Using machine learning to design adeno-associated virus capsids with high likelihood of viral assembly

Cuong T. To *[1], Christian Wirsching[1], Andrew D. Marques[2], and Sergei Zolotukhin[3]

[1]Cinference
[2]Department of Microbiology, Perelman School of Medicine, University of Pennsylvania, Philadelphia, PA 19104, USA
[3]Department of Pediatrics, Division of Cellular and Molecular Therapy, University of Florida, Gainesville, FL 32608, USA

June 10, 2021

Abstract

We study the application of machine learning in designing adeno-associated virus (AAV) capsid sequences with high likelihood of viral assembly, i.e. capsid viability. Specifically, we design and implement Origami, a model-based optimization algorithm, to identify highly viable capsid sequences within the vast space of 20^{33} possibilities. Our evaluation shows that Origami performs well in terms of optimality and diversity of model-designed sequences. Moreover, these sequences are ranked according to their viability score. This helps designing experiments given budget constraint.

1 Introduction

AAVs are the most used viral vectors for gene therapy and considered safe [13]. Nevertheless, it remains a challenge to improve its different properties such as immunogenicity or transduction efficacy. Directed evolution [1] has been used to engineer new recombinant AAVs [5, 8], or more concretely, to design their capsid protein. This is an iterative process which mimics how nature evolves. A library of AAV variants is created by either mutagenesis on one AAV serotype or shuffling of the AAV capsid gene among different serotypes.

*cuong@cinference.net

This library is then subjected to a selection pressure to screen for their function. The top performing variants are selected for the next round of mutagenesis or capsid shuffling. This process is repeated until selected AAV variants satisfy optimization requirements. These variants could then be validated further either in animals or clinical studies.

Unfortunately, the traditional directed evolution has three shortcomings. First, the process does not exploit all the experimental data; the information about the performance of non-selected variants are discarded in each iteration. Second, directed evolution typically starts from a recombinant AAV and a library is built with few mutations per variant. This causes the optimized variants remaining to be similar to the original recombinant AAV. Hence, they are likely to be suboptimal. Finally, because of the nature of experiments, directed evolution is time-consuming and costly. This limits the number of experiment rounds that can be done.

Machine learning (ML)-guided directed evolution [2, 10, 14] could be a solution for the aforementioned problems. We can use the screening dataset to train a predictive model to approximate the mapping from protein sequences to their function, aka fitness landscape [10]. This predictive model can screen virtually variants to save time and cost of experiments, assuming that the approximation is good enough. The virtual screening results are used by model-based design algorithms [3, 9, 12], which explore the sequence space to find optimal sequences. These algorithms are aptly called *explorers*.

In [7], the authors train ML models to predict whether an AAV2 capsid sequence will form a virus particle once produced. The idea is that, with an accurate predictive model, researchers can virtually screen any capsid sequence for its viability without performing actual experiments. This enables them to build better libraries quickly.

While a predictive model helps filtering out non-viable sequences, it does not recommend which sequences to try out to begin with. This is a difficult problem due to the unimaginably large size of the sequence space. In [7], the number of mutational position is 33, which means the corresponding sequence space cardinality is 20^{33}! We definitely need smart algorithms to design de novo protein sequences.

In the remainder of this report, Section 2 discusses our approach in detail. Section 3 presents our evaluation. Finally, Section 4 discusses possible next steps.

2 Our approach

First, we define some notations and introduce our approach of using model-based optimization [2, 3, 9, 12], where a distribution of protein sequences conditioned on desired phenotypes is computed. Second, we adapt this approach to solve the problem of designing viable AAV2 capsid.

2.1 Notation

Let $x \in \mathcal{A}^L$ denote an input sequence, where $\mathcal{A}$ and L are the alphabet and the sequence length, respectively. For example, the alphabet of amino acids is $\{I, L, \ldots, T, P\}$. $h(x)$ denotes an oracle that predicts the phenotype of a given sequence, $\hat{y} = h(x)$. $\Gamma \subset \mathbb{R}$ denotes the requirement set that characterizes the desired phenotype. For example, in optimizing GFP protein, $\Gamma = y_{fluorescence} \geq y^*$, where y^* is a scalar threshold.

