Probabilistic Protein Engineering

Thesis by
Kevin Kaichuang Yang

In Partial Fulfillment of the Requirements for the
Degree of
Doctor of Philosophy in Chemical Engineering

Caltech

CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2019
Defended 14 Dec 2018

© 2019

Kevin Kaichuang Yang
ORCID: 0000-0001-9045-6826

Some rights reserved. This thesis is distributed under a Creative Commons Attribution-NonCommercial-ShareAlike License

ACKNOWLEDGEMENTS

First, I'd like to thank my advisor Frances Arnold for her support and guidance throughout my PhD. I applied to Caltech because I wanted to work for Frances, and she has not disappointed. Even as I took a detour to teach high school, she welcomed me into her lab first for a summer, and then for my PhD. Frances has a keen eye for important but tractable scientific problems and gave me the freedom to pursue them past the boundaries of her discipline. Frances has pushed me to be a better scientist, and her lessons are the foundation for the rest of my career.

One thing Frances has done very well is to build a world-class research group. I would like to specifically thank several group members. During my first stint in the lab, Sabine Brinkmann-Chen, John McIntosh, and Chris Farwell taught me to perform directed evolution. Sabine also keeps us all safe, happy, and productive. When I rejoined the lab, Lukas Herwig reacquainted me with molecular biology. It was a joy to work with Claire Bedbrook and Austin Rice on my first three papers. Thank you for collecting my data for me. Zachary Wu has been a great co-belligerent for the cause of machine learning in the Arnold lab. Thank you for the discussions and advice. Finally, I'd like to thank the two undergraduate researchers who worked with me: Raul Sun Han and Alycia Lee. I hope my mentorship was a fair trade for all the work you did. Philip Romero now leads his own group, but I am indebted to him for starting the machine learning work in the Arnold lab, providing me with his code and data, and being on my candidacy committee.

Almost all of my formal training in machine learning comes from Yisong Yue. Yisong taught me the foundations of machine learning and pointed the way towards successful applications to proteins. Most importantly, he has shown me how to be a machine learning researcher. I was also fortunate to collaborate with Yisong's postdoc Yuxin Chen. Thank you for your patience as I learned new fields and for your good humor through the late nights. Without Yisong's guidance and Yuxin's hands-on help, I would still be stumbling in the dark.

I am grateful to my committee, which in addition to Frances and Yisong includes David Tirrell and William Clemons, for their advice and scientific insights. I am still astonished that I convinced such amazing scientists to serve on my committee.

I would like to acknowledge the chemical engineering option at Caltech. I would especially like to thank Kathy Bubash and Allison Oulette, who keep everything running and answer all my questions, and my classmates Bradley Silverman and Heidi Klumpe, who helped me pass my classes and quals and have been great friends over the last 4 years.

Outside of chemical engineering, Justin Bois and Kimberly Mayer have both made significant contributions to my development as a scientist. Justin is my role model in computational science and thoughtful teaching. Kimberly has been invaluable in helping me navigate grant-writing, communication, and science careers.

My parents, Shang-Tian Yang and I-Ching Tang, were my first teachers and awakened my thirst for knowledge. Thank you for everything you've given me. I would like to thank my brothers: Grant, for numerous discussions and his remarkable ability to vectorize my code; and Hopen, who inspires me with his work ethic and creativity.

Finally, I am grateful for my wife, Katherine Ezawa. Thank you for putting up with my weird hours and for loving me throughout. I'm sorry I'm not very good at explaining what I do to you.

ABSTRACT

Machine learning-guided protein engineering is a new paradigm that enables the optimization of complex protein functions. Machine-learning methods use data to predict protein function without requiring a detailed model of the underlying physics or biological pathways. They accelerate protein engineering by learning from information contained in all measured variants and using it to select variants that are likely to be improved. We begin with a review of the basics of machine learning with a focus on applications to protein engineering and protein sequence-function datasets (Chapter 1). We used the entire machine-learning guided engineering paradigm to engineer the algal-derived light-gated channel channelrhodopsin (ChR), which can be used to modulate neuronal activity with light. We build models that discover ChRs with strong plasma membrane localization in mammalian cells (Chapter 2) and unprecedented light sensitivity and photocurrents for optogenetic applications (Chapter 3). Machine learning-guided evolution requires a machine-learning model that learns the relationship between sequence and function. For machine-learning models to learn about protein sequences, protein sequences must be represented as vectors or matrices of numbers. How each protein sequence is represented determines what can be learned. We learn continuous vector encodings of sequences from patterns in unlabeled sequences (Chapter 4). Learned encodings are low-dimensional, do not require alignments, and may improve performance by transferring information in unlabeled sequences to specific prediction tasks. Alternately, we demonstrate an interpretable Gaussian process kernel tailored to biological sequences (Chapter 6). In addition to a model to predict function from sequence, engineering requires a method to use the model to choose sequences for the next round of evolution. Most machine-learning guided engineering strategies assume that selected sequences can be queried directly. However, in directed evolution it is common to design a library of sequences and then sample stochastic batches from that library. We propose a batched stochastic Bayesian optimization algorithm for iteratively designing and screening site-saturation mutagenesis libraries (Chapter 5).

PUBLISHED CONTENT AND CONTRIBUTIONS

1. Kevin K Yang, Zachary Wu, and Frances H Arnold: Machine learning in protein engineering (2018). eprint: https://arxiv.org/abs/1811.10775.
K.K.Y. participated in the conception of the project and wrote the manuscript.

2. Kevin Yang, Zachary Wu, Claire Bedbrook, and Frances Arnold: Learned Protein Embeddings for Machine Learning. *Bioinformatics* (2018). doi: 10.1093/bioinformatics/bty178.
K.K.Y conceived the project, built and trained machine-learning models, prepared the data, and wrote the manuscript.

3. Claire N Bedbrook, Kevin K Yang, Austin J Rice, Viviana Gradinaru, and Frances H Arnold: Machine learning to design integral membrane channelrhodopsins for efficient eukaryotic expression and plasma membrane localization. *PLoS Computational Biology* **13**(10) (2017), e1005786. doi: 10.1371/journal.pcbi.1005786.
K.K.Y. participated in the conception of the project, built and trained machine-learning models, and participated in the writing of the manuscript.

4. Claire N Bedbrook, Kevin K Yang, J Elliot Robinson, Viviana Gradinaru, and Frances H Arnold: Machine learning-guided channelrhodopsin engineering enables minimally-invasive optogenetics (Submitted).
K.K.Y. participated in the conception of the project, built and trained machine-learning models, and participated in the writing of the manuscript.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

Chapter 1

MACHINE LEARNING FOR PROTEIN ENGINEERING

1. Kevin K Yang, Zachary Wu, and Frances H Arnold: Machine learning in protein engineering (2018). eprint: https://arxiv.org/abs/1811.10775.

1.1 Introduction

Protein engineering seeks to design or discover proteins whose properties, useful for technological, scientific, or medical applications, have not been needed or optimized in nature. We can envision the mapping of protein sequence to a desired function or functions as a "fitness landscape" over the high-dimensional space of possible protein sequences[1]. The fitness represents a protein's performance: expression level, catalytic activity, or other properties of interest to the protein engineer. The landscape determines the range of properties available to different sequences as well as the ease with which they can be optimized. In one limit, convex landscapes that climb smoothly to a global maximum are easy to search. In the other, rugged landscapes with many local maxima are much more difficult to traverse. Protein engineering seeks to identify sequences corresponding to high fitnesses on these landscapes.

Identifying optimal locations on a fitness landscape is extremely challenging. The space of possible protein sequences is too large to be searched exhaustively naturally, in the laboratory, or computationally[2]. The problem of finding optimal sequences is NP-hard, meaning that there is no known polynomial-time method for searching this space[3]. Functional proteins are also extremely scarce in this vast space of sequences. Moreover, as the threshold level of fitness increases, the number of sequences having that fitness decreases exponentially[4,5]. As a result, highly fit sequences are vanishingly rare and overwhelmed by nonfunctional and mediocre sequences.

Until recently, the two main approaches for finding high-fitness protein sequences have been directed evolution and rational design. Rational design uses physics-based models to guide the search for improved sequences. These models typically contain an atomic structural representation of a protein and energy-based scoring functions to quantify the target function[6,7]. Rational design has been successful in identifying sequences that fold into desired static structures[8]. This is an important advance, and useful when a single stable structure dictates function[8]. However, because proteins are marginally stable, even small inaccuracies in energy-based scoring functions can lead to very poor performance[9].

Many protein functions such as binding, catalysis, allostery, and signalling, are mediated through recessed cavities, mobility, or multiple low-energy states[10]. For example, computational enzyme design currently proceeds by designing an idealized active site for the desired reaction, matching the active site residues to stable backbones, and then using molecular dynamics (MD) simulations to winnow designs with flaws not apparent from static evaluations. MD simulations require enormous computational resources (100s of CPU hours for each variant) and are not appropriate for testing many variants. This process generally yields sequences with modest activity that are finally improved with directed evolution[11].

Inspired by natural evolution, directed evolution climbs a fitness landscape by accumulating beneficial mutations in an iterative protocol of mutation and selection, as illustrated in Figure 1.1a. The first step is sequence diversification using techniques such as random mutagenesis, site-saturation mutagenesis, or recombination to generate a library of modified sequences starting from the parent sequence(s). The second step is screening or selection to identify variants with improved properties for the next round of diversification. The steps are repeated until fitness goals are achieved.

Directed evolution is limited by the fact that even the most high-throughput screening or selection methods only sample an insignificant fraction of the sequences that can be made using most diversification methods, and developing efficient screens is nontrivial. Moreover, directed evolution requires at least one minimally-functional parent and a locally-smooth fitness landscape for stepwise optimization[1]. Recombination methods may allow for bigger jumps in sequence space while retaining function[12], but sequences designed using recombination are by definition restricted to exploring combinations of previously-explored mutations. No matter the diversification technique, directed evolution is energy-, time-, and material-intensive, and multiple generations of evolution may be required to achieve meaningful performance improvements.

More recently, researchers have begun using statistical, or machine-learning, methods to approximate sequence-function relationships. Machine-learning methods learn functional relationships from data[13,14]. Machine-learning models of protein

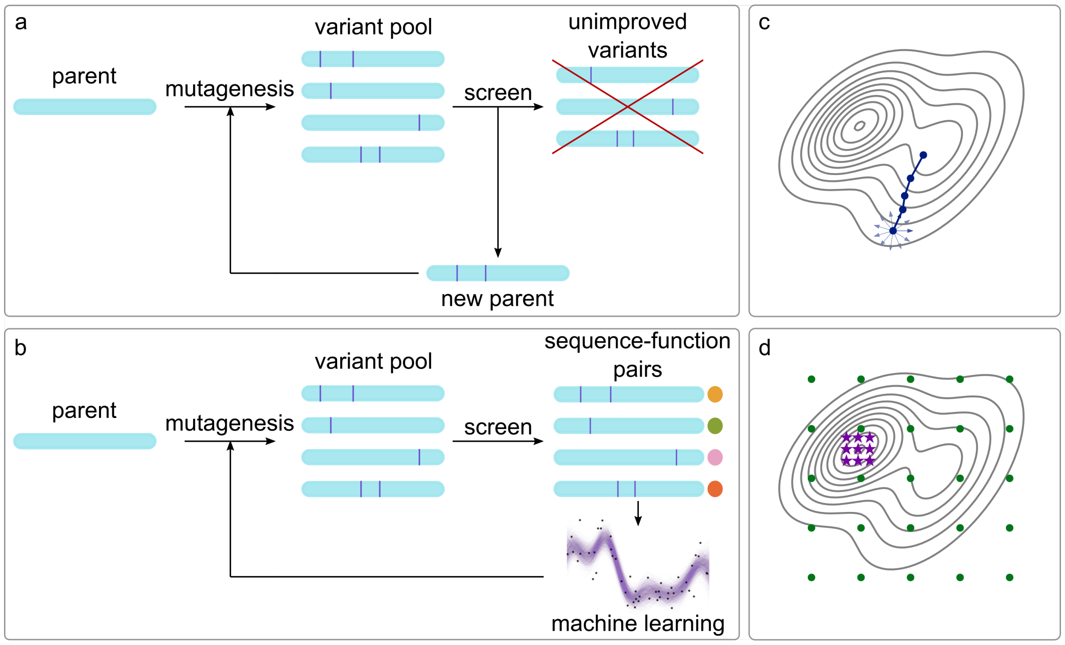


Figure 1.1: Directed evolution with and without machine learning. (a) Directed evolution uses iterative cycles of diversity generation and screening to find improved variants. Information from unimproved variants is discarded. (b) Machine-learning methods use the data collected in each round of directed evolution to choose the mutations to test in the next round. Careful choice of mutations to test decreases the screening burden and improves outcomes. (c) Directed evolution is a series of local searches on the fitness landscape. (d) Machine learning-guided directed evolution often rationally chooses the initial points (green circles) to maximize the information learned from the fitness landscape, allowing future iterations to quickly converge to improved sequences (violet stars).

function can be predictive even when the underlying mechanisms are not well-understood. As shown in Figure 1.1b, machine-learning models can guide directed evolution by learning from the information contained in all measured variants. This new paradigm for protein engineering enables engineering with fewer measurements and fewer generations of evolution. Here, we cover the basics of machine learning with a focus on applications to protein engineering and protein sequence-function datasets, discuss how machine learning can be integrated with directed evolution to accelerate protein engineering, and consider the developments that are required to enable wider applications.

1.2 A brief introduction to machine learning on proteins

In most computational methods, the user provides a hard-coded algorithm and inputs, and the computer executes the steps provided by the human expert. In contrast, machine-learning models infer patterns from data, which can then be used to make predictions on unobserved data. The user must collect the training data, represent it in a form amenable to machine learning, decide what type of machine-learning algorithm to use, and train and interpret the model.

Protein function databases

The broadest protein datasets are UniProt, which aims to catalog all known protein sequences[15], the Protein Data Bank[16], which catalogs all known protein structures, and BRENDA, which catalogs natural enzymes and their functions[17]. These databases do not specifically store sequence-function relationships.

Protein engineering experiments have generated a growing collection of well-characterized libraries. Databases that collect and organize specific categories of sequence-function data include ProTherm and ProNit for protein stability and protein-nucleic acid interactions, respectively[18], and SKEMPI[19], AB-Bind[20], and PROXIMATE[21] for protein-protein complexes. The Protein Mutant Database[22], an early attempt to catalog the effects of mutations from protein engineering studies across different proteins and functions, has not been updated in over a decade. Currently, Protabank is an actively-maintained and updated database for general protein design and engineering data[23].

ProTherm is the oldest and most mature of these databases, but has not been updated since 2013. There have been many efforts to use the sequence-function information in ProTherm to train machine-learning models to predict the effects of mutations on stability[24–35]. However, Yang *et al.* found that ProTherm contains many errors and cautions against using entries as training sets without proper validation[36].

Datasets derived from protein engineering experiments tend to be small (10^2 - 10^3 variants) and focused on high-performing variants derived from a small number of sequences. Furthermore, the variants sampled in a protein engineering study are limited by the bias and intent of the study[23]. This may make it more difficult to generalize models trained on one dataset to other variants of even the same protein generated in different ways or with a different objective. In contrast, datasets of natural variants can be quite large, with examples from many families of proteins. However, the variant distribution in natural datasets is biased by evolution and the fact that not all organisms are equally likely to be studied and sequenced. This can lead to difficulty generalizing to the non-natural sequences often encountered in protein engineering.

Vector representations of proteins

For machine-learning models to learn about protein sequences, protein sequences must be represented as vectors or matrices of numbers. How each protein sequence is represented determines what can be learned[37,38]. Even the most powerful models produce poor results if an inappropriate representation is used. An ideal encoding would perform well across different proteins and functions, not require alignments, structures, or feature selection, and transfer the information contained in unlabeled sequences to specific prediction tasks. Unfortunately, such an encoding does not exist.

In general, a protein sequence is a string of length L where each residue is chosen from an alphabet of size A. The simplest way to encode such a string is to represent each of the A amino acids as a number. However, the assignment of each residue to a number enforces an ordering on the amino acids that has no physical or biological basis. Instead of representing each position as a single number, a one-hot encoding represents each of the L positions as $A - 1$ zeros and one 1, with the position of the 1 within the series denoting the identity of the amino acid at that position. Given structural information, the identity of pairs of amino acids within a certain distance in the structure can also be one-hot encoded[39,40]. Single mutations can also be encoded as a 20-dimensional vector where the original amino acid is denoted by -1, the new amino acid by 1, and all others by zero[41]. One-hot encodings are inherently sparse, memory-inefficient, and high-dimensional. In a one-hot encoding, there is no notion of similarity between sequence or structural elements: they are either identical, or not. Furthermore, one-hot encodings of the primary sequence require that all sequence variants of interest are aligned. This alignment must be updated as sequences are added to the model. Nevertheless, one-hot encodings offer good performance for little complexity and can be considered a good baseline encoding.

A protein can also be encoded by its physical properties by representing each individual amino acid with a collection of physical properties, such as its charge or hydrophobicity, and each protein with a combination of those properties. Properties such as predicted secondary structures can also be used to represent proteins. The challenge is that there are a large number of physical properties that could be used to describe each amino acid or protein. Furthermore, the molecular properties that dictate functional properties are unknown, highly constrained, and dependent on the specific function considered. Therefore, selecting informative properties is challenging because it is difficult to know *a priori* what properties will be predictive for a particular task. Most representations using physical properties also require alignments of the input sequences.

AAIndex[42] attempts to systematically collect descriptors of protein sequences. AAIndex comprises three sections: AAIndex1 contains 554 amino acid features, AAIndex2 contains 94 amino acid substitution matrices, and AAIndex3 contains 47 amino acid contact potential matrices. ProFET is another encoding system that considers physical properties of the bulk protein, the individual residues, and sub-sequences of residues[43]. There have also been attempts to describe each amino acid using two reduced dimensions based on volume and hydrophobicity[44] and to combine physical properties with structural information[30,45] by encoding each position in the sequence as a combination of the properties of amino acids in its geometric neighborhood.

While there are a vast number of known protein sequences, only a tiny fraction are labeled with measured properties relevant to any specific prediction task. The number of unlabeled sequences will continue to rise as more sequences are deposited into public databases. These unlabeled sequences contain information about the frequency and patterns of amino acids selected by evolution to compose proteins, information that may be helpful across prediction tasks. The simplest examples are BLOSUM[46] or AAIndex2-style substitution matrices based on relative amino-acid frequencies. However, more sophisticated continuous vector encodings of sequences can be learned from patterns in unlabeled sequences[47–52]. These representations are known as embedded representations. Conceptually, through sequence context, these representations learn a mapping from a space with one indicator (one-hot) dimension per k-mer or protein sequence to a continuous vector space with a much lower dimension such that similar sequences are close together in the continuous space. When modeling large ($> 10^4$ examples) datasets with neural networks, embeddings for individual amino acids or k-mers can be learned simultaneously with the model weights. Learned encodings are low-dimensional, do not require alignments, and may improve performance by transferring information in unlabeled sequences to specific prediction tasks. However, it is difficult to predict which learned encoding will perform well for any given task.

Just as no model will be optimal across all machine learning tasks, there is no universally optimal vectorization method[53]. Because computational resources are finite, researchers must use a combination of domain expertise and heuristics to select a set of encodings to compare. For small datasets, one-hot encodings offer superior performance to general sets of protein properties[51], although careful feature selection informed by domain knowledge may yield more accurate predictions. If accuracy is insufficient, learned encodings may be able to improve performance. The encoding should ultimately be chosen empirically to maximize predictive performance.

Models for protein data

A wide range of machine-learning algorithms exist, and no single algorithm is optimal across all learning tasks[53]. Machine-learning methods can be broadly classified as supervised, unsupervised, or semi-supervised. In supervised learning, the training data consist of inputs and their associated output values (labels). Supervised methods learn a mapping from input space to output space that enables them to accurately predict outputs from new inputs. Supervised learning can be further divided into regression, which aims to predict real-valued outputs, and classification, which aims to predict class membership. In unsupervised learning, the training data consist only of input values. Unsupervised methods learn to find patterns, such as trends or clustering, in the inputs. In semi-supervised learning, the training data consist of inputs, of which a subset has associated output values. Semi-supervised methods leverage information in the unlabeled training inputs to improve their ability to predict outputs from inputs. Supervised learning is the most developed of these approaches, and is used in the majority of applications to protein engineering. We outline some common supervised machine-learning algorithms below.

The simplest machine-learning models apply a linear transformation of the input features, such as the amino acid at each position, the presence or absence of a mutation[54], or blocks of sequence in a library of chimeric proteins made by recombination[55]. Linear models are simple and the learned parameters are easily interpreted by the user. Linear models are commonly used as baseline predictors before more powerful models are tried.

Classification and regression trees[56] use a decision tree to go from input features (represented as branches) to labels (represented as leaves). Decision tree models are often used in ensemble methods, such as random forests[57] or boosted trees[58], which combine multiple models into a more accurate meta-predictor. For small biological datasets ($< 10^4$ training examples), including those often encountered in protein engineering experiments, random forests are a strong and computationally efficient baseline, and have been used to predict thermostability[24–26].

Kernel methods, such as support vector machines[59] and kernel ridge regression[60], employ a kernel function, which calculates similarities between pairs of inputs, to implicitly project the input features into a high-dimensional feature space without explicitly calculating the coordinates in this new space. The choice of kernel profoundly affects the accuracy of these methods. While general-purpose kernels can be applied to protein inputs, there are also kernels designed for use on proteins, including spectrum and mismatch string kernels[61,62], which count the number of shared sub-sequences between two proteins, and weighted decomposition kernels[33] and other graph kernels, which account for three-dimensional protein structure. Support vector machines have been used to predict protein thermostability[24–31], enantioselectivity[63], and membrane protein expression[64].

Gaussian process regression and classification combine kernel methods with Bayesian learning to produce probabilistic predictions[65]. These models rigorously capture uncertainty, which, combined with methods from Bayesian optimization, can provide principled ways to guide experimental design in optimizing protein properties. The run-time for exact GP regression scales as the number of training examples cubed, making it unsuitable for large ($> 10^3$) datasets, but there are now fast and accurate approximations[66]. Gaussian processes have been used to predict thermostability[32,33,39], enzyme substrates[67], fluorescence[68], membrane localization[40], and channelrhodopsin photo-properties[69].

Deep learning models, also known as neural networks, stack multiple linear layers connected by non-linear activation functions, allowing them to extract high-level features from structured inputs. Neural networks are well-suited for tasks with large labeled datasets with examples from many protein families, such as protein-nucleic acid binding[70–72], protein-MHC binding[73], binding site prediction[74], protein-ligand binding[48,75], solubility[76], thermostability[34,35], subcellular localization[77], secondary structure[78], functional class[79,80], and even 3D structure[81]. Deep learning networks are also particularly useful in metabolic pathway optimization[82] and genome annotation[83–85], which have been reviewed elsewhere.

k-nearest-neighbor[86] methods make predictions on new data points by taking the majority (for classification) or mean (for regression) labels for the *k* nearest training points. The quality of the predictions can be affected by setting the neighborhood size *k* as well as the distance metric used to identify the nearest neighbors. Alternatively, predictions can be made as a linear combination of the training labels weighted by their closeness to the test point. Because calculating distances between protein sequences can be non-intuitive (have little meaning in the problem context) or computationally expensive, *k*-nearest-neighbor methods are not commonly applied to protein datasets.

Model training and evaluation

All model families have hyperparameters that determine the form of the model. Unlike model parameters, hyperparameters cannot be learned directly from the data. These may be set manually by the practitioner or determined using a procedure such as grid search, simulated annealing, random search, or Bayesian optimization. Hyperparameters may be discrete or continuous. For example, in support vector machines, the type of kernel is a hyperparameter, as are the number of layers and learning rate in a deep neural network. The vectorization method is also a hyperparameter. Even modest changes in the values of hyperparameters can improve or diminish accuracy considerably, and the selection of optimal values is often challenging, as each set of hyperparameters considered may require training a new version of the model.

The key test for a machine-learning model is the ability to accurately predict labels for inputs it has not been trained with. Therefore, when training models by learning their parameters and selecting model hyperparameters, it is necessary to estimate the model's performance on data *not* in the training set. Thus it is essential to initially remove a set of data, called the test set, to be set aside until the absolute end for model evaluation. Typically, the test set comprises approximately 20% of the data. In order to compare models and select hyperparameters during a study, the remaining data should be split into a training set and a validation set. The training set is used to learn model parameters, while the validation set is used to choose between models with different hyperparameters. If the training set is small, cross-validation may be used instead of a constant validation set. In *n*-fold cross-validation, the training set is partitioned into *n* complementary subsets. Each subset is then predicted using a model trained on the remaining subsets. Averaging accuracy across the withheld subsets provides an estimate of predictive accuracy over the entire training set.

Care must be taken when selecting the training/validation/test sets that the splits allow an accurate estimate of model performance under the conditions where it will be used. Datasets from mutagenesis studies tend to be small and focused. In this case, the best practice is to train on variants characterized in earlier rounds of mutagenesis and to evaluate model performance on later rounds in order to recapitulate the iterative engineering process. When dealing with large, diverse datasets containing examples from different protein families, the best practice is to ensure that all examples in the validation and test sets are some minimum distance away from all the training examples in order to test the model's ability to generalize to unrelated sequences. In the machine learning context, generalization refers to a model's ability to accurately predict a test set drawn from the same distribution as the training set after training, in contrast to its colloquial meaning of transferring knowledge from one distribution to another, such as from an experimental setting to a real application[87].

Model interpretation

Once a machine-learning model has been built for a certain protein function, the model itself can be a source of knowledge about the underlying physical or biological processes. Model interpretation is the process of determining why or how a model makes its predictions, and allows practitioners to draw biological insights from the knowledge captured by the model. Furthermore, model interpretation can lead to greater user confidence in a model's predictions or a better understanding of when and how a model can fail.

Some machine-learning algorithms are inherently easier to interpret. For example, the learned weights in a linear model indicate which mutations, sequence blocks, or other features are beneficial or detrimental for a function of interest[54,55,89,90], and the splits in a decision tree naturally map to human-interpretable information about the features used to make predictions. However, interpretation is less straightforward for complex models with many parameters (such as neural networks) or non-parametric models (such as Gaussian processes). Unfortunately, this complexity is also what gives these models the capacity to make accurate predictions on complex systems. One way to interpret these complex models is to build local or global approximations. Local approximations, such as LIME (local interpretable model-agnostic explanations)[91] build a simple, interpretable approximation to the original model in the neighborhood of a single example, as illustrated in Figure 1.2a. In contrast, a global approximation attempts to simplify the complex model over all examples; for example, a sparse global linear approximation to a Gaussian process regression model can be used to determine which contacting amino-acid pairs are important for channelrhodopsin membrane localization, as shown in Figure 1.2b[40]. Alternatively, the layer activations and weights within a neural network can be directly examined for inputs of interest. Convolution weights indicate the relative importance of sequence motifs to the property predicted (a convolution layer scans across a

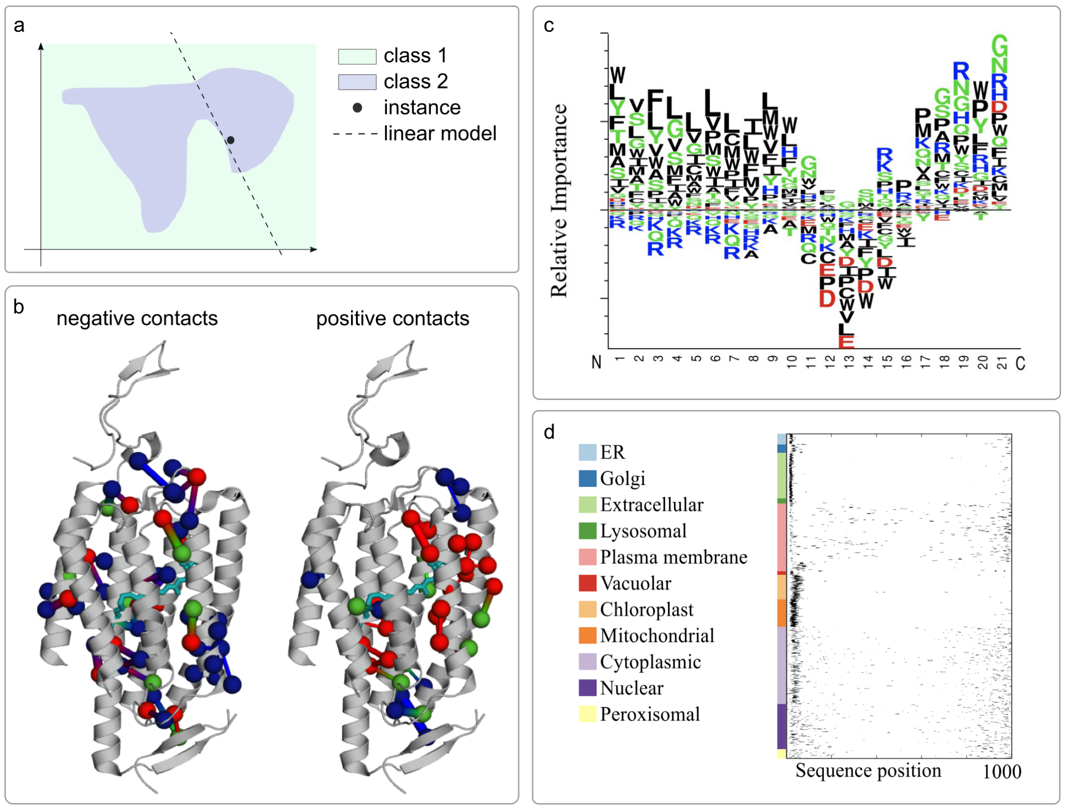


Figure 1.2: Examples of model interpretation. (a) Local approximation. The original model's decision function is represented by the violet/green background, and is clearly nonlinear. The black dot is the instance being explained, and a linear model (dashed line) that approximates the original model well in the vicinity of the instance is learned. Note that the explanation is not accurate globally, but is accurate locally around the instance. (b) Most important contacts for predicting channelrhodopsin localization to the plasma membrane. Contacts with the largest positive and negative weights in a Bayesian ridge regression approximation to a Gaussian process regression model are depicted on the channelrhodopsin crystal structure[40]. Each set of two balls and a stick represents two contacting amino acids. The amino acids are colored according to the source parent. The bound retinal cofactor is shown in cyan. (c) Convolution weights. Visualization of one convolutional filter from a model trained to predict the intracellular location of proteins[88]. The size of each amino acid represents its importance at that position in the k-mer. (d) Attention activations. Importance weights assigned to different regions of proteins in a model trained to predict intracellular location[88].

sequence looking for the presence of a learned motif). Activations within attention layers indicate the sections of each input sequence that were most important to the prediction. Figure 1.2c and d visualize a convolution layer and attention activations, respectively, for predictions of protein subcellular localization[77].

1.3 Machine learning as a guide to directed evolution

Directed evolution accumulates beneficial mutations in iterations of mutagenesis and selection or screening. There are an enormous number of ways to mutate any given protein: for a 300-amino acid protein there are 5700 possible single amino acid substitutions and 32,381,700 ways to make just two substitutions with the 20 canonical amino acids. Additionally, screening is expensive, time-consuming, and often limited in throughput. While directed evolution discards information about unimproved variants, machine-learning methods can use this information to expedite evolution and expand the properties that can be optimized by intelligently selecting new variants to screen. The only added costs are in computation and DNA sequencing, both of which are decreasing rapidly. Figure 1.1 compares directed evolution with and without machine learning as a guide, and Table 1.1 summarizes some studies that have used machine learning to guide directed evolution.

However, machine learning is not beneficial in all applications. In cases where the fitness landscape is smooth enough (i.e. essentially additive), machine learning may not significantly decrease screening burden or find better variants. In these cases, the added cost of sequencing DNA to form sequence-function relationships is unnecessary. Because one major benefit of machine learning is in reducing the quantity of sequences to test, machine learning will be especially useful in cases where the lack of a high-throughput screen limits or precludes directed evolution.

A machine learning-guided evolution strategy requires a method for generating diversity, a screen to evaluate diversity, a machine-learning model that learns the relationship between sequence and function, and a method to use the model to choose mutations for the next round of evolution. Examples of each are discussed here, with the exception of choosing a machine-learning model, which was outlined above.

Generating diversity

A straightforward method of generating diversity is to make random mutations throughout the length of the protein by error-prone polymerase chain reaction (PCR). In most directed evolution strategies, beneficial mutations are discovered by screening and accumulated until a satisfactory level of performance is reached. However, multiple mutations with error-prone PCR can occur, in which case mutations that are generally beneficial may be masked by co-occurrence with (more prevalent) detrimental mutations. Fortunately, even a simple linear model can be sufficient to recover accurate classifications of mutations as beneficial, deleterious, or neutral, allowing more beneficial mutations to be fixed more quickly than by directed evolution alone[54,92,93].

Site-saturation mutagenesis randomizes selected locations within the sequence determined to be most responsible for function or most likely to tolerate mutation. These sites are identified through previous studies or by knowledge of the protein's structure and mechanism. If only one or two pre-identified sites are considered, then machine learning is not necessary, as the entire library can be screened to identify optimal variants. If limited sets of amino acids are tested at each position, more positions can be explored simultaneously[94].

Recombination methods can make larger jumps in sequence space while preserving a large fraction of functional sequences by only considering diversity from within a set of related proteins[95]. The sequence elements to be swapped may be chosen randomly or rationally. Structure-guided recombination uses a 3D structure to choose the boundaries of sequence blocks to swap in order to balance maximizing diversity, minimizing disruption to the structure, and having evenly-sized blocks[96].

Using machine learning to select new variants

An initial set of variants to screen can be selected at random from the library[54], to maximize information about the mutations considered[89,97,98], or to maximize information about the remainder of the library[40,69,99]. Further rounds of learning and selection can be done either by collecting mutations believed to be beneficial or by directly optimizing over sequences in the library. To label mutations as beneficial, linear models of the mutational effects can be learned and the parameters can be directly interpreted to classify mutations as beneficial, neutral, or deleterious. The most beneficial mutations can then be fixed, deleterious mutations can be eliminated from the pool of considered mutations, and new mutations can be added to the pool of considered mutations[54].

Alternatively, learning and selection can be performed directly over sequences in the library. Typically, this is done using a non-parametric model. Instead of choosing the form of a function and learning parameters that best fit the data, non-parametric models directly learn a function that explains the data well without assuming its form. This makes non-parametric models useful for problems such as predicting protein properties from protein sequence, where the form of the function is not obvious (and indeed may not be possible to write down and parameterize). Non-parametric models used in protein engineering include adaptive substituent reordering, or more commonly, Gaussian process models. Adaptive substituent reordering algorithms (ASRAs) account for epistasis by constructing a function-free model of the underlying fitness landscape. In ASRA, a protein of length L resides in an L-dimensional space and can be described by a vector of substituents $x_1, x_2, ..., x_L$. Each substituent x_i is an integer between 1 and 20 indicating the amino acid at that position. Given sequence-function measurements, properties of unmeasured sequences can be estimated by interpolation within this space. The ordering of the substituents, however, defines the smoothness of the space and therefore also the accuracy of the interpolations. In ASRA, an ordering at each site is learned that balances smoothness and training set accuracy[100,101].

