The fast committor machine: Interpretable prediction with kernels

David Aristoff,[1] Mats Johnson,[1] Gideon Simpson,[2] and Robert J. Webber[3]
[1)]*Mathematics, Colorado State University, Fort Collins, CO, USA*[a)]
[2)]*Mathematics, Drexel University, Philadelphia, PA, USA*
[3)]*Mathematics, University of California San Diego, La Jolla, CA, USA*

In the study of stochastic systems, the committor function describes the probability that a system starting from an initial configuration x will reach a set B before a set A. This paper introduces an efficient and interpretable algorithm for approximating the committor, called the "fast committor machine" (FCM). The FCM uses simulated trajectory data to build a kernel-based model of the committor. The kernel function is constructed to emphasize low-dimensional subspaces that optimally describe the A to B transitions. The coefficients in the kernel model are determined using randomized linear algebra, leading to a runtime that scales linearly in the number of data points. In numerical experiments involving a triple-well potential and alanine dipeptide, the FCM yields higher accuracy and trains more quickly than a neural network with the same number of parameters. The FCM is also more interpretable than the neural net.

I. INTRODUCTION

Many physical systems exhibit metastability, which is the tendency to occupy a region A of phase space for a comparatively long time before a quick transition to another region B. Metastability is especially common in molecular dynamics, where states A and B might correspond to the folded and unfolded configurations of a protein. Metastability also arises in continuum mechanics[1], biological systems[2], fluids[3], and the Earth's climate[4].

Scientists can gain insight into metastable systems by numerically simulating and then analyzing the transition paths from A to B. The transition paths can be simulated by a variety of methods including transition path sampling[5,6], forward flux sampling[7,8], the string method[9,10], and adaptive multilevel splitting[11]. This work assumes that a data set of trajectories has already been simulated, including one or more transitions between A and B. A more detailed discussion of sampling methods appears in the conclusion.

Scientists can study the transitions from A to B by evaluating the (forward) committor function[3,4,12–18], which measures the probability that the system starting at state $\boldsymbol{x}$ will reach B before A, thus making a transition. In symbols, the committor is given by

$$q^*(\boldsymbol{x}) = \mathbb{P}_{\boldsymbol{x}}(T_B < T_A),$$

where T_A and T_B are the first hitting times for A and B.

To numerically approximate the committor function, the trajectory data needs to be combined with an appropriate numerical method. Beyond the direct counting of A and B transitions, there are now various machine learning methods available, including neural networks[12–14], diffusion maps[15], Markov state models[19–21], and other tools[16]. As shortcomings, these methods can be slow to apply to large data sets (diffusion maps) or difficult to interpret (neural nets).

This work describes a new method called the fast committor machine (FCM) that is both efficient and interpretable. The FCM approximates the committor q^* using a linear combination of kernel functions, as follows:

- The kernel function is defined in an interpretable way, as the composition of an exponential kernel with an adaptively chosen linear map. The linear map is generated by the recursive feature machine (RFM[22]). The map emphasizes the low-dimensional subspaces that make the greatest contribution to the gradient of the committor and suppresses other subspaces.
- The coefficients in the kernel approximation are optimized using randomly pivoted Cholesky (RPC[23]). RPC is a randomized linear algebra approach that processes N data points using just $\mathscr{O}(N)$ floating point operations and $\mathscr{O}(N)$ storage.

The RFM and RPC were introduced in 2023–2024 and appear to be new in the chemical physics literature. Adapting these techniques to the committor problem requires going beyond the original papers[22,23] and introducing a new functional form based on a variational formulation of the committor.

In summary, the FCM is a method for approximating the committor that is based on an adaptively chosen linear transformation of the phase space. In numerical experiments, the method is more accurate and trains more quickly than a neural network with the same number of parameters. As another advantage, the FCM is interpretable, revealing low-dimensional linear subspaces that optimally describe the transition pathways from A to B.

A. Relationship to past work

Kernel methods are frequently used in chemical machine learning[24–26], especially for calculating force fields[27–30]. Yet the efficiency of the kernel methods depends on finding a suitable distance function between molecular configurations[31].

Two distances are currently being advocated in the chemical physics literature. Diffusion maps[32] provide a sophisticated distance that adapts to the nonlinear manifold structure in the input data, and diffusion maps have recently been used to approximate the committor[15]. Alternatively, the variational approach to conformational dynamics[33] identifies a

[a)]Author to whom correspondence should be addressed: aristoff@colostate.edu

coordinate transformation that amplifies slowly decorrelating dynamical modes, and VAC is often used to help visualize and construct committor estimates[34–36].

As a limitation, diffusion maps and VAC do not adapt to the sets A and B used in the committor definition. Greater accuracy could be potentially obtained by modifying the distance based on A and B. However, this modification requires a fully adaptive learning strategy that redefines the kernel distance for each committor problem.

The kernel machine learning literature has recently developed a flexible approach to distance construction. Many authors apply a *linear* transformation to their data before calculating a standard exponential or square exponential kernel[37,38]. The linear map can be adapted to the input data and also to the specific approximation task[39,40]. A 2024 paper[22] introduced the *recursive feature machine* (RFM), which constructs the linear feature map based on the estimated gradients of the target function. To justify the RFM, the authors compared the linear map with the first layer of a neural net, and they showed empirical and theoretical similarities[22,41,42]. In experiments, they applied the RFM to RGB images with as many as 96×96 pixels, hence $96 \times 96 \times 3 = 27,648$ dimensions. They found that the RFM suppresses irrelevant directions and emphasizes important directions, which is especially important in high dimensions. For further justification of the RFM's effectiveness, see Appendix B.

The RFM provides the template for a kernel-based approximation of the committor. Yet optimizing the coefficients in the kernel approximation remains computationally demanding. A naive optimization would require $\mathcal{O}(N^3)$ operations and $\mathcal{O}(N^2)$ storage, which is prohibitively expensive for $N \geq 10^5$ data points. The high computational cost of kernel methods has been described as a fundamental limitation in the past[43]. Nonetheless, modern strategies in randomized numerical linear algebra are leading to dramatic speed-ups[23,44–46].

This work applies randomly pivoted Cholesky (RPC[23]) to speed up the kernel optimization and extend the numerical experiments to $N = 10^6$ data points. To achieve these speed-ups, RPC generates a randomized rank-r approximation of the kernel matrix, where r is a parameter chosen by the user. The coefficients can be optimized in just $\mathcal{O}(Nr^2)$ operations and $\mathcal{O}(Nr)$ storage. For the experiments in Sec. IV, a constant value of $r = 1000$ yields nearly converged committor results.

B. Outline for the paper

The rest of the paper is organized as follows. Section II provides context for the committor approximation task, Section III describes the new FCM method for calculating the committor, Section IV presents numerical experiments, and Section V offers concluding remarks. Technical derivations are in Appendices A and B.

TABLE I: Definitions of symbols used in this work.

Symbol	Definition
$\boldsymbol{x}$, $\boldsymbol{x}'$	system states
A, B	initial and target sets
Ω	complement of $A \cup B$
$\boldsymbol{x}(t)$	underlying stochastic process
τ	lag time
$q^*(\boldsymbol{x})$	exact committor
$\boldsymbol{\theta}$	linear coefficients in kernel approximation
$q_{\boldsymbol{\theta}}(\boldsymbol{x})$	estimated committor
$k_{\boldsymbol{M}}(\boldsymbol{x}, \boldsymbol{x}')$	kernel function
$\boldsymbol{M}$	scaling matrix
$\boldsymbol{K}$	kernel matrix
$\boldsymbol{I}$	identity matrix
$(\boldsymbol{x}_n, \boldsymbol{y}_n)$	time lagged pairs, $\boldsymbol{x}(0) = \boldsymbol{x}_n$, $\boldsymbol{x}(\tau) = \boldsymbol{y}_n$
N	number of training samples
μ	equilibrium density
ρ	sampling density
$w_n = \mu(\boldsymbol{x}_n)/\rho(\boldsymbol{x}_n)$	weight or likelihood of $\boldsymbol{x}_n$
$1_A, 1_B, 1_\Omega$	characteristic functions of A, B, Ω
ε	bandwidth parameter
γ	regularization parameter

C. Assumptions and notation

The FCM can be applied to any time-reversible Markovian system $\boldsymbol{x}(t) \in \mathbb{R}^d$ which has a discrete time step τ and equilibrium density μ. See Table I for a list of symbols and definitions.

