Discovering substates in protein folding simulations

Alexander Brace
Department of Computer Science
University of Chicago
Chicago, IL 60637
`abrace@uchicago.edu`

Abstract

Proteins take on a variety of intermediate conformations during the process of folding. Characterizing these transient substates and predicting misfolds can elucidate the underlying biophysical basis of protein function. However, due to the short-lived nature of these events, experimental observation remains out of reach requiring researches to use molecular dynamics (MD) simulations to model folding pathways. MD simulations produce vast amounts of data which prompts the use of data-driven techniques to aid in interpretation and analysis. In this work, we develop a principled approach to decompose MD simulation data into biophyiscally relevent clusters enabling the automatic discovery of conformational substates within protein folding simulations. Extending the singular value decomposition with a traditional clustering method, k-means, we design an efficient algorithm for hierarchical clustering of MD data. The code for this work is provided on GitHub[1].

1 Introduction

Molecular dynamics (MD) simulations are an essential tool used for gaining mechanistic insight into biological phenomena such as protein folding or protein ligand complexes lending support to applications such as drug design [1, 2]. MD simulations work by numerically integrating Newton's second law of motion $(F = ma)$ to evolve each atom of a biomolecular system forward in time. Such atomistic and temporal resolution (femtosecond 10^{-15}s) provides valuable insights that are otherwise impossible to observe via physical experiments. Despite great advancement towards simulating large scale systems [3], deriving scientific insight from the resultant terascale (and emerging exascale) datasets remains challenging. This problem is compounded by the fact that MD simulations can become trapped in local minima within the free energy landscape [4] leading to wasted computation – a non-trivial problem as MD simulations are a primary application for supercomputing centers [5]. As such, there is strong motivation to develop efficient algorithms for clustering MD trajectory data into relevant conformational substates which can aid in understanding large simulation data as well as take an active role towards steering ensemble MD simulations to sample portions of the free energy landscape that may be unreachable with traditional sampling [6, 7].

2 Related Work

Several methods have been proposed to learn low dimensional vector representations of MD trajectories which accurately capture conformational substates important to biological function, i.e. low energy states separated by a higher free energy barrier. Ramanathan et al. proposes quasi-anharmonic

[1]`https://github.com/braceal/CMSC-35300-Final-Project`

analysis (QAA) which models the higher-order statistics of the protein's atomic coordinate fluctuations to discover a hierarchy of substates [8]. While structural characterization is important for understanding the range of motion, the temporal dimension of MD simulations can provide further insights into transitions between substates. To resolve the temporal structure of the data, Naritomi et al. use time-structured based independent component analysis (t-ICA) which extends principal component analysis (PCA) by discovering components that are as statistically independent as possible [9]. Building on these approaches, techniques for modeling the nonlinear relationships within MD data have been proposed such as in Bhowmik et al. which develops a variational autoencoder to compress contact maps into a latent manifold using convolutional layers [10]. Taken together, these methods aid scientists in mining large simulation datasets for facts related to protein structure and function that have scientific and medical value.

Similar to Ramanathan et al. [8], in this work, we focus on developing an efficient method for hierarchical clustering of MD trajectory data which leverages the singular value decomposition (SVD) and the k-means clustering algorithm to recursively decompose a full trajectory into it's constituent substates.

3 Methods

Our algorithm computes a hierarchical decomposition of MD simulation data using recursive SVD calls and k-means clustering to uncover potential substates at each level of a hierarchy. We summarize the **algorithm** as follows:

1. Run SVD on the full simulation trajectory
2. Project the trajectory onto r components using the V matrix
3. Cluster the projections into k clusters using k-means
4. Check termination condition to see if recursion has exceeded ℓ levels
5. Repeat steps 1-4 on the clustered trajectory frames recursively

We note that r is typically chosen to be a low-rank approximation to capture the effective rank of the data. We select a fixed value of k to recursively decompose the trajectory into a tree of substates using the k-means algorithm implemented in scikit-learn [11]. In this paper, we set $k = 3$ to minimize the size of the tree while still keeping it large enough to express the underlying substates as separate clusters. We implemented our own version of the SVD using the power method [12] and checked correctness against the SVD implemented in NumPy using Linear Algebra PACKage (LAPACK) [13] as well as by asserting that the input matrix can be reconstructed from the SVD factorization. The power method approach to compute the SVD iteratively decomposes the input data matrix $\mathbf{A} \in \mathbb{R}^{n \times p}$ by first using power iteration to compute $\vec{u_i}, \sigma_i, \vec{v_i}^\top$ and then subtracting off the component projection, i.e. $\mathbf{A} - \sigma_i \vec{u_i} \vec{v_i}^\top$, repeating this until i reaches $\min(n, p)$. We opted to use the power method for ease of implementation, but note that more efficient methods exist such as the one-sided Jacobi algorithm [14].