In addition to being non-parametric, Gaussian processes are also probabilistic: they provide an estimate of uncertainty. This allows a principled trade-off between exploiting the information learned from previous iterations and exploring the remainder of the library at each iteration. For example, the Gaussian Process Upper Confidence Bound (GP-UCB) algorithm balances exploration and exploitation by selecting variants that maximize the weighted sum of the predictive mean and standard deviation[102], and is guaranteed to asymptotically minimize the cumulative regret (difference between sampled variants and best variant) over infinite iterations. Figure 1.3 demonstrates two iterations of the GP-UCB algorithm. Alternatively, the model and data can be fully exploited using the Gaussian Process Lower Confidence Bound algorithm, which selects variants that maximize the weighted difference between the predictive mean and standard deviation. These approaches have been combined with a structure-guided recombination library to optimize cytochrome P450 thermostability[39], channelrhodopsin localization to mammalian cell membranes[40], and channelrhodopsin light-activated conductance[69]. Because there is no high-throughput screen for the channelrhodopsin properties, they would have been difficult or impossible to optimize with directed evolution alone. Alternately, the GP model can be used to select combinations of mutations in a multi-site site saturation library that has a high probability of containing improved variants[68].

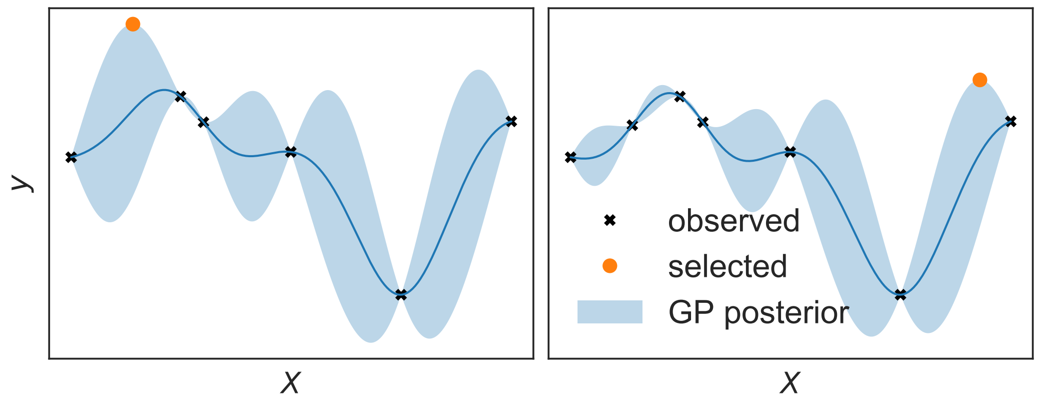


Figure 1.3: Gaussian Process Upper Confidence Bound algorithm. At each iteration, the next point to be sampled is chosen by maximizing the weighted sum of the posterior mean and standard deviation. This balances exploration and exploitation by exploring points that are both uncertain and have a high posterior mean.

Table 1.1: Comparison of selected studies using machine learning to guide directed evolution

Protein	**Property**	***n* screened**	**Model**
halohydrin dehydrogenase	volumetric productivity	60,000	linear[54,92,93]
epoxide hydrolase	enantioselectivity	20,000	ASRA[101]
proteinase K	activity, heat tolerance	95	linear[89]
glutathione transferase	catalytic activity	95	linear[97]
glutathione transferase	catalytic activity	95	linear[98]
cytochrome P450	thermostability	~200	GP[39]
channelrhodopsin	membrane localization	~200	GP[40]
green fluorescent protein	fluorescence	296	GP[68]
channelrhodopsin	spectral properties	119	GP[69]

1.4 Conclusions and future directions

Supervised machine-learning methods have already demonstrated their utility in directed protein evolution. The biggest obstacle to future applications of machine learning to protein engineering is a lack of high-quality data. Protein mutation datasets are heavily influenced by experimental conditions such as buffer components, temperature, expression system, and baseline thresholds. While ProtaBank[23] is spearheading the development of a centralized collection of these variants with their experimental conditions, there is currently no organized way to collect protein mutation data from various experiments to use as benchmarks in machine learning experiments. The collections that do exist are plagued by inconsistencies and errors[36], and significant resources must be dedicated to maintaining the quality of these databases. As demonstrated by Jokinen and coworkers, one way to circumvent limited labeled data is to augment with computationally-generated examples[33]. However, accurate physics-based predictors for properties more complicated than stability and binding do not yet exist. Collections of robust protein sequence-function data would allow researchers to benchmark machine-learning methods for protein functions against a variety of proteins and functions.

Deep mutational scanning[103] combines a high-throughput screen with next-generation sequencing to generate large sequence-function datasets. In deep mutational scanning, variants are sorted by a selection criterion, such as fluorescence or binding affinity. The sequences are sorted into bins, and the frequency of each mutation is compared before and after selection to infer relative fitness values. There is thus no direct measurement of the property of interest for each variant, and if the gene is longer than the sequencing read length, deconvoluting the interactions of multiple distant mutations requires a DNA barcoding scheme. Nevertheless, deep mutational scanning provides large datasets that map a significant fraction of the single and some double mutants to a fitness measure. Alternatively, a deep mutational scanning dataset may map a complete multi-site site-saturation library. There are now deep mutational scanning datasets for a variety of proteins and properties, including green fluorescent protein[104], amidase[105], β-lactamase[106], β-glucosidase[107], HIV envelope protein[108], influenza nucleoprotein[109], influenza hemagglutinin[110], PhoQ[111], GB1[112], the DNA-binding domain of a steroid hormone receptor[113], and Gal4 transcription factor[114]. These datasets provide test beds for machine-learning methods that learn to predict the effects of small numbers of mutations or aim to improve on directed evolution for protein optimization[115].

Large quantities of unlabeled sequence data may also enable machine-learning models to generate artificial protein diversity leading to novel protein functions. Only a tiny fraction of the amino acid landscape encodes a functional protein, and the complete landscape is littered with cliffs and holes, where small changes in sequence result in complete loss of function. Natural and designed proteins are samples from the distribution of functional proteins. Selectively sampling from the distribution of possible proteins would enable large jumps to previously unexplored sections of sequence space that may contain novel functions. Generative models of the distribution of functional proteins would enable these large jumps and provide an attractive alternative to *de novo* design methods[116].

Unlike discriminative models that learn $p(y|x)$ in order to predict labels y given

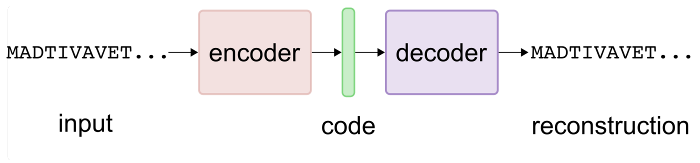


Figure 1.4: Autoencoder. An autoencoder consists of an encoder model and a decoder model. The encoder converts the input to a low-dimensional vector (code). The decoder reconstructs the input from this code. Typically, the encoder and decoder are both neural network models, and the entire autoencoder model is trained end-to-end. The learned code should contain sufficient information to reconstruct the inputs and can be used as input to other machine learning methods, or the autoencoder itself may be used as a generative model.

inputs x, generative models learn to generate examples that are similar to those in the training set but are not in the training set: they learn the generating distribution $p(x)$ for the training data. Tantalizingly, generative models in other fields have been trained to generate new faces[117], sketches[118], and even music[119]. Variational autoencoders additionally allow interpolation between examples or the ability to mix and match properties[120].

Instead of using neural network models to directly learn the mapping from protein sequence to function, Sinai *et al.* and Riesselman *et al.* trained variational autoencoders to learn the distribution of allowed mutations within functional protein families[115,121]. An autoencoder is a neural network that learns to encode an input as a vector (encoding) and then reconstruct the input from the vector (decoding) (Figure 1.4. By learning an encoding with smaller dimensionality than the original input, the model extracts the most important information from the input. The encoding can then be used as an information-dense input to other learning algorithms. In a variational autoencoder[120], the learned encoding is further constrained to encourage the encodings to be densely embedded in the encoding space, allowing smooth interpolations in that space. Applied to aligned families of protein sequences, variational autoencoders can learn complex epistatic relationships among mutations, allowing semi-supervised predictions of variant functionality based only on existing sequences without a need for individual measurements. Recently, recurrent neural networks and generative adverserial networks[122] have been used to generate novel antimicrobial peptides[123,124] and protein structures[125], and there has been an effort to develop a mathematical framework for using a generative model to sample sequences with one or more specified properties[126]. These early examples show the potential of generative models to discover sequences with novel desired functions.

This remains a promising and largely unexplored field.

Machine-learning methods have already expanded the proteins and properties that can be engineered by directed evolution. As researchers continue to collect sequence-function data in engineering experiments and to catalog the natural diversity of proteins, machine learning will be an invaluable tool to extract knowledge from protein data. Advances in both computational and experimental techniques, including generative models and deep mutational scanning, will also allow for better understanding of fitness landscapes and protein diversity.

References

1. Philip A Romero and Frances H Arnold: Exploring protein fitness landscapes by directed evolution. *Nat Rev Mol Cell Biol* **10**(12) (2009), 866. doi: 10.1038/nrm2805 (cit. on pp. 1, 2).
2. Wlodek Mandecki: The game of chess and searches in protein sequence space. *Trends Biotech* **16**(5) (1998), 200–202 (cit. on p. 1).
3. Niles A Pierce and Erik Winfree: Protein design is NP-hard. *Protein Eng* **15**(10) (2002), 779–782 (cit. on p. 1).
4. John Maynard Smith: Natural selection and the concept of a protein space. *Nature* **225**(5232) (1970), 563 (cit. on p. 1).
5. H Allen Orr: The distribution of fitness effects among beneficial mutations in Fisher's geometric model of adaptation. *J Theor Biol* **238**(2) (2006), 279–285 (cit. on p. 1).
6. Bassil I Dahiyat and Stephen L Mayo: De novo protein design: fully automated sequence selection. *Science* **278**(5335) (1997), 82–87 (cit. on p. 1).
7. Justin B. Siegel et al.: Computational design of an enzyme catalyst for a stereoselective bimolecular diels-alder reaction. *Science* **329**(5989) (2010), 309–313. doi: 10.1126/science.1190239 (cit. on p. 1).
8. Jiayi Dou et al.: De novo design of a fluorescence-activating β-barrel. *Nature* (2018), 1. doi: 10.1038/s41586-018-0509-0 (cit. on p. 2).
9. Jiayi Dou et al.: Sampling and energy evaluation challenges in ligand binding protein design. *Protein Sci* **26**(12) (2017), 2426–2437. doi: 10.1002/pro.3317 (cit. on p. 2).
10. Possu Huang, Scott E Boyken, and David Baker: The coming of age of de novo protein design. *Nature* **537**(7620) (2016), 320. doi: 10.1038/nature1994 (cit. on p. 2).
11. Marc Garcia-Borràs, Kendall N Houk, and Gonzalo Jiménez-Osés: Computational Design of Protein Function. *Computational Tools for Chemical Biology* **3** (2017), 87. doi: 10.1039/9781788010139-00087 (cit. on p. 2).

12. D Allan Drummond, Jonathan J Silberg, Michelle M Meyer, Claus O Wilke, and Frances H Arnold: On the conservative nature of intragenic recombination. *Proc Natl Acad Sci USA* **102**(15) (2005), 5380–5385 (cit. on p. 2).

13. Tibshirani Hastie and R Tibshirani: *The Elements of Statistical Learning; Data Mining, Inference and Prediction*. Springer, New York, 2008 (cit. on p. 2).

14. Kevin Murphy: *Machine learning, a probabilistic perspective*.
Murphy's textbook provides a thorough introduction to modern machine learning. MIT Press, 2012 (cit. on p. 2).

15. UniProt Consortium: UniProt: the universal protein knowledgebase. *Nucleic Acids Res* **45**(D1) (2017), D158–D169. doi: 10.1093/nar/gkw1099 (cit. on p. 4).

16. Peter W Rose et al.: The RCSB protein data bank: integrative view of protein, gene and 3D structural information. *Nucleic Acids Res* (2016). doi: 10.1093/nar/gkw952 (cit. on p. 4).

17. Sandra Placzek et al.: BRENDA in 2017: new perspectives and new tools in BRENDA. *Nucleic Acids Res* (2016), D380–D388 (cit. on p. 4).

18. MD Shaji Kumar et al.: ProTherm and ProNIT: thermodynamic databases for proteins and protein–nucleic acid interactions. *Nucleic Acids Res* **34**(suppl_1) (2006), D204–D206 (cit. on p. 4).

19. Iain H Moal and Juan Fernández-Recio: SKEMPI: a Structural Kinetic and Energetic database of Mutant Protein Interactions and its use in empirical models. *Bioinformatics* **28**(20) (2012), 2600–2607. doi: 10.1093/bioinformatics/bts489 (cit. on p. 4).

20. Sarah Sirin, James R Apgar, Eric M Bennett, and Amy E Keating: AB-Bind: antibody binding mutational database for computational affinity predictions. *Protein Sci* **25**(2) (2016), 393–409. doi: 10.1002/pro.2829 (cit. on p. 4).

21. Sherlyn Jemimah, K Yugandhar, and M Michael Gromiha: PROXiMATE: a database of mutant protein–protein complex thermodynamics and kinetics. *Bioinformatics* **33**(17) (2017), 2787–2788. doi: 10.1093/bioinformatics/btx312 (cit. on p. 4).

22. Takeshi Kawabata, Motonori Ota, and Ken Nishikawa: The protein mutant database. *Nucleic Acids Res* **27**(1) (1999), 355–357 (cit. on p. 4).

23. Connie Y Wang et al.: ProtaBank: A repository for protein design and engineering data. *Protein Sci* **27**(6) (2018), 1113–1124. doi: 10.1002/pro.3406 (cit. on pp. 4, 15).

24. Jian Tian, Ningfeng Wu, Xiaoyu Chu, and Yunliu Fan: Predicting changes in protein thermostability brought about by single- or multi-site mutations. *BMC Bioinformatics* **11**(1) (2010), 370. doi: 10.1186/1471-2105-11-37 (cit. on pp. 4, 7, 8).

25. Yunqi Li and Jianwen Fang: PROTS-RF: a robust model for predicting mutation-induced protein stability changes. *PloS One* **7**(10) (2012), e47247. doi: 10.1371/journal.pone.0047247 (cit. on pp. 4, 7, 8).

26. Lei Jia, Ramya Yarlagadda, and Charles C Reed: Structure based thermostability prediction models for protein single point mutations with machine learning tools. *PloS One* **10**(9) (2015), e0138022. doi: 10.1371/journal.pone.0138022 (cit. on pp. 4, 7, 8).

27. Emidio Capriotti, Piero Fariselli, and Rita Casadio: I-Mutant2.0: predicting stability changes upon mutation from the protein sequence or structure. *Nucleic Acids Res* **33**(suppl_2) (2005), W306–W310 (cit. on pp. 4, 8).

28. Emidio Capriotti, Piero Fariselli, Remo Calabrese, and Rita Casadio: Predicting protein stability changes from sequences using support vector machines. *Bioinformatics* **21**(suppl_2) (2005), ii54–ii58 (cit. on pp. 4, 8).

29. Jianlin Cheng, Arlo Randall, and Pierre Baldi: Prediction of protein stability changes for single-site mutations using support vector machines. *Proteins* **62**(4) (2006), 1125–1132 (cit. on pp. 4, 8).

30. Fabian A Buske, Ricarda Their, Elizabeth MJ Gillam, and Mikael Bodén: In silico characterization of protein chimeras: relating sequence and function within the same fold. *Proteins* **77**(1) (2009), 111–120. doi: 10.1002/prot.22422 (cit. on pp. 4, 6, 8).

31. Jianguo Liu and Xianjiang Kang: Grading amino acid properties increased accuracies of single point mutation on protein stability prediction. *BMC Bioinformatics* **13**(1) (2012), 44. doi: 10.1186/1471-2105-13-44 (cit. on pp. 4, 8).

32. Douglas EV Pires, David B Ascher, and Tom L Blundell: mCSM: predicting the effects of mutations in proteins using graph-based signatures. *Bioinformatics* **30**(3) (2013), 335–342. doi: 10.1093/bioinformatics/btt691 (cit. on pp. 4, 8).

33. Emmi Jokinen, Markus Heinonen, and Harri Lähdesmäki: mGPfusion: Predicting protein stability changes with Gaussian process kernel learning and data fusion. *Bioinformatics* **34**(13) (2018), i274–i283. doi: 10.1093/bioinformatics/bty238 (cit. on pp. 4, 8, 16).

34. Yves Dehouck et al.: Fast and accurate predictions of protein stability changes upon mutations using statistical potentials and neural networks: PoPMuSiC-2.0. *Bioinformatics* **25**(19) (2009), 2537–2543. doi: 10.1093/bioinformatics/btp445 (cit. on pp. 4, 8).

35. Manuel Giollo, Alberto JM Martin, Ian Walsh, Carlo Ferrari, and Silvio CE Tosatto: NeEMO: a method using residue interaction networks to improve prediction of protein stability upon mutation. *BMC Genomics* **15**(4) (2014), S7. doi: 10.1186/1471-2164-15-S4-S7 (cit. on pp. 4, 8).

36. Yang Yang et al.: PON-tstab: Protein Variant Stability Predictor. Importance of Training Data Quality. *Int J Mol Sci* **19**(4) (2018), 1009. doi: 10.3390/ijms19041009 (cit. on pp. 4, 15).

37. Pedro Domingos: A few useful things to know about machine learning. *Communications of the ACM* **55**(10) (2012), 78–87 (cit. on p. 5).

38. Yoshua Bengio, Aaron Courville, and Pascal Vincent: Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **35**(8) (2013), 1798–1828 (cit. on p. 5).

39. Philip A Romero, Andreas Krause, and Frances H Arnold: Navigating the protein fitness landscape with Gaussian processes. *Proc Natl Acad Sci USA* **110**(3) (2013). This is the first study to combine SCHEMA recombination with the GP-UCB algorithm to optimize a protein property., E193–E201. doi: 10.1073/pnas.1215251110 (cit. on pp. 5, 8, 14, 15).

40. Claire N Bedbrook, Kevin K Yang, Austin J Rice, Viviana Gradinaru, and Frances H Arnold: Machine learning to design integral membrane channelrhodopsins for efficient eukaryotic expression and plasma membrane localization. *PLoS Comput Biol* **13**(10) (2017), e1005786. doi: 10.1371/journal.pcbi.1005786 (cit. on pp. 5, 8, 10, 11, 13–15).

41. Emidio Capriotti, Piero Fariselli, and Rita Casadio: A neural-network-based method for predicting protein stability changes upon single point mutations. *Bioinformatics* **20**(suppl_1) (2004), i63–i68 (cit. on p. 5).

42. Shuichi Kawashima et al.: AAindex: amino acid index database, progress report 2008. *Nucleic Acids Res* **36**(suppl_1) (2007), D202–D205 (cit. on p. 6).

43. Dan Ofer and Michal Linial: ProFET: Feature engineering captures high-level protein functions. *Bioinformatics* **31**(21) (2015), 3429–3436. doi: 10.1093/bioinformatics/btv345 (cit. on p. 6).

44. Mark H Barley, Nicholas J Turner, and Royston Goodacre: Improved Descriptors for the Quantitative Structure–Activity Relationship Modeling of Peptides and Proteins. *J Chem Inf Model* **58**(2) (2018), 234–243. doi: 10.1021/acs.jcim.7b00488 (cit. on p. 6).

45. Jian Qiu, Martial Hue, Asa Ben-Hur, Jean-Philippe Vert, and William Stafford Noble: A structural alignment kernel for protein structures. *Bioinformatics* **23**(9) (2007), 1090–1098 (cit. on p. 6).

46. Steven Henikoff and Jorja G Henikoff: Amino acid substitution matrices from protein blocks. *Proc Natl Acad Sci USA* **89**(22) (1992), 10915–10919 (cit. on p. 6).

47. Ehsaneddin Asgari and Mohammad RK Mofrad: Continuous distributed representation of biological sequences for deep proteomics and genomics. *PloS One* **10**(11) (2015), e0141287. doi: 10.1371/journal.pone.0141287 (cit. on p. 6).

48. Carlo Mazzaferro: Predicting Protein Binding Affinity With Word Embeddings And Recurrent Neural Networks (2017), 128223. doi: 10.1101/128223. eprint: https://www.biorxiv.org/content/early/2017/04/18/128223 (cit. on pp. 6, 8).

49. Patrick Ng: dna2vec: Consistent vector representations of variable-length k-mers (2017). arXiv: https://arxiv.org/abs/1701.06279 [q-bio] (cit. on p. 6).

50. Dhananjay Kimothi, Akshay Soni, Pravesh Biyani, and James M Hogan: Distributed Representations for Biological Sequence Analysis (2016). eprint: https://arxiv.org/abs/1608.05949 (cit. on p. 6).

51. Kevin Yang, Zachary Wu, Claire Bedbrook, and Frances Arnold: Learned Protein Embeddings for Machine Learning. *Bioinformatics* (2018). doi: 10.1093/bioinformatics/bty178 (cit. on pp. 6, 7).

52. Ariel S Schwartz et al.: Deep Semantic Protein Representation for Annotation, Discovery, and Engineering. *bioRxiv* (2018), 365965. doi: 10.1101/365965. eprint: https://www.biorxiv.org/content/early/2018/07/10/365965 (cit. on p. 6).

53. David H Wolpert: The lack of a priori distinctions between learning algorithms. *Neural Comput* **8**(7) (1996), 1341–1390 (cit. on pp. 6, 7).

54. Richard J Fox et al.: Improving catalytic function by ProSAR-driven enzyme evolution. *Nat Biotechnol* **25**(3) (2007), 338 (cit. on pp. 7, 10, 13, 15).

55. Yougen Li et al.: A diverse family of thermostable cytochrome P450s created by recombination of stabilizing fragments. *Nat Biotechnol* **25**(9) (2007), 1051 (cit. on pp. 7, 10).

56. Leo Breiman: *Classification and Regression Trees*. Routledge, 2017 (cit. on p. 7).

57. Leo Breiman: Random forests. *Machine Learning* **45**(1) (2001), 5–32 (cit. on p. 7).

58. Jerome H Friedman: Stochastic gradient boosting. *Comput Stat Data An* **38**(4) (2002), 367–378. doi: 10.1016/S0167-9473(01)00065-2 (cit. on p. 7).

59. Corinna Cortes and Vladimir Vapnik: Support-vector networks. *Machine learning* **20**(3) (1995), 273–297 (cit. on p. 7).

60. E. Nadaraya: On Estimating Regression. *Theor Probab Appl* **9**(1) (1964), 141–142. doi: 10.1137/1109020 (cit. on p. 7).

61. Christina Leslie, Eleazar Eskin, and William Stafford Noble: The spectrum kernel: A string kernel for SVM protein classification. (2002), 564–575 (cit. on p. 8).

62. Christina S Leslie, Eleazar Eskin, Adiel Cohen, Jason Weston, and William Stafford Noble: Mismatch string kernels for discriminative protein classification. *Bioinformatics* **20**(4) (2004), 467–476 (cit. on p. 8).

63. Julian Zaugg, Yosephine Gumulya, Alpeshkumar K Malde, and Mikael Bodén: Learning epistatic interactions from sequence-activity data to predict enantioselectivity. *J Comput Aided Mol Des* **31**(12) (2017), 1085–1096. doi: 10.1007/s10822-017-0090-x (cit. on p. 8).

64. Shyam M Saladi, Nauman Javed, Axel Müller, and William M Clemons: A statistical model for improved membrane protein expression using sequence-derived features. *J Biol Chem* **293**(13) (2018), 4913–4927. doi: 10.1074/jbc.RA117.001052 (cit. on p. 8).

65. C. E. Rasmussen and C. K. I. Williams: *Gaussian Processes for Machine Learning*. MIT Press, 2006 (cit. on p. 8).

66. Andrew Gordon Wilson and Hannes Nickisch: Kernel Interpolation for Scalable Structured Gaussian Processes (KISS-GP). *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*. 2015, 1775–1784 (cit. on p. 8).

67. Joseph Mellor, Ioana Grigoras, Pablo Carbonell, and Jean-Loup Faulon: Semisupervised Gaussian process for automated enzyme search. *ACS Synth Biol* **5**(6) (2016), 518–528. doi: 10.1021/acssynbio.5b00294 (cit. on p. 8).

68. Yutaka Saito et al.: Machine-Learning-Guided Mutagenesis for Directed Evolution of Fluorescent Proteins. *ACS Synth Biol* (2018). doi: 10.1021/acssynbio.8b00155 (cit. on pp. 8, 14, 15).

69. Claire N Bedbrook, Kevin K Yang, Robinson Eliott J, Gradinaru Viviana, and Frances H Arnold: Machine learning-guided channelrhodopsin engineering enables minimally-invasive optogenetics. *In preparation* (2018) (cit. on pp. 8, 13–15).

70. Sai Zhang et al.: A deep learning framework for modeling structural features of RNA-binding protein targets. *Nucleic Acids Res* **44**(4) (2015), e32–e32. doi: 10.1093/nar/gkv1025 (cit. on p. 8).

71. Babak Alipanahi, Andrew Delong, Matthew T Weirauch, and Brendan J Frey: Predicting the sequence specificities of DNA-and RNA-binding proteins by deep learning. *Nat Biotechnol* **33**(8) (2015), 831. doi: 10.1038/nbt.3300 (cit. on p. 8).

72. Haoyang Zeng, Matthew D Edwards, Ge Liu, and David K Gifford: Convolutional neural network architectures for predicting DNA-protein binding. *Bioinformatics* **32**(12) (2016), i121–i127. doi: 10.1093/bioinformatics/btw255 (cit. on p. 8).

73. Jianjun Hu and Zhonghao Liu: DeepMHC: Deep Convolutional Neural Networks for High-performance peptide-MHC Binding Affinity Prediction (2017), 239236. doi: 10.1101/239236. eprint: https://www.biorxiv.org/content/early/2017/12/24/239236 (cit. on p. 8).

74. J Jiménez, S Doerr, G Martıénez-Rosell, AS Rose, and G De Fabritiis: DeepSite: protein-binding site predictor using 3D-convolutional neural networks. *Bioinformatics* **33**(19) (2017), 3036–3042. doi: 10.1093/bioinformatics/btx350 (cit. on p. 8).

75. Joseph Gomes, Bharath Ramsundar, Evan N Feinberg, and Vijay S Pande: Atomic convolutional networks for predicting protein-ligand binding affinity (2017). eprint: https://arxiv.org/abs/1703.10603 (cit. on p. 8).

76. Sameer Khurana et al.: DeepSol: A Deep Learning Framework for Sequence-Based Protein Solubility Prediction. *Bioinformatics* **1** (2018), 9. doi: 10.1093/bioinformatics/bty166 (cit. on p. 8).

77. Jose Juan Almagro Armenteros, Casper Kaae Sønderby, Søren Kaae Sønderby, Henrik Nielsen, and Ole Winther: DeepLoc: prediction of protein subcellular localization using deep learning. *Bioinformatics* **33**(21) (2017), 3387–3395. doi: 10.1093/bioinformatics/btx548 (cit. on pp. 8, 12).

78. Søren Kaae Sønderby and Ole Winther: Protein secondary structure prediction with long short term memory networks (2014). eprint: https://arxiv.org/abs/1412.7828 (cit. on p. 8).

79. Balázs Szalkai and Vince Grolmusz: Near perfect protein multi-label classification with deep neural networks. *Methods* **132** (2018), 50–56. doi: 10.1016/j.ymeth.2017.06.034 (cit. on p. 8).

80. Renzhi Cao et al.: ProLanGO: Protein Function Prediction Using Neural Machine Translation Based on a Recurrent Neural Network. *Molecules* **22**(10) (2017), 1732. doi: 10.3390/molecules22101732 (cit. on p. 8).

81. Thomas A. Hopf et al.: Three-dimensional structures of membrane proteins from genomic sequencing. *Cell* **149**(7) (2012), 1607–1621. doi: 10.1016/j.cell.2012.04.012 (cit. on p. 8).

82. Tolutola Oyetunde, Forrest Sheng Bao, Jiung-Wen Chen, Hector Garcia Martin, and Yinjie J. Tang: Leveraging knowledge engineering and machine learning for microbial bio-manufacturing. *Biotechnol Adv* **36**(4) (2018), 1308–1315. doi: 10.1016/j.biotechadv.2018.04.008 (cit. on p. 8).

83. Kevin Y. Yip, Chao Cheng, and Mark Gerstein: Machine learning and genome annotation: a match meant to be? *Genome Biol.* **14**(5) (May 2013), 205. doi: 10.1186/gb-2013-14-5-205 (cit. on p. 8).

84. Vanessa Isabell Jurtz et al.: An introduction to deep learning on biological sequence data: examples and solutions. *Bioinformatics* **33**(22) (2017), 3685–3690. doi: 10.1093/bioinformatics/btx531 (cit. on p. 8).

85. Tianwei Yue and Haohan Wang: Deep Learning for Genomics: A Concise Overview (2018). eprint: https://arxiv.org/abs/1802.00810 (cit. on p. 8).

86. Gregory Shakhnarovich, Trevor Darrell, and Piotr Indyk: *Nearest-neighbor methods in learning and vision: theory and practice*. The MIT press, 2006 (cit. on p. 8).

87. Zachary C Lipton and Jacob Steinhardt: Troubling Trends in Machine Learning Scholarship (2018), 1–15. arXiv: https://arxiv.org/abs/1807.03341 (cit. on p. 10).

88. Søren Kaae Sønderby, Casper Kaae Sønderby, Henrik Nielsen, and Ole Winther: Convolutional LSTM networks for subcellular localization of proteins. *International Conference on Algorithms for Computational Biology*. Springer. 2015, 68–80. doi: 10.1007/978-3-319-21233-3_6 (cit. on p. 11).

89. Jun Liao et al.: Engineering proteinase K using machine learning and synthetic genes. *BMC Biotechnol* **7**(1) (2007), 16 (cit. on pp. 10, 13, 15).

90. Dylan Alexander Carlin et al.: Kinetic characterization of 100 glycoside hydrolase mutants enables the discovery of structural features correlated with kinetic constants. *PloS one* **11**(1) (2016), e0147596. doi: 10.1371/journal.pone.0147596 (cit. on p. 10).

91. Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin: Why should I trust you?: Explaining the predictions of any classifier. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. 2016, 1135–1144. doi: 10.1145/2939672.2939778 (cit. on p. 10).

92. Richard Fox et al.: Optimizing the search algorithm for protein engineering by directed evolution. *Protein Eng* **16**(8) (2003).
This study is the first to use machine learning to guide directed evolution., 589–597 (cit. on pp. 13, 15).

93. Richard Fox: Directed molecular evolution by machine learning and the influence of nonlinear interactions. *J Theor Biol* **234**(2) (2005), 187–199 (cit. on pp. 13, 15).

94. Manfred T Reetz, Li-Wen Wang, and Marco Bocola: Directed evolution of enantioselective enzymes: iterative cycles of CASTing for probing protein-sequence space. *Angew Chem Int Ed Engl* **118**(8) (2006), 1258–1263. doi: 10.1002/anie.200502746 (cit. on p. 13).

95. Huimin Zhao, Lori Giver, Zhixin Shao, Joseph A. Affholter, and Frances H. Arnold: Molecular evolution by staggered extension process (StEP) in vitro recombination. *Nat Biotechnol* **16**(3) (1998), 258–261. doi: 10.1038/nbt0398-258 (cit. on p. 13).

96. Jeffrey B Endelman, Jonathan J Silberg, Zhen-Gang Wang, and Frances H Arnold: Site-directed protein recombination as a shortest-path problem. *Protein Eng Des Sel* **17**(7) (2004), 589–594 (cit. on p. 13).

97. Sridhar Govindarajan et al.: Mapping of amino acid substitutions conferring herbicide resistance in wheat glutathione transferase. *ACS Synth Biol* **4**(3) (2014), 221–227. doi: 10.1021/sb500242x (cit. on pp. 13, 15).

98. Yaman Musdal, Sridhar Govindarajan, and Bengt Mannervik: Exploring sequence-function space of a poplar glutathione transferase using designed information-rich gene variants. *Protein Eng Des Sel* **30**(8) (2017), 543–549. doi: 10.1093/protein/gzx045 (cit. on pp. 13, 15).

99. Philip A Romero, Andreas Krause, and Frances H Arnold: Navigating the protein fitness landscape with Gaussian processes. *Proc Natl Acad Sci USA* **110**(3) (2013), E193–E201. doi: 10.1073/pnas.1215251110 (cit. on p. 13).

100. Fan Liang, Xiao-Jiang Feng, Michael Lowry, and Herschel Rabitz: Maximal use of minimal libraries through the adaptive substituent reordering algorithm. *J Phys Chem B* **109**(12) (2005), 5842–5854 (cit. on p. 14).

101. Xiaojiang Feng, Joaquin Sanchis, Manfred T Reetz, and Herschel Rabitz: Enhancing the efficiency of directed evolution in focused enzyme libraries by the adaptive substituent reordering algorithm. *Chemistry* **18**(18) (2012), 5646–5654. doi: 10.1002/chem.201103811 (cit. on pp. 14, 15).

102. Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger: Gaussian process optimization in the bandit setting: No regret and experimental design (2009). eprint: https://arxiv.org/abs/:0912.3995 (cit. on p. 14).

103. Douglas M Fowler and Stanley Fields: Deep mutational scanning: a new style of protein science. *Nat Methods* **11**(8) (2014).
Deep mutational scanning can quickly generate large sequence-function datasets., 801. doi: 10.1038/nmeth.3027 (cit. on p. 16).

104. Karen S Sarkisyan et al.: Local fitness landscape of the green fluorescent protein. *Nature* **533**(7603) (2016), 397. doi: 10.1038/nature17995 (cit. on p. 16).

105. Emily E Wrenbeck, Laura R Azouz, and Timothy A Whitehead: Single-mutation fitness landscapes for an enzyme on multiple substrates reveal specificity is globally encoded. *Nat Commun* **8** (2017), 15695. doi: 10.1038/ncomms15695 (cit. on p. 16).

106. Justin R Klesmith, John-Paul Bacik, Emily E Wrenbeck, Ryszard Michalczyk, and Timothy A Whitehead: Trade-offs between enzyme fitness and solubility illuminated by deep mutational scanning. *Proc Natl Acad Sci USA* **114**(9) (2017), 2265–2270. doi: 10.1073/pnas.1614437114 (cit. on p. 16).

107. Philip A Romero, Tuan M Tran, and Adam R Abate: Dissecting enzyme function with microfluidic-based deep mutational scanning. *Proc Natl Acad Sci USA* **112**(23) (2015), 7159–7164. doi: 10.1073/pnas.1422285112 (cit. on p. 16).

108. Hugh K Haddox, Adam S Dingens, and Jesse D Bloom: Experimental estimation of the effects of all amino-acid mutations to HIV's envelope protein on viral replication in cell culture. *PLoS Pathog* **12**(12) (2016), e1006114. doi: 10.1371/journal.ppat.1006114 (cit. on p. 16).

109. Orr Ashenberg, Jai Padmakumar, Michael B Doud, and Jesse D Bloom: Deep mutational scanning identifies sites in influenza nucleoprotein that affect viral inhibition by MxA. *PLoS Pathog* **13**(3) (2017), e1006288. doi: 10.1371/journal.ppat.1006288 (cit. on p. 16).

110. Michael B Doud and Jesse D Bloom: Accurate measurement of the effects of all amino-acid mutations on influenza hemagglutinin. *Viruses* **8**(6) (2016), 155. doi: 10.3390/v8060155 (cit. on p. 16).

111. Anna I Podgornaia and Michael T Laub: Pervasive degeneracy and epistasis in a protein-protein interface. *Science* **347**(6222) (2015), 673–677. doi: 10.1126/science.1257360 (cit. on p. 16).

112. Nicholas C Wu, Lei Dai, C Anders Olson, James O Lloyd-Smith, and Ren Sun: Adaptation in protein fitness landscapes is facilitated by indirect paths. *Elife* **5** (2016), e16965. doi: 10.7554/eLife.16965 (cit. on p. 16).

113. Tyler N Starr, Lora K Picton, and Joseph W Thornton: Alternative evolutionary histories in the sequence space of an ancient protein. *Nature* **549**(7672) (2017), 409. doi: 10.1038/nature23902 (cit. on p. 16).