II. CONTEXT

The FCM is motivated by a variational principle which states that the committor $q^*(\boldsymbol{x}) = \mathbb{P}_{\boldsymbol{x}}(T_B < T_A)$ is the unique minimizer of

$$\mathscr{L}(q) = \mathbb{E}_\mu \big| q(\boldsymbol{x}(0)) - q(\boldsymbol{x}(\tau)) \big|^2, \tag{1}$$

among square-integrable functions satisfying $q = 0$ on A and $q = 1$ on B. The expectation $\mathbb{E}_\mu$ is an average over all trajectories $\boldsymbol{x}(t)$ that are started from the equilibrium density $\boldsymbol{x}(0) \sim \mu$ and run forward to an end point $\boldsymbol{x}(\tau)$. The functional (1) is known as the discrete-in-time Dirichlet form, and it appears frequently in Markov chain analysis[47–50]. Here, the Dirichlet form serves as a cost function for committor estimation in machine learning[34,35].

Frequently, the Dirichlet form needs to be approximated from data. Assume the data consists of N pairs $(\boldsymbol{x}_n, \boldsymbol{y}_n)_{n=1,2,\ldots,N}$, where the start point $\boldsymbol{x}_n$ is randomly sampled from a density ρ and the end point $\boldsymbol{y}_n$ is the result of running the model forward for τ time units. Then, a data-driven approximation for $\mathscr{L}(q)$ is the weighted average

$$L(q) = \frac{1}{N} \sum_{n=1}^{N} w_n \big| q(\boldsymbol{x}_n) - q(\boldsymbol{y}_n) \big|^2, \tag{2}$$

where the weights are the likelihood ratios between the sam-

pling density ρ and the target density μ:

$$w_n = \frac{\mu(\boldsymbol{x}_n)}{\rho(\boldsymbol{x}_n)}. \tag{3}$$

The weighted average (2) is mathematically justified. If the data pairs $(\boldsymbol{x}_n, \boldsymbol{y}_n)_{n=1,2,\ldots}$ are ergodic and μ is *absolutely continuous* with respect to ρ:

$$\rho(\boldsymbol{x}) > 0 \quad \text{whenever} \quad \mu(\boldsymbol{x}) > 0,$$

then the estimator (2) is unbiased

$$\mathbb{E}[L(q)] = \mathscr{L}(q),$$

and the law of large numbers guarantees $L(q) \to \mathscr{L}(q)$ as $N \to \infty$. Nonetheless, consistent with the law of large numbers, the accuracy of the weighted average may depend on generating a large quantity of data.

When the goal is variational minimization of $L(q)$, there is an option to use alternative weights

$$w_n' = c\frac{\mu(\boldsymbol{x}_n)}{\rho(\boldsymbol{x}_n)}, \tag{4}$$

where the multiplicative constant $c > 0$ can be any positive number based on convenience. This makes the variational approach applicable even when the likelihood ratios $\mu(\boldsymbol{x}_n)/\rho(\boldsymbol{x}_n)$ are only known up to a constant multiplicative factor.

III. NEW FCM METHOD FOR CALCULATING THE COMMITTOR

This section describes the new method for calculating the committor function q^*, called the "fast committor machine" (FCM).

A. Form of the committor approximation

Recall that $(\boldsymbol{x}_n, \boldsymbol{y}_n)_{n=1,2,\ldots,N}$ is a sequence of simulated data pairs. The FCM is based on a new data-driven committor approximation of the form

$$\begin{aligned} q_{\boldsymbol{\theta}}(\boldsymbol{x}) &= 0, & \boldsymbol{x} &\in A, \\ q_{\boldsymbol{\theta}}(\boldsymbol{x}) &= 1, & \boldsymbol{x} &\in B, \\ q_{\boldsymbol{\theta}}(\boldsymbol{x}) &= \sum_{n=1}^{N} \theta_n \big[k_{\boldsymbol{M}}(\boldsymbol{x}_n, \boldsymbol{x}) - k_{\boldsymbol{M}}(\boldsymbol{y}_n, \boldsymbol{x})\big], & \boldsymbol{x} &\in \Omega. \end{aligned} \tag{5}$$

In this definition, $\Omega = (A \cup B)^c$ is the region of unknown committor values, $\boldsymbol{\theta} \in \mathbb{R}^N$ is a vector of coefficients, and

$$k_{\boldsymbol{M}}(\boldsymbol{x}, \boldsymbol{x}') = \begin{cases} \exp\big(-\frac{1}{\varepsilon}\|\boldsymbol{M}^{1/2}(\boldsymbol{x} - \boldsymbol{x}')\|\big), & \boldsymbol{x}, \boldsymbol{x}' \in \Omega \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

is a kernel function with a bandwidth parameter $\varepsilon > 0$ and a positive definite scaling matrix $\boldsymbol{M} \in \mathbb{R}^{d \times d}$.

The approximation $q_{\boldsymbol{\theta}}$ is more parameter-efficient and empirically accurate than other kernel-based parametrizations considered in Appendix A. By construction, $q_{\boldsymbol{\theta}}$ satisfies the appropriate boundary conditions. Also, $q_{\boldsymbol{\theta}}$ changes flexibly in the interior region Ω. Note that the kernel $k_{\boldsymbol{M}}(\boldsymbol{x}, \boldsymbol{y})$ is not differentiable at $\boldsymbol{x} = \boldsymbol{y}$, but this paper uses $\nabla k_{\boldsymbol{M}}(\boldsymbol{x}, \boldsymbol{x}) = \boldsymbol{0}$ as the pseudogradient.

The rest of this section describes the approach for optimizing the scaling matrix $\boldsymbol{M}$, the coefficient vector $\boldsymbol{\theta}$, and the bandwidth ε in the FCM.

B. Optimization of the scaling matrix M

The FCM incorporates a scaling matrix $\boldsymbol{M} \in \mathbb{R}^{d \times d}$ that transforms the state variable $\boldsymbol{x} \in \mathbb{R}^d$ according to $\boldsymbol{x} \mapsto \boldsymbol{M}^{1/2}\boldsymbol{x}$. The optimization of this scaling matrix is key to the performance of the FCM.

Since the early 2000s, researchers have suggested a scaling matrix chosen as the inverse covariance matrix of the input data points[37,38]. This choice of $\boldsymbol{M}$ transforms the data points so they become isotropic. Isotropy can sometimes improve the predictive accuracy, but there is a better approach to optimally select $\boldsymbol{M}$ for kernel learning.

Radhakrishnan et al. recently introduced an approach for tuning the scaling matrix called the "recursive feature machine" (RFM)[22]. The RFM is a generalized kernel method that takes input/output pairs $(\boldsymbol{x}_n, b_n = f(\boldsymbol{x}_n))_{n=1,2,\ldots,N}$ and iteratively learns both a regressor

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{n=1}^{N} \theta_n k_{\boldsymbol{M}}(\boldsymbol{x}_n, \boldsymbol{x}) \tag{7}$$

and a scaling matrix M. The RFM alternates between two steps:

- Update the coefficient vector $\boldsymbol{\theta}$ so that $f_{\boldsymbol{\theta}}$ minimizes the least-squares loss

  $$\boldsymbol{\theta} \leftarrow \arg\min_{\boldsymbol{\theta}} \frac{1}{N}\sum_{n=1}^{N} |f_{\boldsymbol{\theta}}(\boldsymbol{x}_n) - b_n|^2$$

  using the current scaling matrix $\boldsymbol{M}$.

- Update the scaling matrix

  $$\boldsymbol{M} = \sum_{n=1}^{N} \nabla f_{\boldsymbol{\theta}}(\boldsymbol{x}_n) \nabla f_{\boldsymbol{\theta}}(\boldsymbol{x}_n)^T. \tag{8}$$

  using the current regressor $f_{\boldsymbol{\theta}}$.

The method iterates until finding an approximate fixed point for $f_{\boldsymbol{\theta}}$ and $\boldsymbol{M}$, and typically $3-6$ iterations are enough[22].

