Summary and Abstract

Demetrius Rowland

Fall 2019

Terminology

Symbol	Meaning
x_t	An element of a Markov Process at time t.
τ	The lag time between two configurations $x_{t+\tau}$ and x_t in the Markov process.
$\chi(x_t)$	The i^{th} element of χ encodes the probability that x_t occupies metastable state i
$q(y,t)$	The i^{th} element of q encodes the probability that $x_{t+\tau} = y$ given that $x_t \in$ state i.
γ	A trainable weight function that helps estimate q from the empirical distribution.
K	The transition matrix between metastable states in the latent space.
G	A generative function that maps from latent space into configuration space.

Summary

This paper provides a deep learning framework for modeling the long-term behavior of a complex dynamical system and uses this model to generate unobserved potential configurations of the dynamical system. It does so by learning from time-series data, and it accomplishes these goals using the following steps:

1. Defines an encoder $\boldsymbol{\chi}(x_t)$ from the d-dimensional input configurations into an m dimensional latent space, with $m << d$, where the i^{th} element of $\boldsymbol{\chi}(x_t)$ is the probability that x_t occupies state i.
2. Defines a generator q(y, τ) from latent space into configuration space, where the i^{th} element of q(y,τ) encodes the probability that $x_{t+\tau} = y$ given that $x_t \in$ state i.
3. Trains networks χ and q to approximate the encoder and generator, respectively, either according to the DeepResampleMSM or DeepGenMSM methodology.
4. Derives a transition matrix K between metastable states, using the trained networks χ and q.

5. Generates unobserved configurations using the generator q.

Equipped with an encoder, a transition matrix in latent space, and a generator, we can propagate in configuration space or between metastable states at some fixed time lag τ, in order to estimate either the configuration or the state of an element in our dynamical system at some future time.

The transition density in configuration space between configuration x and configuration y can be written as

$$P(x_{t+\tau} = y | x_t = x) = \sum_{i=1}^{m} \chi_i(x) * q_i(y, t)$$

with

$$\chi_i(x) = Pr(x_t \in \text{state i} | x_t = x)$$
$$q_i(y, t) = Pr(x_{t+\tau} = y | x \in \text{state i})$$

Given time series data $x_1, x_2, ..., x_T$, we have two options for estimating $\boldsymbol{\chi}, \boldsymbol{q}$

Option 1: DeepResampleMSM

We estimate $\boldsymbol{q}$ by

$$q_i = \frac{1}{\bar{\gamma}_i} \gamma_i(y) p(y)$$

where $p(y)$ is the empirical distribution function, $\gamma_i(y)$ is a trainable weight function, and $\bar{\gamma}_i$ is a normalization factor. Here, we approximate the function $\boldsymbol{\gamma}$ by a network γ. The likelihood function, which we define to be the probability of observing the pairs $(x_t, x_{t+\tau})$ for $t = 1, 2, ..., T - \tau$, is given by

$$
\begin{aligned}
\mathrm{L} &= \prod_{t=1}^{T-\tau} \mathbb{P}(x_{t+\tau}|x_t) \\
&= \prod_{t=1}^{T-\tau} \sum_{i=1}^{m} \chi_i(x_t) q_i(x_{t+\tau}) \\
\log \mathrm{L} &= \sum_{t=1}^{T-\tau} \log(\sum_{i=1}^{m} \frac{1}{\bar{\gamma}_i} \gamma_i(x_{t+\tau}) p(x_{t+\tau}) \chi_i(x_t)) \\
&= \sum_{t=1}^{T-\tau} \log(p(x_{t+\tau})) + \sum_{t=1}^{T-\tau} \log(\sum_{i=1}^{m} \frac{1}{\bar{\gamma}_i} \gamma_i(x_{t+\tau}) \chi_i(x_t))
\end{aligned}
$$

Because there is no contribution from the weights of the networks χ and γ to the left term, we maximize the right term to find appropriate weights of χ and γ.

