Guiding directed protein evolution with Bayesian Optimization

Yevgen Zainchkovskyy [1 2]

Abstract

We look into a possibility of improving the current direct evolution workflow by reducing the experimental effort associated with directed protein evolution. By incorporating Variational Autoencoders, a novel unsupervised deep generative model and Bayesian Optimization we seek to reduce the number of expensive experiments needed to find the target protein with desired properties. Starting with a discrete sequence of amino acids of our protein of choice (the wild-type) and it's multiple-sequence-aligned neighboring sequences (being proteins with the same function found elsewhere in nature), we use a VAE to learn a non-linear mapping from discrete sequence of aminoacids into a latent continuous space. We then use Bayesian Optimization in the latent space to propose promising changes to the wildtype protein (generating mutants). Finally we map the mutants back to the original sequence space for experiments in the lab. The proposed approach is validated on a large published emperical fitness landscape for all 4997 single mutations in TEM-1 β-lactamase selecting for the wild-type function (Stiffler et al., 2015). We learned that our VAE+BO approach significantly outperforms random mutant selection as it easily finds at least two of the best 9 mutants in less than 200 steps.

1. Background

The field of protein engineering is solving the problem of developing useful and valuable proteins. To do that, two common strategies are employed: rational protein design and directed evolution. Those strategies are not mutually exclusive, however for the sake of brevity, allow us to be very general in our description where we tried to summarize the main differences of both methods.

[1]Section for Cognitive Systems, Department of Applied Mathematics and Computer Science, Technical University of Denmark [2]Novo Nordisk A/S. Correspondence to: Yevgen <yeza@dtu.dk>.

Within **rational protein design** (RPD), the detailed knowledge of the structure and function of a protein is used to design new variants. This has the advantage of being both inexpensive and technically feasible, however the downside of this method is the limited availability of detailed structural information (3D structure). Additionally, this approach suffers from the fact that it is very difficult to predict the effects of various mutations. This is due to structural information only providing a static representation of a protein structure.

Mimicking natural evolution, **directed evolution** (DE) is utilizing random mutations applied to the protein of interest (called *wildtype*). A selection regime is then used to select variants having desired properties. In contrast to rational protein design, directed evolution does not require a prior structural knowledge of a protein and allows for wider exploration, but is expensive due to high cost of high-throughput screening. Not all desired activities can be screened for easily. Also, this technique is not applicable to all proteins.

2. Introduction

Motivated by the idea of reducing the number of costly high-throughput candidate screenings, we setup an experiment where we wish to test the applicability of Bayesian Optimization in the context of candidate selection. Recall that there are an enormous number of ways a protein can be mutated: at every position in the chain, a single aminoacid can be swapped into 19 other possible variants (and that number is even larger if we include non-naturally occurring aminoacids). A quick calculation reveals that for 250-amino-acid protein, there are $19 * 250 = 4,750$ possible substitutions, and for double substitutions, the number is $4750 * (4750 - 19) = 22,472,250$. Further aggravating the problem, we must also note that the vast majority of mutations lead to unfolded, useless proteins. Jumping a little forward, consider Figure 1, where we depict a histogram of the experimentally measured protein fitness for all possible single-site mutations to the TEM-1 β-lactamase enzyme. Here it is clearly seen that almost all mutations lead to a very substantial reduction in protein fitness.

Zooming in on the far right of the histogram in Figure 1 (depicted in the bottom of the figure), we find the top 9 mutations which lead to an increased protein fitness. Now,