3.1 Features and preprocessing

Part of the challenge of analyzing MD datasets is the sheer number of atoms being simulated, which is approaching O(1B) for large scale systems [3]. To featurize the MD trajectory we consider a reduced set of atoms consisting of the C^α atom of each amino-acid residue, producing a dataset with dimension $(N, 3X)$, where N is the number of frames in the trajectory and $3X$ represents the 3D Cartesian coordinates of each C^α atom. Using the central C^α atoms of each residue provides a minimal view of the data which can be used to infer secondary structure important for identifying conformational substates [15]. Once the raw positions are extracted from the trajectory using the MDAnalysis Python package [16], we align them to the mean position using a kabsch-based iterative means alignment algorithm implemented in the mdlearn[2] Python package [17].

[2]The author of this paper wrote the mdlearn package.

4 Results

To evaluate our method, we focus on a O(100 μs) trajectory (approximately 1 million frames) of the fast-folding protein FSD-EY (PDB ID: 1FME), referred to here as BBA, which adopts several intermediate states on it's way to a canonical $\beta\beta\alpha$ fold [18]. BBA has 28 amino-acid residues and undergoes several conformational changes on it's way to it's folded state, making it an ideal candidate to evaluate our approach. To interpret the efficacy of our clustering method, we compute the root mean squared deviation (RMSD) [19] (Eq. 1) of each conformer $\mathbf{x}(t)$ to the native folded state $\mathbf{x}^{ref}$, where t denotes the trajectory time step. In essence, RMSD computes a coordinate normalized euclidean distance between two conformers, hence, in our case a small RMSD indicates being close to the folded state, and a large RMSD indicates being further away in conformational space, e.g. unfolded. We note that the molecular weight of each C^{α} atom is the same which allows us to ignore this factor.

$$\rho(t) = \sqrt{\frac{1}{N_{atoms}} \sum_{i=1}^{N_{atoms}} (\mathbf{x}_i(t) - \mathbf{x}_i^{ref})^2} \tag{1}$$

4.1 Scientific performance

We illustrate our hierarchical decomposition of the BBA folding trajectory in Figure 1 by projecting the conformers onto the first 3 principal components at several levels of the hierarchy painted by the conformers RMSD to the native state. Level-0 represents the full trajectory, from which projections of the data onto the top 20 principal components are clustered with k-means (shown in black circles). Each cluster gets expanded recursively for 3 levels (level-2 omitted for brevity) for which we see that the clusters become further distilled into regions of similar RMSD indicative of structurally similar substates. Sampling several conformers from representative clusters shown in level-3, we observe distinct secondary structure formation along the folded pathway (shown top to bottom and labeled by the RMSD to native state). The bottom most structure shows the canonical $\beta\beta\alpha$ fold, i.e. two β-pleated sheets and a single α-helix, visualized using the PyMOL software [20].

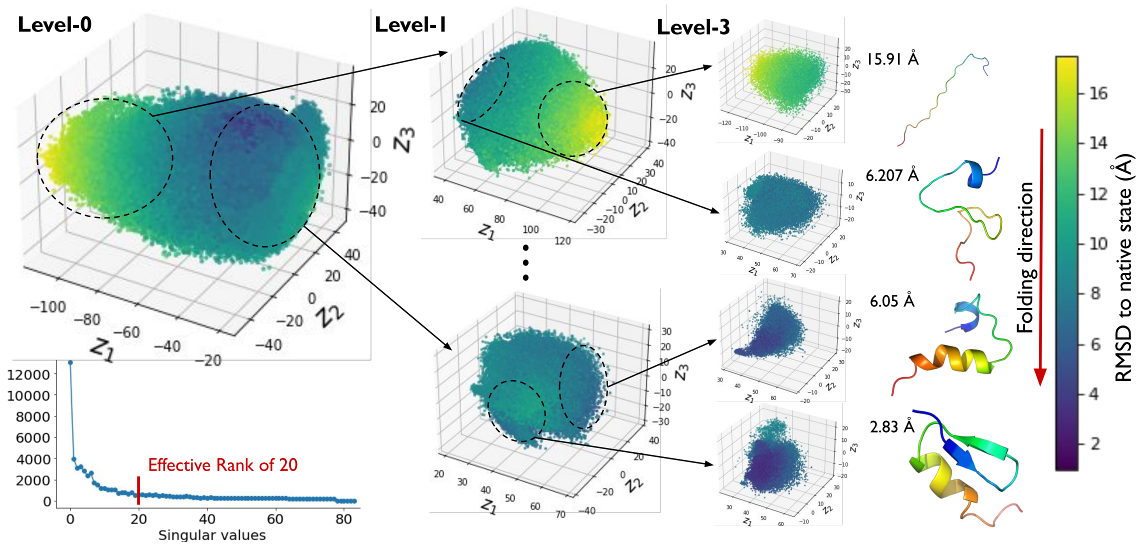
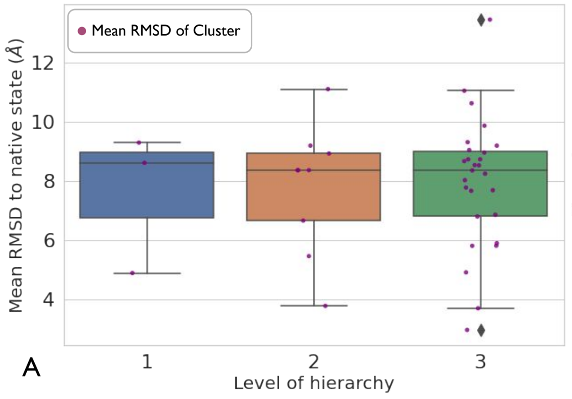


