_\theta(\mathbf{x}) + \mathbf{g}_\theta(\mathbf{x}).$$

  where V_θ and g_θ are feed-forward fully connected neural networks:

$$V_\theta(\mathbf{x}) = S_\theta(\mathbf{x}) + \boldsymbol{\rho} \cdot (\mathbf{x} - \mathbf{c})^2 := S_\theta(\mathbf{x}) + \sum_{i=1}^{d} \rho_i (x_i - c_i)^2.$$

  Activation function: hyperbolic tangent function

- Optimize the trainable parameters (θ, D) by minimizing a loss function (θ here includes $\boldsymbol{\rho}$ and $\boldsymbol{c}$).

Loss function

The loss function:

$$L = L_0(\theta, D; \mathcal{I}) + \lambda L_1(\theta, D; \mathcal{I})$$

- L_0 is the negate of the log-likelihood of the data set:

$$L_0(\theta, D; \mathcal{I}) = \frac{1}{2} \log \det(D) + \frac{h}{2|\mathcal{I}|} \sum_{(i,j)\in\mathcal{I}} \left\| Y_i^j - \mathbf{f}_\theta(X_i^{j,0}) \right\|_{D^{-1}}^2 .$$

- L_1 is the constraint for the decomposition of the force:

$$L_1(\theta, D; \mathcal{I}) = \frac{1}{|\mathcal{I}|} \sum_{(i,j)\in\mathcal{I}} \left| \nabla V_\theta(X_i^{j,0})^T \mathbf{g}_\theta(X_i^{j,0}) - \frac{1}{2} \|D\|_2 \nabla \cdot \mathbf{g}_\theta(X_i^{j,0}) \right|^2 .$$

Here $Y_i^j = (X_i(t_j + h) - X_i(t_j))/h$.

Algorirhm

Algorithm 1 (Training of the parameters θ and D).

Choose the parameter λ, the learning rates η and γ, and the number of training steps K.
Initialize the neural networks V_θ and $\mathbf{g}_\theta$, and set the vectors $\hat{\boldsymbol{\rho}} = \mathbf{c} = \mathbf{0}$.
Initialize the matrix $D = I_d$.
for k = 1 to K **do**
  Sample a small batch $\mathcal{B}$ from the dataset.
  Update the parameters θ using $\theta \leftarrow \theta - \eta \nabla_\theta L(\theta, D; \mathcal{B})$ or the Adam optimizer.
  Update the matrix D using

$$D \leftarrow (1-\gamma)D + \frac{\gamma h}{|\mathcal{B}|} \sum_{(i,j)\in\mathcal{B}} \left(Y_i^j - \mathbf{f}_\theta(X_i^{j,0}; D)\right)\left(Y_i^j - \mathbf{f}_\theta(X_i^{j,0}; D)\right)^T.$$

end for
Output: the force field model $\mathbf{f}_\theta$ and matrix $\tilde{D}$.

Example: a two-dimensional system

Consider the SDE in 2d:

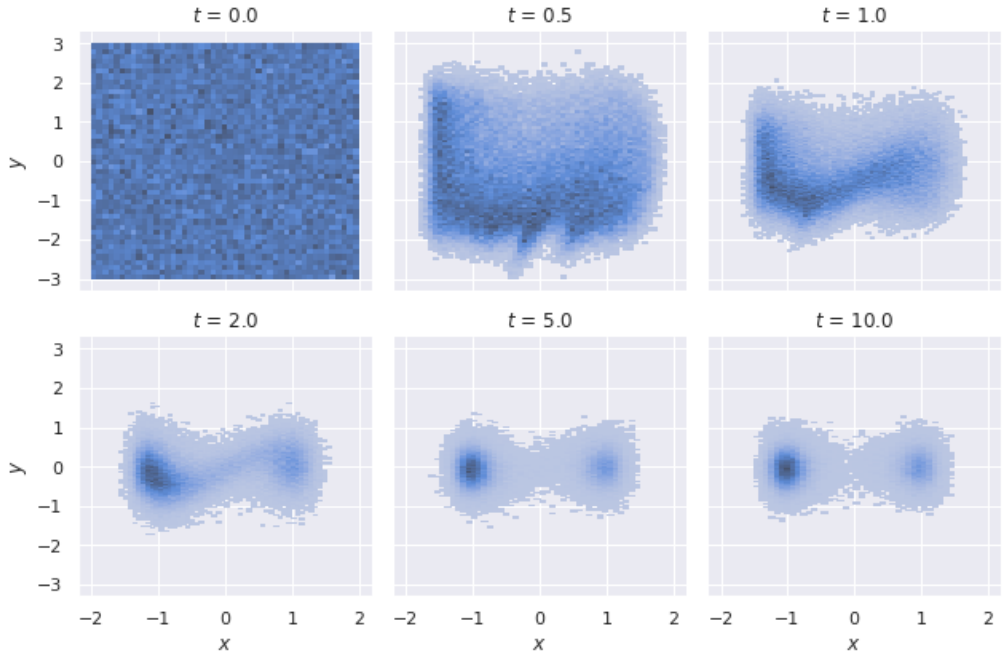
$$\begin{cases} \dot{x} = \frac{1}{5}x(1-x^2) + y(1+\sin x) + \sqrt{\frac{1}{50}}\,\xi_1, \\ \dot{y} = -y + 2x(1-x^2)(1+\sin x) + \sqrt{\frac{1}{5}}\,\xi_2. \end{cases}$$