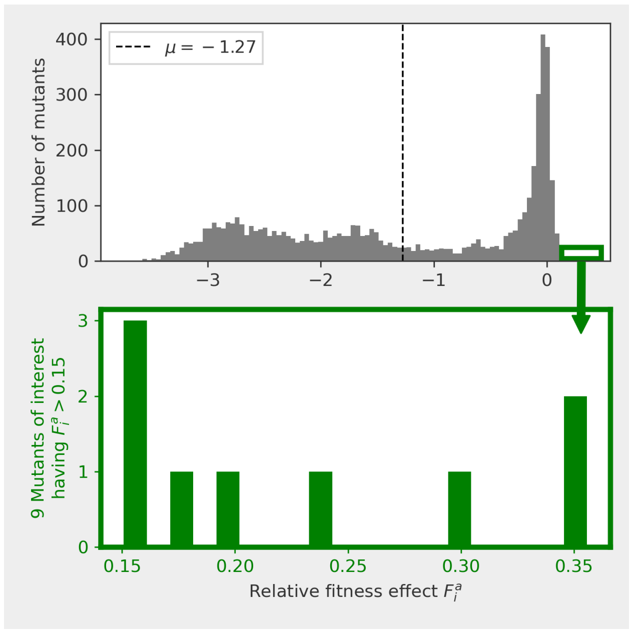

Figure 1. (top) A histogram of the experimentally measured protein fitness for all possible single-site mutations to the TEM-1 β-lactamase enzyme. (bottom) A zoomed in view on the specific region of the histogram where we find the rare mutations which resulted in the increase of the protein fitness.

for TEM-1 β-lactamase enzyme, this specifically means that those mutants provide an increased antibiotic resistance in E-Coli bacteria.

Fantasizing about work in the field of super-bacteria research, we imagine that it can be beneficial to artificially create E-Coli bacteria with those mutations for the purpose of staying ahead in novel antibiotic research. Those specific, 9 best mutations and their experimentally measured fitness values are presented in Table 1. Our tasks here is to see if it is possible to find one or more of those without traversing all possible single point mutations. Before we move on to the topic of efficiently searching for one of those 9 mutants, let us note that the TEM-1 β-lactamase has the length of 263 aminoacids, and the experimental data includes the fitness measurements[1] for all the $263 * 19 = 4997$ single mutations. For complete description of the β-lactamase data and experiments, we refer to the original source (Stiffler et al., 2015).

Before employing Bayesian Optimization, however, we are going to need a more optimal lower dimensional space to search in. As we stated before, the full combinatorial space of all the different mutations is huge, and by naively trying to optimize in the original space, we would not gain much, as the procedure will be completely uninformed, blindly suggesting candidates in an endless void of unfoldable invalid proteins. Additionally, we note the discrete nature of the original space and the problem of applying traditionally formulated BO to the functions of discrete variables. The problem here is that traditional Gaussian Processes based BO employs gradient based methods and those do not work for discrete variables[2].

3. Latent variable model

Proteins have evolved as a result of a long-term evolutionary processes that select for functional molecules. It therefore appears beneficial to try to probabilistically model the underlying process, as this will open up the ability to inference plausibility of other mutations. We model the evolutionary process as a type of a canonical ”protein generator” able to generate a protein x with probability $p(x|\theta)$ where parameters θ are fit to mimic the statistics of evolutionary data. Motivated by the need to reduce the dimensionality of the original space and the fact that it has to be continuous, we will model $p(x|\theta)$ with a nonlinear latent-variable model known as a Variational Autoencoder (VAE). Intuitively, we think of the model as being a generator which samples a hidden variable z from a prior $p(z) \sim \mathcal{N}(0, 1)$ and afterwards generates the actual protein x on the basis of conditional distribution $p(x|\theta, z)$. As z is never observed directly, we will use a marginal likelihood formulation

$$p(x|\theta) = \int p(x|z, \theta)p(z)\, dz$$

which considers all possible instantiations of the hidden variables z and integrates them out. Unfortunately, the integral above is intractable, but using Variational Inference (Bishop, 2006), we can form a lower bound on the log probability of $p(x|\theta)$. This lower bound, $\mathcal{L}$, is known as the evidence lower bound (ELBO):

$$\log p(x|\theta) \geq \mathcal{L}(\phi, x) = \mathbf{E}_q[\log p(x|z, \theta)] - D_{\mathrm{KL}}(q(z|x, \phi)||p(z))$$

Where $q(z|x, \phi)$ is introduced as a variational proxy for the true posterior distribution $p(z|x, \theta)$ of hidden variables given the observations. Looking more closely at the ELBO, we see that it consists of two parts, the reconstruction part $\mathbf{E}_q[\log p(x|z, \theta)]$ which encourages the decoder to learn to reconstruct the data, and the D_{KL} part which acts as a regularizer forcing the latent variables z to be distributed close to a unit Gaussian. Practically speaking, VAE will strive to be general according to it’s reconstructions and arrange similar proteins closely in the latent space as it will be penalized otherwise.