A complete analysis of the RFM lies beyond the scope of the current work. However, as an intuitive explanation, the linear map improves the "fit" between the target function and the approximation. Specifically, the change-of-basis $\boldsymbol{z} = \boldsymbol{M}^{1/2}\boldsymbol{x}$ causes $f_{\boldsymbol{\theta}}$ to have isotropic gradients $\nabla_{\boldsymbol{z}} f_{\boldsymbol{\theta}}(\boldsymbol{z}_1), \ldots, \nabla_{\boldsymbol{z}} f_{\boldsymbol{\theta}}(\boldsymbol{z}_N)$. Thus, it becomes relatively easy to

Algorithm 1: Fast committor machine

Data: Training points: $(\boldsymbol{x}_n, \boldsymbol{y}_n, w_n)_{n=1,\dots,N}$, ;
Solution vector: $\boldsymbol{b}$;
Kernel function: k;
Parameters: bandwidth ε, regularization γ, approximation rank r (multiple of 10);
Result: Scaling matrix $\boldsymbol{M}$, committor estimate $q_{\boldsymbol{\theta}}$;
$\boldsymbol{M} \leftarrow \boldsymbol{I}$;
  // Initialize $\boldsymbol{M}$
for $t = 1, \dots, 5$ **do**
  $\boldsymbol{M} \leftarrow \boldsymbol{M} / \operatorname{tr} \operatorname{cov}(\{\boldsymbol{M}^{1/2}\boldsymbol{x}_1, \dots, \boldsymbol{M}^{1/2}\boldsymbol{x}_N\})$;
    // Rescale $\boldsymbol{M}$
  $k_{mn} \leftarrow k_{\boldsymbol{M}}(\boldsymbol{x}_m, \boldsymbol{x}_n) - k_{\boldsymbol{M}}(\boldsymbol{x}_m, \boldsymbol{y}_n) - k_{\boldsymbol{M}}(\boldsymbol{y}_m, \boldsymbol{x}_n) + k_{\boldsymbol{M}}(\boldsymbol{y}_m, \boldsymbol{x}_n)$;
    // Repopulate $\boldsymbol{K}$ with kernel $k_{\boldsymbol{M}}$
  $k_{mn} \leftarrow \sqrt{w_m w_n}\, k_{mn}$;
    // Reweight $\boldsymbol{K}$
  $S = \{s_1, \dots, s_r\} \leftarrow \texttt{RPCholesky}(\boldsymbol{K}, r)$;
    // Apply Alg. 2
  $\boldsymbol{K}(S,S) \leftarrow \boldsymbol{K}(S,S) + \varepsilon_{\text{mach}} \operatorname{tr}(\boldsymbol{K}(S,S))\boldsymbol{I}$;
    // Regularize
  $\boldsymbol{\eta} \leftarrow$ Solution to $[\boldsymbol{K}(:,S)^T \boldsymbol{K}(:,S) + \gamma N \boldsymbol{K}(S,S)]\boldsymbol{\eta} = \boldsymbol{K}(:,S)^T \boldsymbol{b}$;
    // Optimize coefficients
  $q_{\boldsymbol{\theta}}(\boldsymbol{x}) \leftarrow \sum_{i=1}^{r} \eta_i \sqrt{w_{s_i}} \big[k_{\boldsymbol{M}}(\boldsymbol{x}_{s_i}, \boldsymbol{x}) - k_{\boldsymbol{M}}(\boldsymbol{y}_{s_i}, \boldsymbol{x})\big]$;
    // Update $q_{\boldsymbol{\theta}}$
  $\boldsymbol{M} \leftarrow \sum_{n=1}^{N} \nabla q_{\boldsymbol{\theta}}(\boldsymbol{x}_n) \nabla q_{\boldsymbol{\theta}}(\boldsymbol{x}_n)^T$;
    // Update $\boldsymbol{M}$
end
return $\boldsymbol{M}$, $q_{\boldsymbol{\theta}}$

approximate f using a linear combination of isotropic kernel functions. See Appendix B for a proof of the isotropy property of the RFM.

The main contributions of this work are the extension of the RFM to the committor problem, together with an efficient strategy for optimizing the coefficient vector $\boldsymbol{\theta}$. In homage to the original RFM paper[22], the new method is called the "fast committor machine" (FCM).

Pseudocode for the FCM is provided in Algorithm 1. As a helpful feature, the pseudocode normalizes the scaling matrix $\boldsymbol{M}$ by the trace of the covariance matrix

$$\operatorname{tr} \operatorname{cov}(\{\boldsymbol{M}^{1/2}\boldsymbol{x}_1, \dots, \boldsymbol{M}^{1/2}\boldsymbol{x}_N\}) = \frac{1}{2N(N-1)} \sum_{m,n=1}^{N} \|\boldsymbol{M}^{1/2}(\boldsymbol{x}_m - \boldsymbol{x}_n)\|^2.$$

This makes it easier to select and interpret the bandwidth parameter $\varepsilon > 0$. For the experiments in Sec. IV, setting $\varepsilon = 1$ works well as a default.

C. Optimization of the coefficients θ_n

To derive an efficient procedure for optimizing the coefficient vector $\boldsymbol{\theta} \in \mathbb{R}^N$, the first step is to rewrite the optimization as a standard least-squares problem. Define the rescaled coefficients

$$\overline{\theta}_n = \theta_n / \sqrt{w_n}, \quad n = 1, \dots, N.$$

Also, introduce the kernel matrix $\boldsymbol{K} \in \mathbb{R}^{N \times N}$ and the solution vector $\boldsymbol{b} \in \mathbb{R}^N$ with entries given by

$$\begin{aligned} k_{mn} &= \sqrt{w_m}\sqrt{w_n}\big[k_{\boldsymbol{M}}(\boldsymbol{x}_m, \boldsymbol{x}_n) - k_{\boldsymbol{M}}(\boldsymbol{x}_m, \boldsymbol{y}_n) \\ &\qquad - k_{\boldsymbol{M}}(\boldsymbol{y}_m, \boldsymbol{x}_n) + k_{\boldsymbol{M}}(\boldsymbol{y}_m, \boldsymbol{y}_n)\big], \\ b_n &= \sqrt{w_n}[1_B(\boldsymbol{y}_n) - 1_B(\boldsymbol{x}_n)] \end{aligned}$$

Plugging the FCM into the variational principle (2), the optimal vector $\overline{\boldsymbol{\theta}} \in \mathbb{R}^N$ is the minimizer of the regularized least-squares loss:

$$\min_{\overline{\boldsymbol{\theta}} \in \mathbb{R}^N} \tfrac{1}{N} \|\boldsymbol{K}\overline{\boldsymbol{\theta}} - \boldsymbol{b}\|^2 + \gamma \overline{\boldsymbol{\theta}}^T \boldsymbol{K} \overline{\boldsymbol{\theta}}. \tag{9}$$

Notice that the loss function (9) includes a regularization term $\gamma \overline{\boldsymbol{\theta}}^T \boldsymbol{K} \overline{\boldsymbol{\theta}}$ with $\gamma > 0$ that shrinks the norm of the coefficients to help prevent overfitting. Specifically, $(\overline{\boldsymbol{\theta}}^T \boldsymbol{K} \overline{\boldsymbol{\theta}})^{1/2}$ is the reproducing kernel Hilbert space (RKHS) norm associated with the committor approximation $q_{\boldsymbol{\theta}}$. RKHS norms appear frequently in the kernel literature due to their theoretical properties and computational convenience; see Sec. 2.3 of the paper [51] for an introduction.