114. Jacob O Kitzman, Lea M Starita, Russell S Lo, Stanley Fields, and Jay Shendure: Massively parallel single-amino-acid mutagenesis. *Nat Methods* **12**(3) (2015), 203. doi: 10.1038/nmeth.3223 (cit. on p. 16).

115. Adam J Riesselman, John B Ingraham, and Debora S Marks: Deep generative models of genetic variation capture mutation effects. *Nat Methods* **15** (2018). This study predicts the effects of mutations without using any labeled data., 816–822. doi: 10.1038/s41592-018-0138-4 (cit. on pp. 16, 17).

116. David Baker: An exciting but challenging road ahead for computational enzyme design. *Protein Sci* **19**(10) (2010), 1817–1819. doi: 10.1002/pro.481 (cit. on p. 16).

117. Alec Radford, Luke Metz, and Soumith Chintala: Unsupervised representation learning with deep convolutional generative adversarial networks (2015). eprint: https://arxiv.org/abs/1511.06434 (cit. on p. 17).

118. David Ha and Douglas Eck: A neural representation of sketch drawings (2017). eprint: https://arxiv.org/abs/1704.03477 (cit. on p. 17).

119. Adam Roberts, Jesse Engel, Colin Raffel, Curtis Hawthorne, and Douglas Eck: A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music (2018). eprint: https://arxiv.org/abs/1803.05428 (cit. on p. 17).

120. Diederik P Kingma and Max Welling: Auto-encoding variational Bayes (2013). eprint: https://arxiv.org/abs/1312.6114 (cit. on p. 17).

121. Sam Sinai, Eric Kelsic, George M Church, and Martin A Nowak: Variational auto-encoding of protein sequences (2017). eprint: https://arxiv.org/abs/1712.03346 (cit. on p. 17).

122. Ian Goodfellow et al.: Generative adversarial nets. *Adv Neur In*. 2014, 2672–2680 (cit. on p. 17).

123. Alex T Müller, Jan A Hiss, and Gisbert Schneider: Recurrent Neural Network Model for Constructive Peptide Design. *J Chem Inf Model* (2018). doi: 10.1021/acs.jcim.7b00414 (cit. on p. 17).

124. Anvita Gupta and James Zou: Feedback GAN (FBGAN) for DNA: a Novel Feedback-Loop Architecture for Optimizing Protein Functions (2018). eprint: https://arxiv.org/abs/1804.01694 (cit. on p. 17).

125. Namrata Anand and Possu Huang: Generative modeling for protein structures. *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio et al. Curran Associates, Inc., 2018, 7504–7515 (cit. on p. 17).

126. David H Brookes and Jennifer Listgarten: Design by adaptive sampling (2018). eprint: https://arxiv.org/abs/1810.03714 (cit. on p. 17).

Chapter 2

MACHINE LEARNING TO DESIGN INTEGRAL MEMBRANE CHANNELRHODOPSINS FOR EFFICIENT EUKARYOTIC EXPRESSION AND PLASMA MEMBRANE LOCALIZATION

1. Claire N Bedbrook, Kevin K Yang, Austin J Rice, Viviana Gradinaru, and Frances H Arnold: Machine learning to design integral membrane channelrhodopsins for efficient eukaryotic expression and plasma membrane localization. *PLoS Computational Biology* **13**(10) (2017), e1005786. doi: 10.1371/journal.pcbi.1005786.

2.1 Introduction

As crucial components of regulatory and transport pathways, integral membrane proteins (MPs) are important pharmaceutical and engineering targets[1]. To be functional, MPs must be expressed and localized through a series of elaborate sub-cellular processes that include co-translational insertion, rigorous quality control, and multi-step trafficking to arrive at the correct topology in the correct sub-cellular location[2–4]. With such a complex mechanism for production, it is not surprising that MP engineering has been hampered by poor expression, stability, and localization in heterologous systems[5,6]. To overcome these limitations, protein engineers need a tool to predict how changes in sequence affect MP expression and localization. An accurate predictor would enable us to design and produce MP variants that express and localize correctly, a necessary first step in engineering MP function. A useful predictor would be sensitive to subtle changes in sequence that can lead to drastic changes in expression and localization. Our goal here was to develop data-driven models that predict the likelihood of a MP's expression and plasma membrane localization using the amino acid sequence as the primary input.

For this study, we focus on channelrhodopsins (ChRs), light-gated ion channels that assume a seven transmembrane helix topology with a light-sensitive retinal chromophore bound in an internal pocket. This scaffold is conserved in both microbial rhodopsins (light-driven ion pumps, channels, and light sensors–type I rhodopsins) and animal rhodopsins (light-sensing G-protein coupled receptors–type II rhodopsins)[7]. Found in photosynthetic algae, ChRs function as light sensors in phototaxic and photophobic responses[8,9]. On photon absorption, ChRs undergo a multi-step photo-cycle that allows a flux of ions across the membrane and down the electrochemical gradient[10]. When ChRs are expressed transgenically in neurons, their light-dependent activity can stimulate action potentials, allowing cell-specific control over neuronal activity[11,12] and extensive applications in neuroscience[13]. The functional limitations of available ChRs have spurred efforts to engineer or discover novel ChRs[10]. The utility of a ChR, however, depends on its ability to express and localize to the plasma membrane in eukaryotic cells of interest, and changes to the amino acid sequence frequently abrogate localization[5]. A predictor for ChRs that express and localize would be of great value as a pre-screen for function.

The sequence and structural determinants for membrane localization have been a subject of much scientific investigation[14–16] and have provided some understanding of the MP sequence elements important for localization, such as signal peptide sequence, positive charge at the membrane–cytoplasm interface (the "positive-inside" rule[17]), and increased hydrophobicity in the transmembrane domains. However, these rules are of limited use to a protein engineer: there are too many amino acid sequences that follow these rules but still fail to localize to the plasma membrane (see Results). MP sequence changes that influence expression and localization are highly context-dependent: what eliminates localization in one sequence context has no effect in another, and subtle amino acid changes can have dramatic effects[5,15,18]. In short, sequence determinants of expression and localization are not captured by simple rules.

Accurate atomistic physics-based models relating a sequence to its level of expression and plasma membrane localization currently do not exist, in large measure due to the complexity of the process. Statistical models offer a powerful alternative. Statistical models are useful for predicting the outcomes of complex processes because they do not require prior knowledge of the specific biological mechanisms involved. That being said, statistical models can also be constructed to exploit prior knowledge, such as MP structural information. Statistical models can be trained using empirical data (in this case expression or localization values) collected from known sequences. During training, the model infers relationships between input (sequence) and output (expression or localization) that are then used to predict the properties of unmeasured sequence variants. The process of using empirical data to train and select statistical models is referred to as machine learning.

Machine learning has been applied to predicting various protein properties, including solubility[19,20], trafficking to the periplasm[21], crystallization propensity[22], and function[23]. Generally, these models are trained using large data sets composed of literature data from varied sources with little to no standardization of the experimental conditions, and trained using many protein classes (i.e. proteins with various folds and functions), because their aim is to identify sequence elements across all proteins that contribute to the property of interest. This generalist approach, however, is not useful for identifying subtle sequence features (i.e. amino acids or amino acid interactions) that condition expression and localization for a specific class of related sequences, the ChRs in this case. We focused our model building on ChRs, with training data collected from a range of ChR sequences under standardized conditions. We applied Gaussian process (GP) classification and regression[24] to build models that predict ChR expression and localization directly from these data.

In our previous work, GP models successfully predicted thermal stability, substrate binding affinity, and kinetics for several soluble enzymes[25]. Here, we asked whether GP modeling could accurately predict mammalian expression and localization for heterologous integral membrane ChRs and how much experimental data would be required. For a statistical model to make accurate predictions on a wide range of ChR sequences, it must be trained with a diverse set of ChR sequences[24]. We chose to generate a training set using chimeras produced by SCHEMA recombination, which was previously demonstrated to be useful for producing large sets (libraries) of diverse, functional chimeric sequences from homologous parent proteins[26]. We synthesized and measured expression and localization for only a small subset (0.18%) of sequences from the ChR recombination library. Here we use these data to train GP classification and regression models to predict the expression and localization properties of diverse, untested ChR sequences. We first made predictions on sequences within a large library of chimeric ChRs; we then expanded the predictions to sequences outside that set.

2.2 Results

The ChR training set

The design and characterization of the chimeric ChR sequences used to train our models have been published[5]; we will only briefly describe these results. Two separate, ten-block libraries were designed by recombining three parental ChRs (CsChrimsonR (CsChrimR)[27], C1C2[28], and CheRiff[29]) with 45–55% amino acid sequence identity and a range of expression, localization, and functional properties (Figure 2.S1)[5]. Each chimeric ChR variant in these libraries is composed of blocks of sequence from the parental ChRs. These libraries were prepared by the SCHEMA algorithm to define sequence blocks for recombination that minimize the library-average disruption of tertiary protein structure[30,31]. One library swaps contiguous elements of primary structure (contiguous library), and the second swaps elements that are contiguous in the tertiary structure but not necessarily in the sequence (non-contiguous library[32]). The two libraries have similar, but not identical, element boundaries (Figure 2.S1A) and were constructed in order to test whether one design approach was superior to the other (they gave similar results). These designs generate 118,098 possible chimeras (2×3^{10}), which we will refer to as the recombination library throughout this paper. Each of these chimeras has a full N-terminal signal peptide from one of the three ChR parents.

Two hundred and eighteen chimeras from the recombination library were chosen as a training set, including all the chimeras with single-block swaps (chimeras consisting of 9 blocks of one parent and a single block from one of the other two parents) and multi-block-swap chimera sequences designed to maximize mutual information between the training set and the remainder of the chimeric library. Here, the 'information' a chimera has to offer is how its sequence, relative to all previously tested sequences, changes ChR expression and localization. By maximizing mutual information, we select chimera sequences that provide the most information about the whole library by reducing the uncertainty (Shannon entropy) of prediction for the remainder of the library, as described in[33,34]. The 112 single-block-swap chimeras in the training set have an average of 15 mutations from the most closely related parent, while the 103 multi-block-swap chimeras in the training set have an average of 73 mutations from the most closely related parent (Table 2.1). While the multi-block-swap chimeras provide the most sequence diversity to learn from, they are the least likely to express and localize given their high mutation levels. The single-block-swap chimeras offer less information to learn from due to their sequence redundancies with other chimeras in the training set, but are more likely to express and localize.

Genes for these sequences were synthesized and expressed in human embryonic kidney (HEK) cells, and their expression and membrane localization properties were measured (Figure 2.S1B)[5]. The expression levels were monitored through a fluorescent protein (mKate) fused to the C-termini of the ChRs. Plasma-membrane localization was measured using the SpyTag/SpyCatcher labeling method, which exclusively labels ChR protein that has its N terminus exposed on the extracellular surface of the cell[35]. The training set sequences displayed a wide range of expression

Table 2.1: Comparison of size, diversity, and localization properties of the training set and subsequent sets of chimeras chosen by models in the iterative steps of model development

Set	*n*	Mutations‡	% good localizers*	Localization‡
training – parents	3	0	100	5.6 ± 3.0
training – single-block swap	112	15 ± 9	33	3.2 ± 3.4
training – multi-block swap	103	73 ± 21	12	1.5 ± 2.5
exploration	16	69 ± 12	50	4.8 ± 4.7
verification – high performing	4	29 ± 17	100	8.0 ± 1.6
verification – low performing	7	67 ± 12	0	0.89 ± 0.73
optimization	4	43 ± 6	100	14 ± 3.5

‡mean ± standard deviation
* Localization at or above that of the lowest-performing parent, CheRiff

and localization properties. While the majority of the training set sequences express, only 33% of the single-block-swap chimeras localize well, and an even smaller fraction (12%) of the multi-block-swap chimeras localize well, emphasizing the importance of having a predictive model for membrane localization.

First we explored whether ChR chimera properties could be predicted based on basic biological properties, specifically, signal peptide sequence and hydrophobicity in the transmembrane (TM) domains. Each chimera in the library has one of the three parental signal peptides. Although the signal peptide sequence does affect expression and localization (Figure 2.S2A), chimeras with any parental signal peptide can have high or low expression and localization. Thus, the identity of the signal peptide alone is insufficient for accurate predictions of the ChR chimera properties. We then calculated the level of hydrophobicity within the 7-TM domains of each chimera. With very weak correlation between increasing hydrophobicity and measured expression and localization (Figure 2.S2B), hydrophobicity alone is also insufficient for accurate prediction of ChR chimera properties. These models do not accurately account for the observed levels of expression or localization (Figure 2.S1). Therefore, we need more expressive models to predict expression and localization from the amino acid sequences of these MPs.

Using GP models to learn about ChRs

Our overall strategy for developing predictive machine-learning models is illustrated in Figure 2.1. The goal is to use a set of ChR sequences and their expression and localization measurements to train GP regression and classification models that describe how ChR properties depend on sequence and predict the behavior of untested ChRs. GP models infer predictive values from training examples by assuming that similar inputs (ChR sequence variants) will have similar outputs (expression or localization). We quantify the relatedness of inputs (ChR sequence variants) by comparing both sequence and structure. ChR variants with few differences are considered more similar than ChR variants with many differences. We define the sequence similarity between two chimeras by aligning them and counting the number of positions at which they are identical. For structural comparisons, a residue-residue 'contact map' was built for each ChR variant, where two residues are in contact if they have any non-hydrogen atoms within 4.5 Å. The maps were generated using a ChR parental sequence alignment and the C1C2 crystal structure, which is the only available ChR structure[28], with the assumption that ChR chimeras share the overall contact architecture observed in the C1C2 crystal structure. The structural similarity for any two ChRs was quantified by aligning the contact maps and counting the number of identical contacts[25]. Using these metrics, we calculated the sequence and structural similarity between all ChRs in the training set relative to one another (218×218 ChR comparisons).

These similarity functions are called kernel functions and specify how the functional properties of pairs of sequences are expected to covary (they are also known as covariance functions). In other words, the kernel is a measure of similarity between sequences, and we can draw conclusions about unobserved chimeras on the basis of their similarity to sampled points[24]. The model has high confidence in predicting the properties of sequences that are similar to previously sampled sequences, and the model is less confident in predicting the properties of sequences that are distant from previously sampled sequences.

To build a GP model, we must also specify how the relatedness between sequences will affect the property of interest, in other words how sensitive the ChR properties are to changes in relatedness as defined by the sequence/structure differences between ChRs. This is defined by the form of the kernel used. We tested three different forms of sequence and structure kernels: linear kernels, squared exponential kernels, and Matérn kernels (see Methods). These different forms represent the kinds of functions we expect to observe for the protein's fitness landscape (i.e. the mapping of protein sequence to protein function). The linear kernel corresponds to a simple landscape where the effects of changes in sequence/structure are additive and there is no epistasis. The two non-linear kernels represent more rugged, com-

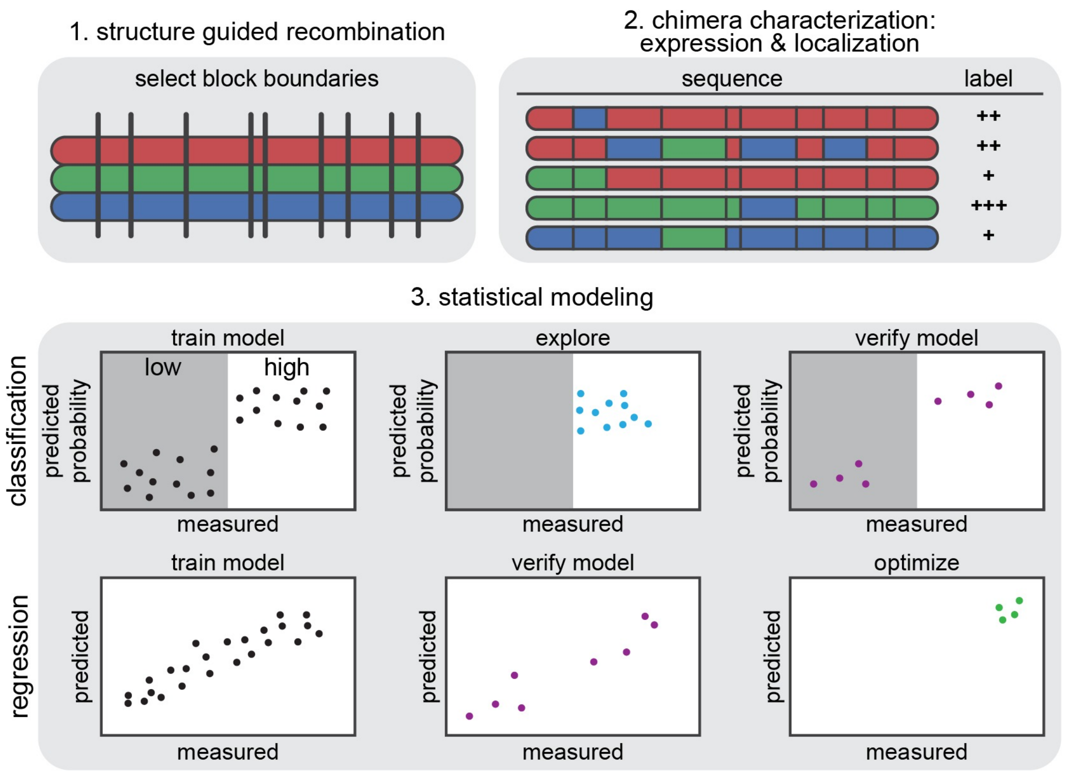


Figure 2.1: General approach to machine learning of protein (ChR) structure-function relationships: diversity generation, measurements on a training set, and modeling. (1) Structure-guided SCHEMA recombination is used to select block boundaries for shuffling protein sequences to generate a sequence-diverse ChR library starting from three parent ChRs (shown in red, green, and blue). (2) A subset of the library serves as the training set. Genes for these chimeras are synthesized and cloned into a mammalian expression vector, and the transfected cells are assayed for ChR expression and localization. (3) Two different models, classification and regression, are trained using the training data and then verified. The classification model is used to explore diverse sequences predicted to have 'high' localization. The regression model is used to design ChRs with optimal localization to the plasma membrane.

plex landscapes where effects may be non-additive. Learning involves optimizing the form of the kernel and its hyperparameters (parameters that influence the form of kernel) to enable accurate predictions. The hyperparameters and the form of the kernel were optimized using the Bayesian method of maximizing the marginal likelihood of the resulting model. The marginal likelihood (i.e. how likely it is to observe the data given the model) rewards models that fit the training data well while penalizing model complexity to prevent overfitting.

Once trained with empirical data, the output of the GP regression model is a predicted mean and variance, or standard deviation, for any given ChR sequence variant. The standard deviation is an indication of how confident the model is in the prediction based on the relatedness of the new input relative to the tested sequences.

We used GP models to infer links between ChR properties and ChR sequence and structure from the training data. We first built GP binary classification models. In binary classification, the outputs are class labels i.e. 'high' or 'low' localization, and the goal is to use the training set data to predict the probability of a sequence falling into one of the two classes (Figure 2.1). We also built a GP regression model that makes real-valued predictions, i.e. amount of localized protein, based on the training data (Figure 2.1). After training these models, we verify that their predictions generalize to sequences outside of the training set. Once validated, these two models can be used in different ways. A classification model trained from localization data can be used to predict the probability of highly diverse sequences falling into the 'high' localization category (Figure 2.1). The classification model can only predict if a sequence has 'high' vs 'low' localization, and it cannot be used to optimize localization. The regression model, on the other hand, can be used to predict sequences with 'optimal' properties; for example, a regression model trained from localization data can predict untested sequences that will have very high levels of localization (Figure 2.1).

Building GP classification models of ChR properties

The training set data (Figure 2.S1) were used to build a GP classification model that predicted which of the 118,098 chimeras in the recombination library would have 'high' vs 'low' expression, localization, and localization efficiency. The training set includes multi-block swaps chosen to be distant from other sequences in the training set in order to provide information on sequences throughout the recombination library. A sequence was considered 'high' if it performed at least as well as the lowest performing parent, and it was considered ‘low’ if it performed worse than the lowest performing parent. Because the lowest performing parent for expression and localization, CheRiff, is produced and localized in sufficient quantities for downstream functional studies, we believe this to be an appropriate threshold for ‘high’ vs ‘low’ performance. For all of the classification models (Figure 2.2 and Figure 2.S3), we used kernels based on structural relatedness. For the expression classification model, we found that a linear kernel performed best, i.e. achieved the highest marginal likelihood. This suggests that expression is best approximated by an additive model weighting each of the structural contacts. Localization and localization efficiency required a non-linear kernel for the model to be predictive. This more expressive kernel allows for non-linear relationships and epistasis and also penalizes differing structural contacts more than the linear kernel. This reflects our intuitive understanding that localization is a more demanding property to tune than expression, with stricter requirements and a non-linear underlying fitness landscape.

Most of the multi-block-swap sequences from the training set did not localize to the membrane[5]. We nonetheless want to be able to design highly mutated ChRs that localize well because these are most likely to have interesting functional properties. We therefore used the localization classification model to identify multi-block-swap chimeras from the library that had a high predicted probability (>0.4) of falling into the ‘high’ localizer category (Figure 2.2D). From the many multi-block-swap chimeras predicted to have ‘high’ localization, we selected a set of 16 highly diverse chimeras with an average of 69 amino acid mutations from the closest parent and called this the ‘exploration’ set (Figure 2.S4). We synthesized and tested these chimeras and found that the model had accurately predicted chimeras with good localization (Figure 2.2 and Figure 2.3): 50% of the exploration set show ‘high’ localization compared to only 12% of the multi-block-swap sequences from the original training set, even though they have similar levels of mutation (Table 2.1 and S1 Data) (chimeras in the exploration set have on average 69 ± 12 amino acid mutations from the closest parent, versus 73 ± 21 for the multi-block-swap chimeras in the training set). The classification model provides a four-fold enrichment in the number of chimeras that localize well when compared to randomly-selected chimeras with equivalent levels of mutation. This accuracy is impressive given that the exploration set was designed to be distant from any sequence the model had seen during training. The model’s performance on this exploration set indicates its ability to predict the properties of sequences distant from the training set.

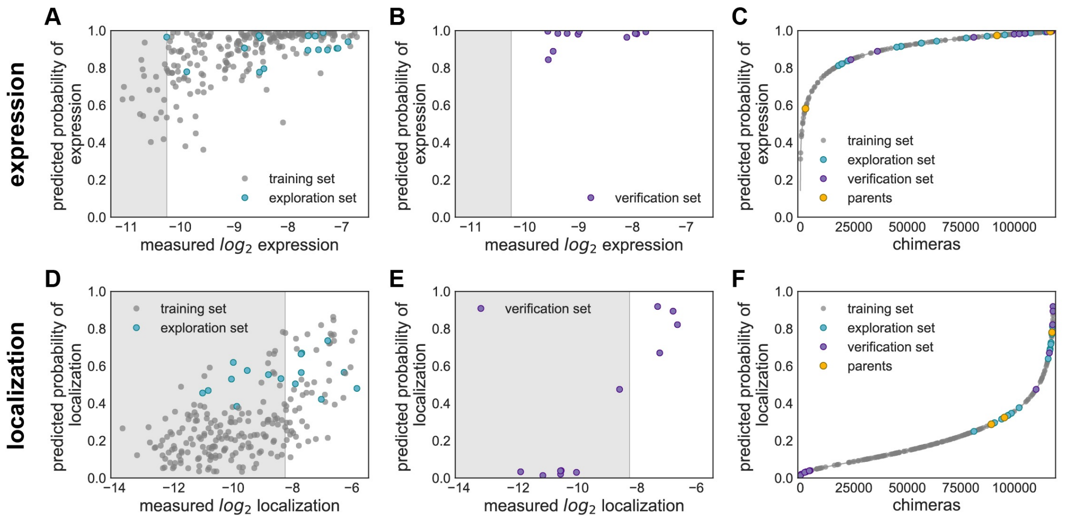


Figure 2.2: GP binary classification models for expression and localization. Plots of predicted probability vs measured properties are divided into 'high' performers (white background) and 'low' performers (gray background) for each property (expression and localization). (A) & (D) Predicted probability vs measured properties for the training set (gray points) and the exploration set (cyan points). Predictions for the training and exploration sets were made using LOO cross-validation. (B) & (E) Predicted probabilities vs measured properties for the verification set. Predictions for the verification set were made by a model trained on the training and exploration sets. (C) & (F) Predicted probability of 'high' expression, and localization for all chimeras in the recombination library (118,098 chimeras) made by models trained on the data from the training and exploration sets. The gray line shows all chimeras in the library, the gray points indicate the training set, the cyan points indicate the exploration set, the purple points indicate the verification set, and the yellow points indicate the parents. (A–C) Show expression and (D–F) show localization. For all plots, the measured property is plotted on a $\log_2$ scale.

The data from the exploration set were then used to better inform our models about highly diverse sequences that localize. To characterize the classification model's performance, we calculated the area under the receiver operating characteristic (ROC) curve (AUC). A poorly performing model would not do better than random chance, resulting in an AUC of 0.5, while a model that perfectly separates the two classes will have an AUC of 1.0. The revised models achieved AUC up to 0.87 for "leave-one-out" (LOO) cross-validation, indicating that there is a high probability that the classifiers will accurately separate 'high' and 'low' performing sequences for the properties measured. The AUC is 0.83 for localization, 0.77 for localization efficiency and 0.87 for expression for LOO cross-validation predictions (Figure 2.S5).

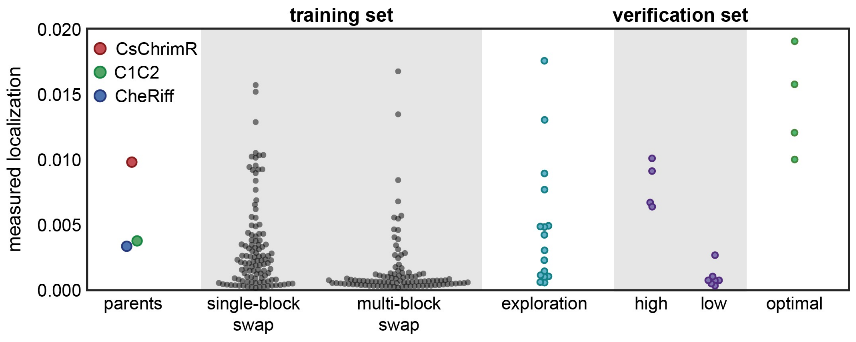


Figure 2.3: Comparison of measured membrane localization for each data set. Swarm plots of localization measurements for each data set compared with parents: training set, exploration set, verification set, and optimization set.

To further test the models, we then built a verification set of eleven chimeras, designed using the localization model. This verification set was composed of four chimeras predicted to be highly likely to localize, six chimeras predicted to be very unlikely to localize, and one chimera with a moderate predicted probability of localizing (Figure 2.S4). The measured localization (Figure 2.2E) and localization efficiency (Figure 2.S3B) of the chimeras in the verification set show clear differences, 'high' vs 'low', consistent with the model predictions (Table 2.1 and S1 Data). The verification sets consist exclusively of chimeras with 'high' measured expression, which is consistent with the model's predictions (Figure 2.2B). The model perfectly classifies the eleven chimeras as either 'high' or 'low' for each property (expression, localization, or localization efficiency) as shown in plots of predicted vs measured properties (Figure 2.2B and 2.2E and Figure 2.S3B) and by perfect separation in ROC curves i.e. AUC = 1.0 (Figure 2.S5). These models are powerful tools that can confidently predict whether a chimera will have 'high' or 'low' expression (Figure 2.2C), localization (Figure 2.2F), and localization efficiency (Figure 2.S3C). Of the 118,098 chimeras in the recombination library, 6,631 (5.6%) are predicted to have a probability > 0.5 of 'high' localization, whereas the vast majority of chimeras (99%) are predicted to have a probability > 0.5 of 'high' expression.

Building a regression model for ChR localization

The classification model predicts the probability that a sequence falls into the 'high' localizer category, but does not give a quantitative prediction as to how well it localizes. Our next goal was to design chimera sequences with optimal localization. Localization is considered optimal if it is at or above the level of CsChrimR, the best localizing parent, which is more than adequate for *in vivo* applications using ChR functionality to control neuronal activity[27]. A regression model for ChR plasma membrane localization is required to predict sequences that have optimal levels of localization. We used the localization data from the training and exploration sets to train a GP regression model (Figure 2.4A). The diversity of sequences in the training data allows the model to generalize well to the remainder of the recombination library. For this regression model, we do not use all of the features from the combined sequence and structure information; instead, we used L1 linear regression to select a subset of these features. The L1 linear regression identifies the sequence and structural features that most strongly influence ChR localization. Using this subset of features instead of all of the features improved the quality of the predictions (as determined by cross-validation). This indicates that not all of the residues and residue-residue contacts have a large influence on localization of ChR. We then used a kernel based on these chosen features (specific contacts and residues) for GP regression. The regression model for localization showed strong predictive ability as indicated by the strong correlation between predicted and measured localization for LOO cross-validation (correlation coefficient, $R > 0.76$) (Figure 2.4A). This was further verified by the strong correlation between predicted and measured values for the previously-discussed verification set ($R > 0.9$) (Figure 2.4A). These cross-validation results suggest that the regression model can be used to predict chimeras with optimal localization.

We used the localization regression model to predict ChR chimeras with optimal localization using the Lower Confidence Bound (LCB) algorithm, in which the predicted mean minus the predicted standard deviation (LB1) is maximized[36]. The LCB algorithm maximally exploits the information learned from the training set by finding sequences the model is most certain will be good localizers. The regression model was used to predict the localization level and standard deviation for all chimeras in the library, and from this the LB1 was calculated for all chimeras (Figure 2.4B). We selected four chimeras whose LB1 predictions for localization were ranked in the top 0.1% of the library (Figure 2.S4). These were constructed and tested (Figure 2.3 and Figure 2.S6 and S1 Data). Measurements showed that they all localize as well as or better than CsChrimR (Figure 2.3 and Figure 2.4A and Table 2.1). Cell population distributions of the optimal set show properties similar to the CsChrimR parent, with one chimera showing a clear shift in the peak of the distribution towards higher levels of localization (Figure 2.S7). These four sequences differ from CsChrimR at 30 to 50 amino acids (Figure 2.S4).

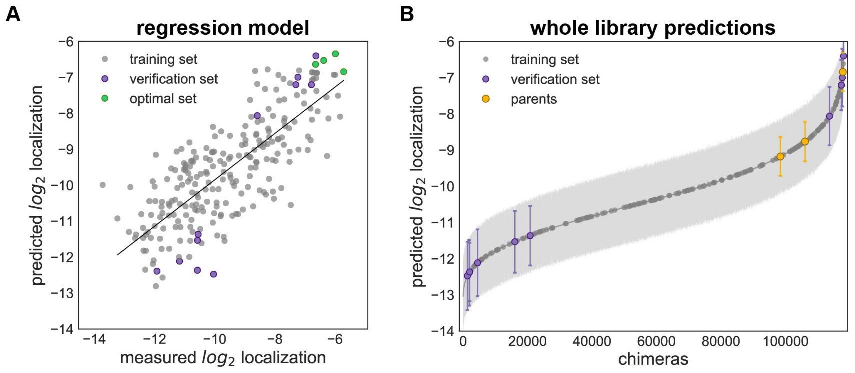


Figure 2.4: GP regression model for localization. (A) Predicted vs measured localization for the combined training and exploration sets (gray points), verification set (purple points), and the optimal set (green points). Predictions for the training and exploration sets were made using LOO cross-validation; predictions for the verification and optimal sets were made by a model trained on data from the training and exploration sets. There is clear correlation between predicted and measured localization. The combined training and exploration sets showed good correlation ($R > 0.73$) as did the verification set ($R > 0.9$). (B) Predicted localization values of all chimeras in the recombination library (118,098 chimeras) based on the GP regression model trained on the training and exploration sets. The gray line shows all chimeras in the library, the gray points indicate the training set and exploration sets, the purple points indicate the verification set, and the yellow points indicate the parents. Error bars (light gray shading) show the standard deviation of the predictions. For all plots, the predicted and measured localization are plotted on a $\log_2$ scale.

We were interested in how predictive the GP localization models could be with fewer training examples. To assess the predictive ability of the GP models as a function of training set size, we sampled random sets of training sequences from the dataset, trained models on these random sets, then evaluated the model's performance on a selected test set (Figure 2.S8). As few as 100 training examples are sufficient for accurate predictions for both the localization regression and classification models. This analysis shows that the models would have been predictive with even fewer training examples than we chose to use.

Sequence and structure features that facilitate prediction of ChR expression and localization

In developing the GP regression model for localization, we used L1-regularized linear regression to identify a limited set of sequence and structural features that strongly influence ChR localization (Figure 2.4). These features include both inter-residue contacts and individual residues and offer insight into the structural determinants of ChR localization. To better gauge the relative importance of these features, L2-regularized linear regression was used to calculate the positive and negative feature weights, which are proportional to each feature's inferred contribution to localization. While not as predictive as the GP regression model because it cannot account for higher-order interactions between features, this linear model has the advantage of being interpretable.

When mapped onto the C1C2 structure, these features highlight parts of the ChR sequence and structural contacts that are important for ChR localization to the plasma membrane (Figure 2.5). Both beneficial and deleterious features are distributed throughout the protein, with no single feature dictating localization properties (Figure 2.5). Clusters of heavily weighted positive contacts suggest that having structurally proximal CsChrimR-residue pairs are important in the N-terminal domain (NTD), between the NTD and TM4, between TM1 and TM7, and between TM3 and TM7. CsChrimR residues at the extracellular side of TM5 also appear to aid localization, although they are weighted less than CheRiff residues in the same area. Beneficial CheRiff contacts and residues are found in the C-terminal domain (CTD), the interface between the CTD and TM5–6, and in TM1. C1C2 residues at the extracellular side of TM6 are also positively weighted for localization, as are C1C2 contacts between the CTD and TM3–4 loop. From the negatively weighted contacts, it is clear that total localization is harmed when CheRiff contributes to the NTD or the intracellular half of TM4 and when CsChrimR contributes to the CTD. Interestingly, positive contacts were formed between TM6 from C1C2 and TM7 from CheRiff, but when the contributions were reversed (TM6 from CheRiff TM7 from C1C2) or if CsChrimR contributed TM6, strong negative weights were observed. Not surprisingly, the sequence and structure of optimal localizers predicted by GP regression (Figure 2.4) largely agree with the L2 weights (Figure 2.S9).

Using this strategy for model interpretation (L1 regression for feature selection followed by L2 regression), we can also weight the contributions of residues and contacts for ChR expression (Figure 2.S10 and Figure 2.S11). There is some overlap between the heavily weighted features for ChR expression and the features for localization, which is expected because more protein expressed means more protein available for localization. For example, both expression and localization models seem to prefer the NTD from CsChrimR and the extracellular half of TM6 from C1C2, and both disfavor the NTD and the intra-cellular half of TM4 from CheRiff. While the heavily-weighted expression features are limited to these isolated sequence regions, localization features are distributed throughout the protein. Moreover, the majority of heavily-weighted features identified for expression are residues rather than contacts. This is in contrast to those weighted features identified for localization, which include heavily-weighted residues and structural contacts. This suggests that sequence is more important in determining expression properties, which is consistent with the largely sequence-dependent mechanisms associated with successful translation and insertion into the ER membrane. In contrast, both sequence and specific structural contacts contribute significantly to whether a ChR will localize to the plasma membrane. Our results demonstrate that the model can 'learn' the features that contribute to localization from the data and make accurate predictions on that property.