[1]Protein fitness is assessed as the logarithm of the allele counts in the selected population

[2]at least without tricks

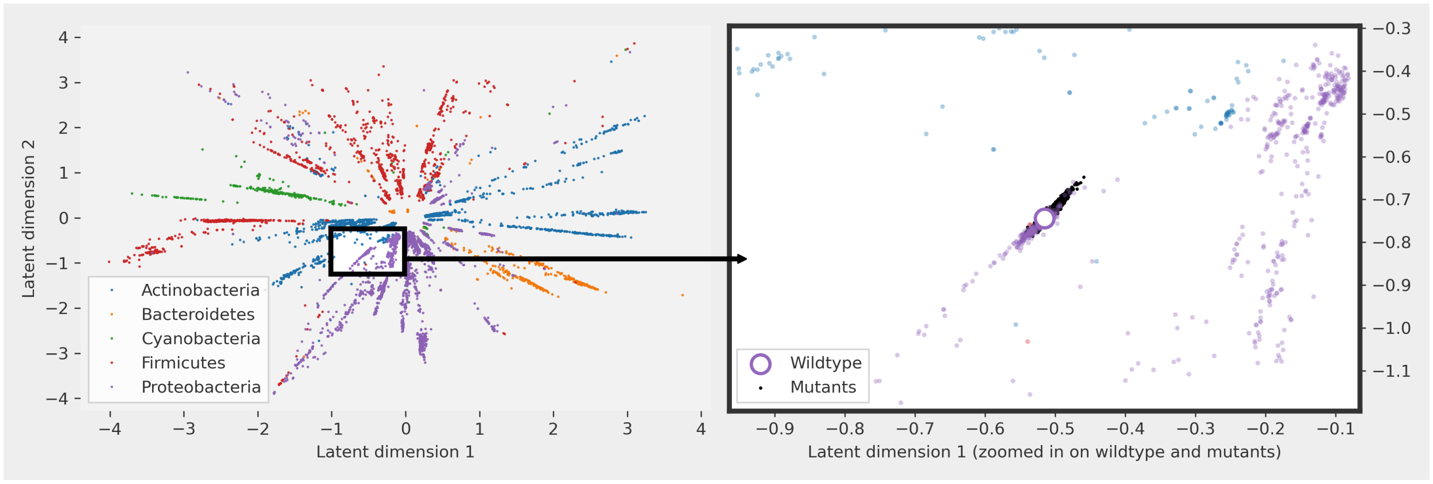

Figure 2. (left) Mapping of the Multiple Sequence Aligned β-lactamase proteins into a two dimensional latent space. (right) Zoomed in portion of the latent space with the wildtype and it's single-site mutations.

To train our VAE, we used a publicly available dataset consisting of 8403 entries of similar proteins found via the Multiple Sequence Aligned procedure (Wikipedia contributors, 2020). Those are the proteins which have the same neutralizing antibiotics-function, however, they originate from different species in nature. It turns out (and this is already well known), VAEs are able to exploit the underlying similarities between similar proteins. For the purpose of demonstration, we train a separate version of VAE with only 2 dimensions in the latent space. The result of the training is shown in Figure 2 where it is clearly seen that proteins which originate from the same species, are clustered together. Note that the training was done in an unsupervised fashion. In other words, the VAE model is trained exclusively on the 1-hot-encoded sequences of aminoacids originating from similar proteins in nature.

To further examine the latent space, we also encoded the mutants from (Stiffler et al., 2015) and were thrilled to discover that they correctly aligned to the wildtype (see the zoomed in portion of the latent space in Figure 2).