Option 2: DeepGenMSM

We define the energy distance D between two probability distributions $\mathbb{P}(x)$ and $\mathbb{P}(y)$ to be $D(\mathbb{P}(x), \mathbb{P}(y)) = \mathbb{E}(2||x-y|| - ||x-x'|| - ||y-y'||)$, where x' and y' are sampled from the distributions of x and y independently.

The generator q is approximated by a neural network G which takes in a one-hot encoding e_i of the metastable state i and a noise vector ϵ, whose components are distributed according to a standard normal distribution.

We now set D to be the energy distance between the probability distributions $\mathbb{P}(\hat{x}_{t+\tau}|x_t)$ and $\mathbb{P}(x_{t+\tau}|x_t)$, where $\hat{x}_{t+\tau}$ is the result of an application of the network G to the latent vector $\chi(x_t)$.

The networks χ and G are then both trained with loss function D.

Training χ and q

We determine $\boldsymbol{\chi}(x_t)$ and $q(y, \tau)$ using either of the two above options given the time series data $x_1, ..., x_T$.

Determining K

We determine the transition matrix K by defining each of its elements as

$$
\begin{aligned}
&k_{ij}(\tau) = Pr(x_{t+\tau} \in \text{state j} | x_t \in \text{state i}) \\
&= \int_y Pr(x_{t+\tau} \in \text{state j} | x_{t+\tau} = y) * Pr(x_{t+\tau} = y | x_t \in \text{state i}) \qquad \text{by (1)} \\
&= \int_y q_i(y,t)\chi_j(y)dy \\
&= \int_y \frac{q_i(y,t)\chi_j(y)}{p(y)} p(y)dy \\
&= \mathbb{E}_{y \sim p(y)}[\frac{q_i(y,t)\chi_j(y)}{p(y)}]
\end{aligned}
$$

Under the law of large numbers, K can then be approximated by

$$K = \frac{1}{T-\tau} \sum_{t=1}^{T-\tau} \frac{\boldsymbol{q}(x_t,\tau)\boldsymbol{\chi}(x_t)^T}{p(x_t)}$$

where p denotes the empirical distribution of $x_1, ..., x_T$.

Propagating in configuration space

We can now use the trained generator q to generate new and potentially unobserved configurations of the elements in the dynamical system.

Data

Prinz

- The Prinz Potential $V(x)$ is defined $V(x) = 4(x^8 + .8\exp(-80x^2) + .2\exp(-80(x - .5)^2) + .5\exp(-40(x + .5)^2))$
- x is a variable encoding one-dimensional position in a diffusion process.
- The positions in the diffusion process are computed using the rule
  $x_{t+\Delta t} = x_t - \frac{\Delta t}{kT}\Delta V(x_t) + \sqrt{2\Delta t D} * e_t$

  with x_t the t^{th} position in the diffusion process, kT the Boltzmann constant multiplied by temperature, $e_t \sim_{iid}$ Normal$(0, 1)$, and D a diffusion constant.
- The authors set $kT = 1.45$, $\Delta t = .01$, and $x_0 = -.75$.
- The metastable states of this diffusion process correspond to the wells of the potential V(x), i.e. the x-coordinates at which the gradient of V is small and so is the change in position.

- Trajectories of 250,000 and 125,000 steps are simulated for training and testing.
- Here, χ, γ are both functions approximated by neural networks with one input node corresponding to the position values x_t and $x_{t+\tau}$ for χ and γ respectively.
- They are trained according to a loss function equal to $-\log L$, where L is the likelihood function given in **Deep Resample MSM**.
- After having been trained each point x_t can be assigned to the metastable state, i.e. a well of the potential function, for which $\chi(x_t)$ assigns the highest probability.
- The K transition probability matrix between wells is learned by applying χ and the induced q (where q_i represents the probability of landing at $x_{t+\tau}$ given that x_t is in well i) to each position $x_{t+\tau}$ in the data set, multiplying and averaging.