Using the GP regression model to engineer novel sequences that localize

We next tested the ChR localization regression model for its ability to predict plasma-membrane localization for ChR sequences outside the recombination library. For this, we chose a natural ChR variant, CbChR1, that expresses in HEK cells and neurons but does not localize to the plasma membrane and thus is non-functional[27]. CbChR1 is distant from the three parental sequences, with 60% identity to CsChrimR and 40% identity to CheRiff and C1C2. We optimized CbChR1 by introducing minor amino acid changes predicted by the localization regression model to be beneficial for membrane localization. To enable measurement of CbChR1 localization with the SpyTag-based labeling method, we substituted the N-terminus of CbChR1 with the CsChrimR N-terminus containing the SpyTag sequence downstream of the signal peptide to make the chimera CsCbChR1[35]. This block swap did not change the membrane localization properties of CbChR1 (Figure 2.6C). Using the regression model, we predicted localization levels for all the possible single-block swaps from the three library parents (CsChrimR, C1C2 and CheRiff) into CsCbChR1 and selected the four chimeras with the highest Upper Confidence Bound (UCB). These chimeras have between 4 and 21 mutations when compared with CsCbChR1. Unlike the LCB algorithm, which seeks to find the safest optimal choices, the UCB algorithm balances exploration and exploitation by maximizing the sum of the predicted mean and standard deviation.

The selected chimeras were assayed for expression, localization, and localization efficiency (S1 Data). One of the four sequences did not express; the other three chimeras expressed and had higher localization levels than CsCbChR1 (Figure 2.6B). Two of the three had localization properties similar to the CheRiff parent (Figure 2.6B).

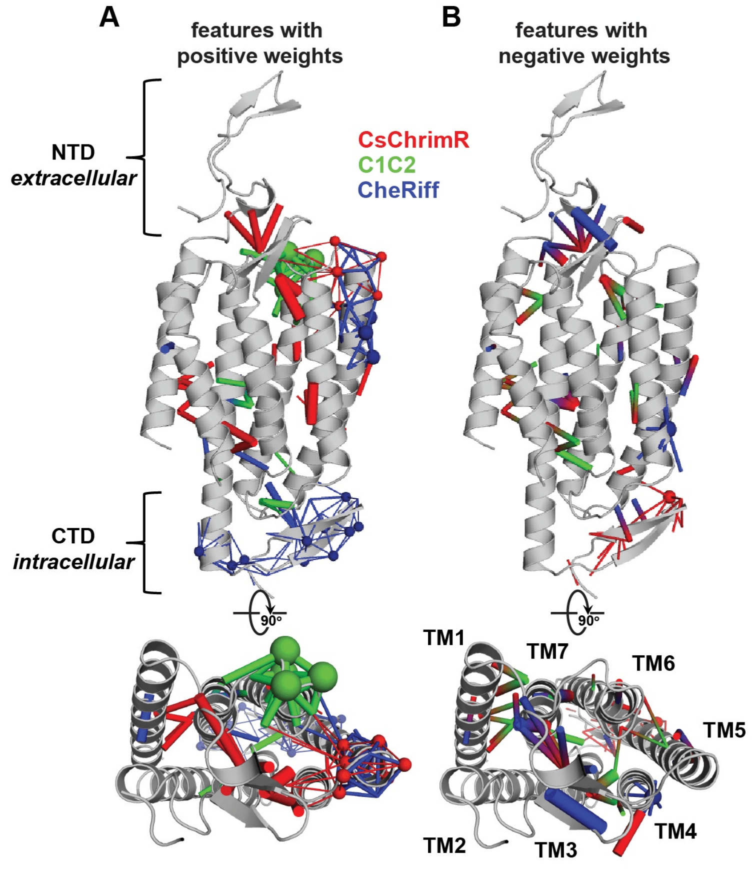


Figure 2.5: Sequence and structural contact features important for prediction of ChR localization. Features with positive (A) and negative (B) weights are displayed on the C1C2 crystal structure (grey). Features can be residues (spheres) or contacts (sticks) from one or more parent ChRs. Features from CsChrimR are shown in red, features from C1C2 are shown in green, and features from CheRiff are shown in blue. In cases where a feature is present in two parents, the following color priorities were used for consistency: red above green above blue. Sticks connect the beta carbons of contacting residues (or alpha carbon in the case of glycine). The size of the spheres and the thickness of the sticks are proportional to the parameter weights. Two residues in contact can be from the same or different parents. Single-color contacts occur when both contributing residues are from the same parent. Multi-color contacts occur when residues from different parents are in contact. The N-terminal domain (NTD), C-terminal domain (CTD), and the seven transmembrane helices (TM1–7) are labeled.

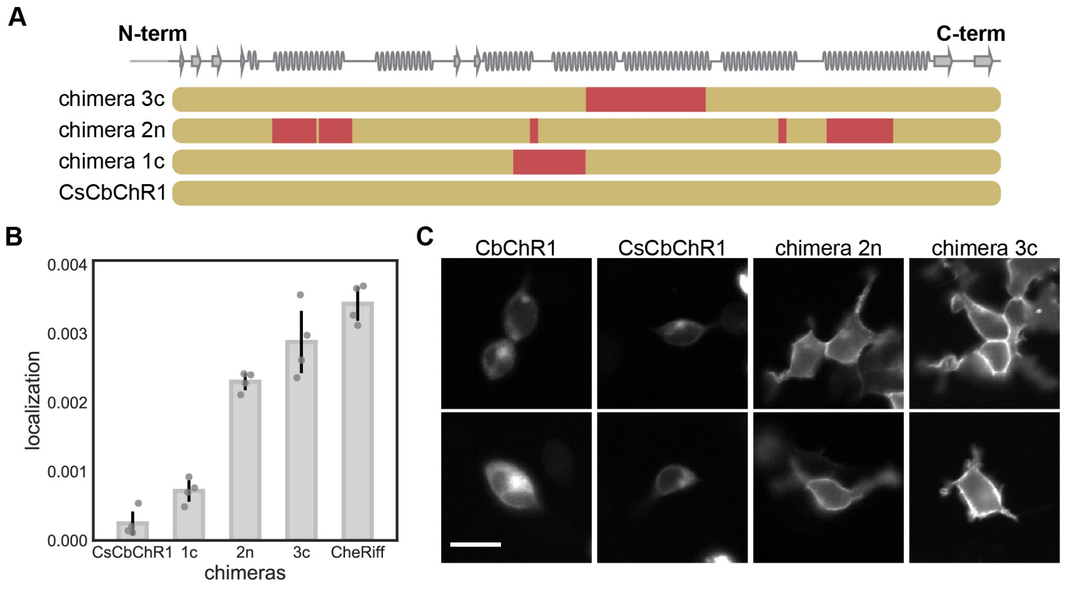


Figure 2.6: GP regression model enables engineering of localization in CbChR1. (A) Block identities of the CsCbChR1 chimeras. Each row represents a chimera. Yellow represents the CbChR1 parent and red represents the CsChrimR parent. Chimeras 1c, 2n, and 3c have 4, 21, and 17 mutations with respect to CsCbChR1, respectively. (B) Plot of measured localization of CsCbChR1 compared to three CsCbChR1 single-block-swap chimeras and the CheRiff parent. (C) Two representative cell images of mKate expression of CbChR1 and CsCbChR1 compared with top-performing CsCbChR1 single-block-swap chimeras show differences in ChR localization properties–chimera 2n and chimera 3c clearly localize to the plasma membrane. Scale bar: 20 μm.

Images of the two best localizing chimeras illustrate the enhancement in localization when compared with CbChR1 and CsCbChR1 (Figure 2.6C and Figure 2.S12). This improvement in localization was achieved through single-block swaps from CsChrimR (17 and 21 amino acid mutations) (Figure 2.6A). These results suggest that this regression model can accurately predict minor sequence changes that will improve the membrane localization of natural ChRs.

2.3 Discussion

The ability to differentiate the functional properties of closely related sequences is extremely powerful for protein design and engineering. This is of particular interest for protein types that have proven to be more recalcitrant to traditional protein design methods, e.g. MPs. We show here that integral membrane protein expression and plasma membrane localization can be predicted for novel, homologous sequences using moderate-throughput data collection and advanced statistical modeling. We have used the models in four ways: 1) to accurately predict which diverse, chimeric ChRs are likely to express and localize at least as well as a moderately-performing native ChR; 2) to design ChR chimeras with optimized membrane localization that matched or exceeded the performance of a very well-localizing ChR (CsChrimR); 3) to identify the structural interactions (contacts) and sequence elements most important for predicting ChR localization; and 4) to identify limited sequence changes that transform a native ChR from a non-localizer to a localizer.

Whereas 99% of the chimeras in the recombination library are predicted to express in HEK cells, only 5.6% are predicted to localize to the membrane at levels equal to or above the lowest parent (CheRiff). This result shows that expression is robust to recombination-based sequence alterations, whereas correct plasma-membrane localization is much more sensitive. The model enables accurate selection of the rare, localization-capable, proteins from the nearly 120,000 possible chimeric library variants. In future work we will show that this diverse set of several thousand variants predicted to localize serves as a highly enriched source of functional ChRs with novel properties.

Although statistical models generalize poorly as one attempts to make predictions on sequences distant from the sequences used in model training, we show that it is possible to train a model that accurately distinguishes between closely related proteins. The tradeoff between making accurate predictions on subtle sequence changes vs generalized predictions for significantly different sequences is one we made intentionally in order to achieve accurate predictions for an important and interesting class of proteins. Accurate statistical models, like the ones described in this paper, could aid in building more expressive physics-based models.

This work details the steps in building machine-learning models and highlights their power in predicting desirable protein properties that arise from the intersection of multiple cellular processes. Combining recombination-based library design with statistical modeling methods, we have scanned a highly functional portion of protein sequence space by training on only 218 sequences. Model development through iterative training, exploration, and verification has yielded a tool that not only predicts optimally performing chimeric proteins, but can also be applied to improve related ChR proteins outside the library. As large-scale gene synthesis and DNA sequencing become more affordable, machine-learning methods such as those described here will become ever more powerful tools for protein engineering offering an alternative to high-throughput assay systems.

Materials and methods

The design, construction, and characterization of recombination library chimeras is described in Bedbrook et al.[5]. Briefly, HEK 293T cells were transfected with purified ChR variant DNA using Fugene6 reagent according to the manufacturer's recommendations. Cells were given 48 hours to express before expression and localization were measured. To assay localization level, transfected cells were subjected to the SpyCatcher-GFP labeling assay, as described in Bedbrook et al.[35]. Transfected HEK cells were then imaged for mKate and GFP fluorescence using a Leica DMI 6000 microscope (for cell populations) or a Zeiss LSM 780 confocal microscope (for single cells: Figure 2.S12). Images were processed using custom image processing scripts for expression (mean mKate fluorescence intensity) and localization (mean GFP fluorescence intensity). All chimeras were assayed under identical conditions.

For each chimera, net hydrophobicity was calculated by summing the hydrophobicity of all residues in the TM domains. The C1C2 crystal structure was used to identify residues within TM domains (Figure 2.S2B), and the Kyte & Doolittle amino acid hydropathicity scale[37] was used to score residue hydrophobicity.

GP modeling

Both the GP regression and classification modeling methods applied in this paper are based on work detailed in[25]. Romero et al. applied GP models to predict protein functions and also defined protein distance using a contact map. We have expanded on this previous work. Regression and classification were performed using open-source packages in the SciPy ecosystem[38–40]. Below are specifics of the GP regression and classification methods used in this paper. The hyperparameters and the form of the kernel were optimized using the Bayesian method of maximizing the marginal likelihood of the resulting model.

GP regression In regression, the problem is to infer the value of an unknown function $f(x)$ at a novel point x_* given observations y at inputs X. Assuming that the observations are subject to independent identically distributed Gaussian noise with variance σ_n^2, the posterior distribution of $f_* = f(x_*)$ for Gaussian process regression is Gaussian with mean

$$\bar{f}_* = k_*^T (K + \sigma_n^2 I)^{-1} \mathbf{y} \tag{2.1}$$

and variance

$$v_* = k(x_*, x_*) - k_*^T (K + \sigma_n^2 I)^{-1} k_* \tag{2.2}$$

Where

1. K is the symmetric, square covariance matrix for the training set, where $K_{ij} = k(x_i, x_j)$ for x_i and x_j in the training set.
2. k_* is the vector of covariances between the novel input and each input in the training set, where $k_{*i} = k(x_*, x_i)$.

We found that results could be improved by first performing feature selection with L1-regularized linear regression and then only training the GP model on features with non-zero weights in the L1 regression. The hyperparameters in the kernel functions, the noise hyperparameter σ_n and the regularization hyperparameter were determined by maximizing the log marginal likelihood:

$$\log p\,(\mathbf{y}|\gamma, X) \propto -\mathbf{y}^T (K_\gamma + \sigma_n^2 I)^{-1} \mathbf{y} - \log|K_\gamma + \sigma_n^2 I| \tag{2.3}$$

where n is the dimensionality of the inputs.

GP classification. In binary classification, instead of continuous outputs y, the outputs are class labels $y_i \in \{+1, -1\}$, and the goal is to use the training data to make probabilistic predictions $\pi(x_*) = p(y_* = +1|x_*)$. Unfortunately, the posterior distribution for classification is analytically intractable. We use Laplace's method to approximate the posterior distribution. There is no noise hyperparameter in the classification case. Hyperparameters in the kernels are also found by maximizing the marginal likelihood.

GP kernels for modeling proteins. Gaussian process regression and classification models require kernel functions that measure the similarity between protein sequences. A protein sequence s of length l is defined by the amino acid present at each location. This information can be encoded as a binary feature vector x_{se} that indicates the presence or absence of each amino acid at each position. The protein's structure can be represented as a residue-residue contact map. The contact-map can be encoded as a binary feature vector x_{st} that indicates the presence or absence of each possible contacting pair. The sequence and structure feature vectors can also be concatenated to form a sequence-structure feature vector.

We considered three types of kernel functions $k(s_i, s_j)$: linear kernels, squared exponential kernels, and Matérn kernels. The linear kernel is defined as:

$$k(s, s') = \sigma_p^2 x^T x' \tag{2.4}$$

where σ_p is a hyperparameter that determines the prior variance of the fitness landscape. The squared exponential kernel is defined as:

$$k(s, s') = \sigma_p^2 \exp\left(\frac{\|x - x'\|_2^2}{l^2}\right) \tag{2.5}$$

where l and σ_p are also hyperparameters and $\| \cdot \|_2$ is the L2 norm. Finally, the Matérn kernel with $\nu = \frac{5}{2}$ is defined as:

$$k(s, s') = \left(1 + \frac{\sqrt{(}5\|x - x'\|_2^2)}{l} + \frac{5\|x - x'\|_2^2}{3l^2}\right) \exp\left(-\frac{5\|x - x'\|_2^2}{l}\right) \tag{2.6}$$

Where l is once again a hyperparameter.

L1 regression feature identification and weighting. To identify those contacts in the ChR structure most important in determining chimera function (here, localization) we used L1 regression. Given the nature of our library design and the limited set of chimeras tested, there are certain residues and contacts that covary within our training set. The effects of these covarying residues and contacts cannot be isolated from one another using this data set and therefore must be weighted together for their overall contribution to ChR function. By using the concatenated sequence and structure binary feature vector for the training set we were able to identify residues and contacts that covary. Each individual set of covarying residues and contacts was combined into a single feature. L1 linear regression was then used to weight features as either zero or non-zero in their contribution to ChR function. The level of regularization was chosen by LOO cross-validation. We then performed Bayesian ridge linear regression on features with non-zero L1 regression weights using the default settings in scikit-learn[41]. The Bayesian ridge linear regression weights were plotted onto the C1C2 structure to highlight positive and negative contributions to ChR localization (Figure 2.5) and ChR expression (Figure 2.S11).

References

1. John P Overington, Bissan Al-Lazikani, and Andrew L Hopkins: How many drug targets are there? *Nat Rev Drug Discov* **5**(12) (2006), 993. doi: 10.1038/nrd2199 (cit. on p. 28).
2. Florian Cymer, Gunnar Von Heijne, and Stephen H White: Mechanisms of integral membrane protein insertion and folding. *J Mol Bio* **427**(5) (2015), 999–1022. doi: 10.1016/j.jmb.2014.09.014 (cit. on p. 28).
3. J Paul Chapple and Michael E Cheetham: The chaperone environment at the cytoplasmic face of the endoplasmic reticulum can modulate rhodopsin processing and inclusion formation. *J Bio Chem* **278**(21) (2003), 19087–19094. doi: 10.1074/jbc.M212349200 (cit. on p. 28).
4. Marcus CS Lee, Elizabeth A Miller, Jonathan Goldberg, Lelio Orci, and Randy Schekman: Bi-directional protein transport between the ER and Golgi. *Annu Rev Cell Dev Biol* **20** (2004), 87–123 (cit. on p. 28).
5. Claire N Bedbrook et al.: Structure-guided SCHEMA recombination generates diverse chimeric channelrhodopsins. *Proc Natl Acad Sci USA* (2017), 201700269. doi: 10.1073/pnas.1700269114 (cit. on pp. 28–31, 36, 46).
6. Marco Schütz et al.: Directed evolution of G protein-coupled receptors in yeast for higher functional production in eukaryotic expression hosts. *Sci Rep* **6** (2016), 21508. doi: 10.1038/srep21508 (cit. on p. 28).
7. John L Spudich, Chii-Shen Yang, Kwang-Hwan Jung, and Elena N Spudich: Retinylidene proteins: structures and functions from archaea to humans. *Annu Rev Cell Dev Bi* **16**(1) (2000), 365–392 (cit. on p. 28).
8. Takeshi Suzuki et al.: Archaeal-type rhodopsins in Chlamydomonas: model structure and intracellular localization. *Biochem Biophys Res Commun* **301**(3) (2003), 711–717 (cit. on p. 28).
9. Oleg A Sineshchekov, Kwang-Hwan Jung, and John L Spudich: Two rhodopsins mediate phototaxis to low-and high-intensity light in Chlamydomonas reinhardtii. *Proc Nat Acad Sci USA* **99**(13) (2002), 8689–8694. doi: 10.1073/pnas.122243399 (cit. on p. 28).
10. Franziska Schneider, Christiane Grimm, and Peter Hegemann: Biophysics of channelrhodopsin. *Annu Rev Biophys* **44** (2015), 167–186. doi: 10.1146/annurev-biophys-060414-034014 (cit. on p. 29).
11. Edward S Boyden, Feng Zhang, Ernst Bamberg, Georg Nagel, and Karl Deisseroth: Millisecond-timescale, genetically targeted optical control of neural activity. *Nat Neurosci* **8**(9) (2005), 1263 (cit. on p. 29).

12. Toru Ishizuka, Masaaki Kakuda, Rikita Araki, and Hiromu Yawo: Kinetic evaluation of photosensitivity in genetically engineered neurons expressing green algae light-gated channels. *Neurosci Res* **54**(2) (2006), 85–94 (cit. on p. 29).

13. Ofer Yizhar, Lief E Fenno, Thomas J Davidson, Murtaza Mogri, and Karl Deisseroth: Optogenetics in neural systems. *Neuron* **71**(1) (2011), 9–34. doi: 10.1016/j.neuron.2011.06.004 (cit. on p. 29).

14. Karen G Fleming: Energetics of membrane protein folding. *Annu Rev Biophys* **43** (2014), 233–255. doi: 10.1146/annurev-biophys-051013-022926 (cit. on p. 29).

15. Assaf Elazar et al.: Mutational scanning reveals the determinants of protein insertion and association energetics in the plasma membrane. *Elife* **5** (2016), e12125. doi: 10.7554/eLife.12125 (cit. on p. 29).

16. Stephen H White and William C Wimley: Membrane protein folding and stability: physical principles. *Annu Rev Biophys Biomol Struct* **28**(1) (1999), 319–365 (cit. on p. 29).

17. Gunnar von Heijne: Control of topology and mode of assembly of a polytopic membrane protein by positively charged residues. *Nature* **341**(6241) (1989), 456 (cit. on p. 29).

18. Mylinh T Duong, Todd M Jaszewski, Karen G Fleming, and Kevin R MacKenzie: Changes in apparent free energy of helix–helix dimerization in a biological membrane due to point mutations. *J Mol Biol* **371**(2) (2007), 422–434. doi: 10.1016/j.jmb.2007.05.026 (cit. on p. 29).

19. Narjeskhatoon Habibi, Siti Z Mohd Hashim, Alireza Norouzi, and Mohammed Razip Samian: A review of machine learning methods to predict the solubility of overexpressed recombinant proteins in Escherichia coli. *BMC bioinformatics* **15**(1) (2014), 134. doi: 10.1186/1471-2105-15-134 (cit. on p. 29).

20. David L Wilkinson and Roger G Harrison: Predicting the solubility of recombinant proteins in Escherichia coli. *Nat Biotech* **9**(5) (1991), 443 (cit. on p. 29).

21. Catherine Ching Han Chang et al.: Periscope: quantitative prediction of soluble protein expression in the periplasm of *Escherichia coli*. *Sci Rep* **6** (2016), 21844. doi: 10.1038/srep2184 (cit. on p. 29).

22. Huilin Wang et al.: Crysalis: an integrated server for computational analysis and design of protein crystallization. *Sci Rep* **6** (2016), 21383. doi: 10.1038/srep21383 (cit. on p. 29).

23. Predrag Radivojac et al.: A large-scale evaluation of computational protein function prediction. *Nat Methods* **10**(3) (2013), 221. doi: 10.1038/nmeth.2340 (cit. on p. 30).

24. C. E. Rasmussen and C. K. I. Williams: *Gaussian Processes for Machine Learning*. MIT Press, 2006 (cit. on pp. 30, 33).

25. Philip A Romero, Andreas Krause, and Frances H Arnold: Navigating the protein fitness landscape with Gaussian processes. *Proc Natl Acad Sci USA* **110**(3) (2013), E193–E201. doi: 10.1073/pnas.1215251110 (cit. on pp. 30, 33, 46).

26. Philip A Romero and Frances H Arnold: Exploring protein fitness landscapes by directed evolution. *Nat Rev Mol Cell Biol* **10**(12) (2009), 866. doi: 10.1038/nrm2805 (cit. on p. 30).

27. Nathan C Klapoetke et al.: Independent optical excitation of distinct neural populations. *Nat Methods* **11**(3) (2014), 338. doi: 10.1038/nmeth.2836 (cit. on pp. 30, 39, 42).

28. Hideaki E Kato et al.: Crystal structure of the channelrhodopsin light-gated cation channel. *Nature* **482**(7385) (2012), 369. doi: 10.1038/nature10870 (cit. on pp. 30, 33).

29. Daniel R Hochbaum et al.: All-optical electrophysiology in mammalian neurons using engineered microbial rhodopsins. *Nat Methods* **11**(8) (2014), 825. doi: 10.1038/nmeth.3000 (cit. on p. 30).

30. Christopher A Voigt, Carlos Martinez, Zhen-Gang Wang, Stephen L Mayo, and Frances H Arnold: Protein building blocks preserved by recombination. *Nat Struct Biol* **9**(7) (2002), 553. doi: 10.1038/nsb805 (cit. on p. 31).

31. Yougen Li et al.: A diverse family of thermostable cytochrome P450s created by recombination of stabilizing fragments. *Nat Biotechnol* **25**(9) (2007), 1051 (cit. on p. 31).

32. Matthew A Smith, Philip A Romero, Timothy Wu, Eric M Brustad, and Frances H Arnold: Chimeragenesis of distantly-related proteins by noncontiguous recombination. *Protein Sci* **22**(2) (2013), 231–238. doi: 10.1002/pro.2202 (cit. on p. 31).

33. Philip A Romero et al.: SCHEMA-designed variants of human arginase I and II reveal sequence elements important to stability and catalysis. *ACS Synth Biol* **1**(6) (2012), 221–228. doi: 10.1021/sb300014t (cit. on p. 31).

34. A Krause and n D Golovi: *Submodular function maximization. Tractability: Practical approaches to hard problems*. Cambridge University Press, 2014 (cit. on p. 31).

35. Claire N Bedbrook et al.: Genetically encoded spy peptide fusion system to detect plasma membrane-localized proteins in vivo. *Chem Bio* **22**(8) (2015), 1108–1121. doi: 10.1016/j.chembiol.2015.06.020 (cit. on pp. 31, 42, 46).

36. Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger: Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design. *Proc. International Conference on Machine Learning (ICML)*. 2010 (cit. on p. 39).

37. Jack Kyte and Russell F Doolittle: A simple method for displaying the hydropathic character of a protein. *J Mol Bio* **157**(1) (1982), 105–132 (cit. on p. 46).

38. Stéfan van der Walt, S Chris Colbert, and Gael Varoquaux: The NumPy array: a structure for efficient numerical computation. *Computing in Science & Engineering* **13**(2) (2011), 22–30 (cit. on p. 46).

39. John D Hunter: Matplotlib: A 2D graphics environment. *Computing in science & engineering* **9**(3) (2007), 90–95 (cit. on p. 46).

40. Travis E Oliphant: Python for scientific computing. *Computing in Science & Engineering* **9**(3) (2007) (cit. on p. 46).

41. Fabian Pedregosa et al.: Scikit-learn: Machine learning in Python. *Journal of machine learning research* **12**(Oct) (2011), 2825–2830 (cit. on p. 48).

Supplemental Information

Dataset 1: Localization and expression characterization of ChR chimeras predicted by the models. Measured localization and expression properties for each chimera tested and associated chimera_name, set, number of mutations, chimera_block_ID, and sequence. Chimera names and chimera_block_ID begin with either ‘c’ or ‘n’ to indicate the contiguous or non-contiguous library. The following 10 digits in the chimera_block_ID indicate, in block order, the parent that contributes each of the 10 blocks (‘0’:CheRiff, ‘1’:C1C2, and ‘2’:CsChrimR). For the contiguous library, blocks in the chimera_block_ID are listed from N- to C-termini; for the non-contiguous library the block order is arbitrary. The set for which the chimera was generated is listed. The number of mutations (m) from the closest parent for each chimera is included. Sequences list only the ChR open reading frame, the C-terminal trafficking and mKate2.5 sequences have been removed. The table shows mean properties (mKate_mean, GFP_mean, and intensity_ratio_mean) and the standard deviation of properties (mKate_std, GFP_std, and intensity_ratio_std). ND: not detected, below the limit of detection for our assay.

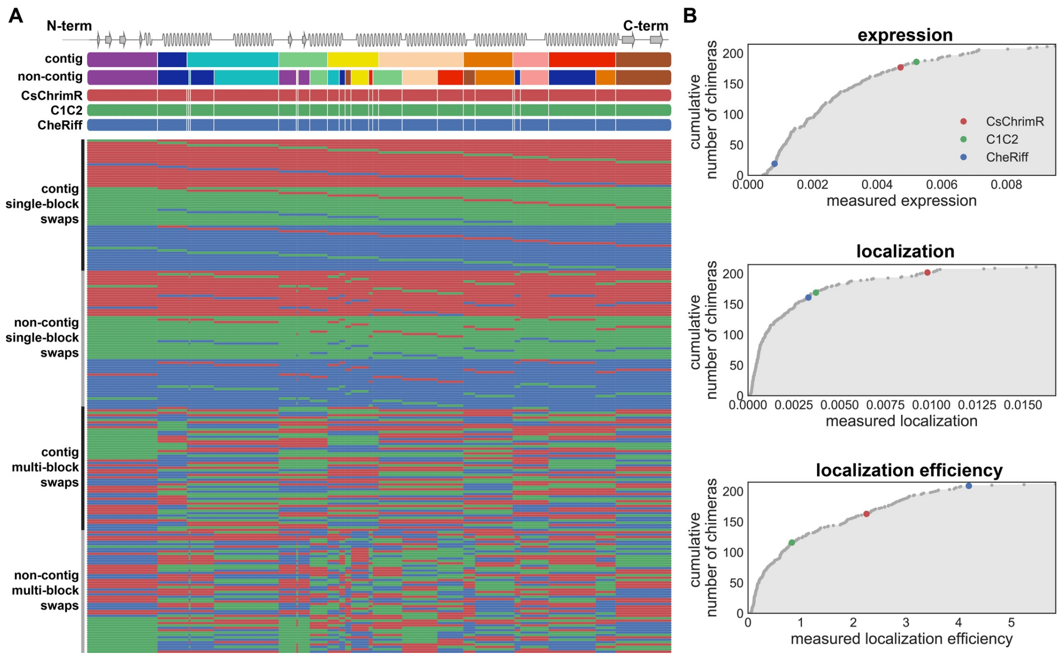


Figure 2.S1: Chimera sequences in training set and their expression, localization, and localization efficiencies. (A) (top) shows blocks (different colors) for the contiguous (contig) and non-contiguous (non-contig) library designs and also shows block boundaries (white lines) for the combined contiguous and non-contiguous library designs on the three parental ChRs aligned with a schematic of the ChR secondary structure. (bottom) Sequences of training set chimeras showing block identities. The colors represent the parental origin of the block (red–CsChrimR, green–C1C2, and blue–CheRiff). (B) Cumulative distributions of the measured expression, localization, and localization efficiency of all 218 chimeras with the three parental constructs highlighted in color (5).

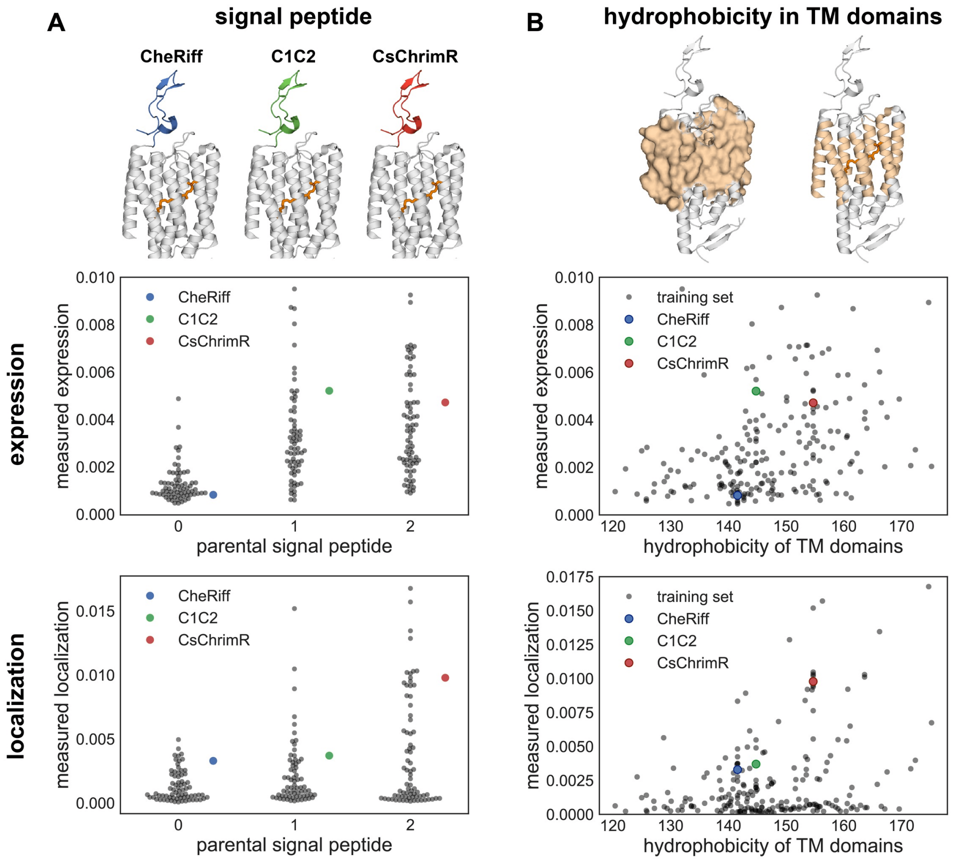


Figure 2.S2: Chimera expression and localization cannot be predicted from simple rules. Expression and localization measurements are plotted with chimeras grouped based on (A) signal peptide sequence identity and (B) hydrophobicity in the transmembrane (TM) domains. (A) Each chimera in the training set is grouped based on its signal peptide identity, which could be the CheRiff (0), C1C2 (1), or CsChrimR (2) signal peptide. The measured expression and localization are shown for each chimera in each of the three groups. (B) The measured expression and localization with respect to the calculated level of hydrophobicity within the 7-TM domains of each chimera. Hydrophobicity was calculated in the region of the protein highlighted in the surface rendering on the ChR structure.

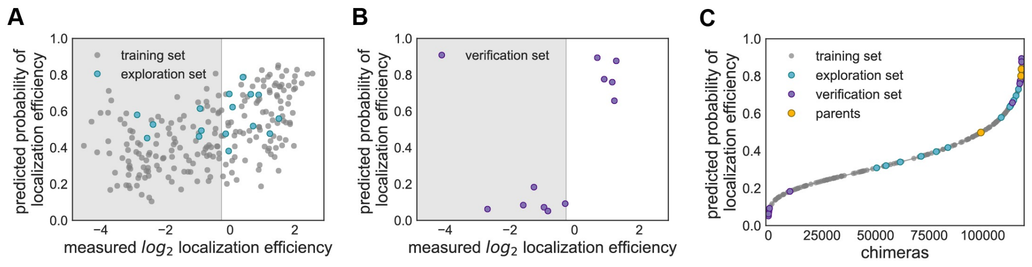


Figure 2.S3: GP binary classification model for localization efficiency. Plots of predicted probability vs measured localization efficiency are divided into ‘high’ performers (white background) and ‘low’ performers (gray background) for localization efficiency. (A) Predicted probability vs measured localization efficiency for the training set (gray points) and the exploration set (cyan points). Predictions for the training and exploration sets were made using LOO cross-validation. (B) Predicted probabilities vs measured localization efficiency for the verification set. Predictions for the verification set were made by a model trained on the training and exploration sets. (C) Probability of ‘high’ localization efficiency for all chimeras in the recombination library (118,098 chimeras) made by a model trained on the data from the training and exploration sets. The gray line shows all chimeras in the library, the gray points indicate the training set, the cyan points indicate the exploration set, the purple points indicate the verification set, and the yellow points indicate the parents. For all plots, the measured localization efficiency is plotted on a $\log_2$ scale.

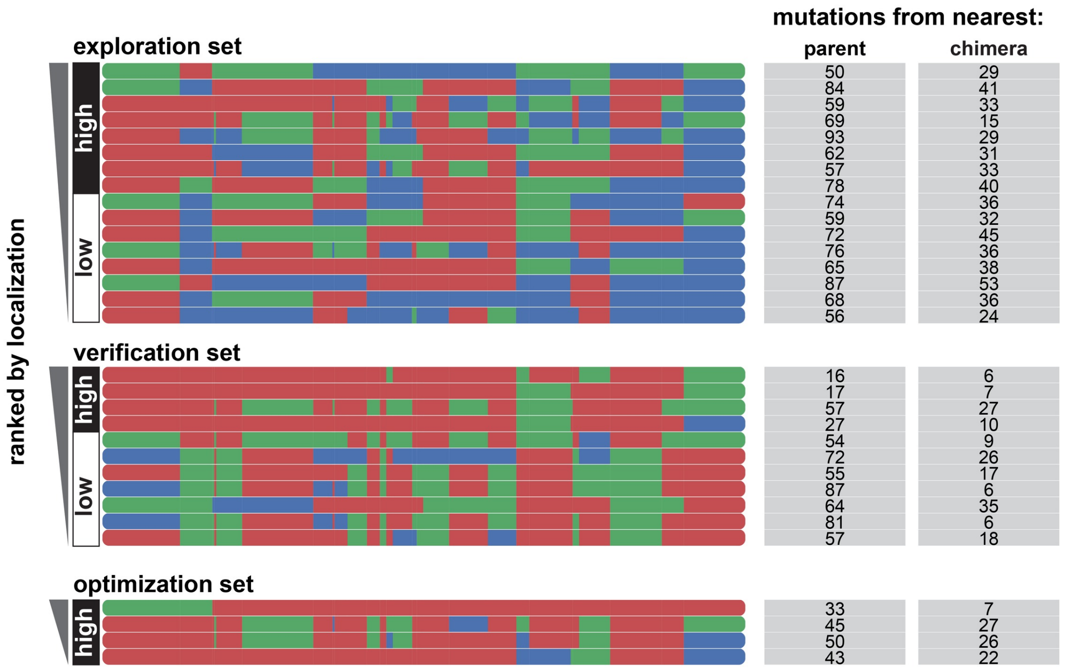


Figure 2.S4: Chimera block identities for exploration, verification, and optimization sets. Block identity of chimeras from each set ranked according to their performance for localization with the best ranking chimera listed at the top of the list. 'High' and 'low' indicates those chimeras had a high predicted probability of localization vs a low predicted probability of localization. Each row represents a chimera. The three different colors represent blocks from the three different parents (red–CsChrimR, green–C1C2, and blue–CheRiff). The number of mutations from the nearest parent and the number of mutations from the nearest previously tested chimera from the library are shown for each chimera.