At this point, we have obtained a promising Latent Variable Model allowing us to encode and decode a protein between a regularized lower-dimensional continuous latent space and the original discrete space. But we are not limited to merely using this bi-directional mapping. By benefitting from probabilistic nature of the VAE model, we will also extract an additional feature to be used in Bayesian Optimization: 'relative favorability' (RF). As described in (Riesselman et al., 2018), the probabilities that the model assigns to any given sequence, can be used as proxy for for the relative plausibility of a molecule satisfying functional constraints. The following ratio

$$\log \frac{p(x^{\text{mutant}}|\theta)}{p(x^{\text{wildtype}}|\theta)}$$

is considered a heuristic metric for the relative favorability of a mutated sequence x^{mutant} compared to the sequence of the wildtype x^{wildtype}. In the following, we denote this metric as MODEL. We note that this metric correlates with the experimentally measured fitness value of the mutant proteins, F_i^a (Spearman $\rho = 0.74$) and is also included in Table 1 for completeness.

Table 1. **9 mutants of interest being the ones with the hightest experimentally measured fitness value (VALUE column).** MUTATION column describes the specific mutation to the wildtype, eg. `E210M` means that a change is made in position 210 where E (glutamic acid) is substituted with Y (tyrosine). Values in the MODEL column is the heuristic metric for the effect of the mutation (it is inferred in the VAE model in an unsupervised way). Columns LAT_DIM_0 to LAT_DIM_30 are the coordinates of the mutants in the latent space. We include those to emphasize their closeness and the fact the latent variable model were able to capture the underlying similarities. Being well aware of the handwavy nature of the last postulate given the non-linearity of the underlying latent space, we deliberately omit to provide a formal definition of similarity.

MUTATION	VALUE	MODEL	LAT_DIM_0	...	LAT_DIM_30
E210Y	0.3557	6.7868	-0.0180	...	0.0719
E210I	0.3485	2.2616	-0.0168	...	0.0718
A211M	0.2997	3.4414	-0.0173	...	0.0722
E210K	0.2328	3.3433	-0.0179	...	0.0721
E210M	0.1941	3.5376	-0.0166	...	0.0715
K190Q	0.1770	1.3670	-0.0176	...	0.0719
K190E	0.1549	3.2577	-0.0176	...	0.0722
E210H	0.1536	6.4048	-0.0186	...	0.0727
E210C	0.1507	6.6831	-0.0179	...	0.0719

4. Bayesian Optimization

As mentioned earlier, protein screening is an expensive and time consuming procedure. Furthermore, we also have the problem of combinatorial explosion in the number of possible mutations to the protein. One of the possible solutions, is to employ an informed searching procedure. Bayesian Optimization is one such technique allowing us to make informed decisions while being constrained by the high cost of the evaluation of the cost function. Recall that *evaluation* in this context constitutes to synthesizing a protein and testing (measuring) it in a lab.

Put a little more formally, Bayesian Optimization is a sequential procedure for global optimization of black-box functions which does not assume any functional forms. The way it works, is as follows.

1. First, we *choose a surrogate model* for modeling the true function f which in our case is the experimentally observed fitness value of a protein F_i^a. This surrogate model will be *a prior* over all possible f's.

2. Next we *observe some proteins* by evaluating the function. This will constitute to making an actual lab experiment where we measure the fitness value F_i^a of given proteins. After we obtain those measurements, the prior is updated (we have constrained the set of all possible f's to a set which now aligns with the observed measurements) resulting in *a posterior*.

3. With a help of an acquisition function $\alpha(x)$, we will make an informed decision on the location of the next promising protein in the latent space x_{t+1}. α itself is constructed by using the mean, μ, and uncertainty, σ, of the surrogate f's. Some acquisition function are parameterized, giving the user an ability to specify a tradeoff between exploitation and exploration.

4. Knowing our next sampling point x_{t+1}, we go to step 2 and continue the procedure from there.

The termination criteria of the above algorithms is up to the user, but there will typically be a budget associated with the experiments. Alternatively, one can terminate if an improvement is not achieved after a number of steps.