Alanine Dipeptide

- The alanine dipeptide trajectories are simulated using the AceMD software with the Amber ff-99SB-ILDN force field and integration time step of 2 fs
- The Temperature is set to 300 K.
- Peptide simulated inside a cubic box with 651 TIP3P water molecules.
- All bonds between hydrogen and heavy atoms are constrained.
- The metastable states of the alanine dipeptide dynamics correspond to the backbone and torsion angles which give a minimum of the force field, i.e. the potential energy function.
- Under these conditions, the authors simulate a 250 ns trajectory of the 3-dimensional coordinates of the ten heavy atoms in alanine dipeptide (30 dimensions in total), and use a 80% / 20% split for training and testing.
- Here, χ, γ are both functions approximated by neural networks with thirty input nodes corresponding to the three dimensional coordinates of the ten heavy atoms x_t and $x_{t+\tau}$ for χ and γ respectively.
- They are trained according to a loss function equal to $-\log L$, where L is the likelihood function given in **Deep Resample MSM**.
- After having been trained each point x_t can be assigned to the metastable state, i.e. a minimum of the potential function, for which $\chi(x_t)$ assigns the highest probability.

- The K transition probability matrix between wells is learned by applying χ and the induced q (where q_i represents the probability of landing at $x_{t+\tau}$ given that x_t is in minimum i) to each position $x_{t+\tau}$ in the data set, multiplying and averaging.

Algorithm

Below we outline the process for approximating the transition matrix K and computing the corresponding implied time scales and stationary distributions under the DeepResampleMSM methodology.

DeepResampleMSM

- We are given configuration data $\boldsymbol{x}_{1:T}$.
- We configure neural networks χ and γ as follows:
  - χ:
    - χ has d input nodes, where d is the dimension of the configuration space.
    - χ has 4 hidden layers with 64 nodes each.
    - Batch normalization is used after each layer.
    - χ has m output nodes, where m is the dimension of the latent space.
    - RELU activations are used everywhere, except in the output layer which uses SoftMax
  - γ:
    - γ has d input nodes, where d is the dimension of the configuration space.
    - γ has 4 hidden layers with 64 nodes each.
    - Batch normalization is used after each layer.
    - γ has m output nodes, where m is the dimension of the latent space.
    - RELU activations are used everywhere. This is acceptable, since we normalize during the computation of the generator q.
- We define an Early Stopping class which prevents the networks χ and γ from overfitting the training data.
- Specifically, after each epoch, we check the performance of the networks χ and γ measured according to the negative log likelihood on the test data

$$-\log(\prod_{t\in\text{test}} P(x_{t+\tau}|x_t)) = -\log(\prod_{t\in\text{test}} \sum_{i=1}^{m} \chi_i(x_t)\frac{1}{\bar{\gamma}_i}\gamma_i(x_{t+\tau}))$$

and see if this negative log likelihood decreases. If it does not, we increment a count and if the count exceeds a user-specified patience parameter p, we stop the training.

- We train the networks χ and γ according to the loss function

  $-\log(\prod_{t=1}^{T-\tau} \sum_{i=1}^{m} \chi_i(x_t) \frac{1}{\bar{\gamma}_i} \gamma_i(x_{t+\tau}))$

  on the training data for 200 epochs.

- Specifically, we use torch.optim.adam to find sets of weights W_P and W_G belonging to networks P and G respectively, that minimize the negative log likelihood function depicted above.

- We check the loss function value on the test data, and if it has not decreased after p epochs (recall that p is our user-specified patience parameter), then we stop the training of the networks.