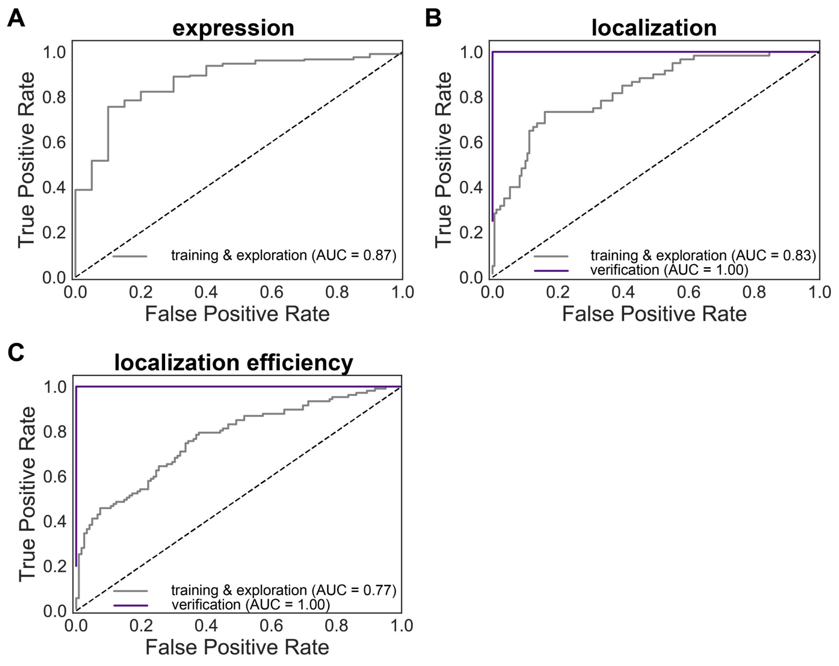


Figure 2.S5: ROC curves for GP classification expression, localization, and localization efficiency models. ROC curves show true positive rate vs false positive rate for predictions from the expression (A), localization (B), and localization efficiency (C) classification models. The gray line shows the ROC for the combined training and exploration sets. The purple line shows the ROC for the verification set. The verification sets consist exclusively of chimeras with 'high' expression so no verification ROC curve for expression is shown. Predictions for the training and exploration sets were made using LOO cross-validation, while predictions for the verification set were made by a model trained on the training and exploration sets. Calculated AUC values are shown in the figure key.

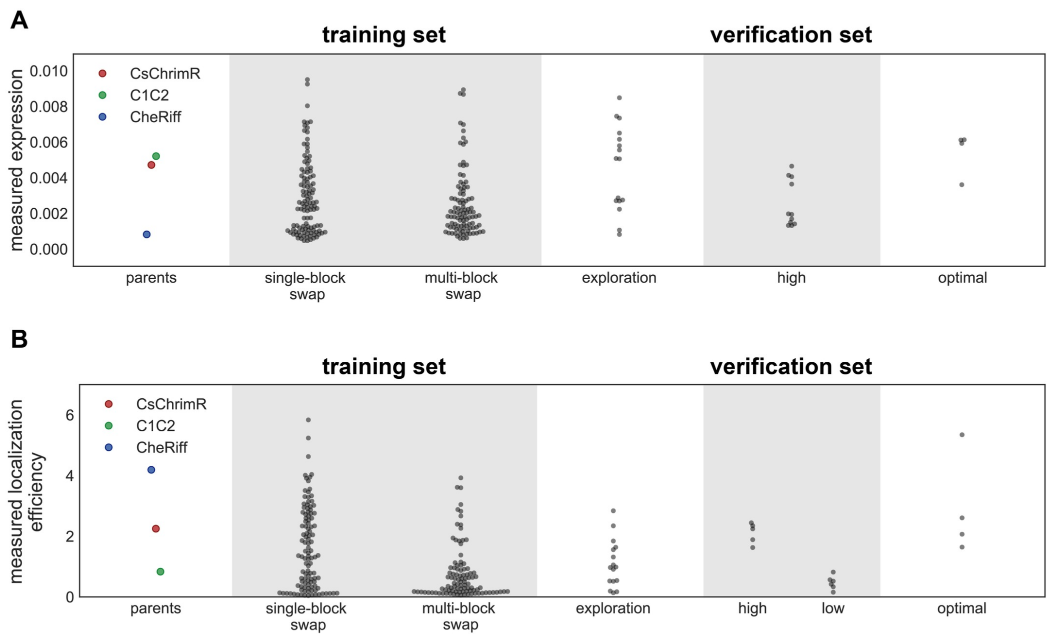


Figure 2.S6: Comparison of measured expression and localization efficiency for each data set. Swarm plots of expression (A) and localization efficiency (B) measurements for each data set compared with parents: training set, exploration set, verification set, and optimization set.

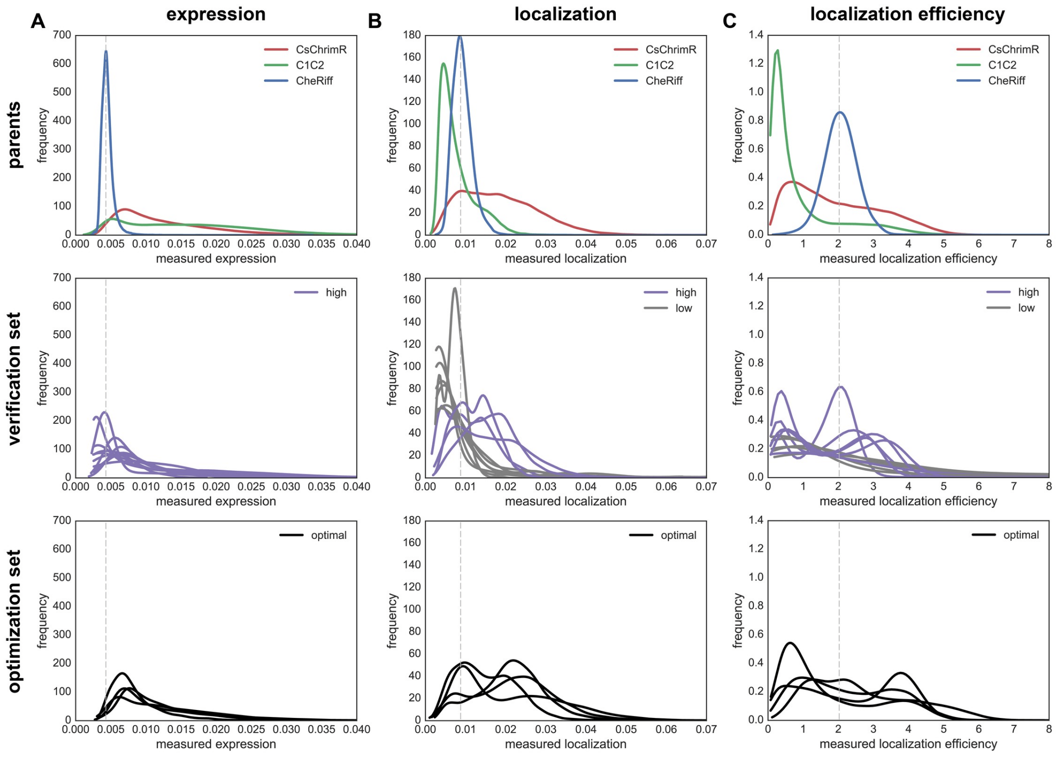


Figure 2.S7: Cell population distributions of expression, localization, and localization efficiency properties for each chimera in the verification and optimization sets compared with parents. The distribution of expression (A), localization (B), and localization efficiency (C) for the population of transfected cells is plotted for each parent (top row), each chimera in the verification set (middle row), and each chimera in the optimization set (bottom row) using kernel density estimation for smoothing. Parents are plotted in red (CsChrimR), green (C1C2), and blue (CheRiff). Chimeras in the verification set are plotted in gray if they were predicted to be 'low' or purple if they were predicted to be 'high' in each property. The vertical, gray, dashed line indicates the mean behavior of the CheRiff parent for each property.

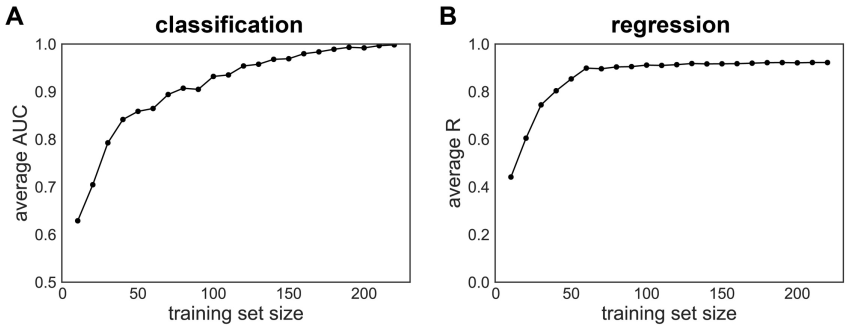


Figure 2.S8: Predictive ability of GP localization models as a function of training set size.We trained GP models on random training sets of various sizes sampled from our data and evaluated their predictive performance on a fixed test set of sequences for the classification (A) and regression (B) localization models. The predictive performance of the classification model is described by AUC for the test set (A), while the predictive performance of the regression model (B) is described by the correlation coefficient (R-value) for the test set. For each training set size, the results are averaged over 100 random samples.

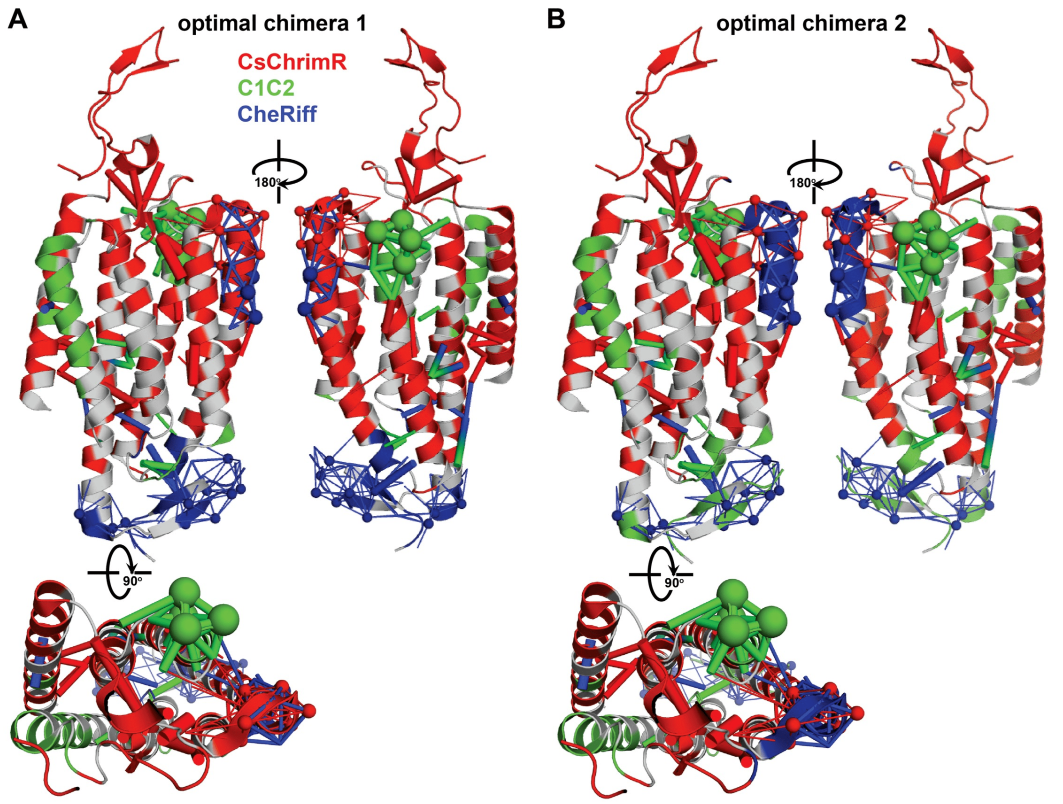


Figure 2.S9: Important features for prediction of ChR localization aligned with chimeras with optimal localization. Features with positive weights from the localization model (Figure 2.5) are displayed on the C1C2 crystal structure which is colored based on the block design of two different chimeras, (A) n1_7 and (B) n4_7, from the optimization set. Features can be residues (spheres) or contacts (sticks) from one or more parent ChRs. Features/blocks from CsChrimR are shown in red, features/blocks from C1C2 are shown in green, and features/blocks from CheRiff are shown in blue. Gray positions are conserved residues. Sticks connect the beta carbons of contacting residues (or alpha carbon in the case of glycine). The size of the spheres and the thickness of the sticks are proportional to the parameter weights.

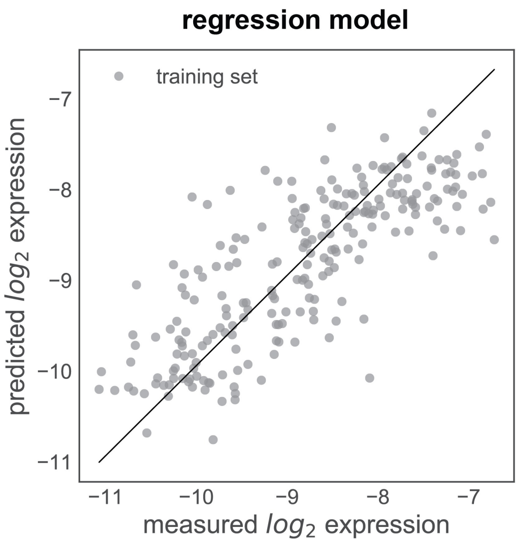


Figure 2.S10: GP regression model for ChR expression. Shows the GP regression model predicted vs measured expression for the combined training and exploration sets (gray points). Predictions for the training and exploration sets were made using LOO cross-validation. The predicted and measured expression are plotted on a $\log_2$ scale. The combined training and exploration sets showed good correlation ($R > 0.70$).

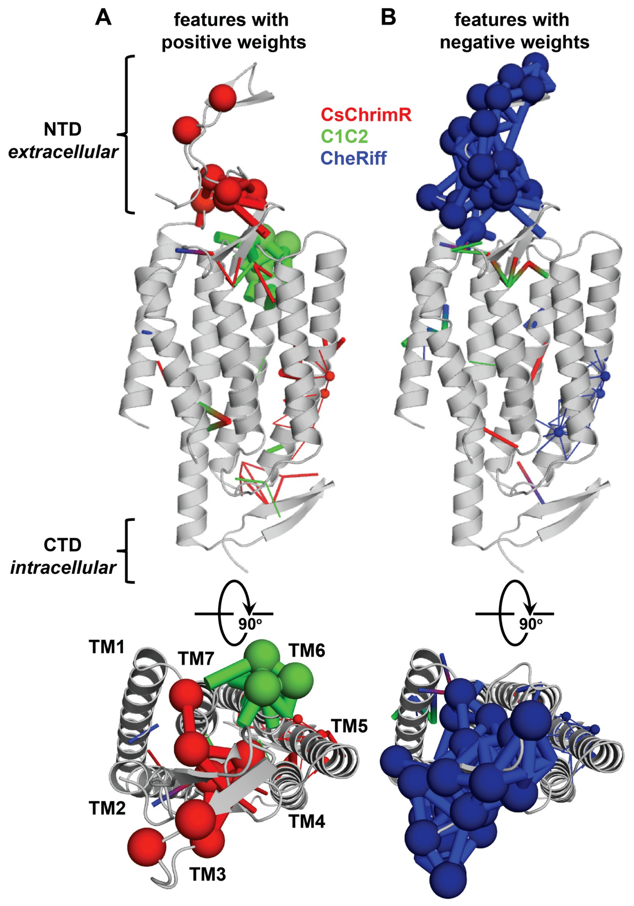


Figure 2.S11: Sequence and structure features important for prediction of ChR expression. Features with positive (A) and negative (B) weights are displayed on the C1C2 crystal structure (grey). Features can be residues (spheres) or contacts (sticks) from one or more parent ChRs. Features from CsChrimR are shown in red, features from C1C2 are shown in green, and features from CheRiff are shown in blue. In cases where a feature is present in two parents, the following color priorities were used for consistency: red above green above blue. Sticks connect the beta carbons of contacting residues (or alpha carbon in the case of glycine). The size of the spheres and the thickness of the sticks are proportional to the parameter weights. Two residues in contact can be from the same or different parents. Single-color contacts occur when both contributing residues are from the same parent. Multi-color contacts occur when residues from different parents are in contact. The N-terminal domain (NTD), C-terminal domain (CTD), and the seven transmembrane helices (TM1-7) are labeled.

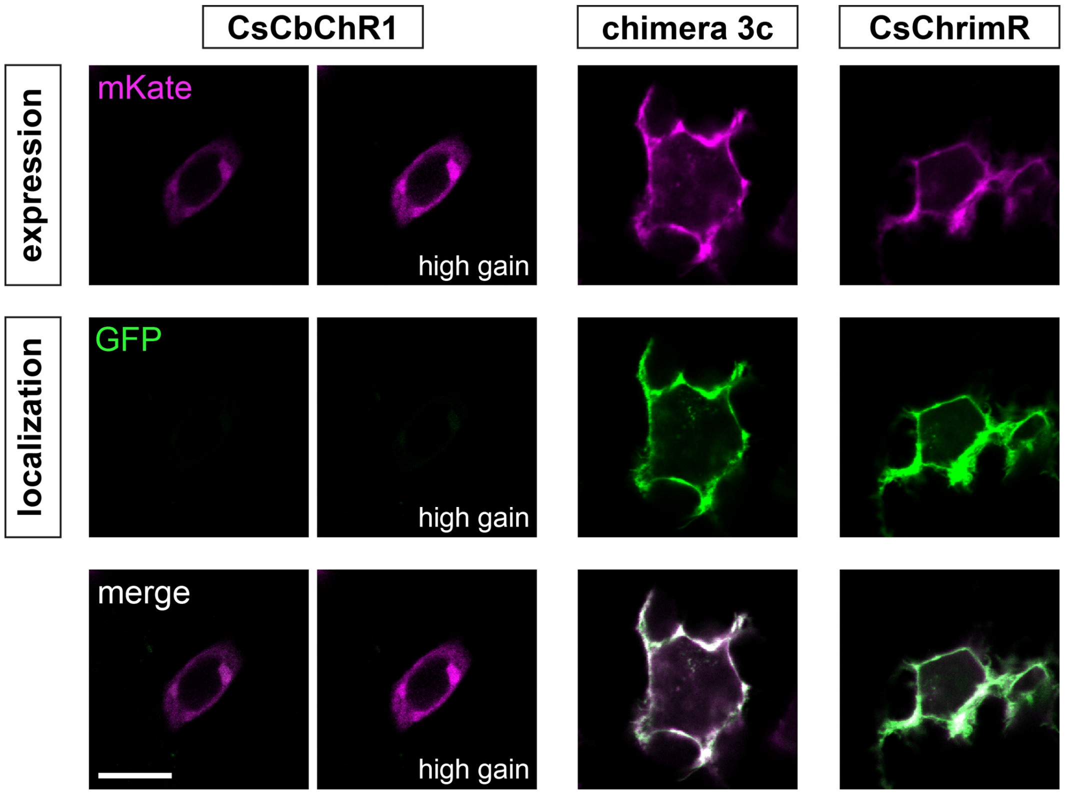


Figure 2.S12: Localization of engineered CbChR1 variant chimera 3c. Representative cell confocal images of mKate expression and GFP labeled localization of CsCbChR1 compared with top-performing CsCbChR1 single-block-swap chimera (chimera 3c), and top-performing parent (CsChrimR). CsCbChR1 shows weak expression and no localization, while chimera 3c expresses well and clearly localizes to the plasma membrane as does CsChrimR. Gain was adjusted in CsCbChR1 images to show any low signal. Scale bar: 10 μm.

Chapter 3

MACHINE LEARNING-GUIDED CHANNELRHODOPSIN ENGINEERING ENABLES MINIMALLY-INVASIVE OPTOGENETICS

1. Claire N Bedbrook, Kevin K Yang, J Elliot Robinson, Viviana Gradinaru, and Frances H Arnold: Machine learning-guided channelrhodopsin engineering enables minimally-invasive optogenetics (Submitted).

3.1 Introduction

Channelrhodopsins (ChRs) are light-gated ion channels found in photosynthetic algae. Transgenic expression of ChRs in the brain enables light-dependent neuronal activation[1]. These channels have been widely applied as tools in neuroscience research[2]; however, functional limitations of available ChRs prohibit a number of optogenetic applications. In their algal hosts, ChRs serve as sunlight sensors in phototaxic and photophobic responses[1]. Because these channels have evolved to use sunlight for functional activation, they have broad activation spectra in the visible range and require high-intensity light for activation [$\sim$1 mW mm^{-2}]. ChRs are naturally low-conductance channels requiring approximately 105 - 106 functional ChRs expressed in the plasma-membrane of a neuron to produce sufficient light-dependent depolarization to induce neuronal activation[3]. When applied to the mouse brain, ChRs require $\sim$1 - 15 mW light delivered $\sim$100 μm from the target cell population to reliably activate action potentials[4–6]. This confines light-dependent activation to a small volume of brain tissue [$\sim$1 mm^3][7]. Enabling optogenetics for large brain volumes without the need to implant invasive optical fibers for light delivery would be highly desirable for neuroscience applications.

Our goal has been to engineer available ChRs to overcome limits in conductance and light sensitivity and extend the reach of optogenetic experiments. Engineering ChRs requires overcoming three major challenges. First, rhodopsins are trans-membrane proteins that are inherently difficult to engineer because the sequence and structural determinants of membrane protein expression and plasma-membrane localization are highly constrained and poorly understood[8,9]. Second, because properties of interest for neuroscience applications are assayed using low-throughput techniques, such as patch-clamp electrophysiology, engineering by directed evolution is not feasible[10]. And third, *in vivo* applications require either retention or optimization of multiple properties in a single protein tool; for example, we must optimize expression and localization in mammalian cells while simultaneously tuning kinetics, photocurrents, and spectral properties[6].

Diverse ChRs have been published, including variants discovered from nature[11], variants engineered through recombination[9,12] and mutagenesis[13,14], as well as variants resulting from rational design[15]. Studies of these coupled with structural information[16] and molecular dynamic simulations[17] have established some understanding of the mechanics and sequence features important for specific ChR properties[1,15]. Despite this, it is still not possible to predict the functional properties of new ChR sequences and therefore not trivial to design new ChRs with a desired combination of functional properties.

Our approach has been to leverage the significant literature of ChRs to train statistical models that enable design of new, highly-functional ChRs. These models take as their input the sequence and structural information for a given ChR variant and then predict its functional properties. The models use training data to learn how sequence and structural elements map to ChR functional properties. Once known, that mapping can be used to predict the functional behavior of untested ChR variants and to select variants predicted to have optimal combinations of desired properties.

We trained models in this manner and found that they accurately predict the functional properties of untested ChR sequences. We used these models to engineer 30 'designer' ChR variants with specific combinations of desired properties. A number of variants identified from this work have unprecedented conductance and light sensitivity. We have characterized these low-light sensitive, high-conductance ChRs for applications in the mammalian brain and demonstrate their potential for minimally-invasive activation of populations of neurons in the brain enabled by systemic transgene delivery with engineered AAV, rAAV-PHP.eB[18]. This work demonstrates of the power of a machine learning-guided approach to engineering this difficult-to-engineer class of proteins.

3.2 Results

Dataset of ChR sequence variants and corresponding functional properties for machine learning

In previous work, we explored structure-guided recombination[19,20] of three highly-functional ChR parents [CsChrimsonR (CsChrimR)[11], C1C2[16], and CheRiff[21] by designing two 10-block recombination libraries with a theoretical size of ~120,000 (i.e. 2×3^{10}) ChR variants[9]. Measuring expression, localization, and photocurrent properties of a subset of these chimeric ChRs showed that these recombination libraries are a rich source of functionally diverse sequences[9]. That work produced 76 ChR variants with measured photocurrent properties, the largest single source of published ChR functional data. In subsequent work, we generated an additional 26 ChR variants selected from the same recombination libraries[8], which we have now characterized for functional properties. Together, these 102 ChR variants from the recombination libraries provide the primary dataset used for model training in this work. We supplemented this dataset with data from other published sources including 19 ChR variants from nature, 14 single-mutant ChR variants, and 28 recombination variants from other libraries (Dataset 1). As the data produced by other labs were not collected under the same experimental conditions as data collected in our hands, they cannot be used for comparison for absolute ChR properties (i.e. photocurrent strength); however, these data do provide useful binary information on whether a sequence variant is functional or not. Thus, we used published data from other sources when training binary classification models for ChR function.

Our primary interest was modeling and optimization of three ChR photocurrent properties: photocurrent strength, wavelength sensitivity, and off-kinetics (Figure 3.1a). Enhancing ChR photocurrent strength would enable reliable neuronal activation even under low-light conditions. As metrics of photocurrent strength, we use peak and steady-state photocurrent (Figure 3.1a). As a metric for the ChR activation spectrum, we use the normalized current strength induced by exposure to green light (560 nm) (Figure 3.1a). Different off-rates can be useful for specific applications: fast off-kinetics enable high-frequency optical stimulation[22], slow off-kinetics is correlated with increased light sensitivity[3,13,14], and very slow off-kinetics can be used for constant depolarization (step-function opsins [SFOs][13]). We use two parameters to characterize the off-kinetics: the time to reach 50% of the light-activated current and the photocurrent decay rate, τ_{off} (Figure 3.1a). In addition to opsin functional properties, it is also necessary to optimize or maintain plasma-membrane localization, a prerequisite for ChR function[8].

As inputs for the machine-learning models, we consider both ChR sequence and structure. ChR sequence information is simply encoded in the amino acid sequence. For structural comparisons, we convert the 3D crystal-structural information into a 'contact map' form that is convenient for modeling. Two residues are considered to

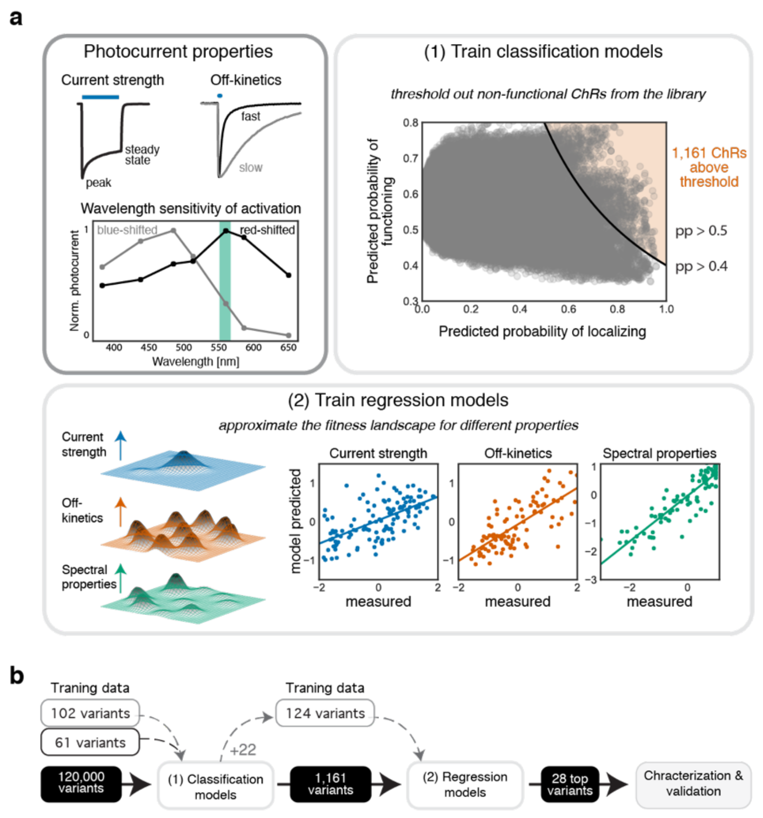

a
Photocurrent properties
Current strength
Off-kinetics
fast
steady state
slow
peak
Wavelength sensitivity of activation
blue-shifted
red-shifted
Norm. photocurrent
Wavelength [nm]
(1) Train classification models
threshold out non-functional ChRs from the library
Predicted probability of functioning
Predicted probability of localizing
1,161 ChRs above threshold
pp > 0.5
pp > 0.4
(2) Train regression models
approximate the fitness landscape for different properties
Current strength
Off-kinetics
Spectral properties
model predicted
measured
b
Traning data
102 variants
61 variants
120,000 variants
(1) Classification models
+22
Traning data
124 variants
1,161 variants
(2) Regression models
28 top variants
Chracterization & validation

Figure 3.1: Machine learning-guided optimization of ChR photocurrent strength, off-kinetics, and wavelength sensitivity of activation. (a) Upon light exposure, ChRs rapidly open and reach a peak inward current; with continuous light exposure, ChRs desensitize reaching a lower steady-state current. Both peak and steady-state current are used as metrics for photocurrent strength. To evaluate ChR off-kinetics, the current decay after a 1 ms light exposure is fit to a monoexponential decay curve and the decay rate (τ_{off}) is used as a metric for off-kinetics. We also use the time to reach 50% of the light-exposed current after light removal as a metric for off-kinetics. ChRs are maximally activated by one wavelength of light and less activated as one shifts the light further from that optimal wavelength. Most ChRs are 'blue shifted,' with their wavelength of peak activation at ~450 – 480 nm. Some ChRs are 'red shifted,' with a wavelength of peak activation between 520 – 650 nm. We use the normalized photocurrent with green (560 nm) light as a metric for wavelength sensitivity of activation. Variant selection was carried out in tiers, (1) using trained classification models to predict whether ChRs would localize correctly to the plasma membrane and function (2) using regression models to approximate the fitness landscape for each property of interest for the recombination library. Sequences from the recombination library predicted to localize and function by the classification models and predicted to have an optimized set of functional properties by the regression models were selected for further characterization. Models are trained with photocurrent properties for each ChR in the training set such that the model predicted properties correlate well with measured properties (plots show 20-fold cross validation on the training set). (b) Schematic of the trajectory of the machine learning-guided engineering of designer ChRs. The classification function model was trained with 102 variants from the recombination libraries (Dataset 2) and 61 previously-published ChRs (Dataset 1). The regression models were trained with 124 variants from the recombination libraries (Dataset 2).

be in contact and potentially important for structural and functional integrity if they have any non-hydrogen atoms within 4.5 Å in the C1C2 crystal structure[16].

Training Gaussian process (GP) classification and regression models

Using the ChR sequence/structure and functional data as inputs, we trained Gaussian process (GP) classification and regression models (Figure 3.1). GP models have successfully predicted thermostability, substrate binding affinity, and kinetics for several soluble enzymes[23], and, more recently, ChR membrane localization8. For a detailed description of the GP model architecture and properties used for protein engineering see refs [8, 23]. Briefly, these models infer predictive values for new sequences from training examples by assuming that similar inputs (ChR sequence variants) will have similar outputs (photocurrent properties). To quantify the relatedness of inputs (ChR sequence variants), we compared both sequence and structure. We defined the sequence and structural similarity between two chimeras by aligning them and counting the number of positions and contacts at which they are identical[23].

We trained a binary classification model to predict if a ChR sequence will be functional using all 102 training sequences from the recombination library as well as data from 61 sequence variants published by others (Dataset 1). A ChR sequence was considered to be functional if its photocurrents were > 0.1 nA upon light exposure, a threshold we set as an approximate lower bound for conductance necessary for neuronal activation. We then used this trained classification model to predict whether uncharacterized ChR sequence variants were functional (Figure 3.1a). To verify that the classification model is capable of accurate predictions, we performed 20-fold cross validation on the training data set and measured an area under the receiver operator curve (AUC) of 0.78, indicating good predictive power (Table 3.1).

Next, we trained three regression models, one for each of the ChR photocurrent properties of interest: photocurrent strength, wavelength sensitivity of photocurrents, and off-kinetics (Figure 3.1a). For these, we exclusively used data collected from our ChR recombination libraries (Dataset 2). Once trained, these models were used to predict photocurrent strength, wavelength sensitivity of photocurrents, and off-kinetics of new, untested ChRs sequence variants. Again, to test whether these models make accurate predictions, we performed 20-fold cross validation on the training dataset and observed high correlation between predicted and measured properties as indicated by Pearson correlations between 0.65 – 0.9 for all models,

Table 3.1: Evaluation of prediction accuracy for different ChR property models. Calculated AUC or Pearson correlation after 20-fold cross validation on training set data for classification and regression models. The test set for both the classification and regression models was the 28 ChR sequences predicted to have useful combinations of diverse properties. Accuracy of model predictions on the test set is evaluated by AUC (for classification model) or Pearson correlation (for the regression models). The Matérn kernel is with $\nu = \frac{5}{2}$.

Model type	ChR property	Kernel	Cross validation	Test
GP classification	function	Matérn	AUC = 0.78	AUC = 1.0
GP regression	current strength	Matérn	$R = 0.65$	AUC = 0.92
GP regression	off-kinetics	Matérn	$R = 0.75$	AUC = 0.97
GP regression	wavelength sensitivity	Matérn	$R = 0.90$	AUC = 0.96

as shown in Table 3.1.

Selection of designer ChRs using trained models

A ‘designer’ ChR is a ChR predicted by our models to have a useful combinations of properties. We used a tiered approach (Figure 3.1b) to select designer ChRs. The first step was to eliminate all ChR sequences predicted to not localize to the plasma membrane or predicted to be non-functional. To do this, we used the ChR function classification model along with our previously published ChR localization classification model[8] to predict the probability of localization and function for each ChR sequence in the 120,000-variant recombination library. Not surprisingly, most ChR variants were predicted to not localize and not function. To focus on ChR variants predicted to localize and function, we set a threshold for the product of the predicted probabilities of localization and function; any ChR sequence above that threshold would be considered for the next tier of the process (Figure 3.1a). We selected a conservative threshold of 0.4 (Figure 3.1a). Only 1,161 sequence variants passed the 0.4 threshold (Figure 3.1).