5. Experiment setup

The first ingredient in our experiment will naturally be the data. We denote our dataset $\mathcal{D}$ consisting of 4807 training pairs $(\mathbf{x}, t)$ where $\mathbf{x} = [z \quad \text{RF}]^\top$ is the input feature vector and y is the target variable (scalar) we seek to predict. The training procedure is then carried as follows: we initialize the BO loop with a sample of a 100 randomly selected observations[3] from the dataset. During the training, on every iteration, the procedure will suggest a new location x_{t+1} to be sampled. In a large-scale experiment, this suggestion would be mapped back to the original space through a decoder of the VAE and tested in the lab. As we do not have this luxury, a limitation will need to be imposed. Instead of obtaining a true value for x_{t+1} through a real experiment, we are going to find the closest, observed x in $\mathcal{D}$ and use it's corresponding target value $\hat{t}$ as a substitute for the real t. For the similarity metric between x_{t+1} and true, x's, we will use Euclidean distance. Obviously, this substitution will artificially constrain the optimization procedure to only see the effects of single-point mutations, but nevertheless, we hope to see some indication of an informed search. To quantify the success of the experiment, we will look at the mean of the target values of suggested mutants $\hat{\mu}_t$. It is clear that the procedure is working if $\hat{\mu}_t$ is high or at least higher than the mean of the targets across the full $\mathcal{D}$.

6. Results

We report selected findings obtained after completing a total of 54 experiments. Across the experiments we varied the dimensionality of the latent space, a choice of the GP kernel, initialization scheme and most importantly, the acquisition function and it's exploitation-exploration hyperparameter.

For the optimal dimensionality of the latent space, we learned that 8 dimensions worked best. This roughly aligned with the explained variance obtained by a PCA analysis on the latent space of 30 dimensions. In alignment with our intuition, running the optimization procedure in 30 dimensions resulted in low μ and large variance $\hat{\sigma}_t$ in the target values of the suggested mutants (See Figure 4). One possible explanation being the poor coverage of the latent space by the GP surrogate function due to the curse of dimensionality (Bishop, 2006).

For the initialization of BO loop, we hypothesized that it could be beneficial to favor samples of high relative favorability (RF, Section 3) as this metric correlates with the target value. In practice, however, it didn't yield better results. When initialized with samples of high RF, BO procedure would perform worse, although it generally recovers in the the training process (Figure 5). We think it might have to do with the need of the surrogate function to unlearn the imposed RF bias.

[3]This aligns with 10×10 sample plate used for screening in the lab.