- Recall that the formula for the generator q under the DeepResampleMSM methodology is

$$q_i = \frac{1}{\bar{\gamma}_i} \gamma_i(y) p(y)$$

- Recall also that K can be computed with

$$\begin{aligned}
&k_{ij}(\tau) = Pr(x_{t+\tau} \in \text{state j} | x_t \in \text{state i}) \\
&= \int_y Pr(x_{t+\tau} \in \text{state j} | x_{t+\tau} = y) * Pr(x_{t+\tau} = y | x_t \in \text{state i}) \\
&= \int_y q_i(y,t) \chi_j(y) dy \\
&= \int_y \frac{1}{\bar{\gamma}_i} \gamma_i(y) \chi_j(y) p(y) dy \\
&= \mathbb{E}_{y \sim p(y)} [\frac{1}{\bar{\gamma}_i} \gamma_i(y) \chi_j(y)]
\end{aligned}$$

- It follows from the law of large numbers that

$$k_{ij} \approx \frac{1}{T-\tau} \sum_{t=1}^{T-\tau} \frac{1}{\bar{\gamma}_i} \gamma_i(x_t) \chi_j(x_t)$$

- With trained networks P and G, we can apply the network P and G to the observations $x_1, ..., x_{T-\tau}$ to obtain two $(T-\tau) \times m$ matrices which we

will call M_P and M_G. We then normalize each column of M_G to obtain $\tilde{M}_G$. We obtain K as

$$K = \frac{1}{T-\tau}\tilde{M}_G^T * M_P$$

- With K computed we can find the implied timescales via the formula

$$t_i(\tau) = -\frac{\tau}{log|\lambda_i(\tau)|}$$

where λ_i are the eigenvalues of the matrix K.

- We can then compute the transition density $\mathbb{P}(x_{t+\tau}|x_t)$ by the formula

$$\mathbb{P}(x_{t+\tau}|x_t) = \boldsymbol{\chi}^T(x_t)\boldsymbol{q}(x_{t+\tau}, \tau)$$

where we estimate $\boldsymbol{\chi}$ and $\boldsymbol{q}$ by the trained networks P and G respectively.

- We compute the stationary density by solving

$$\boldsymbol{\pi} = K^T\boldsymbol{\pi}$$

and taking this distribution $\boldsymbol{\pi}$ into configuration space by an application of the network G.

- We save the matrix K, the implied time scales, the transition densities, and stationary densities.

Extensions

Below I highlight two areas which merit further exploration.

Bayesian Analysis

Given configuration data $\boldsymbol{x}_{1:T}$ and parameters $p_1, ..., p_{n_1}$ for the encoding distribution $\boldsymbol{\chi}$ and parameters $r_1, ..., r_{n_2}$ for the generating distribution $\boldsymbol{q}$, we can perform Bayesian inference for $p_1, ..., p_{n_1}$ and $r_1, ..., r_{n_2}$ under the likelihood function described under the **DeepResampleMSM** section.

We can also define a prior distribution on the matrix K, from which we can obtain the encoder $\boldsymbol{\chi}$ and the generator $\boldsymbol{q}$ by reversing the rewiring trick: Train the weights of the networks χ and q which approximate $\boldsymbol{\chi}$ and $\boldsymbol{q}$, respectively, by mapping each observation x_t into latent space via χ, propagating in latent space via K, and comparing this result with the result of an application of the connected networks q and χ. The loss function we could use might be a sum of squares error between the former and latter results.

Computing the Transition Matrix

In the authors' analysis, the transition matrix K is computed by taking each vector x_t into latent space via the trained network χ, generating a configuration vector $\tilde{x}_{t+\tau}$ via the trained network q, and applying χ once more to obtain two latent vectors, which give us the statistics needed to estimate K.

However, we can also use the trained network χ to take the vectors x_t and $x_{t+\tau}$ into latent space, and learn the matrix K from these two latent vectors. It might prove useful to compare the matrices generated by these two methods.