Figure 1: Hierarchical decomposition of the BBA folding trajectory reveals distinct clusters of structurally similar conformers traversing from extremely unfolded states (top most structure) to folded states (bottom structure).

To effectively cluster the data, we used $k = 3$ along with a $\ell = 3$ level hierarchy, however, visualizing the resultant k^{ℓ} projection plots and spectrum's at each level is tedious. For this reason, we summarize these clusters in panel **A** of Figure 2 by computing the average RMSD across all conformers in a given cluster. Interestingly, we observe that as the level increases, the existence of clusters with



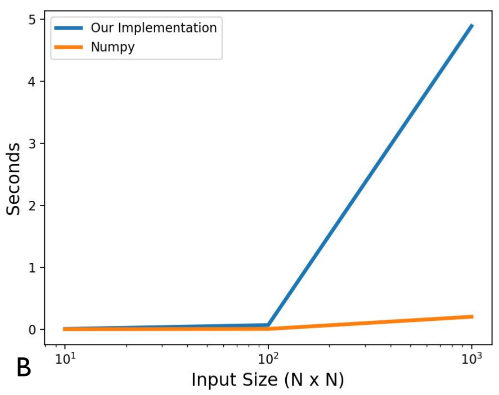


Figure 2: Panel **A** visualizes the distribution of average intra-cluster root mean squared deviation (RMSD) to native state values at each level of the hierarchy. Each level of the hierarchy pushes further into the tails of the RMSD distribution providing evidence that our method accurately discovers biophysically relevent substates. Panel **B** depicts the computational performance of our SVD approach compared to the NumPy implementation on matrices up to dimension $10^3 \times 10^3$. We can see that after $N = 10^2$, the NumPy implementation outperforms ours by a factor of 5.

both high and low mean RMSD values increases. This indicates that the hierarchical clustering is decomposing the simulation trajectory into regions of structural similarity which have biophysical relevance (e.g. close to native state) enabling automatic discovery of conformational substates. We also observe a less favorable trend displaying the mean RMSD values clustering near 7-10Å which may suggest that some clusters are redundant. We leave it to future work to address this issue but posit that a more dynamic selection of k tuned to the spectrum of the SVD for each discovered substate may alleviate this issue.

4.2 Computational performance

To scale our SVD implementation to large input matrices, we utilized the Numba Python package [21] which uses the LLVM compiler to run just-in-time (JIT) compilation of certain NumPy operations. We also enabled automatic parallelization of core NumPy kernels, such as matrix multiplication and arithmetic operations, and activated automatic relaxation of numerical rigour to enhance performance. Despite these optimizations, for large input sizes $(N > 10^6)$ our method crashes when trying to allocate the U matrix. To avoid out-of-memory errors, we evenly subsample the data before running SVD until our first input dimension is less than 10^5 (the feature dimension is always 3×28 in the case of BBA). Shown in panel **B** of Figure 2, we measure performance against the NumPy SVD implementation by comparing the time it takes to compute the SVD of square matrices up to $N = 10^3$. We attribute the majority of the difference in computational performance of our algorithm and the NumPy version to the efficiency of the divide and conquer algorithm used by the LAPACK dgesdd kernel versus the iterative power method approach we employ. However, we note that with subsampling our algorithm still runs efficiently for BBA and future work with this method can safely interchange the NumPy SVD function with ours.

5 Discussion & Conclusion

In this paper, we have developed a hierarchical approach to decompose MD trajectories into biophysically relevant clusters. We demonstrate our approach on the BBA folding trajectory for which we show that a combination of SVD and k-means can be used to automatically discover diverse conformational substates implicated in protein folding. While our implementation was sufficient to analyze BBA, we note that scaling up to larger systems may benefit from computing a truncated SVD which stops after the effective rank has been detected. We also posit that the addition of intra-cluster coordinate alignment would likely allow variability within substates to be more accurately modeled which may help to characterize transient events important for protein folding. We leave these improvements to future work.