2.2 AAV2 viral assembly problem

With the defined notation, we view the protein design problem as one of computing a distribution over the protein sequence space conditioned on the desired phenotype, $p(\mathbf{x}|\Gamma)$. Since we want our AAV2 capsid sequences to be viable, our requirement set takes the form $\Gamma = \{y = 1\}$. The conditional distribution $p(\mathbf{x}|\Gamma)$ needs to be computed, and then desired sequences are sampled from it. Let us rewrite $p(\mathbf{x}|\Gamma)$ as follows.

$$p(\mathbf{x}|\Gamma, \theta^{(0)}) = \frac{p(\Gamma|\mathbf{x})p(\mathbf{x}|\theta^{(0)})}{p(\Gamma|\theta^{(0)})},$$

where $p(\Gamma = \{y = 1\}|\mathbf{x})$ can be computed by an oracle. $p(\mathbf{x}|\theta^0)$ is a generative model that can be learnt from AAV2 capsid sequences. Unfortunately, it is not feasible to compute this condition distribution exactly as the denominator $p(\Gamma|\theta^0) = \int p(\Gamma|\mathbf{x})p(\mathbf{x}|\theta^{(0)})d\mathbf{x}$ is intractable due the the high dimensionality of the sequence space.

To overcome the aforementioned intractability, we use the Cross Entropy Method [11], which produces a sequence of generative models, $p(\mathbf{x}|\theta^{(t)})$. Specifically, at each step t, it solves the optimization problem:

$$\theta^* = \arg\max_{\theta^{(t)}} w(\mathbf{x}) \sum_{\mathbf{x} \in S_{t-1}} \log p(\mathbf{x}|\theta^{(t)}),$$

where $S_{t-1} = \{\mathbf{x}|\mathbf{x} \sim p(\mathbf{x}|\theta^{(t-1)})\}$ and $w(\mathbf{x})$ weighting function. A popular choice is $w(\mathbf{x}) = 1|\mathbf{x} \in U_{t-1}$, where U_{t-1} is a set of sequences whose scores, predicted by the oracle, are above a threshold.

Once we have the final generative model, $p(\mathbf{x}|\theta^{(t)})$, we can sample from it to generate AAV2 capsid sequences with good viability.

3 Evaluation

3.1 Dataset

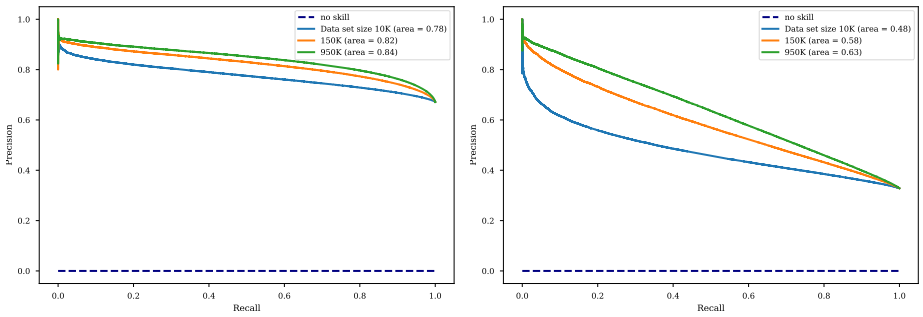
The original CapLib8 data set [7] has about 21M data points of the form (x, y), where $\mathbf{x}$ is a sequence of 33 amino acids. These are mutations on the VP1 capsid protein of AAV2. $y \in \{1, 0\}$ where 1 and 0 stands for viral assembly and non-viral assembly, respectively.