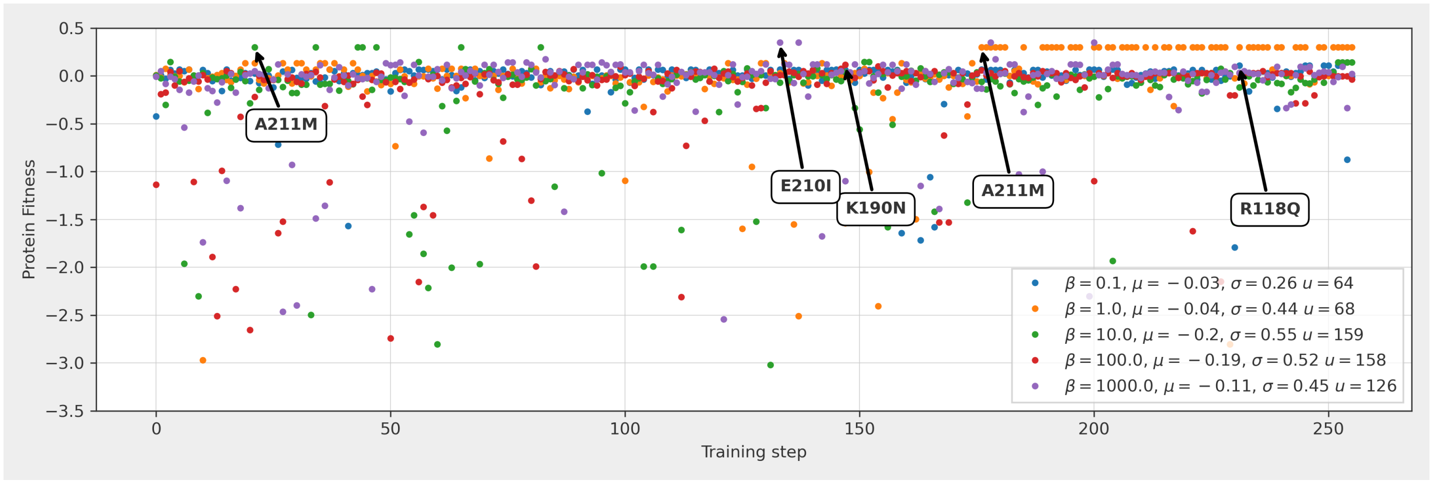

Figure 3. Optimization progress across 5 experiments with varying β (exploitation-exploration tradeoff hyperparameter). In all experiments, the Bayesian Optimization procedure manages to suggest highly relevant mutations as seen by the high value of μ - the mean value of the fitness of the suggested targets. Arrows point to the first occurrence of the best mutant in the experiment. Comparing with Table 1, we note that the procedure successfully manages to find 2 out of 3 top mutants.

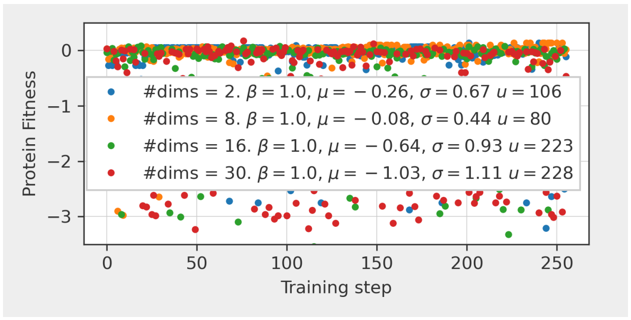

Figure 4. Finding the optimal size of the latent space. As seen on the high mean μ and low variance σ of the suggested targets, the optimal size of the latent space is 8.

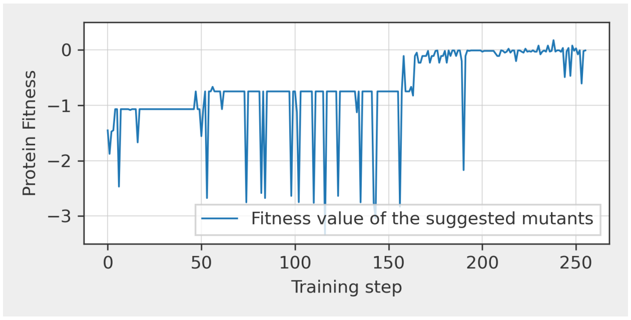

Figure 5. Optimization progress when initialized with a bad heuristic.

Trying different GP kernels for the surrogate function (RBF, Matern$^{\frac{1}{2}}$, Matern$^{\frac{3}{2}}$ and Matern$^{\frac{5}{2}}$), we learned that for our problem, Matern$^{\frac{5}{2}}$ performed best. This is not entirely surprising, as this kernel, being a generalization of the RBF) allows to capture more structure due to an extra parameter (ν). For all the kernels above, both the general and the ARD (automatic relevance determination) formulations were tried. Again, not surprisingly, ARD led to better results, due to it's ability to expresses each dimension as being independent from the other though a separate set of hyperparameters.

For the acquisition function and the exploitation-exploration hyperparameter, a grid search was performed. Here, we went through the most popular acquisition functions: Expected Improvement (EI), Probability of Improvement (POI) and finally Upper Confidence Bound (UCB). We learned that UCB performed best, however we cannot clearly pinpoint an optimal value for it's β hyperparameter as it depends on the actual goal of an experiment. By observing the mean of the mutants for the different values of β, we confirmed that the search is getting more exploratory for higher values of β. This can be seen in the general decrease in the mean target of the mutants μ an increase in the variance σ and a number of uniquely queried samples (u).

In Figure 3 we present the result of 5 experiments with varying β. In all experiments, the Bayesian Optimization procedure manages to suggest highly relevant mutations as seen by the high value of μ - the mean of the suggested targets (recall that the mean of targets in the full dataset is $\mu_t = -1.27$). While we did not manage to "find" the top `E210Y` mutation in any of our experiments, we did successfully locate the second and third. Interestingly, `A211M`, was found in two separate experiments.