The model training data made clear that the higher the mutation level (mutation distance from one of the three parent proteins), the less likely it was that a sequence would be functional; however, we expect that the more diverse sequences would also have the most diverse functional properties. We wanted to explore diverse sequences predicted to function by the classification models. We selected 22 ChR variants that passed the 0.4 threshold and were diverse multi-block-swap sequences (i.e. containing on average 70 mutations from the closest parent). After these 22 sequences were synthesized, cloned in the expression vector, and expressed in HEK

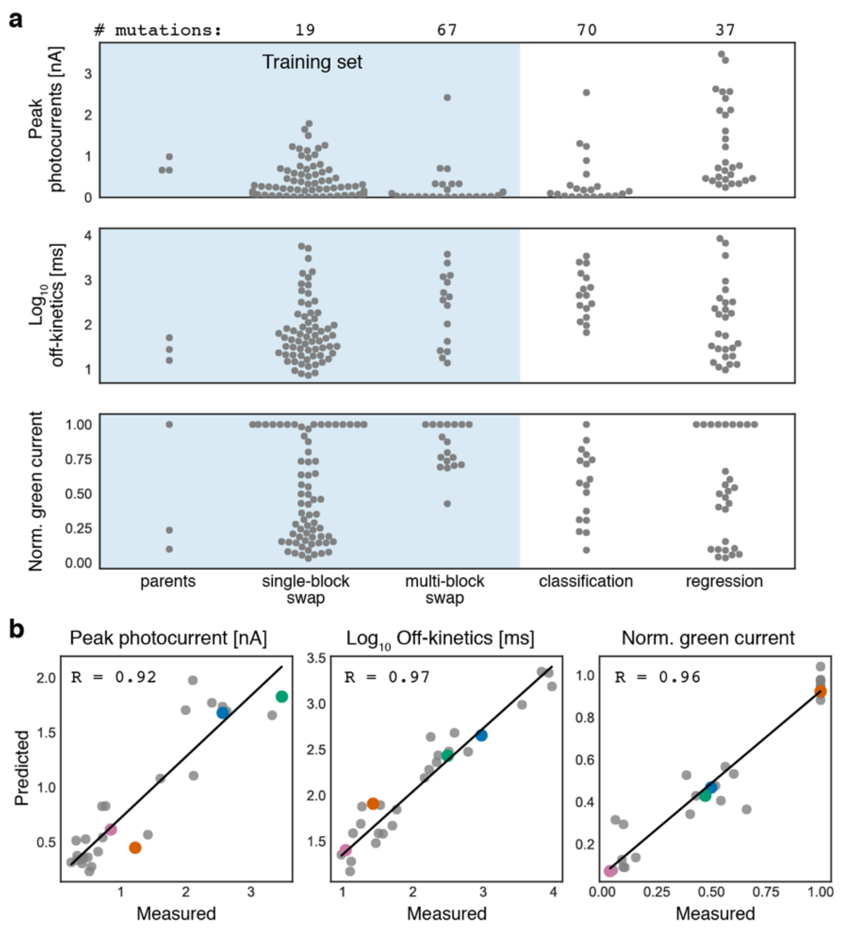


Figure 3.2: Training machine-learning models to predict ChR properties of interest based on sequence and structure enables design of ChR variants with collections of desirable properties. (a) Measurements of training set ChR and model-predicted ChR, peak photocurrent, off-kinetics, and normalized green current. Each gray-colored point is a ChR variant. Training set data are shaded in blue. Mean number of mutations for each set is above the plots. (b) Model predictions vs measured property for peak photocurrent, off-kinetics, and normalized green current of the 28 designer ChRs shows strong correlation. Specific ChR variants are highlighted to show predicted and measured properties for all three models: blue, ChR_12_10, green, ChR_11_10, orange, ChR_28_10, pink, ChR_5_10.

cells, their photocurrent properties were measured with patch-clamp electrophysiology. 59% of the tested sequences were functional (Figure 3.2a), compared to 38% of the multi-block swap sequences not selected by the model and having the same average mutation level. This validates the classification model's ability to make useful predictions about novel functional sequences, even for sequences that are very distant from those previously tested.

For the second tier of the selection process, we used the three regression models trained on all functional variants collected up to this point to predict the photocurrent strength, wavelength sensitivity of photocurrents, and off-kinetics for each of the remaining 1,161 ChR sequence variants above the predicted localization and function thresholds. From these predictions, we selected ChR sequence variants predicted to have the highest photocurrent strength, most red-shifted or blue-shifted activation wavelengths, and variants with a range of off-kinetics from very fast to very slow. We selected 28 designer ChRs with different combinations of properties that were all predicted to be highly-functional (photocurrents > 0.2 nA) and capable of good membrane localization (Figure 3.S1).

Genes encoding the 28 selected designer ChR variants were synthesized and cloned into expression vectors, expressed in HEK cells, and characterized for their photocurrent properties with patch-clamp electrophysiology. For each of the designer ChR variants, the three measured photocurrent properties correlated very well with the model predictions ($R > 0.9$ for all models) (Figure 3.2b, Table 3.1). This outstanding performance on a novel set of sequences demonstrates the power of this data-driven predictive method for engineering designer ChRs. As a negative control, we selected two ChR variant sequences from the recombination library that the model predicted would be non-functional (ChR_29_10 and ChR_30_10). These sequences resulted from a single-block swap from two of the most highly functional ChR recombination variants tested. As predicted, these sequences were non-functional (Figure 3.3b), demonstrating how easily ChR functionality can be attenuated by incorporating even minimal diversity at certain positions.

Sequence and structural determinants of ChR functional properties

We used L1-regularized linear regression models to identify a limited set of residues and structural contacts that strongly influence ChR photocurrent strength, spectral properties, and off-kinetics. We can assess the relative importance of these sequence and structural features by weighting their contributions using L2-regularized linear regression and have included important features and their weights in Dataset 3 and Figures 3.S2-3.S3. For each functional property, we identified a set of important residues and contacts. Residues and contacts most important for tuning spectral properties are generally proximal to the retinal-binding pocket, with some exceptions (Supplemental Figure 3.3). Residues important for conductance reside between transmembrane (TM) helix 1 and 7 (Figure 3.S2). The C1C2 crystal structure shows TM helices 1, 2 and 7 form a cavity which allows water influx for the cation-translocation pathway. Interestingly, residues important for ion conductance also appear in to be important for kinetic properties (Figure 3.S2), consistent

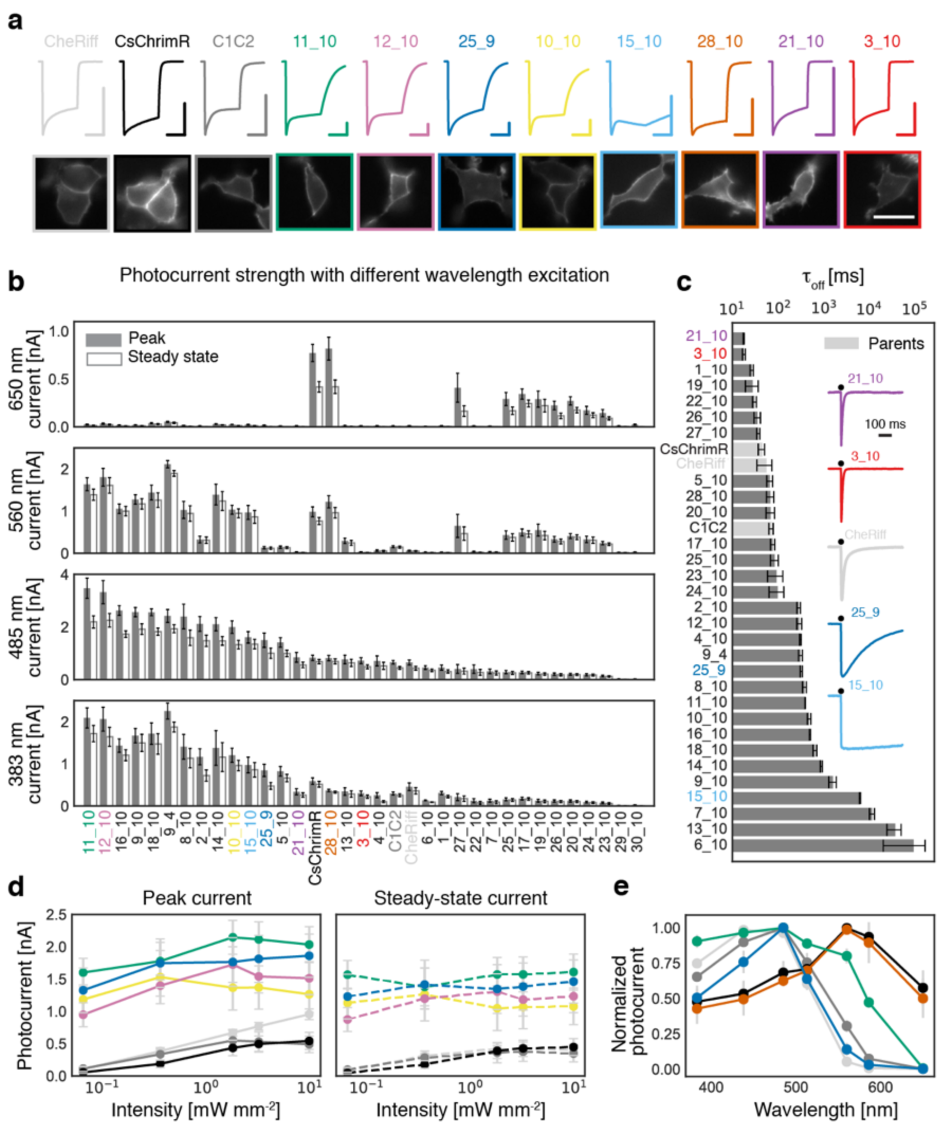


Figure 3.3: The model-predicted ChRs exhibit a large range of functional properties often far exceeding the parents. (a) Current trace after 0.5 s light exposure for select designer ChR variants with corresponding expression and localization in HEK cells. Vertical colored scale bar for each ChR current trace represents 0.5 nA, and horizontal scale bar represents 250 ms. Different color traces are labeled with each variant's name. The variant color presented in (a) is kept constant for all other panels. (b) Designer ChR measured peak and steady-state photocurrent with different wavelengths of light. 383 nm light at 1.5 mW mm^{-2}, 485 nm light at 2.3 mW mm^{-2}, 560 nm light at 2.8 mW mm^{-2}, and 650 nm light at 2.2 mW mm^{-2}. (c) Designer ChR off-kinetics decay rate (τ_{off}) following a 1 ms exposure to 485 nm light (2.3 mW mm^{-2}). Parent ChRs are highlighted in light gray. Inset shows current traces with 1 ms light exposure for select ChRs compared with CheRiff. (d) Selected ChR variants' peak and steady-state photocurrent strength with varying light irradiances compared with parental ChRs. (e) Wavelength sensitivity of activation for select ChRs compared with parental ChRs.

with previous findings that light sensitivity is inversely proportional to off-kinetic speed[3,13,14].

Machine-guided search identifies ChRs with a range of useful functional properties

We assessed photocurrent amplitude, wavelength sensitivity, and off-kinetics of the designer ChRs and the three parental ChRs [CsChrimR[11], CheRiff[21], and C1C2[16] (Figure 3.3). In addition to the 28 regression model-predicted ChRs, we also assessed the top performing ChRs from the classification models' predictions [ChR_9_4 (predicted from the classification localization model) and ChR_25_9 (classification function model)], for a total of 30 highly-functional model-predicted ChRs as well as the two negative control ChRs (ChR_29_10, ChR_30_10). Of the 30 model-predicted ChRs, we found 13 variants with significantly higher blue-light activated photocurrents than the top-performing parent (CheRiff) (Figure 3.3b). Six variants exhibit significantly higher green-light activated photocurrents than CsChrimR (Figure 3.3b). Eight variants have larger red-light activated photocurrents when compared with the blue-light activated parents (CheRiff and C1C2), though none significantly out-perform CsChrimR (Figure 3.3b). Both ChR variants predicted to be non-functional by the models produce < 0.03 nA currents.

Characterization of the 30 designer ChRs revealed that their off-kinetics span 4 orders of magnitude (τ_{off} = 10 ms – 1 min) (Figure 3.3c). This range is quite remarkable given that all designer ChRs are built from sequence blocks of three parents that have very similar off-kinetics (τ_{off} = 30 – 50 ms). We found that 5 designer ChRs have significantly faster off-kinetics than the fastest parent, while 16 are significantly slower (Figure 3.3c). Four ChRs have particularly slow off-kinetics with $\tau_{\text{off}} > 1$ s. Short 1 ms-exposures to blue light elicits distinct profiles from selected ChRs: ChR_21_10 turns off rapidly, ChR_25_9 and ChR_11_10 turn off more slowly, and ChR_15_10 exhibits little decrease in photocurrent 0.5 s after the light was turned off (Figure 3.3c). Three designer ChRs exhibit interesting spectral properties. ChR_28_10's red-shifted spectrum matches that of CsChrimR, demonstrating that incorporating sequence elements from blue-shifted ChRs into CsChrimR can still generate a red-shifted activation spectrum (Figure 3.3e). Two of the designer ChRs exhibit novel spectral properties: ChR_11_10 has a broad activation spectrum relative to the parental spectra, with similar steady-state current strength from 400 – 560 nm light and even maintain strong currents (0.7 ± 0.1 nA) when activated with 586 nm light (Figure 3.3e). ChR_25_9, on the other hand, exhibits a narrow activation spectrum relative to the parental spectra, with a peak at 485 nm light (Figure 3.3e).

We assessed the light sensitivity of the designer ChRs with enhanced photocurrents by measuring photocurrent strength at various irradiances (Figure 3.3d). We refer to these high-photocurrent ChRs as ‘high-conductance’ ChRs. All high-conductance ChRs have $\geq$ 9-times larger currents at the lowest intensity of light tested (10^{-1} mW mm^{-2}) as well as larger currents at all intensities of light tested. The high-conductance ChRs also demonstrate minimal decrease in photocurrent over the range of intensities tested (10^{-1} – 10^{1} mW mm^{-2}), suggesting that photocurrents were saturated at these intensities and would only attenuate at much lower light intensities (Figure 3.3d). The high-conductance ChRs are expressed at levels similar to the CsChrimR parent (the highest expressing parent) (Figure 3.S4).

We also compared high-conductance designer ChRs with ChR2(H134R)[6,24], an enhanced photocurrent single mutant of ChR2 commonly used for *in vivo* optogenetics, and CoChR (from *Chloromonas oogama*)[11], which was reported to be one of the highest conducting ChRs with blue light. Three of the top high-conductance ChRs (ChR_9_4, ChR_25_9, and ChR_11_10) show significantly larger peak and steady-state currents compared with ChR2 and significantly larger steady-state currents when compared with CoChR when exposed to 2 mW mm^{-2} 485 nm light (Figure 3.S5). Although CoChR produced peak currents of similar magnitude to the high-conductance ChRs, rapid decay to a much lower steady-state level was observed for this opsin (Figure 3.S5). At lower light intensities (6.5×10^{-2} mW mm^{-2}), the high-conductance ChRs show significantly larger peak and steady-state photocurrents than both ChR2(H134R) and CoChR (Figure 3.S5). These opsins have the potential for optogenetic activation with very low light levels.

Validation of designer ChRs for neuroscience applications

For further validation in neurons we selected three of the top high-conductance ChRs, ChR_9_4, ChR_25_9, and ChR_11_10, and renamed them hi-ChR1, hi-ChR2, and hi-ChR3, respectively (Figure 3.S6). For validation in cultured neurons and acute brain slices, the hi-ChRs and ChR2(H134R) were cloned into AAV vectors with a hSyn promoter, Golgi export trafficking signal (TS) sequence[5], and enhanced fluorescent protein (eYFP) marker and packaged in the engineered rAAV-PHP.eB capsid[18] (Figure 3.4a and Table 3.S1). When expressed in cultured neurons, the hi-ChRs display robust membrane localization and expression throughout the

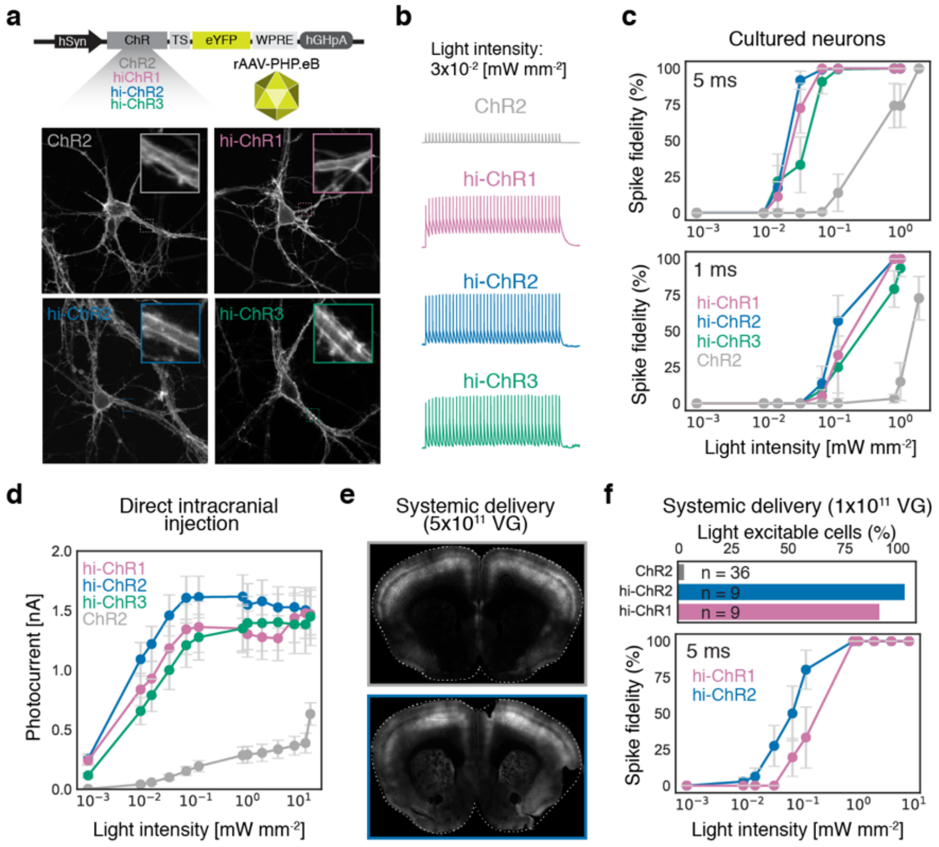


Figure 3.4: High-conductance ChR variants in cultured neurons (a – c) and in acute brain slices (d –f) outperform the commonly used ChR2(H134R). (a) High-conductance ChRs and the ChR2(H134R) control were cloned into an AAV vector with a trafficking signal (TS), eYFP, and WPRE and then packaged into rAAV-PHP.eB for expression in culture and *in vivo*. Cultured neurons expressing hi-ChR1, hi-ChR2, hi-ChR3, and ChR2(H134R). (b) Voltage traces of hi-ChR1, hi-ChR2, hi-ChR3, and ChR2(H134R) at 2 Hz with 5 ms pulsed low-intensity blue light stimulation (3×10^{-2} mW mm^{-2}) shows robust neuronal firing for the high-conductance ChRs while ChR2(H134R) exhibits only sub-threshold light-induced depolarization. (c) Spike fidelity with varying intensity light of high-conductance ChRs and ChR2(H134R) for 5 ms and 1 ms light pulses with 2 Hz stimulation. (d) High-conductance ChR and ChR2(H134R) photocurrent strength with varying light irradiances in acute brain slice after direct injection of rAAV-PHP.eB packaged ChR constructs into the PFC. (e) Systemic delivery of rAAV-PHP.eB packaged hi-ChR2 or ChR2(H134R) (5×10^{11} vg/animal) results in broad expression throughout the cortex. (f) The fraction of light excitable neurons in the PFC after systemic delivery of rAAV-PHP.eB packaged hi-ChR2 or ChR2(H134R) (1×10^{11} vg/animal) measured by cell-attached recording in acute slice targeting only neurons expressing the eYFP marker. Spike fidelity with varying intensity light of high-conductance ChRs after systemic delivery (1×10^{11} vg/animal). vg, viral genomes.

neuron soma and neurites (Figure 3.4a). We assessed neuronal spike fidelity with varying irradiance using ChR2(H134R) for comparison and observed a 10 – 100-fold decrease in the light intensity required to induce reliable spiking by 1 and 5 ms 485 nm light pulses (Figure 3.4b,c). These results demonstrate that the designer hi-ChRs require 1 – 2 orders of magnitude lower light intensity than ChR2(H134R) for neuronal activation. The hi-ChRs permit robust optically-induced firing at rates between 2 – 20 Hz, although spike fidelity was reduced at higher frequency stimulation.

Next, we performed direct intracranial injections into the mouse prefrontal cortex (PFC) of rAAV-PHP.eB packaging either hi-ChR1, hi-ChR2, hi-ChR3, or ChR2(H134R). After 3 – 5 weeks of expression, we measured light sensitivity in ChR-expressing neurons in acute brain slices. Consistent with the pervious experiments, we observe a large increase in the light sensitivity for the hi-ChRs compared with ChR2(H134R) (Figure 3.4d). All high-conductance opsins tested exhibit $>$ 200 pA photocurrent at the lowest irradiance tested, 10^{-3} mW mm^{-2}, while at the equivalent irradiance ChR2(H134R) exhibits undetectable photocurrents (Figure 3.4d). The hi-ChRs reach $>$ 1 nA photocurrents with $\sim 10^{-2}$ mW mm^{-2} light, a four-fold improvement over ChR2(H134R)'s irradiance-matched photocurrents (Figure 3.4d). Our characterization of ChR2(H134R)'s light sensitivity and photocurrent strength is consistent with previously published results from other labs[6,21].

Designer ChRs and systemic AAVs enable minimally-invasive optogenetic excitation We investigated whether these light-sensitive, high-conductance ChRs could provide optogenetic activation coupled with minimally-invasive gene delivery. Recently, we described the novel AAV capsid rAAV-PHP.eB18 that produces broad transduction throughout the central nervous system with a single minimally-invasive intravenous injection in the adult mouse[25]. Systemic delivery of rAAV-PHP.eB vectors results in brain-wide transgene delivery with expression throughout large brain volumes without the need for invasive intracranial injections[18,25]. The use of rAAV-PHP.eB for optogenetic applications has been limited, however, by the lower multiplicity of infection with systemically delivered viral vectors than with direct injection. This results in insufficient opsin expression and light-evoked currents to evoke neuronal firing with lower-conductance channels (e.g. ChR2). We hypothesized that the high-conductance ChRs could overcome this limitation and allow large-volume optogenetic excitation following systemic transgene delivery. We systemically delivered rAAV-PHP.eB packaging either hi-ChR1-TS-eYFP, hi-

ChR2-TS-eYFP, or ChR2(H134R)-TS-eYFP under the hSyn promoter and observed broad expression throughout the brain with expression strongest in the cortex (Figure 3.4e). We then measured the fraction of opsin-expressing cells with sufficient opsin-mediated currents for light-induced firing (Figure 3.4f). Only 1/36 neurons expressing ChR2(H134R) produced light-induced firing, while 8/9 neurons expressing hi-ChR1 produced light-induced activity and 9/9 neurons expressing hi-ChR2 produced light-induced activity. We also observed high spike fidelity with low light levels in hi-ChR1 and hi-ChR2, consistent with observations in neuronal cultures (Figure 3.4f). These results demonstrate the need for high-conductance opsins for applications where systemic delivery is desired.

We next evaluated the optogenetic efficiency of the high-conductance opsins after systemic delivery using a well-established behavioral paradigm: optogenetic intracranial self-stimulation (oICSS) of dopaminergic neurons of the ventral tegmental area (VTA)[26]. We used systemic delivery of rAAV-PHP.eB packaging a double-floxed inverted open reading frame (DIO) containing either hi-ChR2-TS-eYFP or ChR2(H134R)-TS-eYFP into *Dat*-Cre mice (Figure 3.5a and Table 3.S1). Three weeks after systemic viral delivery and stereotaxic implantation of fiber-optic cannulas above the VTA, mice were placed in an operant box and were conditioned to trigger a burst of 447 nm laser stimulation via nose poke. Animals expressing hi-ChR2 displayed robust optogenetic self-stimulation in a frequency-dependent and laser power-dependent manner. Higher frequencies (up to 20 Hz) and higher light power (up to 10 mW) promoting greater maximum operant response rate (Figure 3.5a). Conversely, laser stimulation failed to reinforce operant responding in ChR2(H134R)-expressing animals (Figure 3.5a); these results were consistent with results in acute slice where the light-induced currents of ChR2(H134R) are too weak at the low copy number produced by systemic delivery for robust neuronal activation.

In order to determine if hi-ChR2 would enable both minimally-invasive transgene delivery and minimally-invasive optical excitation, we assayed directional control of locomotion in freely moving animals by optogenetic stimulation of the right secondary motor cortex (M2), a well-established behavioral paradigm previously used to validate optogenetic tools[27]. In this assay, unilateral stimulation of M2 disrupts motor function in the contralateral lower extremities, causing mice to turn away from the stimulation side. We systemically administered rAAV-PHP.eB packaging either hi-ChR2-TS-eYFP or ChR2(H134R)-TS-eYFP under a CaMKIIa promoter for transgene expression in excitatory pyramidal neurons in the cortex (Figure 3.5b and Table 3.S1). We secured a fiber-optic cannula guide to the surface of the thinned skull above M2 without puncturing the dura and therefore leaving the brain intact (Figure 3.5b). Despite the presence of the highly optically scattering calavarial bone, stimulation with 20 mW 447 nm light induced left-turning behavior in animals expressing hi-ChR2 but not in animals expressing ChR2(H134R) (Figure 3.5b and Supplemental Video 1 – 2). Left-turning behavior terminated upon conclusion of optical stimulation (Supplemental Video 1). Behavioral effects were seen at powers as low as 10 mW, but the most consistent turning phenotypes were seen with 20 mW laser power. In order to ensure that turning behavior was not due to unexpected visual stimuli or heating caused by the stimulation laser, we repeated treadmill experiments using 671 nm light, which is outside the excitation spectrum of both opsins. 20 mW 671 nm light failed to induce turning in both hiChR2 and ChR2(H124R). Overall, these experiments demonstrate that hi-ChR2 in compatible with minimally-invasive systemic gene delivery and can enable minimally-invasive optogenetic excitation.

3.3 Discussion

We have outlined and demonstrated a data-driven approach to engineering ChR properties that enables efficient discovery of highly functional ChR variants based on data from relatively few variants. In this approach we approximate the ChR fitness landscape and use it to efficiently search sequence space and select top-performing variants for a given property[10,23]. By first eliminating the vast majority of non-functional sequences, we can focus on local peaks scattered throughout the landscape. Then, using regression models, we predict which sequences lie on the fitness peaks.

Designing useful ChRs for *in vivo* applications requires simultaneous optimization of multiple properties; machine learning provides a platform for such optimization and allows us to identify designer variants with combinations of properties that follow engineering specifications. Using a limited sequence space of ~120,000 chimeric ChRs, we were able to generate variants with large variations in off-kinetics (10 ms to 1 min) and photocurrents that far exceed any of the parental or other commonly used ChRs. We also use the machine-learning models to identify the residues and contacts most important for ChR function. We have designed high-conductance ChRs (hi-ChR1, hi-ChR2, and hi-ChR3) with unprecedented light sensitivity and have validated hi-ChR2's application for *in vivo* optogenetics. The

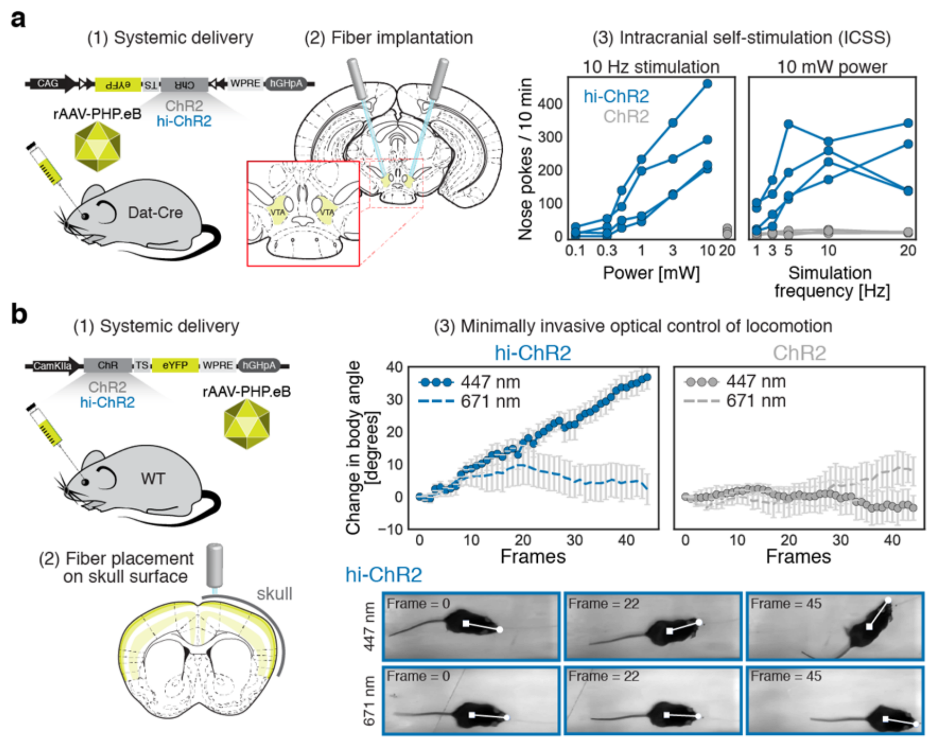


Figure 3.5: Validation of high conductance hi-ChR2 for minimally-invasive optogenetic behavioral modulation. (a) Minimally-invasive, systemic delivery of rAAV-PHP.eB packaged CAG-DIO hi-ChR2-TS-eYFP or ChR2(H134R)-TS-eYFP (3×10^{11} vg/mouse) into *Dat*-Cre animals coupled with fiber optic implantation above the VTA enabled blue light-induced intracranial self-stimulation (ten 5 ms laser pulses) exclusively with hi-ChR2 and not ChR2(H134R) with varying light power and varying stimulation frequencies. hi-ChR2, $n = 4$; ChR2(H134R), $n = 4$. (b) Minimally-invasive, systemic delivery of rAAV-PHP.eB packaged CaMKIIa hi-ChR2-TS-eYFP or ChR2(H134R)-TS-eYFP (5×10^{11} vg/mouse) into wild type (WT) animals coupled with surgically secured 2 mm long, 400 μm fiber-optic cannula guide to the surface of the skull above the right M2 that had been thinned to create a level surface for the fiber-skull interface. Three weeks later, mice were trained to walk on a linear-track treadmill at fixed velocity. Unilateral blue light stimulation of M2 induced turning behavior exclusively with hi-ChR2 and not ChR2(H134R) (10 Hz stimulation with 5 ms 447 nm light pulses at 20 mW). hi-ChR2, $n = 5$; ChR2(H134R), $n = 5$. No turning behavior was observed in any animal with 10 Hz stimulation with 5 ms 671 nm light pulses (20 mW). Error bars represent standard error of the mean. vg, viral genomes.

high-conductance properties of these ChRs have overcome limitations of low per-cell copy number after systemic delivery. hi-ChR2 enabled neuronal excitation with high temporal precision without invasive intracranial surgery for virus delivery or fiber optic implantation for superficial brain areas extending what is currently possible with optogenetics experiments.

Methods

Construct design and characterization

The design, construction, and characterization of the recombination library of chimeras is described in detail in Bedbrook *et al.*[9]. The 10-block contiguous and 10-block noncontiguous recombination libraries were designed and built using SCHEMA recombination[9]. Software packages for calculating SCHEMA energies are openly available at cheme.che.caltech.edu/groups/fha/Software.htm. Selected ChR variant genes were inserted into a constant vector backbone [pFCK from Addgene plasmid #51693[21]] with a CMV promoter, Golgi export trafficking signal (TS) sequence (KSRITSEGEYIPLDQIDINV)[5], and fluorescent protein (mKate). All ChR variants contain the SpyTag sequence following the N-terminal signal peptide for the SpyTag/SpyCatcher labeling assays used to characterize ChR membrane localization[9,28]. For characterization in neurons, selected ChR variants [hi-ChR1, hi-ChR2, hi-ChR3, CoChR11, and hChR2(H134R)] were inserted into a pAAV-hSyn vector backbone [Addgene plasmid #26973], a pAAV-CamKIIa vector backbone [Addgene plasmid #51087], and a pAAV-CAG-DIO vector backbone [Addgene plasmid #104052]. In all backbones, each ChR was inserted with a Golgi export trafficking signal (TS) sequence (KSRITSEGEYIPLDQIDINV)[5], and fluorescent protein (eYFP). ChR variant sequences used in this study are documented in Dataset 2. All selected ChR genes were synthesized and cloned in the pFCK mammalian expression vector by Twist Bioscience. HEK293T cells were transfected with purified ChR variant DNA using FuGENE®6 reagent according to the manufacturer's (Promega) recommendations. Cells were given 48 hours to express the ChRs before photocurrent measurements. Imaging of ChR variants expression in HEK cells was performed using an Andor Neo 5.5 sCMOS camera and Micro-Manager Open Source Microscopy Software. Imaging of ChR expression in neuronal cultures and in brain slices was performed using a Zeiss LSM 880 confocal microscope and Zen software.

Primary neuronal cultures

Primary hippocampal neuronal cultures were prepped from C57BL/6N mouse embryos 16-18 days post-fertilization (E16-E18 Charles-River Labs) and cultured at 37 °C in the presence of 5% CO_2 in Neurobasal media supplemented with glutamine and B27. Cells were transduced 3 - 4 days after plating with rAAV-PHP.eB packaging ChR2(H134R), hi-ChR1, hi-ChR2, or hi-ChR3. Whole-cell recordings were performed 10 - 14 days after transduction.

Patch-clamp electrophysiology

Whole-cell patch-clamp and cell-attached recordings were performed in transfected HEK cells, transduced neurons, and acute brain slices to measure light-activated inward currents or neuronal firing. For electrophysiological recordings, cultured cells were continuously perfused with extracellular solution at room temperature (in mM: 140 NaCl, 5 KCl, 10 HEPES, 2 $MgCl_2$, 2 $CaCl_2$, 10 glucose; pH 7.35) while mounted on the microscope stage. For slice recordings, 32 °C artificial cerebrospinal fluid (ACSF) was continuously perfused over slices. ACSF contained 127 mM NaCl, 2.5 mM KCl, 25 mM $NaHCO_3$, 1.25 mM NaH_2PO_4, 12 mM d-glucose, 0.4 mM sodium ascorbate, 2 mM $CaCl_2$, and 1 mM $MgCl_2$ and was bubbled continuously with 95% oxygen / 5% CO_2.

Patch pipettes were fabricated from borosilicate capillary glass tubing (1B150-4; World Precision Instruments) using a model P-2000 laser puller (Sutter Instruments) to resistances of 3–6 $M\Omega$. Pipettes were filled with K-gluconate intracellular solution containing the following (in mM): 134 K gluconate, 5 EGTA, 10 HEPES, 2 $MgCl_2$, 0.5 $CaCl_2$, 3 ATP, and 0.2 GTP. Whole-cell patch-clamp and cell-attached recordings were made using a Multiclamp 700B amplifier (Molecular Devices), a Digidata 1440 digitizer (Molecular Devices), and a PC running pClamp (version 10.4) software (Molecular Devices) to generate current injection waveforms and to record voltage and current traces. Patch-clamp recordings were done with short light pulses to measure photocurrents. Light pulse duration, wavelength, and power were varied depending on the experiment (as described in the text). Light pulses were generated using a Lumencor SPECTRAX light engine and quad band 387/485/ 559/649 nm excitation filter, quad band 410/504/582/669 nm dichroic mirror, and quad band 440/521/607/700 nm emission filter (all SEMROCK). Photocurrents were recorded from cells in voltage clamp held at -70 mV. Neuronal firing was measured in current clamp mode with current injection for a -70 mV holding potential.

Electrophysiology data were analyzed using custom data-processing scripts written using open-source packages in the Python programming language to do baseline adjustments, find the peak and steady state inward currents, perform monoexponential fits of photocurrent decay for off-kinetic properties, and quantify spike fidelity. Plotting and statistical analysis were done in Python and GraphPad Prism 7.01. For statistical comparisons, we performed one-way ANOVA (which assumes a Gaussian distribution) and corrected for multiple comparisons with statistical hypothesis testing using Dunnett's multiple comparisons *post hoc* test.

AAV production and purification

Production of recombinant AAV-PHP.eB packaging pAAV-hSyn-X-TS-eYFP-WPRE, pAAV-CAG-DIO[X-TS-eYFP]-WPRE, and pAAV-CaMKIIa-X-TS-eYFP-WPRE (X = ChR2(H134R), hi-ChR1, hi-ChR2, and hi-ChR3) was done following the method described in Deverman *et al.*[29]. Briefly, triple transfection of HEK293T cells (ATCC) was performed using polyethylenimine (PEI). Viral particles were harvested from the media and cells. Virus was then purified over iodixanol (Optiprep, Sigma; D1556) step gradients (15%, 25%, 40% and 60%). Viruses were concentrated and formulated in phosphate buffered saline (PBS). Virus titers were determined by measuring the number of DNase I–resistant viral genomes using qPCR with linearized genome plasmid as a standard.