References

[1] Ron O Dror, Robert M Dirks, JP Grossman, Huafeng Xu, and David E Shaw. Biomolecular simulation: a computational microscope for molecular biology. *Annual review of biophysics*, 41:429–452, 2012.

[2] Scott A Hollingsworth and Ron O Dror. Molecular dynamics simulation for all. *Neuron*, 99(6):1129–1143, 2018.

[3] Abigail Dommer, Lorenzo Casalino, Fiona Kearns, Mia Rosenfeld, Nicholas Wauer, Surl-Hee Ahn, John Russo, Sofia Oliveira, Clare Morris, Anthony Bogetti, et al. # covidisairborne: Ai-enabled multiscale computational microscopy of delta sars-cov-2 in a respiratory aerosol. *bioRxiv*, 2021.

[4] Fabio Pietrucci. Strategies for the exploration of free energy landscapes: Unity in diversity and challenges ahead. *Reviews in Physics*, 2:32–45, 2017.

[5] Lorenzo Casalino, Abigail C Dommer, Zied Gaieb, Emilia P Barros, Terra Sztain, Surl-Hee Ahn, Anda Trifan, Alexander Brace, Anthony T Bogetti, Austin Clyde, et al. Ai-driven multiscale simulations illuminate mechanisms of sars-cov-2 spike dynamics. *The International Journal of High Performance Computing Applications*, 35(5):432–451, 2021.

[6] Alexander Brace, Igor Yakushin, Heng Ma, Anda Trifan, Todd Munson, Ian Foster, Arvind Ramanathan, Hyungro Lee, Matteo Turilli, and Shantenu Jha. Coupling streaming ai and hpc ensembles to achieve 100-1000x faster biomolecular simulations, 2021.

[7] Matthew C. Zwier, Joshua L. Adelman, Joseph W. Kaus, Adam J. Pratt, Kim F. Wong, Nicholas B. Rego, Ernesto Suárez, Steven Lettieri, David W. Wang, Michael Grabe, Daniel M. Zuckerman, and Lillian T. Chong. WESTPA: An interoperable, highly scalable software package for weighted ensemble simulation and analysis. *Journal of Chemical Theory and Computation*, 11(2):800–809, 2015.

[8] Arvind Ramanathan, Andrej J Savol, Christopher J Langmead, Pratul K Agarwal, and Chakra S Chennubhotla. Discovering conformational sub-states relevant to protein function. *PLoS One*, 6(1):e15827, 2011.

[9] Yusuke Naritomi and Sotaro Fuchigami. Slow dynamics in protein fluctuations revealed by time-structure based independent component analysis: the case of domain motions. *The Journal of chemical physics*, 134(6):02B617, 2011.

[10] Debsindhu Bhowmik, Shang Gao, Michael T Young, and Arvind Ramanathan. Deep clustering of protein folding simulations. *BMC bioinformatics*, 19(18):47–58, 2018.

[11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[12] Edo Liberty. Lecture 7: Singular value decomposition, 2013. `https://www.cs.yale.edu/homes/el327/datamining2013aFiles/07_singular_value_decomposition.pdf`.

[13] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition, 1999.

[14] Wajih Halim Boukaram, George Turkiyyah, Hatem Ltaief, and David E Keyes. Batched qr and svd algorithms on gpus with applications in hierarchical matrix compression. *Parallel Computing*, 74:19–33, 2018.

[15] Yang Zhang and Jeffrey Skolnick. Tm-align: a protein structure alignment algorithm based on the tm-score. *Nucleic acids research*, 33(7):2302–2309, 2005.

[16] Richard J Gowers, Max Linke, Jonathan Barnoud, Tyler John Edward Reddy, Manuel N Melo, Sean L Seyler, Jan Domanski, David L Dotson, Sébastien Buchoux, Ian M Kenney, et al. Mdanalysis: a python package for the rapid analysis of molecular dynamics simulations. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2019.

[17] Alexander Brace, Heng Ma, Austin Clyde, Debsindhu Bhowmik, Chakra Chennubhotla, and Arvind Ramanathan. mdlearn: Machine learning for molecular dynamics. `https://github.com/ramanathanlab/mdlearn`, 2021.

[18] Kresten Lindorff-Larsen, Stefano Piana, Ron O Dror, and David E Shaw. How fast-folding proteins fold. *Science*, 334(6055):517–520, 2011.

[19] Douglas L Theobald. Rapid calculation of rmsds using a quaternion-based characteristic polynomial. *Acta Crystallographica Section A: Foundations of Crystallography*, 61(4):478–480, 2005.

[20] Schrödinger, LLC. The PyMOL molecular graphics system, version 1.8. November 2015.

[21] Siu Kwan Lam, Antoine Pitrou, and Stanley Seibert. Numba: A llvm-based python jit compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, pages 1–6, 2015.