7. Conclusions and future directions

Although we carried a rather hypothetical and constrained experiment, the potential of combining Variational Autoencoders and Bayesian Optimization is not to be dismissed. In this work, we have successfully shown that there are clear benefits of guiding the directed protein evolution workflow with novel machine learning models thereby potentially reducing costly experimental effort. Our central finding being the fact that the procedure, for the most part, allowed the exploration phase to focus on potentially functional mutations thereby possibly reducing the wasteful exploration of non-functional (unfoldable, invalid) proteins. Compared to strictly random single point mutations, the procedure was able to successfully propose new mutants in the area of the space where functional proteins reside. Generally, our suggested mutatants had an experimentally measured Fitness value $\mu \approx 0$ while the fitness level of underlying population was $\mu_t = -1.27$. We additionally note that the underlying true distribution is very heavily skewed towards low-fitness variants (see Figure 1).

Given the limitations imposed by the restricted scope of experimental data (only single point mutations), it is of our greatest interest to conduct a similar experiment on a larger scale. We strongly believe that a similar model will be able to propose good candidates even further away, in other words, we hypothesize that it is possible to suggest beneficial mutations in several positions in the chain at a single optimization step!

Related to the above, we also predict that improvements can be made given we augment input data with additional information. Similar to the 'relative favorability' feature (Section 3 (Riesselman et al., 2018)), potential other features could be employed through eg. molecular dynamics simulations (Gelpí et al., 2015) or protein vectors (Bepler & Berger, 2019).

Finally, we note an immediate improvement to the model itself. As we have described in Sections 3 and 5, the latent space is implicitly assumed to be Euclidean. In recent work it has been shown that this assumption decreases the capacity of the model, leading to lower performance. Therefore, by modifying the VAE to support Riemannian structure over the latent space, a potential improvement can be achieved (Kalatzis et al., 2020). Given the new latent structure, however, the GP-based surrogate model in Bayesian Optimization, will most likely also require an update.

8. Data and code

The β-lactamase multiple sequence alignment dataset is taken from (Riesselman et al., 2018) and the single-point mutations are from (Stiffler et al., 2015). After training a VAE and mapping all the single-point mutations, an aggregated dataset was created (`data/df-*.csv`, where $\star$ refers to the dimensionality of the latent space: 2, 5, 8, 16, 30). The resulting experiment data resides in `results/` and includes a total 54 experiments. Across experiments we vary: latent space dimensionality, exploitation-exploration trade-off and initialization scheme (random or model-informed). The code is written in Python using the excellent Botorch library (an intermediate version based on `fmfn/BayesianOptimization` is also included). Using a GPU, expect a training time of around 5 minutes for a single experiment. See `http://github.com/eugene/prot-bo-0` for details.

References

Bepler, T. and Berger, B. Learning protein sequence embeddings using information from structure, 2019.

Bishop, C. M. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006. ISBN 0387310738.

Gelpí, J., Hospital, A., Goñi, R., and Orozco, M. Molecular dynamics simulations: Advances and applications. *Advances and Applications in Bioinformatics and Chemistry*, 10:37, 11 2015. doi: 10.2147/AABC.S70333.

Kalatzis, D., Eklund, D., Arvanitidis, G., and Hauberg, S. Variational autoencoders with riemannian brownian motion priors, 2020.

Riesselman, A. J., Ingraham, J. B., and Marks, D. S. Deep generative models of genetic variation capture the effects of mutations. *Nature Methods*, 15(10): 816–822, 2018. ISSN 15487105. doi: 10.1038/s41592-018-0138-4. URL `https://doi.org/10.1038/s41592-018-0138-4`.

Stiffler, M., Hekstra, D., and Ranganathan, R. Evolvability as a function of purifying selection in tem-1 beta-lactamase. *Cell*, 160:882–892, 02 2015. doi: 10.1016/j.cell.2015.01.035.

Wikipedia contributors. Multiple sequence alignment, 2020. URL `https://en.wikipedia.org/wiki/Multiple_sequence_alignment`. [Online; accessed 12-Sep-2020].