LLP Research Final Report: Webb Lab

Pulkith Paruchuri

I. Introduction

Enzymes are vital proteins that accelerate chemical reactions. They act as biological catalysts and can be found in nearly all life forms. Certain non-native enzymes can be introduced into human bodies, for neural regeneration, for example, although these enzymes often cannot survive long in non-native environments. Polymers, chains of repeating macromolecules, can bind with enzymes to create Polymer-Protein Hybrids that bolster stability, allowing enzymes to survive, or even thrive in hostile or non-native environments. However, a certain polymer can be effective in one enzyme, and ineffective in another. Thus, efficiently discovering fruitful polymers for each enzyme presents a challenge. A Machine Learning (ML) model is a collection of algorithms, that makes predictions based on provided data. ML models have proven to be effective in this regard. However, the performance of Machine Learning algorithms correlates strongly with the initial data provided. The goal of this research is to discover, implement, and compare various approaches for generating strong datasets that would ideally engender strong model performance.

II. Methods

Each polymer can be represented as an aggregation of a certain number of monomers. For this study, polymers were described uniquely as the fraction of incorporation (FOI) of their constituent monomers. The FOI of a monomer, x, in a polymer, y, is the ratio of the number of x monomers to the total number of monomers in y. Consequently, a polymer can be distinctively represented as a point in an *M*-dimension Elucidian space, where *M* denotes the number of unique monomers in the polymer. More formally, each polymer, z, satisfies the conditions:

$$z \in [0,1]^M, \sum_{i=1}^{M} z_i = 1$$

The space of all possible polymers makes a unit $(M-1)$-*simplex* or a generalized unit triangle in M-dimensional hyperspace. To generate a dataset for the ML model, a sample of points from the simplex is required. A strong dataset, in this case, was expected to be one that had points distributed relatively evenly across the space. Thus the engineering goal of the study was reduced to sampling well-spaced points on a unit simplex. Random sampling points would not be effective, since random sampling tends to create clusters.

For this study, various algorithms were used to generate N points, including:

I. Maximin(Construction): generating N points sequentially, maximizing the distance from any point to its neighbors. At a point where k points have been generated, this can be described as satisfying the condition below to generate the $(K+1)$th point $z^{(k+1)}$

$$\begin{aligned} &\text{Maximize} \min_{i=1}^{k} \left\| \mathbf{z}^{(k+1)} - \mathbf{z}^{(i)} \right\| \\ &\text{subject to} \sum_{m=1}^{M} z_m^{(k+1)} = 1 \\ &\qquad z_m^{(k+1)} \geq 0, \quad m = 1, \ldots, M \end{aligned}$$

II. Latin HyperCube: discretizing the space into N regions in each dimension, and sampling once within each region of each dimension. Then the samples in each dimension of the same index of the N partitions are merged to generate N points. This devolves to being similar to a Latin Square in a 2-dimensional example, which ensures there is only one point in each row and column.

III. Reduction: Generating simplex boundary points via Das-Dennis, generating excessive points by random, removing points until N points remain via Maximin and improving the final distribution through K-means.

IV. Energy: using energy functions that mimic natural phenomena, using Adam Optimization to accelerate computation. Two energy methods were used: Lennard-Jones Potentials, and Reisz S-Energy. Reisz S-Energy is a generalization of potential energy, where the potential between two points, z^i and z^j is defined as

$$U\left(\mathbf{z}^{(i)}, \mathbf{z}^{(j)}\right) = \frac{1}{\left\| \mathbf{z}^{(i)} - \mathbf{z}^{(j)} \right\|^s}$$

With the overall s-energy of a region being written as

$$U_T(\mathbf{z}) = \frac{1}{2} \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{1}{\left\| \mathbf{z}^{(i)} - \mathbf{z}^{(j)} \right\|^s}, \quad \mathbf{z} \in \mathbb{R}^{n \times M}$$

We then aim to minimize U_T. To reduce the computational cost, we take the logarithm of U_T and compute the partial derivative $F_E = log(U_T)$ with respect to $z_m^{(i)}$, as shown below

$$\frac{\partial F_E}{\partial z_m^{(i)}} = -\frac{d}{U_T} \left[\sum_{\substack{j=1, \\ j \neq i}}^{n} \frac{\left(z_m^{(i)} - z_m^{(j)} \right)}{\left\| \mathbf{z}^{(i)} - \mathbf{z}^{(j)} \right\|^{s+2}} \right]$$

Between the 3,000 iterations of Adam gradient-based optimizations, points outside the simplex were projected back onto the simplex, such that $z_m^{(i)} \geq 0 \forall i, m$. Furthermore, within each iteration, each gradient was projected onto the unit simplex to ensure $\sum_{m=1}^{M} z_m^{(i)} = 1 \forall i$.