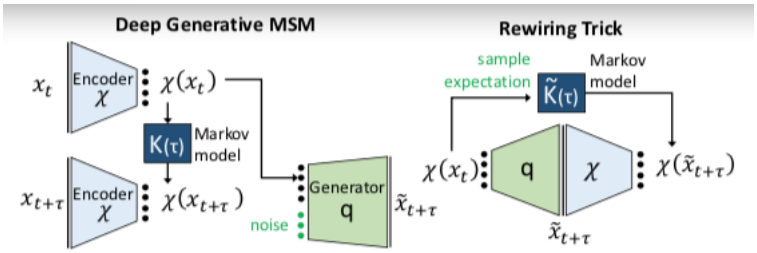

Figure 1

- Suppose we are studying a set of configurations indexed by time describing a dynamical system.
- We let x_t represent a configuration x at time t, we let the user-specified time lag $\tau \in \mathbb{R}^+$ and we let $T \in \mathbb{Z}^+$ be the number of observed x_t in the time series data.
- We build an encoding network χ to approximate the function $\chi_i(x) = \mathbb{P}(x_t \in \text{state i} | x_t = x)$.
- We build a generative network q to approximate the function $q_i(x) = \mathbb{P}(x_{t+\tau} = y | x_t \in \text{state i})$.
- The networks χ and q are trained, either according to the negative log likelihood loss function (outlined under the **DeepResampleMSM** section), or according to an energy distance loss function (outlined under the **DeepGenMSM** section).
- With trained networks χ and q, we can estimate the transition matrix K by taking each point x_t, mapping it into latent space using χ, taking the result into configuration space via q, and applying χ again. We now have two latent vectors $\chi(x_t)$ and $\chi(\hat{x}_{t+\tau})$ from which we can estimate K. The full process is detailed under the **Summary** section.
- This is the rewiring trick depicted on the right.
- With trained networks χ and q and estimated transition matrix K, we can now propagate x_t in configuration space by taking x_t into latent space via χ and mapping the result into configuration space via q. We can propagate in latent space by taking x_t into latent space via χ and taking the result to another latent vector by an application of K.

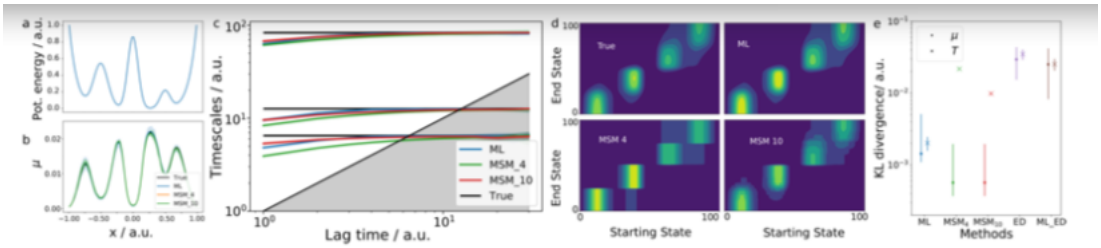

Figure 2

(a) Plots the Prinz potential V(x) where x denotes position, and $V(x) = 4(x^8 + .8\exp(-80x^2) + .2\exp(-80(x-.5)^2) + .5\exp(-40(x+.5)^2))$.

(b) For each method, plots $\mu(x) = \boldsymbol{\pi}^T \boldsymbol{q}(x, \tau)$, which gives the stationary distribution in configuration space, against the true stationary distribution, where $\boldsymbol{\pi}$ is the solution to the eigenvector equation $\boldsymbol{\pi} = K^T \boldsymbol{\pi}$. Here, as afterwards, ML denotes the DeepResampleMSM method with four states in latent space.

(c) For each method, plots the estimated relaxation timescale against the true relaxation timescale. For large values of τ each estimate converges to the true relaxation timescale. The estimates are computed as $t_i(\tau) = -\dfrac{\tau}{log|\lambda_i(\tau)|}$. The λ_i, with $i \in 2, 3, 4$, are the nontrivial eigenvalues of the transition matrix K, supplied by each method. Thus each eigenvalue corresponds to a different timescale, and hence a different plot on the graph.

(d) First we scale and shift the x-values for the diffusion process to match the interval [0, 100]. We define the transition density to be $P(x, y) = \mathbb{P}(x_{t+\tau} = y | x_t = x)$. This is equivalent to $P(x, y) = \chi(x)^T q(y, \tau)$, by our definitions for χ and q. These can be computed using our trained networks for χ and q. For each method and lag time $\tau = 5$, plots the transition density.

(e) Computes the KL-divergence between the stationary and transition densities estimated by each method and the true stationary and transition densities.