For large data sets with $N \geq 10^5$ data points, it is computationally convenient to use a randomized strategy for solving (9). The randomly pivoted Cholesky algorithm (Algorithm 2) selects a set of "landmark" indices

$$S = \{s_1, s_2, \dots, s_r\}.$$

Then, the coefficient vector $\overline{\boldsymbol{\theta}} \in \mathbb{R}^N$ is restricted to satisfy $\overline{\theta}_i = 0$ for all $i \notin S$ and

$$\overline{\boldsymbol{\theta}}(S) = \boldsymbol{\eta}.$$

Last, the vector $\boldsymbol{\eta} \in \mathbb{R}^r$ is optimized by solving

$$\min_{\boldsymbol{\eta} \in \mathbb{R}^r} \tfrac{1}{N} \|\boldsymbol{K}(:,S)\boldsymbol{\eta} - \boldsymbol{b}\|^2 + \gamma \boldsymbol{\eta}^T \boldsymbol{K}(S,S)\boldsymbol{\eta},$$

which is equivalent to the linear system

$$[\boldsymbol{K}(:,S)^T \boldsymbol{K}(:,S) + \gamma N \boldsymbol{K}(S,S)]\,\boldsymbol{\eta} = \boldsymbol{K}(:,S)^T \boldsymbol{b}.$$

It takes just $\mathscr{O}(Nr^2)$ operations to form and solve this linear system by a direct method. Moreover, there is no need to generate the complete kernel matrix $\boldsymbol{K} \in \mathbb{R}^{N \times N}$; it suffices to generate the r-column submatrix $\boldsymbol{K}(:,S)$, and the storage requirements are thus $\mathscr{O}(Nr)$.

D. Optimization of the hyperparameters

The last parameters to optimize are the bandwidth $\varepsilon > 0$, the regularization $\gamma > 0$, and the approximation rank r. To select these parameters, a simple but effective approach is a grid search. In the grid search, 20% of the data is set aside

Algorithm 2: RPCholesky[23]

Data: Formula for looking up entries of $\boldsymbol{K}$, approximation rank r (multiple of 10);
Result: Index set S;
Initialize: $\boldsymbol{F} \leftarrow \boldsymbol{0}$, $S \leftarrow \emptyset$, $T \leftarrow r/10$, and $\boldsymbol{d} \leftarrow \mathrm{diag}(\boldsymbol{K})$;
// Initialize parameters
for $i = 0$ to 9 **do**
  $s_{iT+1}, \dots, s_{iT+T} \sim \boldsymbol{d} / \sum_{j=1} d_j$;
  // Randomly sample indices
  $S' \leftarrow \texttt{Unique}(\{s_{iT+1}, \dots, s_{iT+T}\})$;
  // Identify unique indices
  $S \leftarrow S \cup S'$;
  // Add new indices to landmark set
  $\boldsymbol{G} \leftarrow \boldsymbol{K}(:, S') - \boldsymbol{F}\boldsymbol{F}(S', :)^*$;
  // Evaluate new columns
  $\boldsymbol{G}(S', :) \leftarrow \boldsymbol{G}(S', :) + \varepsilon_{\mathrm{mach}} \mathrm{tr}(\boldsymbol{G}(S', :))\boldsymbol{I}$;
  // Regularize
  $\boldsymbol{R} \leftarrow \texttt{Cholesky}(\boldsymbol{G}(S', :))$;
  // Upper triangular Cholesky factor
  $\boldsymbol{F}(:, iT+1 : iT+|S'|) \leftarrow \boldsymbol{G}\boldsymbol{R}^{-1}$;
  // Update approximation
  $\boldsymbol{d} \leftarrow \boldsymbol{d} - \texttt{SquaredRowNorms}(\boldsymbol{G}\boldsymbol{R}^{-1})$;
  // Update sampling probabilities
  $\boldsymbol{d} \leftarrow \max\{\boldsymbol{d}, 0\}$;
  // Ensure nonnegative probabilities
  $\boldsymbol{d}(S') \leftarrow 0$;
  // Prevent double sampling
end
return S

as validation data and the rest is training data. The FCM is optimized using the training data. Then the loss function (2) is evaluated using the validation data, and the best parameters are the ones that minimize the loss.

Based on the grid search results for the numerical experiments in Sec. IV, a good default bandwidth is $\varepsilon = 1$. This bandwidth is intuitively reasonable since it corresponds to one standard deviation unit for the transformed data points $\boldsymbol{M}^{1/2}\boldsymbol{x}_1, \dots, \boldsymbol{M}^{1/2}\boldsymbol{x}_n$.

Based on the grid search results, the experiments use a regularization parameter of $\gamma = 10^{-6}$, but this value is not intuitive and it remains unclear whether this would be a good default for other problems. Any value of γ smaller than $\gamma = 10^{-6}$ also leads to a similar loss (less than 1% change). To be conservative, the highest value $\gamma = 10^{-6}$ was selected.

The last parameter to optimize is the approximation rank r that is used in RPCholesky. Raising r increases the accuracy but also increases the linear algebra cost since the FCM optimization requires $\mathcal{O}(Nr^2)$ floating point operations. Typical values of r range from 10^2–10^4, and the theoretical optimum depends on the number of large eigenvalues in the full-data kernel matrix[23]. In the Sec. IV experiments, the rank is set to $r = 10^3$, at which point the results are nearly converged (see Fig. 3).

IV. NUMERICAL RESULTS

This section describes numerical results from applying the fast committor machine (FCM) to two Markovian systems: the overdamped Langevin system with a triple-well potential (Sec. IV B) and a stochastic simulation of alanine dipeptide (Sec. IV C). The code to run the experiments is available on Github[52].

A. Neural network comparison

In each experiment, the FCM was compared against a fully connected feedforward neural network. To make the comparisons fair, the models were designed with nearly the same number of parameters. The FCM has r linear coefficients and $d(d+1)/2$ free parameters in the scaling matrix (d is the phase space dimension). This adds up to 1055 parameters for the triple-well experiment ($r = 1000$, $d = 10$) and 1465 parameters for the alanine dipeptide experiment ($r = 1000$, $d = 30$). The neural architecture includes ℓ hidden layers with p neurons per layer and a tanh nonlinearity, together with an outer layer using a sigmoid nonlinearity. The total number of parameters in the neural network is thus

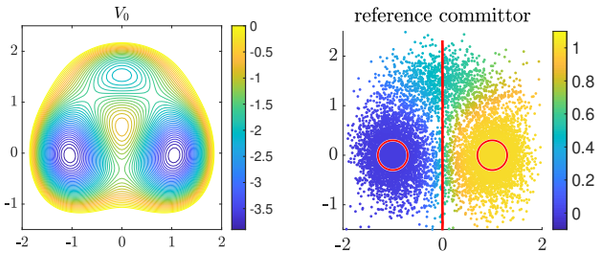
$$p(d+1) + (\ell - 1)p(p+1) + (p+1)$$

and includes 1081 parameters for the triple-well experiment ($\ell = 2$, $p = 27$, $d = 10$) and 1481 parameters for the alanine dipeptide experiment ($\ell = 3$, $p = 20$, $d = 30$). Larger neural nets could potentially improve the accuracy[53], but they would be more challenging and costly to train.

The neural nets were trained using the PyTorch package[54] and the AdamW optimizer[55]. Before the training, 20% of the data was set aside for validation, and the remaining 80% was training data. During each epoch, the optimizer evaluated all the training data points and applied a sequence of stochastic gradient updates using mini-batches of 500 points. After each epoch, the validation data was used to estimate the loss function (2). After 20 epochs with no improvement to the loss function, the training was halted and the parameter set leading to the lowest loss function was selected.

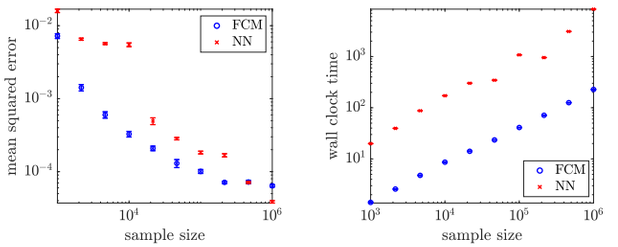
One crucial parameter when training neural nets is the learning rate, which was set to 10^{-4} for the triple-well experiment and 5×10^{-4} for the alanine dipeptide experiment. When the learning rate is too small, the training is excessively slow. When the learning rate is too large, the model exhibits a significant decrease in accuracy or even fails to converge. As a complication, changing the learning rate may require changing the patience parameter (number of epochs with no improvement before halting the training).