Points were generated not only on a unit simplex, but also on a M-dimesional unit hypercube, and a M-dimensional unit hypersphere, then mapped onto the unit simplex. These were to be used for comparative analysis.

Furthermore, various methods were used to evaluate how well-spaced a selected sample of points was. This includes (i) Spheres: generating hyperspheres around each point and calculating the hypervolume filled, (ii) VGM: calculating the variance of the geometric means of each point to its M nearest neighbors, and (iii) D-min: calculating the minimum distance to the closest point for any point. For Sphere Evaluation, Monto Caro Approximation was used to reduce computational cost. All evaluation metrics were compared against a uniform random sampling of the simplex as a baseline.

Finally, each polymer was extended to not only represent the FOIs of its constituent monomers, but also the Degree of Polymerization(DP) of the polymer, which was normalized to be within $[0, 1]$. The DP is independent of the FOIs. This converts each polymer from the previously described representation of a $(M-1)$-simplex in a M-dimensional hyperspace, to a $(M-1)$ simplex-prism in a $(M+1)$-dimensional hyperspace. Space-Filling on the simplex-prism was done in two manners, (i) sampling the entire simplex-prism via energy-based methods, and (ii) sampling on the simplex via energy-based methods, and sampling on the prism portion via Reduction, Maximin, Duplications, or Latin Hypercube.

All methods were simulated in Python with hardware GPU acceleration, using Mathplotlib for visualization, and Pymoo for certain methods to generate points.

III. Results

Generating points directly on the simplex was found to generally outperform scenarios where points were generated on a hypersphere or hypercube, then mapped onto the simplex. Marginal distributions when sampling on a hypersphere can be seen in Figure 5. Thus, the rest of this section discusses performance when generating directly on a simplex. While most methods outperformed uniform random sampling, energy-based methods typically performed the best on all evaluation metrics, as seen in Figure 1. Specifically, Riesz s-energy performed the best, with Lennard Jones potentials being close behind. These energy methods typically created quite well-spaced and uniform distributions. Maximin, and variations of Maximin, tended to generate points that were near the edges, which is suboptimal, although this behavior tended to be less detrimental as $N \to \infty$. Latin Hypercube could not be implemented effectively on a Simplex to yield results comparable to energy-based methods.

When space was altered to include the DP dimension, energy methods continued to perform the best. Sampling on only the simplex via Energy, then subsequently sampling on the prism portion via other methods, tended to create subprime distributions as seen in Figure 3, while sampling on the entire simplex-prism via energy, created well-distributed points, as seen in Figure 4.

Furthermore, generation was bounded by $O(N^2)$ time complexity, usually taking just a few seconds for the number of dimensions($M < 7$) used in this study. Evaluation of the metrics can be seen for various selections of N for each method in Figure 2.

IV. Conclusions

Energy-based methods, such as Reisz s-energy and Lennard-Jones Potentials, which are inspired by nature, have proven to be effective for generating points on a simplex and simplex-prism for use in ML dataset generation and seeding. The success of such methods is expected to be tested in specific regard to finding polymers for Polymer-Protein Hyrbid creation in collaboration with Rutgers University at a later date. However, it is expected that such an approach is to yield more performant machine learning models relative to current approaches based on the completed simulations.

V. Figures

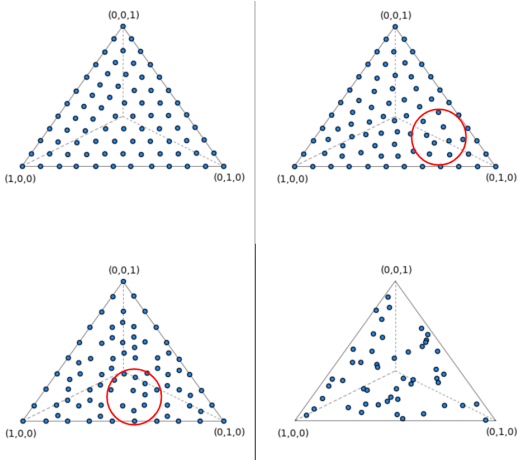

Figure 1. Examples of points generated via the various methods. Top-Left is Energy, Top-Right is Reduction, Bottom-Left is Maximin(Construction), and bottom-Right is uniform Random. Red circles indicate regions of significant non-uniformity.