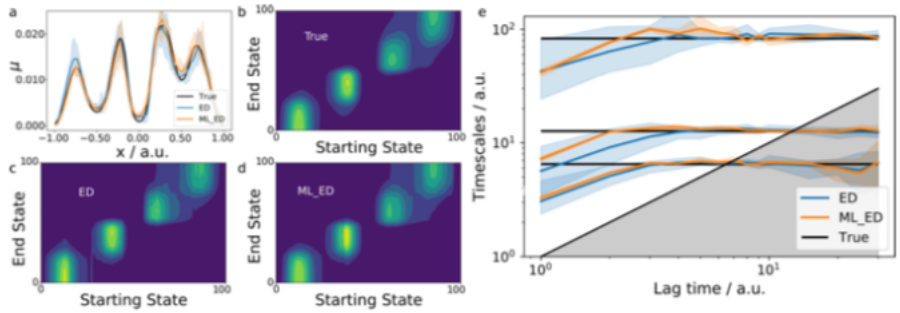

Figure 3
Repeats the same procedures as in Figure 2, but for two DeepGenMSM methods. The first is ED, which trains both χ and q using an energy distance loss function described under **DeepGenMSM**. The second is ML-ED, which uses the network χ from an ML-trained DeepResampleMSM, and trains q using the energy distance loss function.

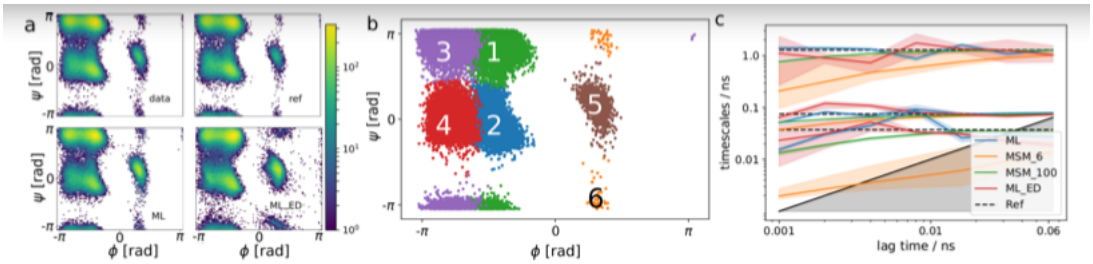

Figure 4
The data set for Alanine Dipeptide consists of the 3-dimensional coordinates of the ten heavy atoms. However, it is known that the metastable states, corresponding to the minima of the potential function underlying the dynamics for Alanine Dipeptide, can be understood entirely in terms of the backbone and torsion angles (ϕ, ψ) of the molecule. It is therefore the task of DeepGenMSM to learn the mapping between the cartesian coordinates and the (ϕ, ψ) space and then to learn which (ϕ, ψ) correspond to the metastable states.

(a) Plots the empirical distribution of the time series data $\{x_t\}_{t=1}^{125,000}$, where each x_t is a 30-dimensional vector of cartesian coordinates, as well as the stationary distributions for a standard MSM model, DeepResampleMSM, DeepGenMSM.

(b) Illustrates the DeepResampleMSM categorization of the (ϕ, ψ) pair into each of the six metastable states.

(c) As before, plots the estimated relaxation timescales for each method against the true relaxation timescale.