Two metastable states: $x_a = (-1, 0)$, $x_b = (1, 0)$.

Example: a two-dimensional system

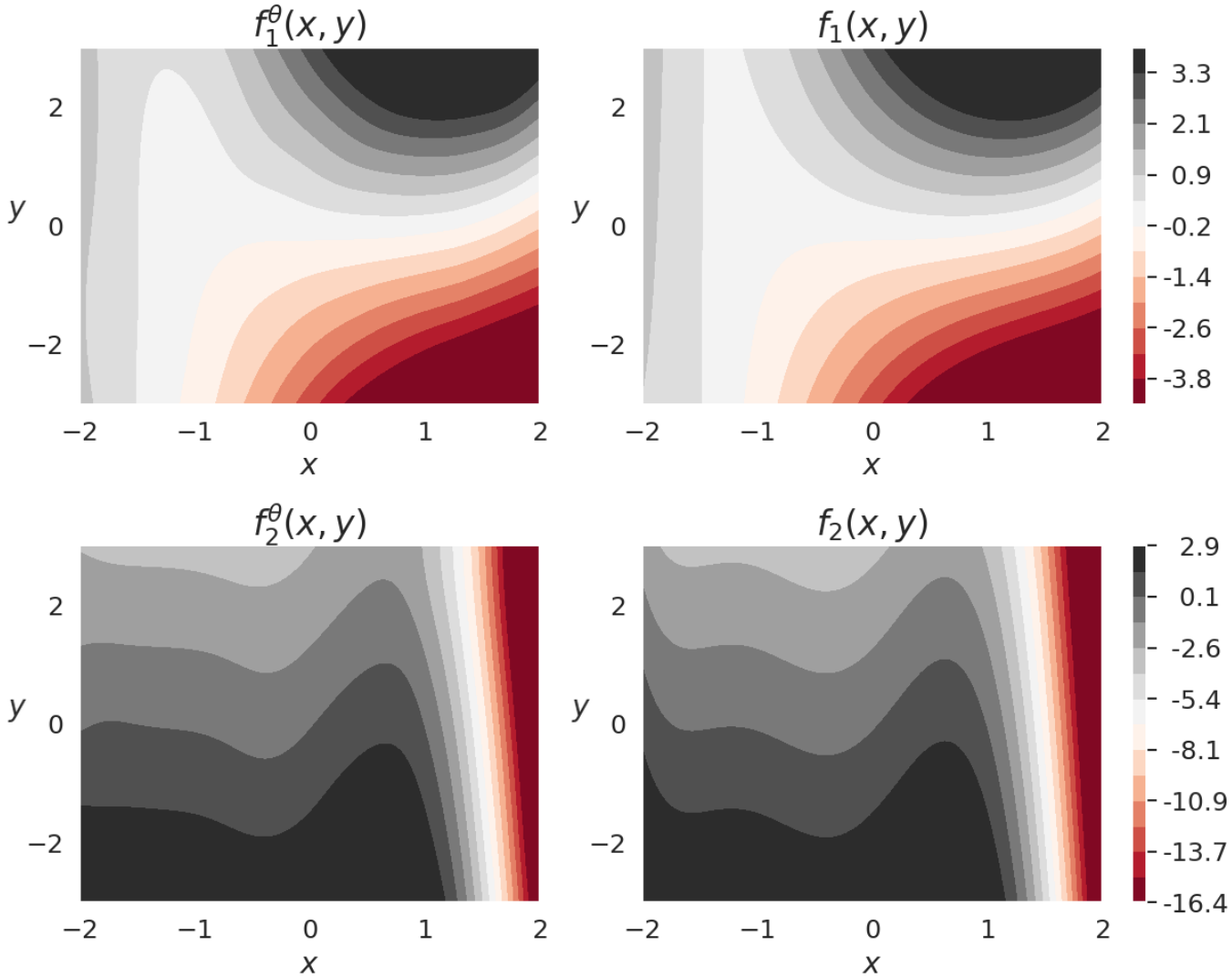
Sampling data from trajectories:

- Distribution of $N = 10^5$ initial states (upper-left) and its evolution following the dynamics.
- 10 pairs of data points are sampled from each trajectory during the transient period $[0, 1]$.
- neural network: two hidden layers with 50 nodes on each layer for both S_θ and $\mathbf{g}_\theta$.



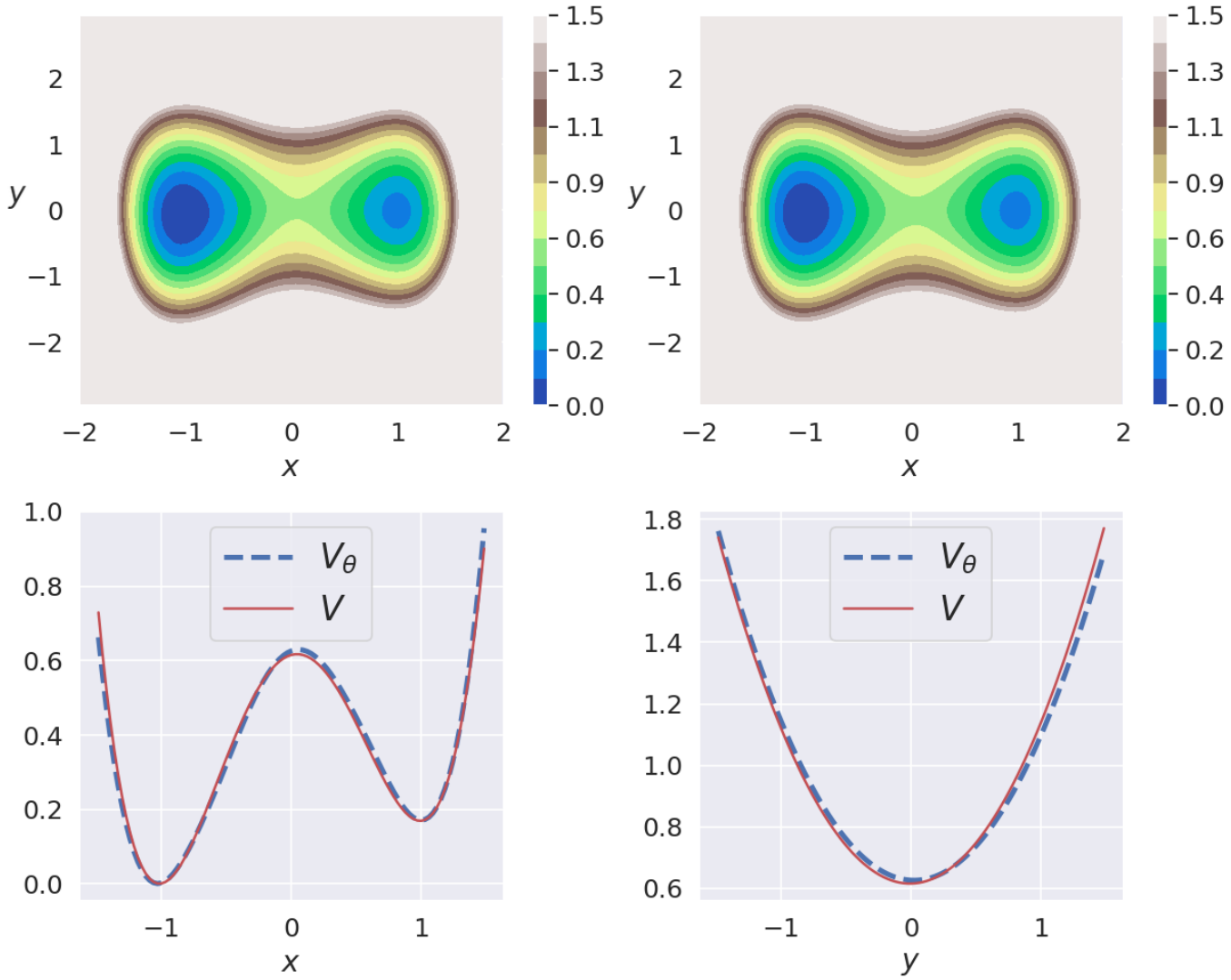
Example: a two-dimensional system

Learned force field (left) vs. true force field (right):



