1 ver March 2021 (29 Mar 2021)

- A configuration file in INI format can be used for advanced analysis, which allows a more flexible control of the training process. In this advanced mode, the input data can be multiple trajectories.

2 ver June 2021 (25 Jun 2021)

- The predicted state population is used to measure the convergence of training for each iteration.
- Both the linear encoder and nonlinear encoder are implemented.

3 ver July 2021 (30 Jul 2021)

- The pseudo-inputs are replaced with the representative-inputs and the hyper-parameter K is removed.

  For a given unbiased trajectory $\{\boldsymbol{X}^1, \cdots, \boldsymbol{X}^{M+s}\}$ and its corresponding state labels $\{\boldsymbol{y}^1, \cdots, \boldsymbol{y}^{M+s}\}$ with large enough M, we can employ the deep variational information bottleneck framework and construct an ANN that is trained to maximize the following objective function:

  $$\mathcal{L} = \frac{1}{M} \sum_{n=1}^{M} \left[\log q_\theta(\boldsymbol{y}^{n+s}|\boldsymbol{z}^n) - \beta \log \frac{p_\theta(\boldsymbol{z}^n|\boldsymbol{X}^n)}{r_\theta(\boldsymbol{z}^n)} \right] \tag{1}$$

  To implement a multi-modal distribution for the prior $r_\theta(\boldsymbol{z})$, we employed the variational mixture of posteriors prior (VampPrior) in our original publication. The approximate prior $r_\theta(\boldsymbol{z})$ is a weighted mixture of different posteriors $p_\theta(\boldsymbol{z}|\boldsymbol{X})$ with pseudo-inputs $\{\boldsymbol{u}^k\}_{k=1}^K$ in lieu of $\boldsymbol{X}$:

  $$r_\theta(\boldsymbol{z}) = \sum_{k=1}^{K} \omega_k \; p_\theta(\boldsymbol{z}|\boldsymbol{u}^k) \tag{2}$$

  where K is the number of pseudo-inputs, $\boldsymbol{u}^k$ is a vector which has the same dimension as input $\boldsymbol{X}$, and ω_k represents the weight of $p_\theta(\boldsymbol{z}|\boldsymbol{u}^k)$ under the constraint $\sum_k \omega_k = 1$. The pseudo-inputs $\{\boldsymbol{u}^k\}$ and weights $\{\omega_k\}$ can be thought of the parameters of the prior, which are learned through backpropagation of the objective function (Eq. 1). However, in practice, we find that using representative-inputs $\{\boldsymbol{X}^k_{\text{rep}}\}_{k=1}^K$ instead of pseudo-inputs $\{\boldsymbol{u}^k\}_{k=1}^K$ has better performance. Thus, we modify the VampPrior algorithm and obtain the new approximate prior $r_\theta(\boldsymbol{z})$:

  $$r_\theta(\boldsymbol{z}) = \sum_{k=1}^{K} \omega_k \; p_\theta(\boldsymbol{z}|\boldsymbol{X}^k_{\text{rep}}). \tag{3}$$

  The algorithm to find the representative-inputs $\{\boldsymbol{X}^k_{\text{rep}}\}_{k=1}^K$ is as follow: Initially, randomly pick up one sample for each initial state as our initial representative-inputs $\{\boldsymbol{X}^k_{\text{rep}}\}_{k=1}^K$, where K will be equal to the number of initial states. After each iteration of model training and state label refinement, we can map all the input samples to the learned RC space. In this way, we can calculate the center of each newly refined non-empty metastable state in the learned RC space, and identify the nearest sample of each center based on the Euclidean distance in the RC space. These input samples will then be treated as new representative inputs $\{\boldsymbol{X}^k_{\text{rep}}\}_{k=1}^K$ in next iteration. In this way, our algorithm can automatically adjust the representative inputs including K.

- The new code will stop the training if only one metastable state is found.
- A demo is also added to better illustrate how to use the code.