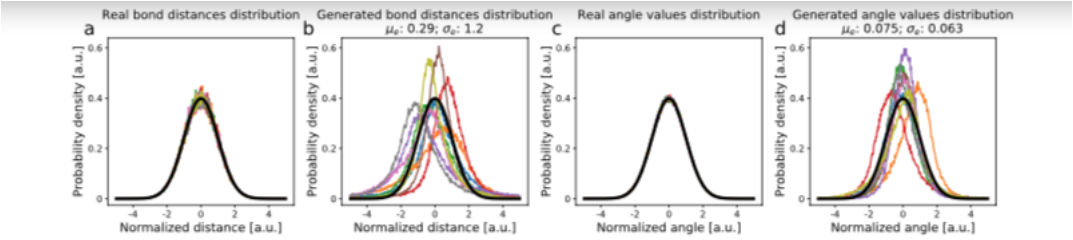

Figure 5

- We now want to check if the generated configurations under DeepGenMSM are physically meaningful.
- So we take the true data, we normalize it, and we plot the bond distances and angle distributions against a standard normal distribution.
- We then take the generated trajectories under DeepGenMSM, we normalize according to the mean and variance of the true data, and we plot the bond and angle distributions for these against a standard normal distribution.
- The results are encouraging, because we can see that the bond and angle distributions for the generated trajectories under DeepGenMSM closely mimic the bond and angle distributions for the true data.

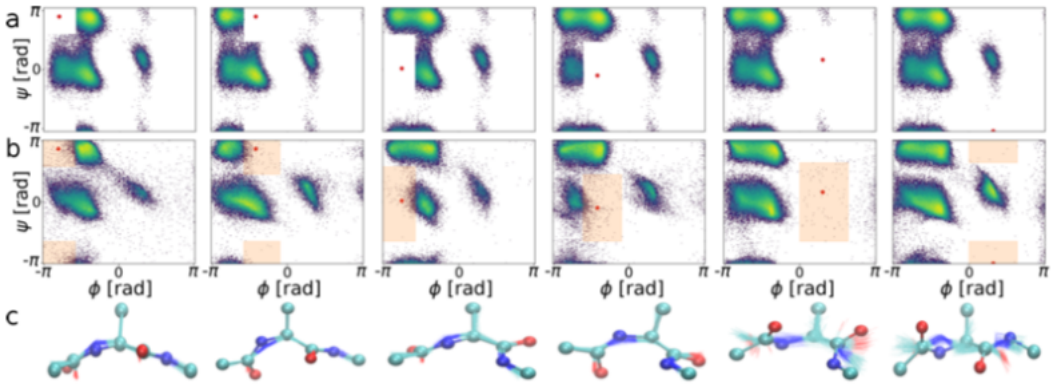

Figure 6

(a) Removes one of the six metastable states for alanine dipeptide and plots the distributions for the true data.

(b) Trains the DeepGenMSM on each dataset that is missing one of the six metastable states. Some configurations are predicted even where no data was present. This provides the hope of discovering new, physically meaningful metastable states with DeepGenMSM.

(c) Shows a physical representation of the molecule occupying the removed metastable state, along with the 100 closest configurations predicted by DeepGenMSM.

Papers

VAMPnets for deep learning of molecular kinetics

- Variational Approach to Markov Processes
- Encodes a mapping of molecular coordinates to the Markov States.
- VAMP provides a framework for evaluating the accuracy of a model given the MD operator that governs the kinetics underlying the data. As a result, it requires little expertise from the modeler.
- The fundamental idea is that molecular dynamics can be described by a Markov Process $\boldsymbol{x}_t$, and that there exist transformations $\boldsymbol{\chi}_0, \boldsymbol{\chi}_1$ such that for large lag time τ, $\mathrm{E}[\boldsymbol{\chi}_1(\boldsymbol{x}_{t+\tau})] \approx K^T \mathrm{E}[\boldsymbol{\chi}_0(\boldsymbol{x}_t)]$
- We then define a VAMP-2 score, which we can maximize to find appropriate transformations $\boldsymbol{\chi}_0, \boldsymbol{\chi}_1$ and transition matrix K.

Kinetic distance and kinetic maps from molecular dynamics simulation

- Important aims of MD simulation are characterizing metastable states and the probabilities of transitioning between them.
- Here, the authors define a kinetic distance that quantifies how slowly the molecular conformations interconvert.
- This distance $D_\tau(\mathbf{x}_1, \mathbf{x}_2)$ is defined as the distance between the system?s probability densities at time τ, given that we have initialized the system either at state $\mathbf{x}_1$ or at state $\mathbf{x}_2$ at time 0.
- This can be converted into a formula involving eigenvalues λ and eigenfunctions ψ of the dynamical propagator P, which can be learned using the TICA method.