Example: a two-dimensional system

Learned generalized potential (left) vs. finite difference solution (right):



Example: a two-dimensional system

Learned diffusion matrix:

$$\tilde{D} = \begin{bmatrix} 1.989 \times 10^{-2} & 5.616 \times 10^{-4} \\ 5.616 \times 10^{-4} & 1.985 \times 10^{-1} \end{bmatrix}.$$

- The true diffusion matrix is $D = diag(1/50, 1/5)$.
- Relative error: 8.3×10^{-3}.

Example: Lorenz system

Dynamical equations:

$$\begin{cases} \dot{x} = \beta_1(y - x) + \sqrt{2\epsilon}\,\xi_1, \\ \dot{y} = x(\beta_2 - z) - y + \sqrt{2\epsilon}\,\xi_2, \\ \dot{z} = xy - \beta_3 z + \sqrt{2\epsilon}\,\xi_3, \end{cases}$$

where

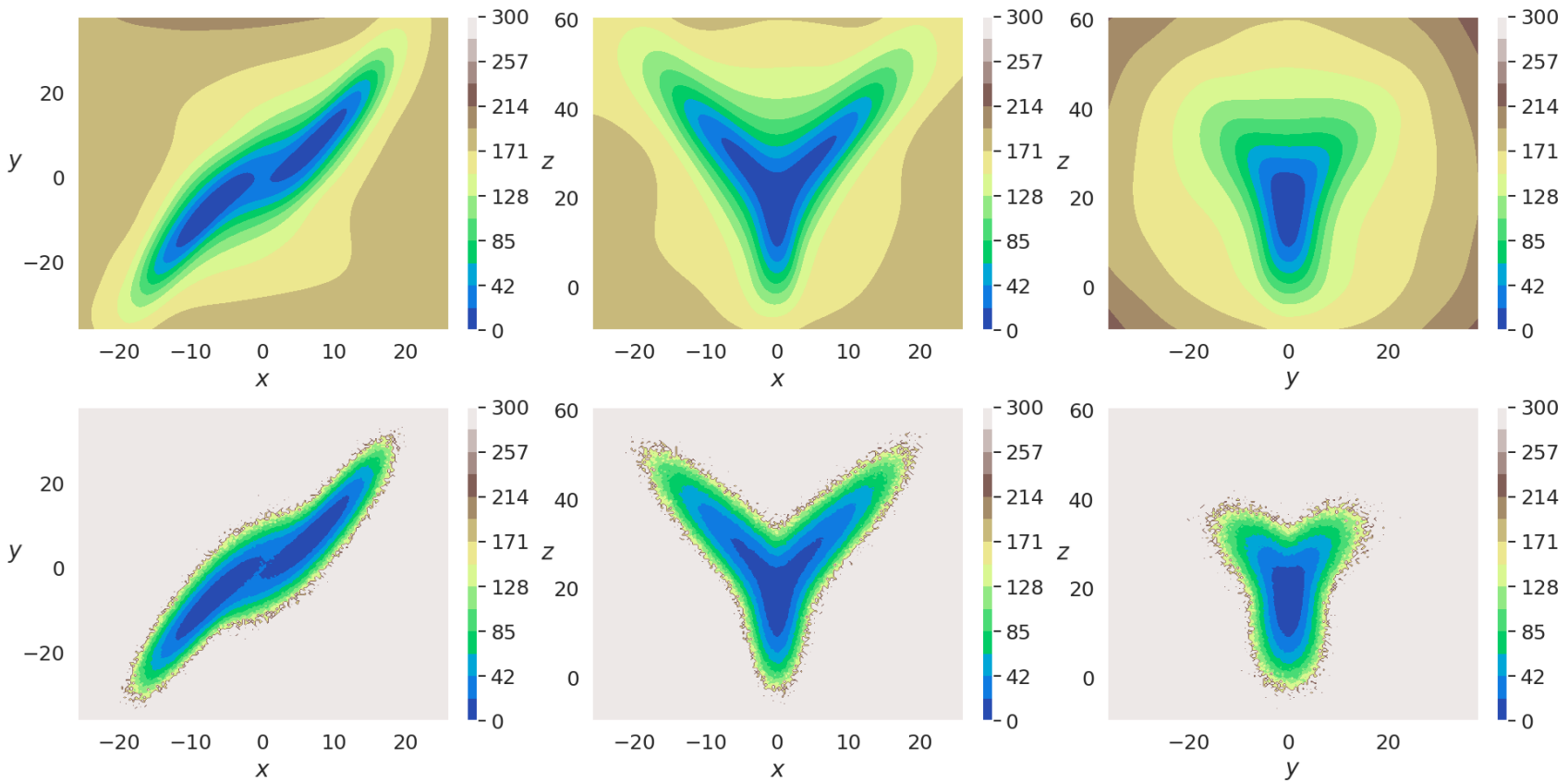
- $\xi = (\xi_1, \xi_2, \xi_3)$ is a 3d white noise.
- The diffusion matrix $D = 2\epsilon I_3$, $\epsilon = 20$.
- $\beta_1 = 10, \beta_2 = 28, \beta_3 = 8/3$.