Animals

All procedures were approved by the California Institute of Technology Institutional Animal Care and Use Committee (IACUC). *Dat*-Cre mice (006660) and C57Bl/6J mice (000664) were purchased from Jackson Laboratory.

Intravenous injections, stereotactic injections, and cannula implantation

Intravenous administration of rAAV vectors was performed by injecting the virus into the retro-orbital sinus at viral titers indicated in the text. Local expression in the prefrontal cortex (PFC) was performed by direct stereotactic injection of 1 μl of purified AAV vectors at 5×10^{12} vg ml^{-1} targeting the following coordinates: anterior-posterior (AP), -1.7; media-lateral (ML), ±0.5; and dorsal-ventral (DV), -2.2. For stimulation of the VTA, stereotaxic implantation of 300 μm outer diameter fiber-optic cannulas 200 μm above the VTA was targeted to the following coordinates: AP, -3.44 mm; ML, ±0.48 mm; DV, 4.4 mm. For stimulation of the right secondary motor cortex (M2), 2 mm long, 400 μm fiber-optic cannula guide were surgically secured to the surface of the skull above M2 (unilaterally) targeted to the following coordinates: AP, 1 mm; ML, 0.5 mm. Skull was thinned ~40 – 50% with a standard drill to create a level surface for the fiber-skull interface. Light was delivered from either a 447 nm or 671 nm laser (Changchun New Industries [CNI] Model with PSU-H-LED) via optical fiber through a fiber-guide cannula. Cannula guides were secured to the skull with Metabond (Parkel, SKU S396) and dental cement. Before each behavioral session, animals were connected to an optical fiber for optical stimulation through the previously implanted cannula guide.

Analysis of behavioral experiments was performed using the open-source MATLAB program OptiMouse[30] to track mouse nose, body, and tail position while the mouse was running on the treadmill.

Gaussian process modeling

Both the GP regression and classification modeling methods applied in this paper are based on work detailed in ref [8, 23]. For modeling, all sequences were aligned using MUltiple Sequence Comparison by Log-Expectation (MUSCLE) (https://www.ebi.ac.uk/Tools/msa/muscle/). For modeling, aligned sequences were truncated to match the length of the C1C2 sequence, eliminating N- and C-terminal fragments with poor alignment quality due to high sequence diversity (Dataset 1 and Dataset 2). Structural encodings use the C1C2 crystal structure (3UG9.pdb) and assume that ChR chimeras share the contact architecture observed in the C1C2 crystal structure. For a given ChR, the contact map is simply a list of contacting amino acids with their positions. For example, a contact between alanine at position 134 and methionine at position 1 of the amino acid sequence would be encoded by [(‘A134’), (‘M1’)]. Both sequence and structural information were one-hot encoded. Regression models for ChR properties were trained to predict the logarithm of the measured properties. All training data was normalized to have mean zero and standard deviation one.

Gaussian process regression and classification models require kernel functions that measure the similarity between protein sequences. Learning involves optimizing the form of the kernel and its hyperparameters (Table 3.S2). The Matérn kernel was found to be optimal for all ChR properties (Table 3.1).

GP regression In regression, the goal is to infer the value of an unknown function $f(x)$ at a novel point x_* given observations y at inputs X. Assuming that the observations are subject to independent and identically distributed Gaussian noise with variance σ_n^2, the posterior distribution of $f_* = f(x_*)$ for Gaussian process regression is Gaussian with mean

$$\bar{f}_* = k_*^T (K + \sigma_n^2 I)^{-1} y \tag{3.1}$$

and variance

$$v_* = k(x_*, x_*) - k_*^T (K + \sigma_n^2 I)^{-1} k_* \tag{3.2}$$

Where K is the symmetric, square covariance matrix for the training set: $K_{ij} = k(x_i, x_j)$ for x_i and x_j in the training set. k_* is the vector of covariances between the novel input and each input in the training set, and $k_{*i} = k(x_*, x_i)$. The hyperparameters in the kernel functions and the noise hyperparameter σ_n were determined by maximizing the log marginal likelihood:

$$\log p\,(y|X) = -\frac{1}{2} y^T (K + \sigma_n^2 I)^{-1} y - \frac{1}{2} \log |K + \sigma_n^2 I| - \frac{n}{2} 2\pi \tag{3.3}$$

where n is the dimensionality of the inputs. Regression was implemented using open-source packages in the SciPy ecosystem[31-33].

GP classification In binary classification, instead of continuous outputs y, the outputs are class labels $y_i \in \{+1, -1\}$, and the goal is to use the training data to make probabilistic predictions $\pi(x_*) = p(y_* = +1|x_*)$. We use Laplace's method to approximate the posterior distribution. Hyperparameters in the kernels are found by maximizing the marginal likelihood. Classification was implemented using open-source packages in the SciPy ecosystem[31–33].

GP kernels for modeling proteins Gaussian process regression and classification models require kernel functions that measure the similarity between protein sequences. A protein sequence s of length L is defined by the amino acid present at each location. This can be encoded as a binary feature vector x_{se} that indicates the presence or absence of each amino acid at each position resulting in a vector of length $20L$ (for 20 possible amino acids). Likewise, the protein's structure can be represented as a residue-residue contact map. The contact map can be encoded as a binary feature vector x_{st} that indicates the presence or absence of each possible contacting pair. We used both the sequence and structure feature vectors by concatenating them to form a sequence-structure feature vector.

We considered three types of kernel functions $k(s_i, s_j)$: polynomial kernels, squared exponential kernels, and Matérn kernels. These different forms represent possible functions for the protein's fitness landscape. The polynomial kernel is defined as:

$$k(s, s') = (\sigma_0^2 + \sigma_p^2 x^T x')^d \tag{3.4}$$

where σ_0 and σ_p are hyperparameters. We considered polynomial kernels with $d = 3$. The squared exponential kernel is defined as:

$$k(s, s') = \sigma_p^2 \exp\left(\frac{\|x - x'\|_2^2}{l^2}\right) \tag{3.5}$$

where l and σ_p are also hyperparameters and $\|\cdot\|_2$ is the L2 norm. Finally, the Matérn kernel with $\nu = \frac{5}{2}$ is defined as:

$$k(s, s') = \left(1 + \frac{\sqrt(5\|x - x'\|_2^2)}{l} + \frac{5\|x - x'\|_2^2}{3l^2}\right) \exp\left(-\frac{5\|x - x'\|_2^2}{l}\right) \tag{3.6}$$

Where l is once again a hyperparameter.

L1 regression feature identification and weighting We used L1 regression to identify residues and contacts in the ChR structure most important for each ChR functional property of interest. Using the concatenated sequence and structure binary feature vector for each of the training set ChR variants, we identified residues and contacts that covary. Each set of covarying residues and contacts was combined into a single feature. L1 linear regression was used to select the features that contribute most to each ChR functional property of interest. The level of regularization was chosen by maximizing the log marginal likelihood of the Gaussian process regression model trained on the features selected at that level of regularization. We then performed Bayesian ridge regression on the selected features using the default settings in scikit-learn[34]. Residues and contacts with the largest absolute Bayesian ridge linear regression weights were plotted onto the C1C2 structure (Figures 3.S2 - 3.S3).

References

1. Karl Deisseroth and Peter Hegemann: The form and function of channelrhodopsin. *Science* **357**(6356) (2017), eaan5544. doi: 10.1126/science.aan5544 (cit. on pp. 66, 67).

2. Ofer Yizhar, Lief E Fenno, Thomas J Davidson, Murtaza Mogri, and Karl Deisseroth: Optogenetics in neural systems. *Neuron* **71**(1) (2011), 9–34. doi: 10.1016/j.neuron.2011.06.004 (cit. on p. 66).
3. John Y Lin: A user's guide to channelrhodopsin variants: features, limitations and future developments. *Exp Physiol* **96**(1) (2011), 19–25. doi: 10.1113/expphysiol.2009.051961 (cit. on pp. 66, 68, 75).
4. Feng Zhang et al.: Optogenetic interrogation of neural circuits: technology for probing mammalian brain structures. *Nature Protocols* **5**(3) (2010), 439 (cit. on p. 66).
5. Viviana Gradinaru et al.: Molecular and cellular approaches for diversifying and extending optogenetics. *Cell* **141**(1) (2010), 154–165. doi: 10.1016/j.cell.2010.02.037 (cit. on pp. 66, 76, 82).
6. Joanna Mattis et al.: Principles for applying optogenetic tools derived from direct comparative analysis of microbial opsins. *Nat Methods* **9**(2) (2012), 159. doi: 10.1038/nmeth.1808. (cit. on pp. 66, 67, 76, 78).
7. Amy S Chuong et al.: Noninvasive optical inhibition with a red-shifted microbial rhodopsin. *Nat Neurosci* **17**(8) (2014), 1123. doi: 10.1038/nn.3752 (cit. on p. 66).
8. Claire N Bedbrook, Kevin K Yang, Austin J Rice, Viviana Gradinaru, and Frances H Arnold: Machine learning to design integral membrane channelrhodopsins for efficient eukaryotic expression and plasma membrane localization. *PLoS Comput Biol* **13**(10) (2017), e1005786. doi: 10.1371/journal.pcbi.1005786 (cit. on pp. 66, 68, 70, 71, 85).
9. Claire N Bedbrook et al.: Structure-guided SCHEMA recombination generates diverse chimeric channelrhodopsins. *Proc Natl Acad Sci USA* (2017), 201700269. doi: 10.1073/pnas.1700269114 (cit. on pp. 66–68, 82).
10. Philip A Romero and Frances H Arnold: Exploring protein fitness landscapes by directed evolution. *Nat Rev Mol Cell Biol* **10**(12) (2009), 866. doi: 10.1038/nrm2805 (cit. on pp. 67, 80).
11. Nathan C Klapoetke et al.: Independent optical excitation of distinct neural populations. *Nat Methods* **11**(3) (2014), 338. doi: 10.1038/nmeth.2836 (cit. on pp. 67, 75, 76).
12. John Y Lin, Per Magne Knutsen, Arnaud Muller, David Kleinfeld, and Roger Y Tsien: ReaChR: a red-shifted variant of channelrhodopsin enables deep transcranial optogenetic excitation. *Nat Neurosci* **16**(10) (2013), 1499. doi: 10.1038/nn.3502 (cit. on p. 67).
13. André Berndt, Ofer Yizhar, Lisa A Gunaydin, Peter Hegemann, and Karl Deisseroth: Bi-stable neural state switches. *Nat Neurosci* **12**(2) (2009), 229. doi: 10.1038/nn.2247 (cit. on pp. 67, 68, 75).
14. John Y Lin, Michael Z Lin, Paul Steinbach, and Roger Y Tsien: Characterization of engineered channelrhodopsin variants with improved properties and kinetics. *Biophys J* **96**(5) (2009), 1803–1814. doi: 10.1016/j.bpj.2008.11.034 (cit. on pp. 67, 68, 75).

15. Andre Berndt et al.: Structural foundations of optogenetics: Determinants of channelrhodopsin ion selectivity. *Proc Natl Acad Sci USA* **113**(4) (2016), 822–829. doi: 10.1073/pnas.1523341113 (cit. on p. 67).

16. Hideaki E Kato et al.: Crystal structure of the channelrhodopsin light-gated cation channel. *Nature* **482**(7385) (2012), 369. doi: 10.1038/nature10870 (cit. on pp. 67, 70, 75).

17. Jonas Wietek et al.: Conversion of channelrhodopsin into a light-gated chloride channel. *Science* **344**(6182) (2014), 409–412. doi: 10.1126/science.1249375 (cit. on p. 67).

18. Ken Y Chan et al.: Engineered AAVs for efficient noninvasive gene delivery to the central and peripheral nervous systems. *Nat Neurosci* **20**(8) (2017), 1172. doi: 10.1038/nn.4593 (cit. on pp. 67, 76, 78).

19. Matthew A Smith, Philip A Romero, Timothy Wu, Eric M Brustad, and Frances H Arnold: Chimeragenesis of distantly-related proteins by noncontiguous recombination. *Protein Sci* **22**(2) (2013), 231–238. doi: 10.1002/pro.2202 (cit. on p. 67).

20. Christopher A Voigt, Carlos Martinez, Zhen-Gang Wang, Stephen L Mayo, and Frances H Arnold: Protein building blocks preserved by recombination. *Nat Struct Biol* **9**(7) (2002), 553. doi: 10.1038/nsb805 (cit. on p. 67).

21. Daniel R Hochbaum et al.: All-optical electrophysiology in mammalian neurons using engineered microbial rhodopsins. *Nat Methods* **11**(8) (2014), 825. doi: 10.1038/nmeth.3000 (cit. on pp. 67, 75, 78, 82).

22. Lisa A Gunaydin et al.: Ultrafast optogenetic control. *Nat Neurosci* **13**(3) (2010), 387. doi: 10.1038/nn.2495 (cit. on p. 68).

23. Philip A Romero, Andreas Krause, and Frances H Arnold: Navigating the protein fitness landscape with Gaussian processes. *Proc Natl Acad Sci USA* **110**(3) (2013), E193–E201. doi: 10.1073/pnas.1215251110 (cit. on pp. 70, 80, 85).

24. Georg Nagel et al.: Light activation of channelrhodopsin-2 in excitable cells of Caenorhabditis elegans triggers rapid behavioral responses. *Curr Biol* **15**(24) (2005), 2279–2284. doi: 10.1016/j.cub.2005.11.032 (cit. on p. 76).

25. Claire N Bedbrook, Benjamin E Deverman, and Viviana Gradinaru: Viral strategies for targeting the central and peripheral nervous systems. *Annu Rev Neurosci* (2018). doi: 10.1146/annurev-neuro-080317-062048 (cit. on p. 78).

26. Vincent Pascoli, Jean Terrier, Agnès Hiver, and Christian Lüscher: Sufficiency of mesolimbic dopamine neuron stimulation for the progression to addiction. *Neuron* **88**(5) (2015), 1054–1066. doi: 10.1016/j.neuron.2015.10.017 (cit. on p. 79).

27. Viviana Gradinaru et al.: Targeting and readout strategies for fast optical neural control in vitro and in vivo. *J Neurosci* **27**(52) (2007), 14231–14238. doi: 10.1523/JNEUROSCI.3578-07.2007 (cit. on p. 79).

28. Claire N Bedbrook et al.: Genetically encoded spy peptide fusion system to detect plasma membrane-localized proteins in vivo. *Chem Bio* **22**(8) (2015), 1108–1121. doi: 10.1016/j.chembiol.2015.06.020 (cit. on p. 82).

29. Benjamin E Deverman et al.: Cre-dependent selection yields AAV variants for widespread gene transfer to the adult brain. *Nat Biotechnol* **34**(2) (2016), 204. doi: 10.1038/nbt.3440 (cit. on p. 84).

30. Yoram Ben-Shaul: OptiMouse: a comprehensive open source program for reliable detection and analysis of mouse body and nose positions. *BMC Biol* **15**(1) (2017), 41. doi: 10.1186/s12915-017-0377-3 (cit. on p. 85).

31. Stéfan van der Walt, S Chris Colbert, and Gael Varoquaux: The NumPy array: a structure for efficient numerical computation. *Computing in Science & Engineering* **13**(2) (2011), 22–30 (cit. on p. 86).

32. John D Hunter: Matplotlib: A 2D graphics environment. *Computing in science & engineering* **9**(3) (2007), 90–95 (cit. on p. 86).

33. Travis E Oliphant: Python for scientific computing. *Computing in Science & Engineering* **9**(3) (2007) (cit. on p. 86).

34. Fabian Pedregosa et al.: Scikit-learn: Machine learning in Python. *Journal of machine learning research* **12**(Oct) (2011), 2825–2830 (cit. on p. 87).

Supplemental Information

Dataset 1. ChR sequence and photocurrent data from published sources including 19 natural ChR variants, 14 point-mutant ChR variants, and 28 recombination variants from various recombination libraries. The source of the photocurrent data is included (‘Reference’). When possible, we use references with side-by-side measurements of multiple ChRs. For modeling, all sequences were aligned and truncated to match the length of the C1C2 sequence (Methods). The truncated and aligned sequences are included (‘Aligned_amino_acid_sequence’) as well as the full-length sequence (‘Amino_acid_sequence’).

Dataset 2. ChR sequences and functional properties for designed chimeric sequences from our ChR recombination libraries. Functional properties were tested in HEK cells. Measurements of peak and steady-state photocurrent (nA) with 485 nm light at 2.3 mW mm^{-2} (‘cyan peak’ & ‘cyan_ss’), 560 nm light at 2.8 mW mm^{-2} (‘green_peak’ & ‘green_ss’), and 650 nm light at 2.2 mW mm^{-2} (‘red_peak & ‘red_ss’) are included. The maximum peak (‘max_peak‘) and maximum steady-state (‘max_ss’) photocurrent (nA) obtained with any wavelength are included. Measurement of the time (ms) to reach 50% of the light-exposed photocurrent after light removal is included (‘kinetics_off’). The ratio of peak photocurrent with 560 nm light to maximum photocurrent was calculated per each cell and average for each ChR variant (‘norm_green’). Off-kinetics (‘kinetics_off’) and spectral properties (‘norm_green’) were only included for ChR variants with photocurrent strength $>$ 0.02 nA. Each ChR recombination variant has a chimera identity (‘block_ID’) beginning with either ‘c’ or ‘n’ to indicate the contiguous or non-contiguous library followed by 10 digits indicating the parent that contributes each of the 10 blocks (‘0’: CheRiff, ‘1’:C1C2, and ‘2’:CsChrimR). Each ChR variant’s number of mutations away from the nearest parent (‘m’) is included. For modeling, all sequences were aligned and truncated to match the length of the C1C2 sequence (Methods). The truncated and aligned sequences are included (‘Aligned amino acid sequence’) as well as the full-length sequence (‘Amino acid sequence’).

Dataset 3. Limited set of amino acid residues and structural contacts important for model predictions identified with L1-regularized linear regression. The relative importance (‘weight’) of these sequence and structural features is learned using Bayesian ridge regression. We found a different limited set of features for each of the three functional properties of interest (‘norm_green’, ‘off_kinetics’, and ‘peak_photocurrent’). Features are either amino acid residues (i.e. a sequence feature [‘seq’]) or contacts. The feature position is indicated with numbering according to the aligned and truncated ChR sequence. We also include the parental features at each position with numbering according the parental sequence. Highly-weighted features highlighted in color in Figures 3.S2 – 3.S3 are indicated by their corresponding color. Features not highlighted in Supplemental Figures 3.S2 – 3.S3 are listed as gray.

Supplementary Video 1. hi-ChR2-expressing mouse running on a treadmill while receiving optogenetic stimulation exhibits clear left-turning behavior [10 Hz stimulation with 5 ms 447 nm light pulse (20 mW)]. Minimally-invasive, systemic delivery of rAAV-PHP.eB packaged CaMKIIa hi-ChR2-TS-eYFP (5×10^{11} vg/mouse) into wild type (WT) animals coupled with surgically secured 2 mm long, 400 μm fiber-optic cannula guide to the surface of the skull above the right M2 that had been thinned to create a level surface for the fiber-skull interface (~40 – 50%). Video shows multiple runs with the same animal.

Supplementary Video 2. ChR2(H134R)-expressing mouse running on a treadmill while receiving optogenetic stimulation does not exhibit left-turning behavior [10 Hz stimulation with 5 ms 447 nm light pulse (20 mW)]. Minimally-invasive, systemic delivery of rAAV-PHP.eB packaged CaMKIIa ChR2(H134R)-TS-eYFP (5×10^{11} vg/mouse) into wild type (WT) animals coupled with surgically secured 2 mm long, 400 μm fiber-optic cannula guide to the surface of the skull above the right M2 that had been thinned to create a level surface for the fiber-skull interface (~40 – 50%). Video shows multiple runs with the same animal.

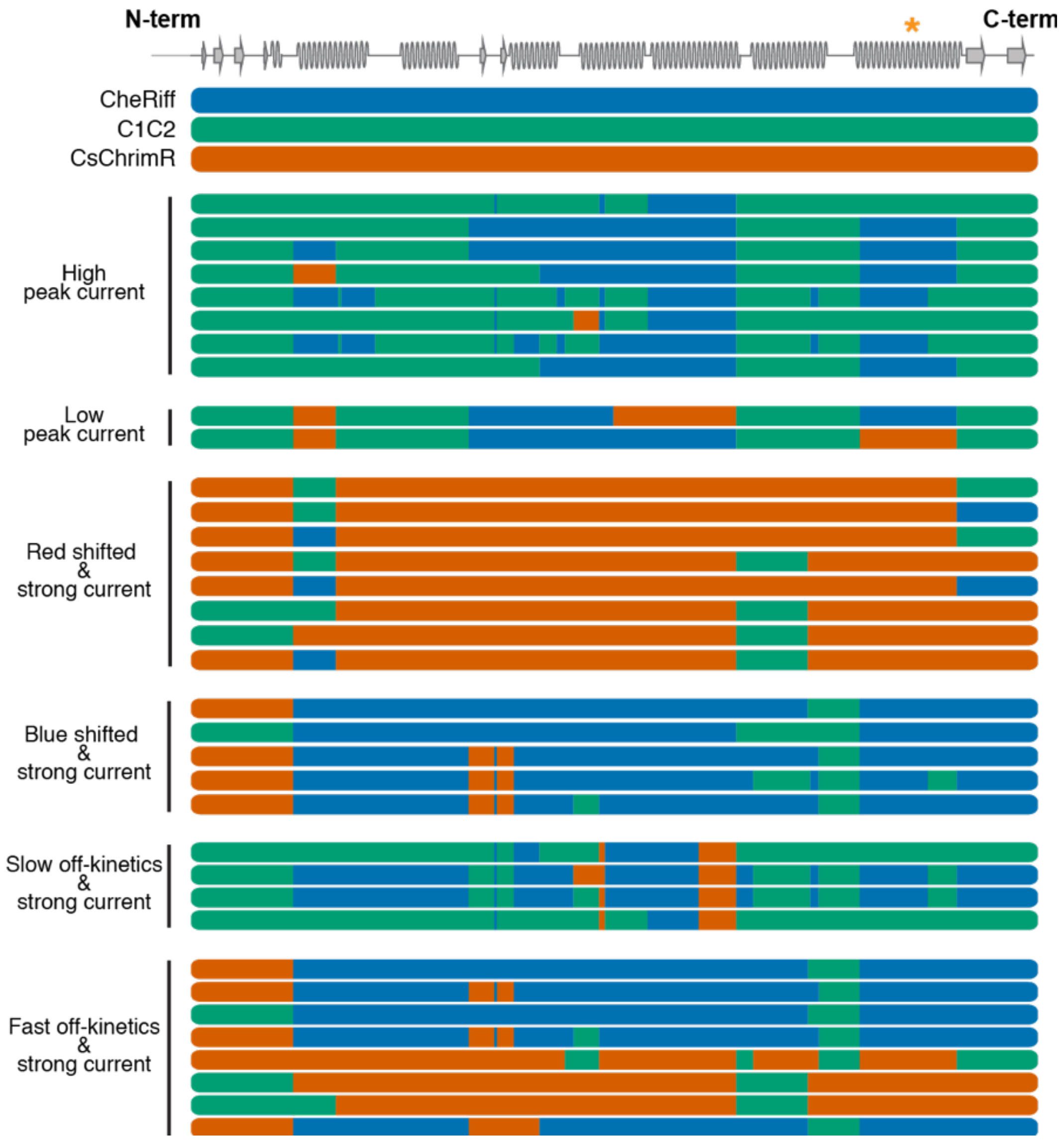

N-term
*
C-term
CheRiff
C1C2
CsChrimR
High peak current
Low peak current
Red shifted & strong current
Blue shifted & strong current
Slow off-kinetics & strong current
Fast off-kinetics & strong current

Figure 3.S1: Thirty model-predicted ChR chimeras aligned with the three parents and the secondary structure. Blocks of ChR chimeras are colored according to which parent each block came from. CsChrimR is red, CheRiff is blue, and C1C2 is green. (*) highlights the Schiff base. ChRs are divided into categories based on their predicted properties. Twenty-eight ChR chimeras are predicted to be optimized for one or more properties. Two ChR chimeras are predicted to be non-optimal and produce low currents. A number of chimeras appear twice because they were optimal for multiple categories.

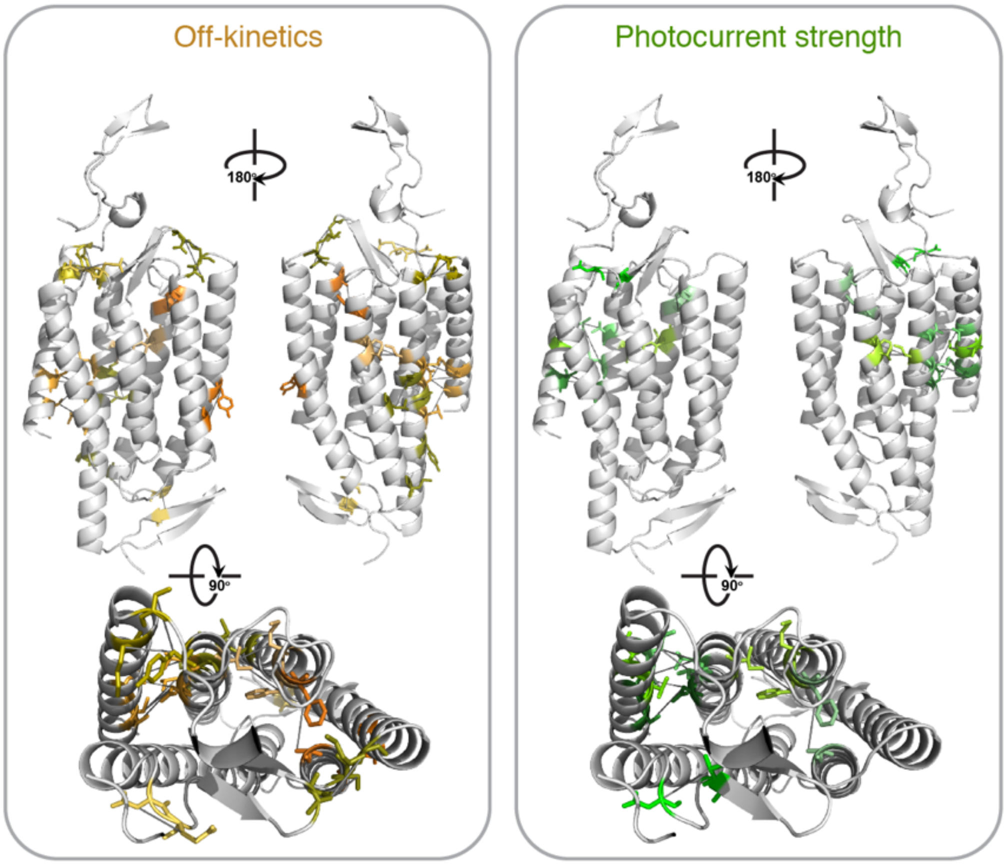


Figure 3.S2: Specific residues (amino-acid sticks) and contacts (dark gray lines) most important for model prediction of off-kinetics and photocurrent strength overlaid on the C1C2 crystal structure in light gray (3ug9.pdb).

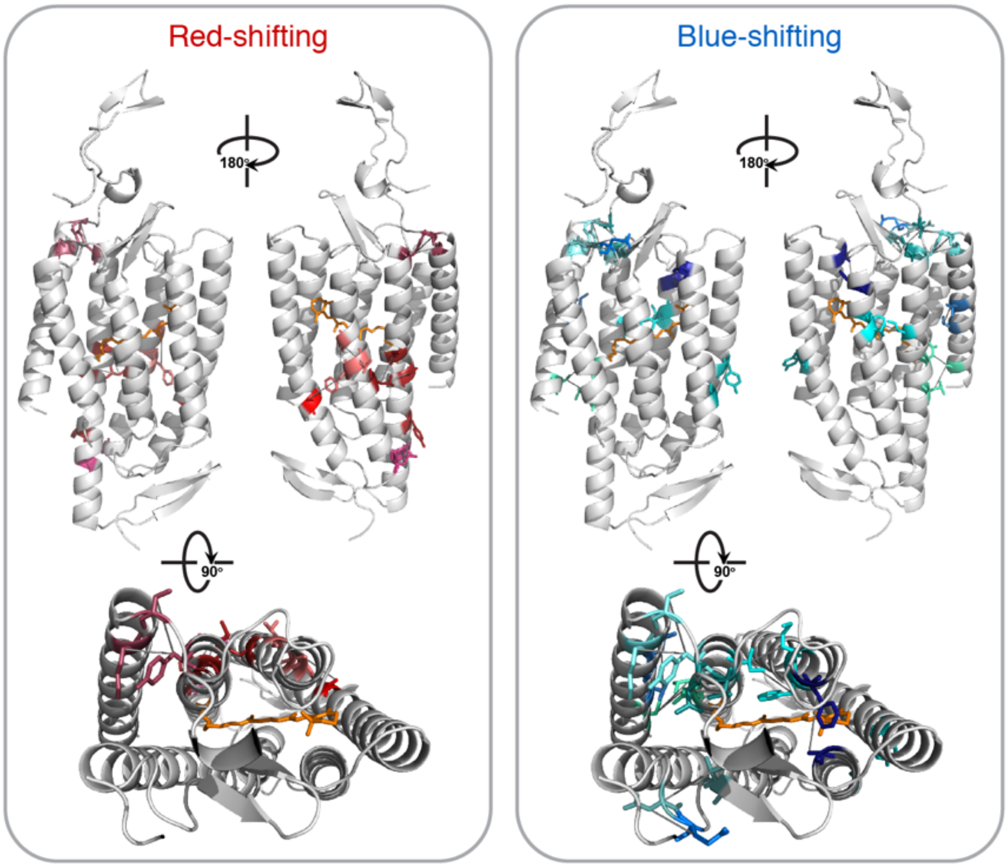


Figure 3.S3: Specific residues (amino-acid sticks) and contacts (dark gray lines) most important for model prediction of red-shifted light sensitivity and blue-shifted light sensitivity overlaid on the C1C2 crystal structure in light gray (3ug9.pdb).

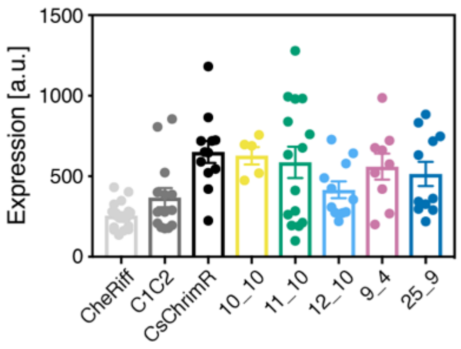
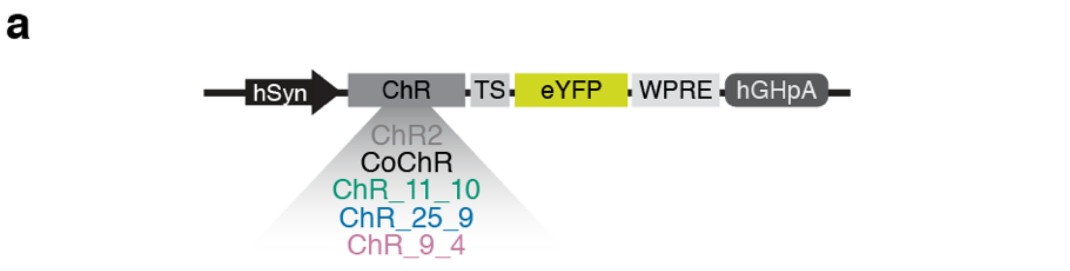


Figure 3.S4: HEK cell expression of selected high-conductance ChR variants. Plot of measured expression in HEK cells for each variant (CheRiff, n = 17 cells; C1C2, n = 14 cells; CsChrimR, n = 12 cells; ChR_10_10, n = 5 cells; ChR_11_10, n = 15 cells; ChR_12_10, n = 11 cells; ChR_9_4, n = 9 cells; ChR_25_9, n = 11 cells). Plotted data are mean ± SEM. No significant difference between CsChrimR and the high-conductance ChR variants with ANOVA and Dunnett's *post hoc* test.



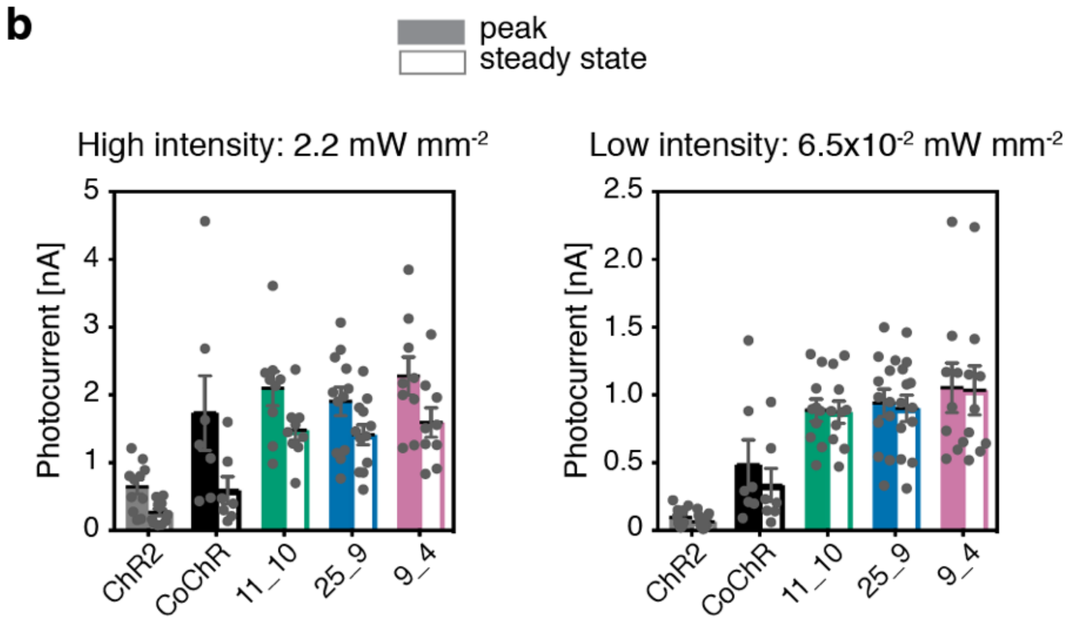


Figure 3.S5: (a) Construct design for each ChR tested with a trafficking sequence, eYFP, and WPRE under the hSyn promoter. (b) Peak and steady-state photocurrent comparison between high-conductance ChRs, ChR2(H134R), and CoChR with both high-intensity and low-intensity light. Multiple HEK cells were recorded from for each ChR: ChR2(H134R), *n* = 11 cells; CoChR, *n* = 7 cells; 11_10, *n* = 9 cells; 25_9, *n* = 12 cells; 9_4, *n* = 9 cells. Plotted data are mean ± SEM. There is a significant difference between the high-conductance ChR variants and ChR2 with low intensity light ($P < 0.0001$ for 11_10, 25_9, and 9_4); ANOVA and Dunnett's *post hoc* test, with ChR2 as a reference. There is a significant difference between the high-conductance ChR variants and CoChR with low intensity light ($P = 0.007$ for 11_10, $P < 0.003$ for 25_9, and $P < 0.0005$ for 9_4); ANOVA and Dunnett's *post hoc* test, with CoChR as a reference.