The CapLib8 data set has a class imbalance between non-viable and viable sequences; there are more viable sequences than non-viable ones. As we will see, this affects the oracle performance with respect to the non-viable capsid sequences.

3.2 Oracle

We use an LSTM [4] for our predictive model. We sample from the CapLib8 dataset randomly to create datasets of different sizes. The same LSTM architecture is trained on them to evaluate how the performance varies with the size of training data set.

We notice that the recall score for non-viable capsid sequences improves with more data (Figure 1b). This is important because for Origami to work, we need to prevent training our generative model with false positive sequences, i.e., sequences that are predicted to be viable but are in fact non-viable. The effect of the data set size is not significant for the oracle performance with respect to viable sequences (Figure 1a).



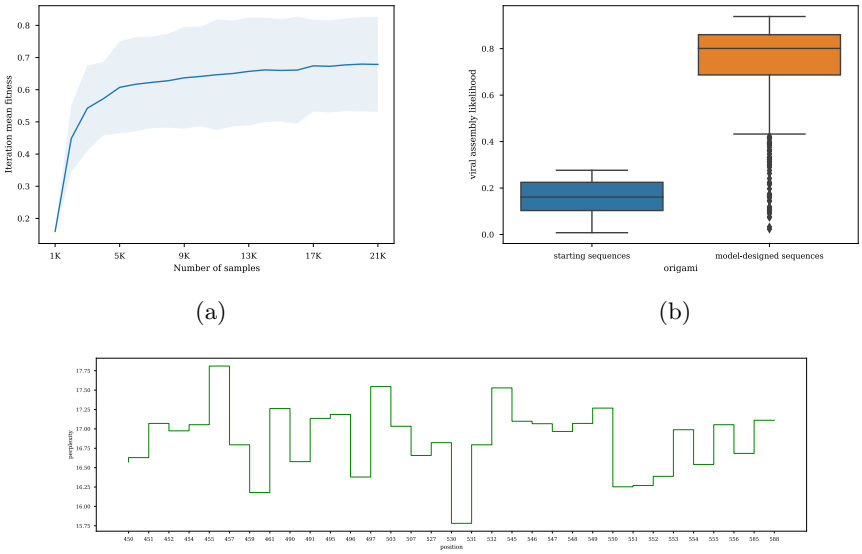
(a) (b)

Figure 1: The quality of oracles with different training dataset sizes.(a) Precision recall curve with respect to viral assembly. (b) Precision recall curve with respect to non-viral assembly.

Finally, we create a balanced data set of total 700K data points with equal split between viable and non-viable AAV2 capsid sequences. We use this data set to train an oracle that will be used by our Origami algorithm to screen for capsid viability.

3.3 Viable sequences

We train a variational auto encoder (VAE) [6] as our generative model with tuned hyperparameters. To validate Origami, we perform viral assembly predictions on a test data set of 150K data points. Then we take those sequences in the bottom 20th percentile of prediction scores. These sequences are used as the starting sequences for Origami. We run Origami for 20 iterations. At the



(a) (b)

(c)

Figure 2: (a) Model-designed sequence mean fitness improves with each iteration. Note that the x-axis shows the cumulative number of samples over iterations. The shade area shows the 95% confidence interval. (b) Starting sequences vs. model-designed ones in terms of viability score, i.e. the probability of viral assembly provided by the oracle. (c) The perplexity of distribution of amino acids at each of the 33 mutational positions. The higher the perplexity, the higher the diversity.

end, the model-designed sequences have an average assembly likelihood of 0.75, which is 4.7 times better than that of the starting ones (Figure 2b).

Figure 2c the perplexity at each position of the mutated AAV2 capsid sequence. The perplexity value can be viewed as "branching factor". In other words, it measures the diversity in terms of amino acids found at each position. We can see that the model-designed sequences are quite diversified.