On the whole, training neural networks for committor approximation involves choosing an appropriate architecture and an appropriate learning rate with minimal guidance about the best choice of these parameters. In contrast, the parameter-tuning for the FCM is more straightforward: the default bandwidth $\varepsilon = 1$ is sufficient for these experiments. Moreover, adjusting the approximation rank r or regularization parame-



(a) (b)

FIG. 1: (a) The potential function V_0. (b) The reference committor evaluated on the validation data points with states A and B and the committor one-half surface indicated in red.



(a) (b)

FIG. 2: Comparison of neural net (NN) and FCM performance for the triple-well system, with standard error bars computed from 10 independent runs of the FCM. (a) Mean squared error computed using the reference committor. (b) Runtime in seconds.

ter γ has direct and predictable impacts on training speed and performance.

B. Triple-well potential energy

The first numerical experiment is based on the overdamped Langevin dynamics

$$d\boldsymbol{x}(t) = -\nabla V(\boldsymbol{x}(t))\,dt + \sqrt{2\beta^{-1}}\,d\boldsymbol{w}(t). \tag{10}$$

The potential function $V(x)$ is constructed as

$$V(\boldsymbol{x}) = V_0(\boldsymbol{e}_1^T\boldsymbol{x}, \boldsymbol{e}_2^T\boldsymbol{x}) + 2\sum_{i=3}^{10}(\boldsymbol{e}_i^T\boldsymbol{x})^2,$$

where V_0 is the three-well potential[11] that is illustrated in Figure 1, and $\boldsymbol{e}_i$ is the ith basis vector. The equilibrium density for this dynamics is the Gibbs measure

$$\mu(\boldsymbol{x}) = \frac{\mathrm{e}^{-\beta V(\boldsymbol{x})}}{\int \mathrm{e}^{-\beta V(\boldsymbol{y})}\,d\boldsymbol{y}},$$

where $\beta > 0$ is the inverse-temperature parameter.

The goal of the experiment is to calculate the committor function associated with the inverse temperature $\beta = 2$ and the states

$$A = \{\boldsymbol{x} \in \mathbb{R}^{10} \colon (\boldsymbol{e}_1^T\boldsymbol{x} + 1)^2 + (\boldsymbol{e}_2^T\boldsymbol{x})^2 \le 0.3^2\},$$
$$B = \{\boldsymbol{x} \in \mathbb{R}^{10} \colon (\boldsymbol{e}_1^T\boldsymbol{x} - 1)^2 + (\boldsymbol{e}_2^T\boldsymbol{x})^2 \le 0.3^2\}.$$

The definitions for A and B depend only on coordinates 1 and 2. Coordinates 3–10 are nuisance coordinates which have no effect on the committor but increase difficulty of the committor approximation. Because the committor only depends on coordinates 1 and 2, the finite elements method can solve the two-dimensional PDE formulation of the committor problem to generate a highly accurate reference; see Figure 1 for an illustration.

The data set was generated by a two-stage process. In the first stage, the initial states $(\boldsymbol{x}_n)_{1\le n\le N}$ were sampled by running the Langevin dynamics (10) at the inverse temperature $\beta_s = 1$ and storing the positions after $N = 10^6$ uniformly spaced time intervals. The low β_s value was needed to ensure adequate phase space coverage. Since the sampling distribution diverged from the target distribution, the initial states were weighted according to $w_n = \mathrm{e}^{(\beta_s-\beta)V(\boldsymbol{x}_n)}$. This weight definition is consistent with eq. (4), which states that the weights are likelihood ratios multiplied by an arbitrary constant factor.

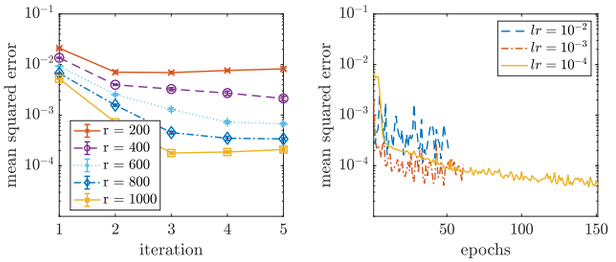
In the second stage, the model was run forward for an additional $\tau = 10^{-2}$ time units at the target inverse temperature $\beta = 2$, starting from $\boldsymbol{x}(0) = \boldsymbol{x}_n$ and ending at a new point $\boldsymbol{x}(\tau) = \boldsymbol{y}_n$. This process was repeated for each data point for $n = 1, \ldots, 10^6$.

Figure 2 evaluate the performance of the FCM and the feed-forward neural network across 10 data sizes logarithmically spaced between $N = 10^3$ and $N = 10^6$. The largest experiments use the full data set with $N = 10^6$ data points, while the other experiments use data pairs chosen uniformly at random. For all sample sizes $N < 10^6$, the FCM achieves higher accuracy than the neural net and also trains more quickly.

The detailed runtime and accuracy comparisons between the FCM and the neural net committor approximation may depend on implementation choices (neural net structure, stopping criteria, optimization method, etc.). Nonetheless, these tests suggest that the FCM is a competitive method for the triple-well experiment.

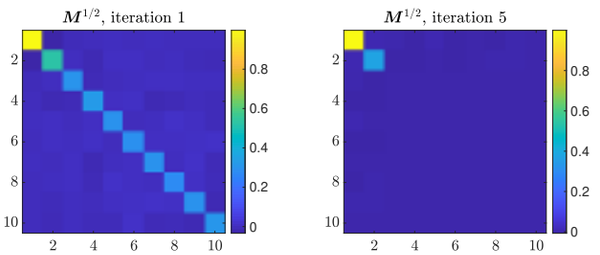
As an additional advantage, the FCM exhibits robustness during training. Figure 3 shows that the FCM error decreases and then stabilizes after a few iterations. In contrast, the neural net error behaves unpredictably with epochs unless the learning rate is very small. To interpret the plot, recall that the FCM updates the scaling matrix $\boldsymbol{M}$ once per iteration and runs for 5 iterations, while the neural net updates the neural net parameters many times per epoch and runs for a variable number of epochs based on the stopping rule.

Last, Figure 4 displays the square root of the scaling matrix, which shows the most significant subspaces for committor estimation. After convergence, the transformation $\boldsymbol{M}^{1/2}$ maps away the nuisance coordinates 3–10. The leading eigenvector of $\boldsymbol{M}^{1/2}$ nearly aligns with coordinate 1, signaling that coordinate 1 is the most essential and coordinate 2 is the sec-



(a) (b)

FIG. 3: Training performance for single instances of the triple-well experiment, with error computed using the reference committor from Fig. 1. (a) The FCM with different approximation ranks r. (b) The neural net with different learning rates lr.



(a) (b)

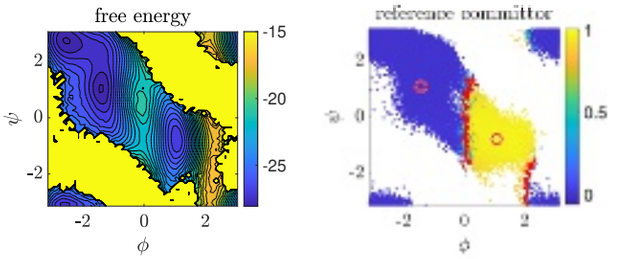
FIG. 4: Square root of scaling matrix for the triple-well system when $N = 10^6$ and $r = 1000$. (a) After 1 iteration. (b) After 5 iterations, corresponding to convergence.

ond most essential. While the two-dimensional figures make it seem that the FCM is solving an easy problem in two-dimensional space, the FCM is actually solving a harder problem in ten-dimensional space. Nonetheless, the FCM is reducing the problem to two dimensions through the automatic identification of the active subspaces.

C. Alanine dipeptide

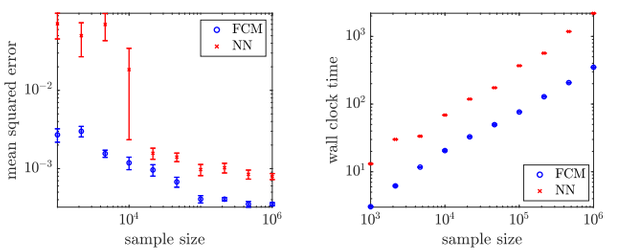
The previous example may seem cherry-picked for the FCM's success since there is so clearly a reduction to two dimensions. Yet many stochastic systems can be described using low-dimensional subspaces, even high-dimensional biomolecular systems[21]. As a concrete example, alanine dipeptide is a small molecule whose dynamics can be reduced to a low-dimensional subspace.