A variational approach to modeling slow processes in stochastic dynamical systems

- Often slow processes are the most difficult to simulate due to their extremely long simulation times.
- Consider a dynamical system with propagation function $\mathrm{P}(\tau)$.
- We can estimate the eigenvalues λ and eigenfunctions ψ corresponding to the spectral decomposition of P by maximizing the Rayleigh coefficient, which we can estimate from data.

On the approximation quality of markov state models

- Provides an upper bound for the error between a Markov State Model and a dynamical process.
- We see that under certain ergodicity conditions for the dynamical process, $\mathrm{E(k)} = \mathrm{dist}(T_{k\tau}, P_{k\tau}) \leq 2^{*}\rho^{k}$ for some $\rho \in (0, 1)$.
- A stronger statement can be made using a partitioning of the state space, which allows the user to reduce the error bound by choosing an appropriately large τ.

Time-lagged autoencoders: Deep learning of slow collective variables for molecular kinetics

- Applies an autoencoder type neural network to find low-dimensional embeddings for high dimensional feature space.
- In many cases, the metastable states are not linearly separable, and hence traditional MSMs can have high prediction errors.
- This requires that we use a deep Time-lagged Autoencoder to learn the nonlinear transformation of the features which would allow for separation.
- Using toy models and MD data, the authors have found that deepTAE outperforms TICA and PCA.

Variational approach for learning markov processes from time series data

- Defines the VAMP approach, which learns a nonlinear transformation of given features that can induce a linear Markovian Model in the transformed feature space.
- Optimal transformations and optimal Markov models are found by maximizing the VAMP-E score, which is determined by the singular value decomposition of a Koopman operator.

Plan

- Find neural time series data $x_1, ..., x_T$.
- Determine appropriate metastable states.
- Build a normal location scale Hidden Markov Model with hidden states $z_1, ..., z_T$ taking on values in the set of metastable states, and $y_t | z_t = k \sim \text{Norm}(\mu_k, \sigma_k^2)$ for $k = 1, 2, ..., m$.
- Construct priors for μ_k, σ_k^2.
- Learn a transition matrix K between metastable states using the DeepResampleMSM methodology.
- Find posteriors for μ_k, σ_k^2 using the learned transition matrix K.
- Find the predictive density $\mathbb{P}(x_{T+\tau} | x_1, ..., x_T)$ using the posteriors for μ_k, σ_k^2.
- Determine error rate on test data and compare with DeepResampleMSM, DeepGenMSM, and standard MSMs.

Schedule

20 November 2019

- Update abstract, goals, and summary.

27 November 2019 -

- TBD

References

[1] Andreas Mardt, Luca Pasquali, Hao Wu, and Frank Noe. *Vampnets for deep learning of molecular kinetics*. Nat. Commun., 2018.

[2] Frank Noe and Cecilia Clementi. *Kinetic distance and kinetic maps from molecular dynamics simulation*. J. Chem. Theory. Comput., 2015.

[3] Frank Noe and Feliks Nuske. *A variational approach to modeling slow processes in stochastic dynamical systems*. Multiscale Model. & Simul., 2013.

[4] M. Sarich and C. Schutte. *Metastability and Markov State Models in Molecular Dynamics*. Courant Lecture Notes. American Mathematical Society, 2013.

[5] Marco Sarich, Frank Noe, and Christof Schutte. *On the approximation quality of markov state models*. Multiscale Model. Simul., 2010.

[6] Christoph Wehmeyer and Frank Noe. *Time-lagged autoencoders: Deep learning of slow collective variables for molecular kinetics*. J. Chem. Phys., 2018.

[7] Hao Wu and Frank Noe. *Variational approach for learning markov processes from time series data*. arXiv:1707.04659, 2017.

[8] Hao Wu, Andreas Mardt, Luca Pasquali, and Frank Noe. *Deep generative markov state models*. arXiv:1805.07601, 2019.