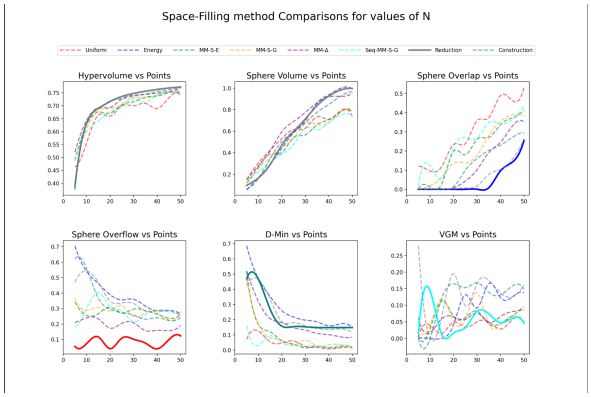

Figure 2: Metrics of Evaluation for various values of N. Bold, solid lines represent the best-performing method at $N = 50$. MM-S-E, MM-S-G, MM-Δ, and Seq-MM-S-G, are variations of Maximin that generate points on hypercubes and hyperspheres. Sphere Volume, Sphere Overlap, and Sphere Overflow are child metrics that derive from the Sphere Hypervolume metric of evaluation.

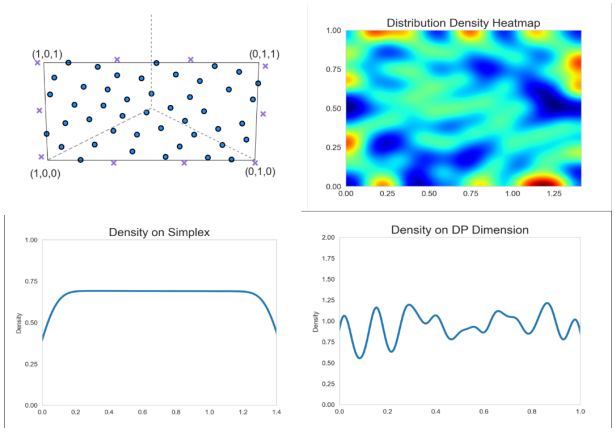

Figure 3: Distributions after generating points on a 1- simplex via energy, and on the DP dimension via other functions. The X-axis on the heatmaps and density distributions represents the length of the 1-simplex, while the y-axis on the density distributions represents the density. The y-axis on the heatmap represents the DP dimension.

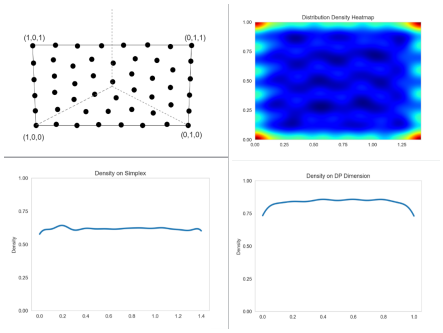

Figure 4: Distributions after generating points on a 1- simplex via energy, and on the DP dimension via energy. The X-axis on the heatmaps and density distributions represents the length of the 1-simplex, while the y-axis on the density distributions represents the density. The y-axis on the heatmap represents the DP dimension.

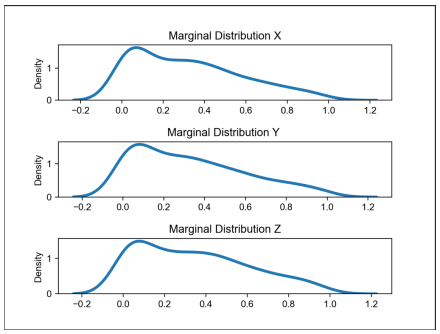

Figure 5: Marginal Distrubtiions of the X, Y, and Z axes after generating points on a hypersphere via Maximin, and mapping it onto a simplex. A well-spaced distribution would have more uniform marginal distributions.