The experiments in this section are drawn from a metadynamics simulation of alanine dipeptide based on the tutorial[56]. Details of the simulation can be found on Github[57]. After excluding the hydrogen atoms, the alanine dipeptide data set contains (x, y, z)-coordinates for the ten backbone atoms, leading to 30-dimensional data points. Figure 5 shows that ϕ and ψ dihedral angles give a simple description of the free energy



(a) (b)

FIG. 5: (a) Free energy surface of alanine dipeptide in ϕ and ψ coordinates, compared to reference committor half-surface (red). (b) The reference committor in ϕ and ψ coordinates with states A and B and the committor one-half surface indicated in red.



(a) (b)

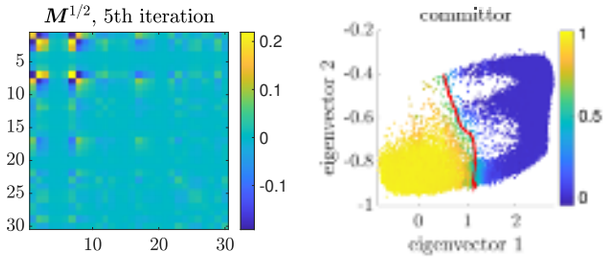
FIG. 6: Comparison of neural net (NN) and FCM performance for alanine dipeptide, with standard error bars computed from 10 independent simulations. (a) Mean squared error, computed with respect to the reference. (b) Runtime in seconds.

surface, containing two prominent metastable states.

The goal of the alanine dipeptide experiment is to estimate the committor function for the two metastable states, labeled as A and B. See Figure 5 for a picture of the precise A and B definitions and a reference committor which is generated by running the FCM with the largest data set ($N = 10^6$) and approximation rank $r = 2000$.

Figure 6 shows the mean squared error and runtimes of the neural network and the FCM with $r = 1000$, as compared to the reference committor. The results again show the FCM is more accurate than the neural network and trains more quickly. These results may depend on the details of the neural network optimization, but they imply the FCM is a competitive method for the alanine dipeptide experiment.

Last, Figure 7 shows the square root of the FCM scaling matrix. The matrix has two eigenvalues that are much larger than the rest, signaling that a two-dimensional linear subspace is closely aligned with the gradients of the committor. Linear regression confirms that the top 2 eigenvectors explain 95% of the variance in the committor values, whereas the nonlinear ϕ and ψ coordinates only explain 62% of the variance. See the right panel of Fig. 7 for a picture of the reference committor



(a) (b)

FIG. 7: Results of applying FCM to alanine dipeptide when $N = 10^6$ and $r = 1000$. (a) $M^{1/2}$ after 5 iterations of Algorithm 1, corresponding to convergence. (b) Reference committor mapped onto top 2 eigenvectors of M with reference committor one-half surface indicated in red.

mapped onto the top 2 eigenvectors.

V. CONCLUSION

The FCM is a method for efficiently solving the committor problem which shows promising results when applied to triple-well and alanine dipeptide systems. As the main conceptual feature, the method identifies a scaling matrix that emphasizes low-dimensional subspaces with maximal variation in the committor values. The method also uses randomized numerical linear algebra to achieve a training time faster than a neural net with the same number of parameters.

The next research goal is testing the FCM on high-dimensional stochastic systems arising in molecular dynamics and other areas of science. In these systems, it may be challenging to generate an appropriate data set, since the states A and B might be separated by large free energy barriers, which make transition events rare and justify the use of enhanced sampling. Even for the 10- and 30-dimensional experiments in this paper, importance sampling and metadynamics were needed to ensure a large density of sample points in the transition region where the committor is 0.1–0.9. These sampling approaches were applied heuristically, and future progress will require a more careful mathematical and empirical study. To that end, a promising approach that was recently developed for committor approximation[58] involves umbrella sampling with a sequence of bins emphasizing different ranges of committor values, including many bins near the committor one-half surface. Building on this strategy, there is hope to develop a fully adaptive FCM method that generates data and a corresponding committor estimate using only the A and B definitions.

Last, there remain challenging mathematical questions about the FCM's performance. For example, why does the method take so few iterations to converge and what makes the exponential kernel (6) preferable over other choices? Addditionally, there is a deep question at the core of the FCM about why the linear rescaling is so successful, how to describe it mathematically, and what nonlinear extensions would be possible.

ACKNOWLEDGEMENTS

D. Aristoff and G. Simpson gratefully acknowledge support from the National Science Foundation via Award No. DMS 2111277. R.J.W. was supported by the Office of Naval Research through BRC Award N00014-18-1-2363, the National Science Foundation through FRG Award 1952777, and Caltech through the Carver Mead New Adventures Fund, under the aegis of Joel A. Tropp.

Appendix A: Justification for the kernel approximation

The FCM is based on two choices regarding the shape of the kernel function and the coefficients used in the optimization. This appendix justifies the choices that were made.

The exponential kernel (6) is used throughout the paper. In the Sec. IV experiments, the exponential kernel outperforms the popular square exponential kernel

$$k_M(\boldsymbol{x},\boldsymbol{x}') = \exp\big(-\varepsilon^{-2}\|M^{1/2}(\boldsymbol{x}-\boldsymbol{x}')\|^2\big).$$

Nonetheless, there may be an opportunity to further improve the FCM by using an alternative kernel of the form

$$k_M(\boldsymbol{x},\boldsymbol{x}') = \phi(\varepsilon^{-1}\|M^{1/2}(\boldsymbol{x}-\boldsymbol{x}')\|)$$

for an optimized univariate function $\phi : \mathbb{R} \to \mathbb{R}$.

The most general data-driven approximation using the exponential kernel is a linear combination of kernel functions centered on the input data points $\boldsymbol{x}_n$ or the output data points $\boldsymbol{y}_n$. The functional form can be written as

$$q_{\boldsymbol{c},\boldsymbol{d}}(\boldsymbol{x}) = \sum_{n=1}^{N} c_n k_M(\boldsymbol{x}_n,\boldsymbol{x}) + \sum_{n=1}^{N} d_n k_M(\boldsymbol{y}_n,\boldsymbol{x}),$$

where $\boldsymbol{c} \in \mathbb{R}^N$ and $\boldsymbol{d} \in \mathbb{R}^N$ are variational parameters to be optimized. Yet, this form can be simplified. Theorem A.1 shows the optimal coefficients must satisfy $\boldsymbol{c} = -\boldsymbol{d}$, which leads to a more specific and concise committor approximation

$$q_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{n=1}^{N} \theta_n\big[k_M(\boldsymbol{x}_n,\boldsymbol{x}) - k_M(\boldsymbol{y}_n,\boldsymbol{x})\big].$$

The proof relies on direct linear algebra calculations.

Theorem A.1. *Define the least-squares loss function*

$$L_\gamma(\boldsymbol{c},\boldsymbol{d}) = \frac{1}{N}\left\|\begin{bmatrix}\mathbf{I}\\-\mathbf{I}\end{bmatrix}^T \boldsymbol{K}\begin{bmatrix}\boldsymbol{c}\\\boldsymbol{d}\end{bmatrix} - \boldsymbol{b}\right\|^2 + \gamma\begin{bmatrix}\boldsymbol{c}\\\boldsymbol{d}\end{bmatrix}^T \boldsymbol{K}\begin{bmatrix}\boldsymbol{c}\\\boldsymbol{d}\end{bmatrix},$$

where the positive semidefinite kernel matrix

$$\boldsymbol{K} = \begin{bmatrix}\boldsymbol{K}^{11} & \boldsymbol{K}^{12}\\ \boldsymbol{K}^{21} & \boldsymbol{K}^{22}\end{bmatrix},$$

consists of four blocks with entries

$$k_{mn}^{11} = \sqrt{w_m}\sqrt{w_n}k_{\boldsymbol{M}}(\boldsymbol{x}_m,\boldsymbol{x}_n),$$
$$k_{mn}^{12} = \sqrt{w_m}\sqrt{w_n}k_{\boldsymbol{M}}(\boldsymbol{x}_m,\boldsymbol{y}_n),$$
$$k_{mn}^{21} = \sqrt{w_m}\sqrt{w_n}k_{\boldsymbol{M}}(\boldsymbol{y}_m,\boldsymbol{x}_n),$$
$$k_{mn}^{22} = \sqrt{w_m}\sqrt{w_n}k_{\boldsymbol{M}}(\boldsymbol{y}_m,\boldsymbol{y}_n).$$

Then, the loss function $\boldsymbol{L}_\gamma(\boldsymbol{c},\boldsymbol{d})$ has a minimizer that satisfies $\boldsymbol{c}+\boldsymbol{d}=\boldsymbol{0}$.