We also evaluate our optimization procedure using the metric *iteration mean fitness*, which is the average fitness score of the current batch, $S_{t-1} = \{\mathbf{x}|\mathbf{x} \sim p(\mathbf{x}|\theta^{t-1})\}$. Figure 2a shows that Origami can generate sequences with increasing viral assembly likelihood over iterations.

4 Future work

We must admit that the CapLib8 dataset is not an ideal data set to evaluate Origami. First, as per private communication with authors of [7], to build the CapLib8 dataset, the authors have to sequence the initial library creating a

sequence set called $\mathcal{F}_{plasmid}$, and then after the viral production, sequence it again to create another sequence set called $\mathcal{F}_{virus}$. All the sequences in $\mathcal{F}_{virus}$ are considered viable whereas all found in $\mathcal{F}_{plasmid} - \mathcal{F}_{virus}$ are considered non-viable. Unfortunately $\mathcal{F}_{plasmid}$ was not fully sequenced, hence CapLib8 does not have a lot of mutants that did not assemble in virus production. This creates a class imbalance that affects the performance of oracles as discussed in Section 3.2.

Second, Origami was designed to optimize under the assumption that sequences with good phenotypes are rare events. But since the "good" event of viral assembly is abundant in the CapLib8 dataset, it is therefore not an ideal one to benchmark Origami.

Finally, CapLib8 provides only one single screening dimension, namely viral assembly. Therefore, we could not evaluate Origami in a multi-objective optimization scenario. The next step could be using a multi-organ bio-distribution data set to evaluate Origami further.

References

[1] Frances H Arnold. Design by directed evolution. *Accounts of chemical research*, 31(3):125–131, 1998.

[2] David H. Brookes, Hahnbeom Park, and Jennifer Listgarten. Conditioning by adaptive sampling for robust design. *36th International Conference on Machine Learning, ICML 2019*, 2019-June:1214–1223, 2019.

[3] Anvita Gupta and James Zou. Feedback GAN for DNA optimizes protein functions. *Nature Machine Intelligence*, 1(2):105–111, 2019.

[4] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[5] Jae-Hyung Jang, James T Koerber, Jung-Suk Kim, Prashanth Asuri, Tandis Vazin, Melissa Bartel, Albert Keung, Inchan Kwon, Kook In Park, and David V Schaffer. An evolved adeno-associated viral variant enhances gene delivery and gene targeting in neural stem cells. *Molecular Therapy*, 19(4):667–675, 2011.

[6] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[7] Andrew D. Marques, Michael Kummer, Oleksandr Kondratov, Arunava Banerjee, Oleksandr Moskalenko, and Sergei Zolotukhin. Applying machine learning to predict viral assembly for adeno-associated virus capsid libraries. *Molecular Therapy - Methods and Clinical Development*, 20(March):276–286, 2021.

[8] David S. Ojala, Sabrina Sun, Jorge L. Santiago-Ortiz, Mikhail G. Shapiro, Philip A. Romero, and David V. Schaffer. In Vivo Selection of a Computationally Designed SCHEMA AAV Library Yields a Novel Variant for Infection of Adult Neural Stem Cells in the SVZ. *Molecular Therapy*, 26(1):304–319, 2018.

[9] Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th international conference on Machine learning*, pages 745–750, 2007.

[10] Philip A. Romero, Andreas Krause, and Frances H. Arnold. Navigating the protein fitness landscape with Gaussian processes. *Proceedings of the National Academy of Sciences of the United States of America*, 110(3), 2013.

[11] Reuven Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and computing in applied probability*, 1(2):127–190, 1999.

[12] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *arXiv preprint arXiv:1206.2944*, 2012.

[13] Dan Wang, Phillip W.L. Tai, and Guangping Gao. Adeno-associated virus vector as a platform for gene therapy delivery. *Nature Reviews Drug Discovery*, 18(5):358–378, 2019.

[14] Kevin K. Yang, Zachary Wu, and Frances H. Arnold. Machine-learning-guided directed evolution for protein engineering. *Nature Methods*, 16(8):687–694, 2019.