MENG 25620 Final Assignment Write Up

Achyu Menon, Allen Hu, Joe Aulicino, Daniel Mark

March 2022

1 Description of Implementation

Our group has set out to investigate and implement how reinforcement learning can be used in the context of protein folding to to find the free energy minimizing configuration of hydrophobic and hydrophilic residues. According to the original paper by Jafari et al., the problem of predicting the structure of a protein is currently a computationally expensive operation, whereby new methods based on algorithms such as RL is being implemented as a more efficient prediction method. In this implementation, we are able to replicate a simple version of the agent-critic reinforcement learning algorithm in a $n \times n$ grid. In our environment, we set up a grid dimension that utilizes a $n \times n$ space where $n =$ (Length of Residue List $\times 2 + 1$). The agent starts in the center of the grid and begins to move into any of the 4 adjacent directions to 'place' either a hydrophobic or hydrophilic residue and is punished or rewarded at each step. An epoch ends when the agent exhausts all the residues provided, and epochs continue until the agent converges.

In our formulation, the critic (or reward) space is as follows:

Action	Reward
If move is invalid	0.01
If the move is valid	$-E$
If the move is anything else	0.1

Notice that we do not "block" invalid moves, rather we allow the agent to play invalid moves, but give a relatively negligible reward, which incentivizes the agent away from these types of invalid plays (Known as self-avoidance). Finally, we will be mainly be using the result of the valid action (with reward $-E$) in order to train an agent to move in a way that minimizes the energy of the given residues.[1]

Finally, the agent is trained using Q-learning in our value function iteration process, whereby mappings of state/action pairs are found with reference to their rewards using the metrics above. In this way, the agent then not only looks at the immediate reward of a given step, rather also maximizes future rewards given a discount value γ.

[1] This energy $-E$ is calculated using the energy functional of a particular move. For any residue $P_a \in \{P_{-1}, P_{+1}\}$ with their energy $E_a \in \{-1, 1\}$ in a $n \times n$ grid, the free energy that is calculated (E) for placing a specific residue P_i is defined by $E = \sum_{j=1}^{n} \sum_{k=1}^{n} E_{j,k}$, where $E_{j,k} = E_i$ if $P_{j,k} = P_i$, and $E_{j,k} = 0$ otherwise.

2 Description of the Experiments

We are able to create an experiment set where we essentially applied a 'horse-race' between two a random placement method and our agent. The main metric that we are interested in looking at is the relative MAE against a globally optimizing solution (whereby we calculate all possibilities and take the minimizing value). Specifically, the comparison is done by selecting 10 random proteins with lengths 3, 5, and 7; then folding them using the agent and a random benchmark which does arbitrary actions. The error is mean absolute relative error and is compared against the global minimum best fold. The model is clearing converging, and definitely learns the self-avoiding behavior.

3 Results and Further Discussions

The main results are visualized displayed below

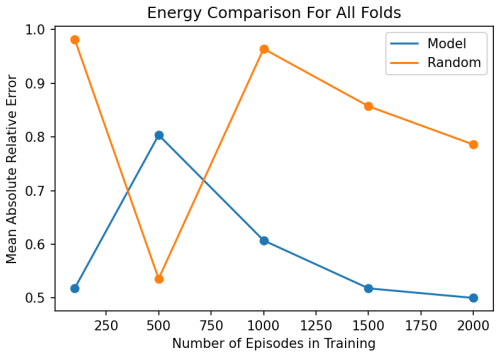

Figure 1: the MAE of the two models relative to the globally optimized solution.

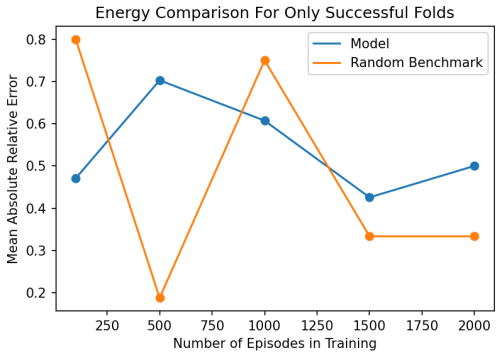

Figure 2: the MAE of the two models filtering out invalid folding

In the "all folds" graph, folds by the random benchmark and the model which are invalid are given an energy of 0. Whereas in the "only successful folds" these are screened out. One important fact presented in the second figures is that the the model still needs to learn how to fold better, as random folds (when they are at least valid) is often more successful, so higher training epochs might be needed in this exercise. Nevertheless, unlike the random folds, we can see how the model grows in a predictable manner and generally gets better with increasing the epochs.

Furthermore, the number of invalid folds in the 10 different random proteins between the two models are given below. From the graph below, we can see how our agent is able to learn how to avoid invalid folding based on the reward mechanisms that were implemented. In particular, we see that while the agent as still not learned how to more efficiently fold different proteins in our model, we are able to see how the agent already has learned how to not place multiple residues in the same cell. This shows that the 'invalid move' portion of our reward mechanism works as intended.

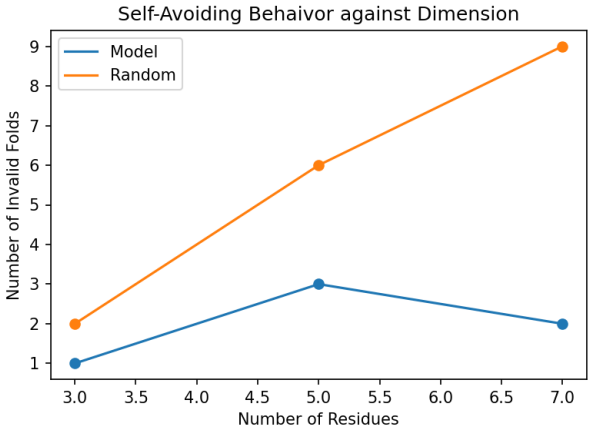

Figure 3: The number of invalid folds with respect to different protein lengths for the two different models

3.1 What have we learned

One key area that we have learned in this exercise is the importance of implementing a computationally efficient environment. As our implementation was essentially a 'fool proof' implementation, it was quite computationally expensive when actually training the model. For example, when we have 7 residues, that would imply that our environment would be a 15×15 grid already. It is easy to see how this increase in environment size can easily become very expensive when training on longer residue sets.

That being said, it is also worth noting the time advantages that comes with training an agent relative to the globally optimal solution. This echos the theme presented in class, whereby globally optimal solutions are computationally expensive.

Furthermore, unlike solving for the configurations in brute force, our agent is able to apply what it has 'learned' in a training residue list of length l, and residues r and predict the structure of a residue list with length $l' \neq l$ and residue set $r' \neq r$.

3.2 What we would do next

In line with our original proposal, a natural extension of this project is to go from 2D to 3D. This was difficult to implement in this short time frame as the curse of dimensionality made this extension nearly impossible with the hardware we have on hand. However, it is needless to say that because a 3D space is a more accurate representation of protein folding, this would allow us to create more accurate representations.

Another easy improvement of our design which will make the previous extension more feasible is by redesigning the environment space. As noted in our "what did we learn" section, the main issue of our environment was the large grid which later became quite expensive. This large grid means that we could have situations where the final structure could be linearly translated around the grid, which gives us "new" results without actually looking at new conformations.

4 References

Jafari, R., amp; Javidi, M. M. (2020). Solving the protein folding problem in hydrophobic-polar model using Deep Reinforcement Learning. *SN Applied Sciences* , 2(2). https://doi.org/10.1007/s42452-020-2012-0