Proof. The loss function is convex, so any minimizers can be identified by setting the gradient equal to zero:

$$\frac{2}{N}\boldsymbol{K}\begin{bmatrix}\mathbf{I}\\-\mathbf{I}\end{bmatrix}\left[\begin{bmatrix}\mathbf{I}\\-\mathbf{I}\end{bmatrix}^T\boldsymbol{K}\begin{bmatrix}\boldsymbol{c}\\\boldsymbol{d}\end{bmatrix}-\boldsymbol{b}\right]+2\gamma\boldsymbol{K}\begin{bmatrix}\boldsymbol{c}\\\boldsymbol{d}\end{bmatrix}=\boldsymbol{0}.$$

The equation can be rearranged to yield

$$\boldsymbol{K}\left[\begin{bmatrix}\mathbf{I}\\-\mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{I}\\-\mathbf{I}\end{bmatrix}^T\boldsymbol{K}+\gamma N\begin{bmatrix}\boldsymbol{I}&\\&\boldsymbol{I}\end{bmatrix}\right]\begin{bmatrix}\boldsymbol{c}\\\boldsymbol{d}\end{bmatrix}=\boldsymbol{K}\begin{bmatrix}\mathbf{I}\\-\mathbf{I}\end{bmatrix}\boldsymbol{b}$$

Therefore, there is a minimizer which comes from choosing $\boldsymbol{c}$ and $\boldsymbol{d}$ to satisfy the positive definite linear system

$$\left[\begin{bmatrix}\mathbf{I}\\-\mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{I}\\-\mathbf{I}\end{bmatrix}^T\boldsymbol{K}+\gamma N\begin{bmatrix}\boldsymbol{I}&\\&\boldsymbol{I}\end{bmatrix}\right]\begin{bmatrix}\boldsymbol{c}\\\boldsymbol{d}\end{bmatrix}=\begin{bmatrix}\mathbf{I}\\-\mathbf{I}\end{bmatrix}\boldsymbol{b}. \tag{A1}$$

(All other minimizers come from adding a vector in the nullspace of $\boldsymbol{K}$). Last, multiply the system (A1) on the left by $\begin{bmatrix}\boldsymbol{I} & \boldsymbol{I}\end{bmatrix}$ to reveal $\boldsymbol{c}+\boldsymbol{d}=\boldsymbol{0}$. □

Appendix B: The scaling matrix

In any learning task, it can be helpful to model a function $f:\mathbb{R}^d\to\mathbb{R}$ as the composition of an invertible linear map $\boldsymbol{A}:\mathbb{R}^d\to\mathbb{R}^d$ and a function $g_{\boldsymbol{A}}$ that is more amenable to learning:

$$f(\boldsymbol{x}) = g_{\boldsymbol{A}}(\boldsymbol{A}\boldsymbol{x}), \quad \boldsymbol{x}\in\mathbb{R}^d.$$

One way to ensure the learnability of the function $g_{\boldsymbol{A}}$ is by selecting the matrix $\boldsymbol{A}$ so that the gradients

$$\nabla g_{\boldsymbol{A}}(\boldsymbol{z}_1),\dots,\nabla g_{\boldsymbol{A}}(\boldsymbol{z}_N)$$

are isotropic, where

$$\boldsymbol{z}_i = \boldsymbol{A}\boldsymbol{x}_i, \quad i=1,2,\dots,N$$

are the linearly transformed data points. The optimal transformation can be characterized as follows:

Proposition B.1. *The following are equivalent:*

(i) $\boldsymbol{A}$ is invertible, and the sample gradients $\nabla g_{\boldsymbol{A}}(\boldsymbol{z}_1),\dots,\nabla g_{\boldsymbol{A}}(\boldsymbol{z}_N)$ are isotropic, that is,

$$\frac{1}{N}\sum_{i=1}^N|\boldsymbol{u}^T\nabla g_{\boldsymbol{A}}(\boldsymbol{z}_i)|^2 = 1$$

for any unit vector $\boldsymbol{u}\in\mathbb{R}^d$.

(ii) The average gradient product

$$\boldsymbol{M} = \frac{1}{N}\sum_{i=1}^N\nabla f(\boldsymbol{x}_i)\nabla f(\boldsymbol{x}_i)^T.$$

is invertible, and $\boldsymbol{A}=\boldsymbol{Q}\boldsymbol{M}^{1/2}$ for an orthogonal matrix $\boldsymbol{Q}\in\mathbb{R}^{d\times d}$.

(iii) $\boldsymbol{M}$ is invertible, and $\boldsymbol{A}$ transforms distances according to

$$\|\boldsymbol{A}(\boldsymbol{x}-\boldsymbol{x}')\| = \|\boldsymbol{M}^{1/2}(\boldsymbol{x}-\boldsymbol{x}')\|,$$

for each $\boldsymbol{x},\boldsymbol{x}'\in\mathbb{R}^d$.

Proof. Either of the conditions (i)-(ii) implies the linear map $\boldsymbol{A}$ is invertible. Therefore calculate

$$\begin{aligned}\frac{1}{N}\sum_{i=1}^N|\boldsymbol{u}^T\nabla g_{\boldsymbol{A}}(\boldsymbol{z}_i)|^2 &= \frac{1}{N}\sum_{i=1}^N|\boldsymbol{u}^T\boldsymbol{A}^{-T}\nabla f(\boldsymbol{x}_i)|^2\\ &= \boldsymbol{u}^T\boldsymbol{A}^{-T}\boldsymbol{M}\boldsymbol{A}^{-1}\boldsymbol{u}.\end{aligned}$$

The above display is 1 for each unit vector $\boldsymbol{u}\in\mathbb{R}^d$ if and only if $\boldsymbol{A}^{-T}\boldsymbol{M}\boldsymbol{A}^{-1}=\mathbf{I}$ and $\boldsymbol{M}^{1/2}\boldsymbol{A}^{-1}$ is an orthogonal matrix. Thus, (i) and (ii) are equivalent.

Clearly, (ii) implies (iii). Conversely, if $\|\boldsymbol{A}\boldsymbol{x}\| = \|\boldsymbol{M}^{1/2}\boldsymbol{x}\|$ for each $\boldsymbol{x}\in\mathbb{R}^d$, it follows that

$$\boldsymbol{M} = \boldsymbol{A}^T\boldsymbol{A}.$$

Consequently, if $\boldsymbol{A}=\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ is a singular value decomposition, $\boldsymbol{M}=\boldsymbol{V}\boldsymbol{\Sigma}^2\boldsymbol{V}^T$ is an eigenvalue decomposition and

$$\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{M}^{1/2}, \quad \text{where } \boldsymbol{Q}=\boldsymbol{U}\boldsymbol{V}^T \text{ is orthogonal.} \tag{B1}$$

This shows that (ii) and (iii) are equivalent. □

[1] K. A. Newhall and E. Vanden-Eijnden, Journal of Nonlinear Science **27**, 1007 (2017).
[2] T. Grafke and E. Vanden-Eijnden, Chaos: An Interdisciplinary Journal of Nonlinear Science **29**, 063118 (2019).
[3] V. Jacques-Dumas, R. M. van Westen, F. Bouchet, and H. A. Dijkstra, Nonlinear Processes in Geophysics **30**, 195 (2023).
[4] D. Lucente, C. Herbert, and F. Bouchet, Journal of the Atmospheric Sciences **79**, 2387 (2022).
[5] P. G. Bolhuis, D. Chandler, C. Dellago, and P. L. Geissler, Annual Review of Physical Chemistry **53**, 291 (2002).
[6] J. Rogal and P. G. Bolhuis, The Journal of Chemical Physics **129**, 224107 (2008).
[7] R. J. Allen, D. Frenkel, and P. R. ten Wolde, The Journal of Chemical Physics **124**, 194111 (2006).
[8] F. A. Escobedo, E. E. Borrero, and J. C. Araque, Journal of Physics: Condensed Matter **21**, 333101 (2009).
[9] W. E, W. Ren, and E. Vanden-Eijnden, The Journal of Physical Chemistry B **109**, 6688 (2005).
[10] E. Vanden-Eijnden and M. Venturoli, The Journal of Chemical Physics **130**, 194103 (2009).
[11] F. Cérou, A. Guyader, T. Lelièvre, and D. Pommier, The Journal of Chemical Physics **134**, 054108 (2011).
[12] Y. Khoo, J. Lu, and L. Ying, Research in the Mathematical Sciences **6**, 1 (2019).
[13] Q. Li, B. Lin, and W. Ren, The Journal of Chemical Physics **151**, 054112 (2019).