Example: Lorenz system

- Data are sampled from $N = 10^4$ trajectories over the time interval $[0, 2]$.
- $M = 200$ pairs of data points are sampled from each trajectory.
- Neural networks: two hidden layers with 100 nodes on each layer for both S_θ and $\mathbf{g}_\theta$.

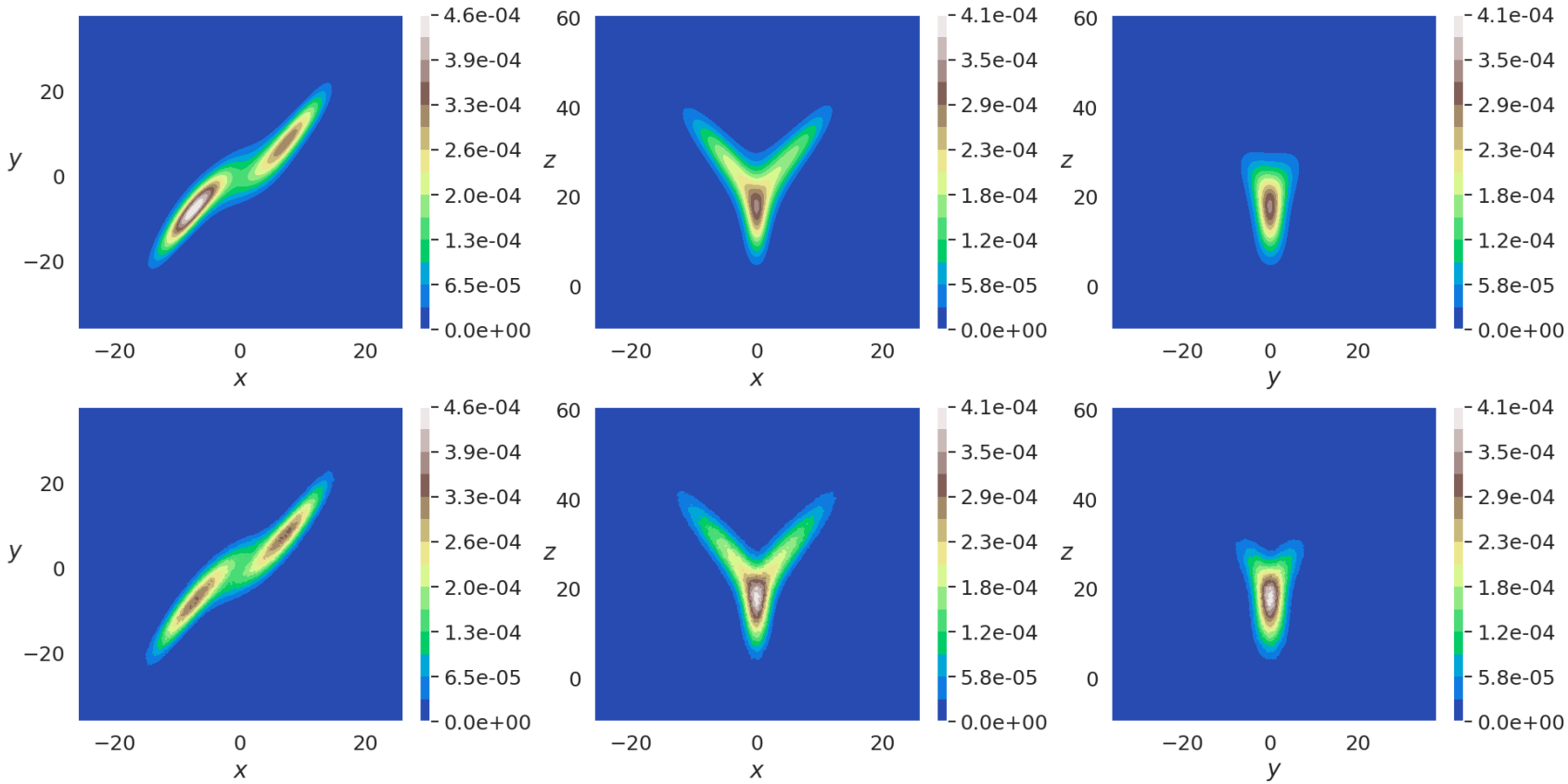
Example: Lorenz system

Learned potential V_θ (upper panels) vs. $V = -\epsilon \log p(x)$ (lower panels), where p is estimated using MC:



Example: Lorenz system

Learned distribution p_θ (upper panels) vs. MC solution (lower panels) :

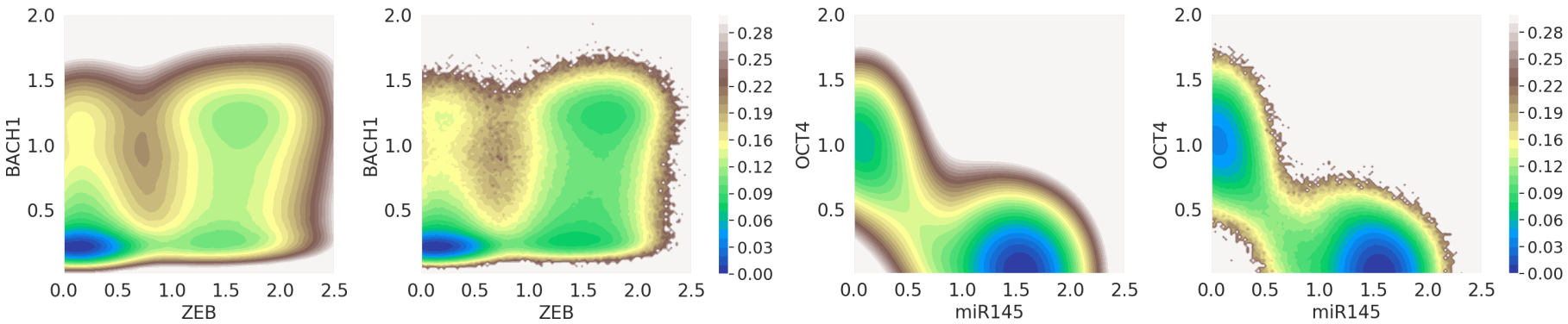


Example: EMT-metastasis network

- Ten-dim dynamical system, modelling a biological network.
- Earlier work used various methods to estimate the invariant distribution. For example, the distribution was estimated under the assumption that the different variables are independent.
- In the current method, the data are sampled from $N = 10^5$ trajectories, 36 pairs from each.
- Neural network: two hidden layers with 100 nodes on each layer for both S_θ and $\mathbf{g}_\theta$.

Example: EMT-metastasis network

The free energy landscape $V^{\theta}_{2,10}$ (left) and $V^{\theta}_{6,3}$ (right) vs. MC solutions:



Free energy:

$$V^{\theta}_{2,10}(x_2, x_{10}) = -\tilde{\epsilon} \log \int_{\mathbb{R}^8} \exp\left(-\frac{1}{\tilde{\epsilon}} V_{\theta}(x_1, \ldots, x_{10})\right) dx_1 dx_3 \cdots dx_9,$$

$$V_{2,10}(x_2, x_{10}) = -\epsilon \log \int_{\mathbb{R}^8} p(x_1, \ldots, x_{10}) dx_1 dx_3 \cdots dx_9.$$

Summary

- The method for the committor function combines deep learning with *data sampling* and *feature engineering* .

  It provides an alternative approach to study high-d complex systems with rough energy landscapes.

- For the problem of computing the invariant distribution, the dynamics is assumed to be in the form of SDEs, but the deterministic force field and the diffusion are both unknown.

  The method learns the force field, in particular the generalized potential, and the diffusion from noisy data sampled from short trajectories.

Summary

Ref: Q. Li, B. Lin and W. Ren, Computing committor functions for the study of rare events using deep learning, J. Chem. Phys. 151, 054112 (2019)

Ref: B. Lin, Q. Li and W. Ren Computing high-dimensional invariant distributions from noisy data , J. Comput. Phys. 474, 111783 (2023)