[14]H. Li, Y. Khoo, Y. Ren, and L. Ying, in *Proceedings of the 2nd Mathematical and Scientific Machine Learning Conference* (2022).
[15]L. Evans, M. K. Cameron, and P. Tiwary, The Journal of Chemical Physics **157**, 214107 (2022).
[16]R. Lai and J. Lu, Multiscale Modeling & Simulation **16**, 710 (2018).
[17]D. Lucente, J. Rolland, C. Herbert, and F. Bouchet, Journal of Statistical Mechanics: Theory and Experiment **2022**, 083201 (2022).
[18]A. M. Berezhkovskii and A. Szabo, The Journal of Chemical Physics **150**, 054106 (2019).
[19]V. S. Pande, K. Beauchamp, and G. R. Bowman, Methods **52**, 99 (2010).
[20]J. D. Chodera and F. Noé, Current Opinion in Structural Biology **25**, 135 (2014).
[21]B. E. Husic and V. S. Pande, Journal of the American Chemical Society **140**, 2386 (2018).
[22]A. Radhakrishnan, D. Beaglehole, P. Pandit, and M. Belkin, Science **383**, 1461 (2024).
[23]Y. Chen, E. N. Epperly, J. A. Tropp, and R. J. Webber, "Randomly pivoted Cholesky: Practical approximation of a kernel matrix with few entry evaluations," (2023), arXiv:2207.06503 [cs, math, stat].
[24]A. L. Ferguson, A. Z. Panagiotopoulos, I. G. Kevrekidis, and P. G. Debenedetti, Chemical Physics Letters **509**, 1 (2011).
[25]M. A. Rohrdanz, W. Zheng, M. Maggioni, and C. Clementi, The Journal of Chemical Physics **134**, 124116 (2011).
[26]R.-R. Griffiths, L. Klarner, H. Moss, A. Ravuri, S. T. Truong, Y. Du, S. D. Stanton, G. Tom, B. Ranković, A. R. Jamasb, A. Deshwal, J. Schwartz, A. Tripp, G. Kell, S. Frieder, A. Bourached, A. J. Chan, J. Moss, C. Guo, J. P. Dürholt, S. Chaurasia, J. W. Park, F. Strieth-Kalthoff, A. Lee, B. Cheng, A. Aspuru-Guzik, P. Schwaller, and J. Tang, in *Thirty-seventh Conference on Neural Information Processing Systems* (2023).
[27]M. J. Burn and P. L. A. Popelier, The Journal of Chemical Physics **153**, 054111 (2020).
[28]V. L. Deringer, A. P. Bartók, N. Bernstein, D. M. Wilkins, M. Ceriotti, and G. Csányi, Chemical Reviews **121**, 10073 (2021).
[29]M. J. Burn and P. L. A. Popelier, Materials Advances **3**, 8729 (2022).
[30]G. Tom, R. J. Hickman, A. Zinzuwadia, A. Mohajeri, B. Sanchez-Lengeling, and A. Aspuru-Guzik, Digital Discovery **2**, 759 (2023).
[31]F. Musil, A. Grisafi, A. P. Bartók, C. Ortner, G. Csányi, and M. Ceriotti, Chemical Reviews **121**, 9759 (2021).
[32]R. R. Coifman, I. G. Kevrekidis, S. Lafon, M. Maggioni, and B. Nadler, Multiscale Modeling & Simulation **7**, 842 (2008).
[33]F. Noé and F. Nüske, Multiscale Modeling & Simulation **11**, 635 (2013).
[34]H. Chen, B. Roux, and C. Chipot, Journal of Chemical Theory and Computation **19**, 4414 (2023).
[35]H. Chen and C. Chipot, QRB Discovery **4**, e2 (2023).
[36]C. Schütte, S. Klus, and C. Hartmann, Acta Numerica **32**, 517–673 (2023).
[37]Y. Kamada and S. Abe, in *IAPR Workshop on Artificial Neural Networks in Pattern Recognition* (2006).
[38]G. Camps-Valls, A. Rodrigo-Gonzalez, J. Munoz-Mari, L. Gomez-Chova, and J. Calpe-Maravilla, in *IEEE International Geoscience and Remote Sensing Symposium* (2007).
[39]Q. Wu, J. Guinney, M. Maggioni, and S. Mukherjee, Journal of Machine Learning Research **11**, 2175 (2010).
[40]S. Trivedi, J. Wang, S. Kpotufe, and G. Shakhnarovich, in *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence* (2014).
[41]D. Beaglehole, A. Radhakrishnan, P. Pandit, and M. Belkin, "Mechanism of feature learning in convolutional neural networks," (2023), arXiv:2309.00570 [stat.ML].
[42]D. Beaglehole, P. Súkeník, M. Mondelli, and M. Belkin, "Average gradient outer product as a mechanism for deep neural collapse," (2024), arXiv:2402.13728 [cs.LG].
[43]O. T. Unke, S. Chmiela, H. E. Sauceda, M. Gastegger, I. Poltavsky, K. T. Schütt, A. Tkatchenko, and K.-R. Müller, Chemical Reviews **121**, 10142 (2021).
[44]C. K. I. Williams and M. Seeger, in *Proceedings of the 13th International Conference on Neural Information Processing Systems* (2000).
[45]A. Rudi, L. Carratino, and L. Rosasco, in *Proceedings of the 31st International Conference on Neural Information Processing Systems* (2017).
[46]M. Díaz, E. N. Epperly, Z. Frangella, J. A. Tropp, and R. J. Webber, "Robust, randomized preconditioning for kernel ridge regression," (2023), arXiv:2304.12465 [math.NA].
[47]P. Doyle and J. Snell, *Random Walks and Electric Networks* (American Mathematical Society, 1984).
[48]R. Hersh and R. J. Griego, Scientific American **220**, 66 (1969).
[49]B. Schmuland, The Canadian Journal of Statistics / La Revue Canadienne de Statistique **27**, 683 (1999).
[50]M. Fukushima, Y. Oshima, and M. Takeda, *Dirichlet forms and symmetric Markov processes*, 2nd ed. (De Gruyter, New York, 2011).
[51]M. Kanagawa, P. Hennig, D. Sejdinovic, and B. K. Sriperumbudur, "Gaussian processes and kernel methods: A review on connections and equivalences," (2018), arXiv:1807.02582 [stat.ML].
[52]Https://github.com/davidaristoff/Fast-Committor-Machine/.
[53]M. Belkin, D. Hsu, S. Ma, and S. Mandal, Proceedings of the National Academy of Sciences **116**, 15849 (2019).
[54]A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, in *Proceedings of the 33rd International Conference on Neural Information Processing Systems* (2019).
[55]I. Loshchilov and F. Hutter, in *Proceedings of the Seventh International Conference on Learning Representations* (2019).
[56]Https://www.plumed.org/doc-v2.7/user-doc/html/masterclass-21-4.html.
[57]Https://github.com/davidaristoff/Fast-Committor-Machine/.
[58]G. M. Rotskoff, A. R. Mitchell, and E. Vanden-Eijnden, in *Proceedings of the 2nd Mathematical and Scientific Machine Learning Conference* (2022).