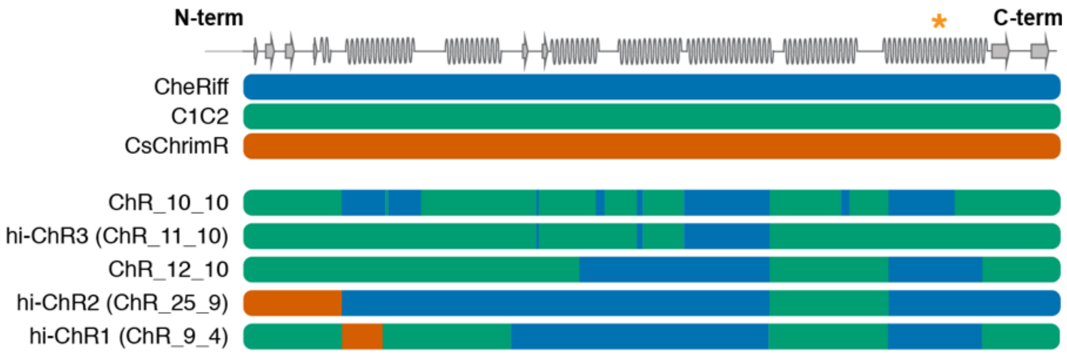


Figure 3.S6: Top five high-conductance ChRs predicted by the machine-learning models aligned with the three parents and the secondary structure. (*) highlights the Schiff base. Blocks of ChR chimeras are colored according to which parent each block came from. CsChrimR is red, CheRiff is blue, and C1C2 is green.

Table 3.S1: List of different constructs made for validation of the high-conductance ChRs.

Vector	Insert (X)	Virus tested
pAAV-hSyn-X-TS-eYFP-WPRE	hChR2(H134R)	Yes
	CoChR	
	hi-ChR1	
	hi-ChR2	
	hi-ChR3	
pAAV-CaMKIIa-X-TS-eYFP-WPRE	hChR2(H134R)	Yes
	hi-ChR1	
	hi-ChR2	
	hi-ChR3	
pAAV-CAG-DIO[X-TS-eYFP]-WPRE	hChR2(H134R)	Yes
	hi-ChR1	
	hi-ChR2	
	hi-ChR3	

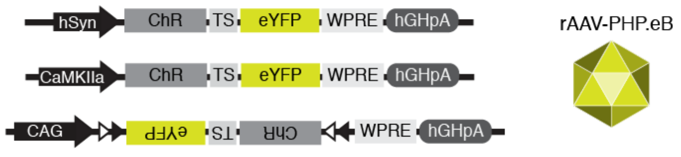


Table 3.S2: GP regression model hyperparameters for each ChR property of interest for the Matérn kernel.

Model type	ChR property	Noise hyperparameter: σ_n	Length hyperparameter: l
GP regression	current strength	0.04848652	19.65389071
GP regression	off-kinetics	0.02902597	19.72715834
GP regression	off-kinetics	0.10927067	37.7883682

Chapter 4

LEARNED PROTEIN EMBEDDINGS FOR MACHINE LEARNING

1. Kevin Yang, Zachary Wu, Claire Bedbrook, and Frances Arnold: Learned Protein Embeddings for Machine Learning. *Bioinformatics* (2018). doi: 10.1093/bioinformatics/bty178.

4.1 Introduction

Machine learning (ML) has been used to predict protein properties from protein sequences to enable protein design and engineering[1–3]. ML models are useful for predicting the outcomes of complex processes, such as how a protein sequence encodes function, because they do not require prior knowledge of specific physical or biological mechanisms. Instead, after training with measured sequences, ML models infer the properties of unseen sequences. A model capable of predicting the properties of unseen protein sequences enables prediction and discovery of sequences with optimal properties. For ML models to learn about protein sequences, we must encode the protein sequence in a form compatible with the mathematical operations used in ML models. Generally, this requires that the protein sequence be encoded as a vector or matrix of numbers. How each protein sequence is encoded determines what can be learned[4]. Even the most powerful models produce poor results if an inappropriate encoding is used. We show that learning these encodings from data can streamline machine-learning pipelines while achieving high predictive accuracies.

A protein sequence can be encoded by its physical properties or directly by its amino acids[1–3,5–8]. When using physical properties to encode a protein sequence, each individual amino acid is represented by a collection of physical properties, such as its charge or hydrophobicity, and each protein is taken to be a combination of those properties. Properties of the bulk protein, such as its predicted secondary structures, can also be used to represent the protein. However, there are countless physical properties that could be used to describe each amino acid/protein, and the molecular properties that dictate functional properties are unknown, highly constrained, and differ between different functional properties. Therefore, selecting informative properties is challenging because it is difficult to know *a priori* what properties will be predictive for a particular task.

Instead of representing a protein with physical properties, one can directly encode its amino acid sequence. A protein sequence of length L can be encoded as an $L\mathbf{y}n$ matrix, where n is the number of amino acids. Each row in the matrix consists of (n – 1) 0s and a single 1, with the position of the 1 indicating the amino acid residue at that position in the protein. This vectorization method for categorical data is known as one-hot encoding. One-hot encodings are inherently sparse, memory-inefficient, and high-dimensional. In a one-hot encoding, there is no notion of similarity between sequence or structural elements: they are either identical, or not. For example, in a one-hot encoding of words, the words "king", "prince", and "pot" are all not identical and thus equidistant from each other even though "king" and "prince" are intuitively more similar in meaning than "king" and "pot" or "prince" and "pot." Similarly, to the biologist, an amino acid sequence of DDD is more similar in meaning to EEE than to PPP or HHH. Furthermore, one-hot encodings of the primary sequence require that all sequence variants of interest are aligned. This alignment must be updated as sequences are added to the model. If updating the alignment changes its length, or even if amino acids are added or removed, the dimensionality of the encoding changes. Multiple sequence alignments between distantly-related proteins require visual validation because there is no universal standard for choosing the best alignment. Even with visual validation, it is challenging to confidently align distantly-related sequences. If the sequences are misaligned, the inputs to ML model are flawed, and there can be little expectation of success.

While there are a massive number of known protein sequences, only a tiny fraction have measured properties relevant to any specific task. Sequences with a measurement for the prediction task are known as labeled sequences, while those that do not are unlabeled sequences. The number of known unlabeled sequences will continue to rise as the cost of DNA-sequencing decreases, but there is no universal method for measuring all relevant protein properties. Therefore, the gap between the number of unlabeled and labeled sequences will continue to grow. However, even unlabeled sequences contain information about the frequency and patterns of amino acids selected by evolution to compose proteins. Information contained in unlabeled sequences may be helpful when predicting properties for a specific set of sequences, especially if the set in question is small. Specifically, instead of selecting physical properties or using a one-hot encoding of the sequence, a continuous vector encoding of each sequence can be learned from unlabeled sequences. This representation contains relevant information about the protein sequence learned from the distribution of sequences in the unlabeled set and is known as an embedded representation because it embeds the protein sequences in a vector space.

The process of using unlabeled data to learn an embedded representation has been well-established by recent work in natural language processing, where word and document embeddings are used as an efficient way to encode text for use in sentiment analysis, machine translation, and other tasks[9]. These examples learn an embedded representation from a large collection of unlabeled texts by assuming that words that appear in similar contexts have similar meanings. The unlabeled texts are analogous to the large number of unlabeled protein sequences. For example, the word2vec model[10,11] uses a shallow two-layer neural network to learn embeddings using one of two architectures: skip-gram and continuous bag-of-words. In the skip-gram architecture, the model uses the current word to predict its surrounding context words. In contrast, in the continuous bag-of-words architecture, the current word is predicted from its surrounding context words. The doc2vec model[12] extends word2vec by learning embeddings for entire sentences, paragraphs, or documents.

There have been efforts to apply word2vec and doc2vec to represent protein sequences[13–16]. These embeddings treat the amino acid sequence as a document and fragments of the amino acid sequence of constant length k (k-mers) as words. As shown in Figure 4.1, a sequence of 9 amino acids can be divided into 3 sets of non-overlapping 3-mers. The learned k-mer embeddings place k-mers that occur in similar contexts near each other in the embedded space by learning to predict a k-mer from its surrounding context k-mers and the sequence embedding. These embeddings have achieved high accuracy in differentiating ordered and disordered proteins and modest accuracy in classifying proteins from SwissProt into families based only on their primary sequence[13]. Our goal was to test if such embeddings can be used in ML to predict specific properties of related proteins. This is a fundamentally different problem than classifying proteins into families or predicting a universal binary property across all proteins because the model must tease apart the effects of subtle sequence changes from limited labeled data for a specific property.

In this work, we train embedded representations for four protein property prediction tasks. These tasks cover a range of protein families, measured properties, and library designs. We show that the predictive power of models trained using these embeddings is comparable to and sometimes exceeds those trained on one-hot encodings,

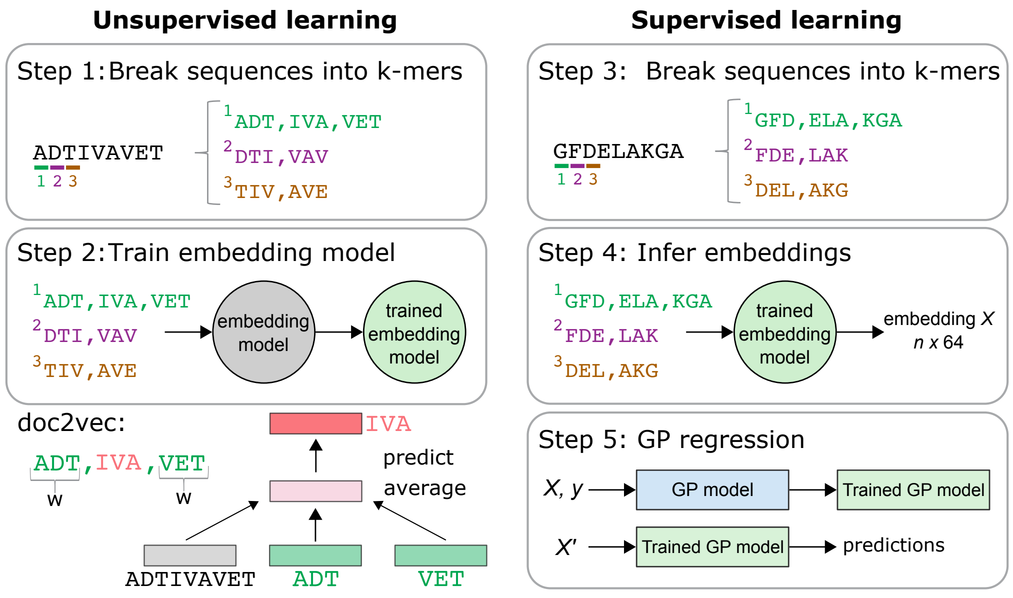


Figure 4.1: The modeling scheme. First, an unsupervised embedding model is trained on 524,529 unlabeled sequences pulled from the UniProt database. The UniProt sequences are broken into k lists of non-overlapping k-mers (Step 1), and then the lists are used to train the embedding model (Step 2). The doc2vec embedding model learns to predict the vectors for center k-mers from the vectors for their surrounding context k-mers and the sequence vectors. These sequence vectors are then the embedded representations of the sequences. Next, information learned during the unsupervised phase is applied during supervised learning with labeled sequences. The labeled sequences for each task (localiza-tion, T50, absorption, and enantioselectivity) are first broken into k lists of non-overlapping k-mers (Step 3). An embedding is then inferred for each se-quence using the trained embedding model (Step 4). n is the number of labeled sequences. Finally, during GP regression (Step 5), the inferred training embeddings X' and the training labels y are used to train a GP regression model, which can then be used to make predictions.

physical amino acid properties, or string mismatch kernels[17]. This suggests that embeddings enable accurate predictions despite having orders of magnitude fewer dimensions and being simpler to obtain because they do not require alignments, structural data, or selection of relevant amino-acid properties. Finally, we visualize the geometry of the embedding vectors, which captures meaningful relationships be-tween the embedded proteins.

4.2 Methods

Modeling Scheme

Figure 4.1 shows the two-part modeling scheme. Unsupervised doc2vec embedding models were trained on 524,529 protein sequences with lengths between 50 and 999 amino acids (mean length 326) obtained from UniProt using the distributed memory architecture[18]. In the distributed memory architecture, the model learns to predict the central k-mer based on the sequence embedding and the embeddings for a context window of k-mers on either side of the central k-mer. The size of the context, i.e. how many k-mers on either side to consider, is the window width (w), which can be adjusted in the embedding model. Each sequence was broken into k lists of non-overlapping k-mers. For example, for $k = 3$, there are three lists and each list begins at one of the first three amino acid positions of the sequence, as shown in Figure 4.1. Unsupervised embedding model train-ing was performed using the lists derived from the UniProt sequences. After unsupervised embedding model training, the embedding model was used to infer encodings of sequences for input to supervised Gaussian process (GP) regression models[19]. Embeddings for the sequences relevant to each task were de-termined by averaging the embeddings for the k lists of k-mers corresponding to each task sequence. It was found that GP performance was highly dependent on the order in which the embeddings for these sequences were inferred. Therefore, embeddings for each of the three tasks studied were calculated as the average of 100 inference runs with random input orders. These embeddings represent each task sequence in a very compact, low-dimensional form. We learn embeddings with between 4 and 128 dimensions. By comparison, the other representations used for comparison in this work have between 103 and 105 dimensions. In addition, sequences from disparate protein families are embedded in the same vector space, allowing comparisons between distant sequences and streamlining down-stream modeling. All doc2vec training and inference was performed in Gensim[20].

For some tasks, it was found that randomizing the UniProt sequences by shuffling or resampling before unsupervised embedding model training improved down-stream performance. Shuffling refers to scrambling the order of amino acids for each sequence. Alternatively, resampling refers to drawing sequences of the original lengths according to the overall observed amino acid frequency for the UniProt sequences (resample-UniProt) or according to uniform amino acid frequency (resample-uniform). The embedding model is then trained on these randomized sequences instead of the original UniProt sequences. We suspect that this has a regularizing effect on the embedding model: randomization prevents the embedding model from overfitting to a set of protein sequences that is not representative of those in the task. This also suggests that one of the key pieces of information the unsupervised embedding model learns is the frequency with which different amino acids occur in the same proteins.

The data for each task are taken from different protein engineering projects. When building a model that must generalize across diverse families of proteins, the best practice is to minimize sequence redundancy between the training and test sets[21]. However, protein-engineering projects typically generate data in a stepwise manner, where each subsequent set of sequences characterized is determined by previously characterized sequences. Therefore, we split the training and test sets such that the training sets contain sequences from earlier steps than those in the test sets, which come from later steps. This provides a realistic simulation of machine learning usage in protein engineering.

All embedding models were trained for 25 epochs. Embedding hyperparameters were chosen using 20-fold cross-validation on the training set. We set the dimension to 64 and considered values of k between 1 and 5, and values of w between 1 and 7. We used GP regression models with Matérn kernels with $\nu = \frac{5}{2}$. The noise and kernel hyperparameters were optimized by maximizing the marginal likelihood[19]. A GP model trained on the entire training set was then used to predict the relevant properties for test set sequences. GP models trained on embedded representations were com-pared to models trained on one-hot representations of amino acid sequence, mismatch string kernels with $k = 5$ and $m = 1$, ProFET[8], and a subset of AAIndex[22]. ProFET represents each sequence by extracting elementary biophysical and sequence-derived features. AAIndex is a set of 553 properties for each of the 20 amino acids. 64 of these properties were chosen by greedily maximizing the average cosine distance between the chosen properties. Each amino acid is therefore represented by a vector of 64 properties, and each protein is represented by concatenating the property vectors for its amino acid sequence. For two of the three tasks, structural information was available. For those tasks, models were also compared to a GP model trained on a one-hot representation of both the sequence and the structure. The structure was encoded in these cases by a binary indicator vector for the identity of each pair of amino acids within 4.5 Å in the crystal structure[1].

Tasks

We tested embeddings on three tasks with diverse proteins, different measured properties, and various methods of generating the original library. The data for these tasks were collected from previous studies and will only briefly be described here.

Channelrhodopsin (ChR) localization ('Localization') Two separate, ten-block recombination libraries were designed from three parental ChRs (CheRiff, C1C2, and CsChrimsonR). Each chimeric ChR variant in these libraries is composed of blocks of sequence from the parental ChRs. The data for this task comprise a total of 248 sequences. Genes for these sequences were synthesized and expressed in human embryonic kidney (HEK) cells, and their membrane localization was measured[2].

Cytochrome P450 thermostability ('T50') An eight-block recombination library was designed from three parental cytochrome P450s (CYP102A1, CYP102A2, and CYP102A3)[23]. The data for this task include 242 sequences from this library and 19 chimeric cytochrome P450s generated from other parents or cross-over points[1], for a total of 261 sequences. Genes for these sequences were expressed in *Escherichia coli* and their T50s (temperature at which half of the protein was irreversibly inactivated after a 10-minute incubation) were measured.

Rhodopsin absorption wavelength ('Absorption') Amino acid substitutions were made in the retinal-binding pocket of *Gloeobacter violaceus* rhodopsin (GR) in order to tune its peak absorption wavelength. GR is a light-activated proton pump[24]. The data for this task consist of GR and 80 blue- and red-shifted variants with 1-5 mutations generated in the course of tuning its absorption wavelength, for a total of 81 sequences.

Epoxide hydrolase enantioselectivity ('Enantioselectivity') Amino-acid substitutions were made in the binding pocket of the epoxide hydrolase (EH) from *Aspergillus niger* in order to improve its preference for the (S)-enantiomer of glycidyl phenyl ether. The data for this task consist of EH and 151 variants with 1-8 mutations generated in the course of improving its enantioselectivity, for a total of 152 variants[25].

These three tasks include light-sensitive integral membrane proteins (ChR and GR) and soluble enzymes (cytochrome P450 and EH). The tasks include libraries

Table 4.1: Summary of tasks used to evaluate embedded representations

Task	n	Protein	Library	Property
Localization	248	Channelrhodopsin	Recom.	Plasma membrane localization
T50	261	Cytochrome P450	Recom.	Thermostability
Absorption	81	Bacterial rhodopsin	SSM	Peak absorption wavelength
Enantioselectivity	152	Epoxide hydrolase	SSM	Enantioselectivity

Notes: Recom. and SSM denote library design by recombination and site-saturation mutagenesis, respectively.

constructed via recombination and site-directed mutagenesis and examine a variety of protein properties. The diversity of tasks allows us to evaluate the generality of embedded representations. Table 4.1 summarizes the tasks. Sequences and measurements are provided as Datasets 1-5 in the Supplementary Information.

4.3 Results and Discussion

We compared the quality of predictions for GP models trained on different encodings. Table 4.2 compares the GP regression results on the test set for each task using embeddings, physical properties from AAIndex, ProFET, a mismatch kernel with $k = 5$ and $m = 1$, and one-hot encodings. Figures 4.S1 – 4.S4 compare the actual test values to those predicted by GP regression models trained using each encoding. The embedding hyperparameters chosen for localization are shuffled, $k = 3$, and $w = 5$. For T50, they are no randomization, $k = 3$, and $w = 7$. For absorption, they are resample-uniform, $k = 4$, and $w = 1$. For enantioselectivity, they are resample-UniProt, $k = 3$, and $w = 7$. The cross-validation metrics for each task and each set of embedding hyperparameters are included as Datasets 5-8 in the Supplementary Information. GP regression predicts a Gaussian distribution, defined by its mean and variance, for each evaluation sequence. Predictions were evaluated using the mean absolute error (MAE), Kendall's τ (τ), and the Gaussian log-likelihood ($\log P$). The MAE measures deviation between predicted and actual values, τ measures ordinal accuracy, and log-likelihood provides a probabilistic measurement of model fit. Together, these three metrics provide a multifaceted comparison between different models.

For localization, embeddings trained on UniProt sequences slightly outperform one-hot encodings of sequence and structure. Previously, we showed that models built on one-hot encodings were sufficiently accurate to identify sequences that maximize localization[2]. For T50, embeddings achieve the best MAE and τ, while AAIndex

Table 4.2: Comparison of learned, dense, embedded representations, ProFET, AAIndex properties, mismatch string kernels and one-hot representations of sequence and structure for predicting protein properties using GP regression

Task	n_{train}	n_{test}	Representation	d	MAE	τ	log P
Localization	215	33	Embedding	**75**	**0.73**	**0.60**	-43.5
			Seq. and struct.	600747	0.76	**0.60**	**-43.2**
			Sequence	7161	0.76	0.59	-43.7
			Mismatch kernel	-	0.86	0.55	-54.6
			ProFET	1173	1.03	0.32	-54.9
			AAIndex	21824	0.76	0.55	-44.3
T50	242	19	Embedding	**64**	**2.91**	**0.61**	-59.5
			Seq. and struct.	994890	2.98	0.53	-57.3
			Sequence	9786	2.94	0.57	-57.2
			Mismatch kernel	-	4.03	0.38	-58.5
			ProFET	1173	4.93	0.43	-63.7
			AAIndex	29824	2.95	0.51	**-56.2**
Absorption	62	19	Embedding	**64**	23.3	0.57	-109.2
			Sequence	6258	22.1	0.63	-111.0
			Mismatch kernel	-	**17.8**	**0.68**	**-103.9**
			ProFET	1173	53.5	0.32	-174.7
			AAIndex	19072	30.1	0.35	-116.4
Enantioselectivity	136	16	Embedding	**64**	9.14	**0.64**	-64.5
			Sequence	8358	8.16	0.50	-63.3
			Mismatch kernel	-	**7.50**	0.46	-65.1
			ProFET	1173	27.9	0.27	-76.7
			AAIndex	25472	12.5	0.25	-65.7

Notes: n_{train} and n_{test} are the number of training and test examples, respectively. d is the dimension of the representation. MAE is the mean absolute error between predicted test values and the actual test values. τ is the Kendall's between the predicted test values and the actual test values. log P is the log Gaussian likelihood of the actual test values given the predicted distributions. All reported metrics are for the held-out test set. All embedding hyperparameters were chosen using 20-fold cross-validation on the training set. The best performance on each metric for each task is shown in bold.

achieves the highest log-likelihood. The one-hot encodings are comparable, while the mismatch kernel and ProFET perform much worse. Likewise, models built on one-hot encodings were previously shown to be sufficiently accurate in identifying sequences that maximize the T50[1]. For absorption, the mismatch kernel achieves the best performance across metrics, embeddings and the one-hot sequence encoding are comparable, while ProFET and AAIndex perform much worse. Finally, for enantioselectivity, embeddings achieve comparable performance to the one-hot sequence encoding and the mismatch kernel while ProFET and AAIndex are much worse.

For three of the four tasks, the embeddings are the most accurate by at least one metric even though they have several orders of magnitude fewer dimensions than the other representations. Mismatch string kernels are calculated directly from the amino acid sequences without an intermediate vector representation and therefore have no dimension. This shows that embeddings can be used as a low-dimensional representation of protein sequences for building machine-learning models of protein function. The training time for GP regression is dominated by the $O(n^3)$ time to invert the covariance matrix. However, on a 2016 Macbook Pro, models using 64-dimensional embeddings train approximately 10 times faster than those using one-hot embeddings of sequence and structure.

To better evaluate the information gained by the embedding model, we performed three negative controls, which are summarized in Table 4.3. First, we trained embedding models only on those sequences used in the task: during unsupervised embedding model training, we replaced the 500,000 UniProt sequences with the 81 – 261 sequences to be inferred. This decreased GP regression performance, suggesting that information from the unlabeled sequences improves predictions and therefore that the unsupervised embedding model is learning sequence-specific information from the unlabeled sequence data. Second, we confirmed that scrambling the order of the amino acids in the task-specific sequences before inferring their embeddings also decreases regression performance. This demonstrates that the embedding model is encoding useful information about the task sequences during the inference step, including information related to the order of the amino acids. Finally, we shuffled the training labels (i.e. the measured properties) for each sequence in the training set but not the test labels, which should remove the model's ability to learn anything about the test set from the training set. These negative controls show that the embedding model is applying information from the unlabeled sequences to learn

Table 4.3: Negative controls

Task	Control	MAE	τ	log P
Localization	-	0.73	0.60	-43.5
	Task sequences only	0.86	0.50	-50.0
	Shuffled task sequences	1.21	0.16	-57.4
	Shuffled training labels	1.16	-0.39	-58.3
T50	-	2.91	0.61	-59.5
	Task sequences only	5.02	0.45	-63.3
	Shuffled task sequences	4.49	0.31	-61.8
	Shuffled training labels	5.72	-0.35	-67.1
Absorption	-	23.3	0.57	-109.2
	Task sequences only	61.4	0.34	-162.1
	Shuffled task sequences	61.4	-0.03	-162.0
	Shuffled training labels	61.4	-0.43	-162.0
Enantioselectivity	-	9.14	0.64	-64.5
	Task sequences only	41.3	-0.06	-85.2
	Shuffled task sequences	42.7	0.27	-84.7
	Shuffled training labels	42.8	0.06	-84.8

meaningful embeddings for the labeled sequences.

In order to determine how many dimensions are required to represent a protein sequence, we compared GP model performance for embeddings inferred from lower-dimensional models with other hyperparameters held constant. Figure 4.2 shows that τ and MAE tend to worsen gradually as d decreases until $d = 16$, and then worsen very steeply. It is likely that predictive performance could be improved by optimizing d simultaneously with the other embedding hyperparameters. These results suggest that ~32 dimensions encode enough information about a 250 – 500 amino acid sequence to make predictions of the protein's functional properties.

Likewise, we compared GP model performance for embeddings inferred from subsets of the UniProt sequences with other hyperparameters held constant in order to determine the number of unlabeled sequences necessary for unsupervised embedding model training. Figure 4.3 illustrates that for localization and T50, both τ and MAE show little improvement as the number of unlabeled sequences increases past 100,000. However, for absorption, MAE continues to decrease as the number of unlabeled sequences increases. For enantioselectivity, τ continues to increase as the number of unlabeled sequences increases. The training sets for absorption and enantioselectivity are smaller than those for localization and T50. In addition, the

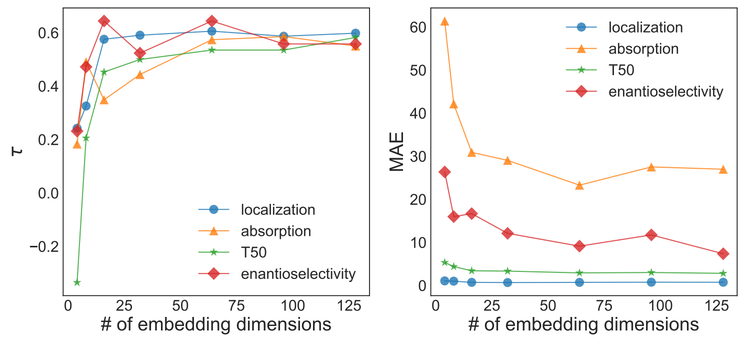


Figure 4.2: Effect of embedding dimension on predictive accuracy. For each task, embeddings of varying dimensions were trained and then used for GP regression. The resulting model quality was then evaluated using the Kendall's τ and MAE.

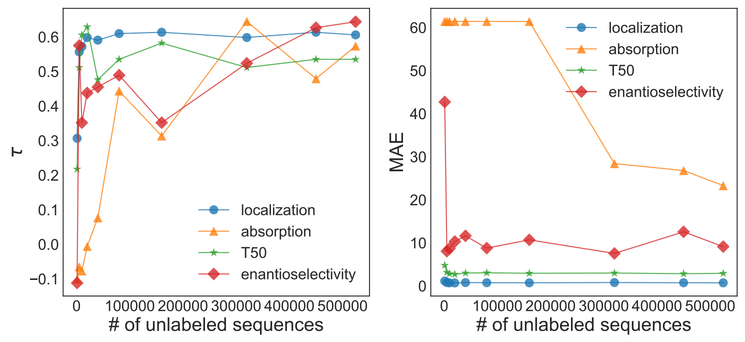


Figure 4.3: Effect of number of unlabeled sequences on predictive accuracy. For each task, embeddings were trained on subsets of the UniProt sequences and then used for GP regression. The resulting model quality was then evaluated using the Kendall's τ and MAE.

localization and T50 tasks use data from recombination libraries, and the training sets for these tasks are chosen to maximize information about the unseen members of these libraries, including those in the test sets. However, the absorption and enantioselectivity tasks use data from site-directed mutagenesis experiments, and the training sets are not designed to be in-formative about the test sets. Therefore, these tasks may benefit more from the additional information gained by unsupervised model training.

To visualize the geometry of the learned embeddings, we used t-distributed stochastic neighbor embedding (t-SNE)[26] to project the inferred embeddings, AAIndex, ProFET, and one-hot encodings of sequence onto a 2-dimensional space. Projections for ProFET use perplexity 10; the other projections use perplexity 50. Compared to other methods for dimensionality reduction, t-SNE focuses on local structure and tends to extract clustered local groups. Projections were calculated

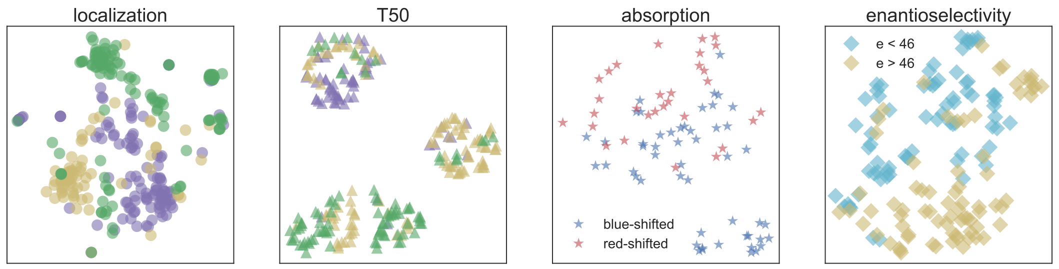


Figure 4.4: Visualization of learned vector representations of protein sequences. Vector representations projected onto 2 dimensions using t-SNE with perplexity 50 (embeddings, AAIndex, sequence) or 10 (ProFET). The sequences for the localization, the T50 and the enantioselectivity tasks are colored by the number of mutations from the nearest parent. The sequences for the absorption task are colored by peak absorption wavelength. Parents for localization, T50 and enantioselectivity are indicated by red triangles.

using scikit-learn's implementation of t-SNE with default parameters except where otherwise specified. Figure 4.4 shows these 2-dimensional projections.

The embeddings for localization cluster around each of the three re-combination parents, and variants with fewer mutations from the parents are closer to the parents. The projections for the AAIndex properties, ProFET, and one-hot encoding for localization show a similar pattern. The embeddings for T50 also cluster around each of the three recombination parents, and variants with fewer mutations are also closer to the parents. The projections for the AAIndex properties and one-hot encoding for T50 also place variants with fewer mutations closer to the parents, but there are not three clear clusters. The projection for ProFET does not show any clear structure. The embeddings for absorption roughly separate red-shifted and blue-shifted sequences, with the most blue-shifted sequences in a separate cluster. The projections for the AAIndex properties, ProFET, and one-hot encoding for absorption show the same blue-shifted cluster and rough separation. The embeddings for enantioselectivity place the sequences with the fewest mutations closest to the parent. The projections for the AAIndex properties, ProFET, and one-hot encoding also place variants with fewer mutations closer to the parent. Across the four diverse tasks, the inferred embeddings capture relationships between the sequences in the library.

The embedding model embeds sequences for all four tasks into the same vector space, so relationships between all the task sequences can also be interrogated. Figure 4.5 shows a 2-dimensional projection obtained using t-SNE with perplexity

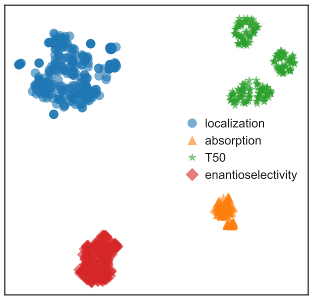


Figure 4.5: Combined visualization of vector representations for each of the four tasks. Sequences are colored to show separation between the embeddings for each task.

50 for all of the embedded representations. The embeddings for each protein family form their own cluster. Figure 4.S5 shows that the clustering of sequences most similar to each parent can still be observed for localization and T50, the absorption sequences still roughly separate by whether they are blue- or red-shifted, and the enantioselectivity sequences are roughly separated by their enantioselectivity.

4.4 Conclusions

This work shows that embedding models trained on proteins from UniProt can be applied to predict the functional properties of a small number of related proteins, such as those often encountered in protein engineering. Models trained using embeddings are comparable to and often outperform those trained on one-hot encodings of sequence and structural contacts, mismatch string kernels, or amino acid physical properties across four tasks, showing that embeddings generalize across protein families, library designs, and protein properties. As few as 32 dimensions are sufficient to achieve competitive model performance. However, the optimal embedding hyperparameters are highly dependent on the specific task. Negative controls show that the unsupervised embedding model incorporates information from the unlabeled sequences. Furthermore, the inferred embeddings show patterns consistent with the library designs when visualized in a 2-dimensional space. While the number of known protein sequences is rapidly increasing, it remains time-consuming and difficult to measure many protein properties of interest. By first training an unsupervised embedding model on unlabeled protein sequences, we are able to transfer information encoded in these unlabeled sequences to a specific task. This allows predictive models while bypassing many of the difficulties associated with using one-hot encodings and physical properties to represent protein sequences.

References

1. Philip A Romero, Andreas Krause, and Frances H Arnold: Navigating the protein fitness landscape with Gaussian processes. *Proc Natl Acad Sci USA* **110**(3) (2013), E193–E201. doi: 10.1073/pnas.1215251110 (cit. on pp. 98, 103, 104, 107).

2. Claire N Bedbrook, Kevin K Yang, Austin J Rice, Viviana Gradinaru, and Frances H Arnold: Machine learning to design integral membrane channelrhodopsins for efficient eukaryotic expression and plasma membrane localization. *PLoS Comput Biol* **13**(10) (2017), e1005786. doi: 10.1371/journal.pcbi.1005786 (cit. on pp. 98, 104, 105).

3. Richard J Fox et al.: Improving catalytic function by ProSAR-driven enzyme evolution. *Nat Biotechnol* **25**(3) (2007), 338 (cit. on p. 98).

4. Pedro Domingos: A few useful things to know about machine learning. *Communications of the ACM* **55**(10) (2012), 78–87 (cit. on p. 98).

5. Babak Alipanahi, Andrew Delong, Matthew T Weirauch, and Brendan J Frey: Predicting the sequence specificities of DNA-and RNA-binding proteins by deep learning. *Nat Biotechnol* **33**(8) (2015), 831. doi: 10.1038/nbt.3300 (cit. on p. 98).

6. Shyam M Saladi, Nauman Javed, Axel Müller, and William M Clemons: A statistical model for improved membrane protein expression using sequence-derived features. *J Biol Chem* **293**(13) (2018), 4913–4927. doi: 10.1074/jbc.RA117.001052 (cit. on p. 98).

7. Catherine Ching Han Chang et al.: Periscope: quantitative prediction of soluble protein expression in the periplasm of *Escherichia coli*. *Sci Rep* **6** (2016), 21844. doi: 10.1038/srep2184 (cit. on p. 98).

8. Dan Ofer and Michal Linial: ProFET: Feature engineering captures high-level protein functions. *Bioinformatics* **31**(21) (2015), 3429–3436. doi: 10.1093/bioinformatics/btv345 (cit. on pp. 98, 103).

9. Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria: Recent trends in deep learning based natural language processing (2017). eprint: https://arxiv.org/abs/1708.02709 (cit. on p. 100).

10. Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean: Distributed representations of words and phrases and their compositionality. *Adv Neur In*. 2013, 3111–3119 (cit. on p. 100).

11. Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean: Efficient estimation of word representations in vector space (2013). eprint: https://arxiv.org/abs/1301.3781 (cit. on p. 100).

12. Quoc Le and Tomas Mikolov: Distributed representations of sentences and documents. *ICML*. 2014, 1188–1196 (cit. on p. 100).

13. Ehsaneddin Asgari and Mohammad RK Mofrad: Continuous distributed representation of biological sequences for deep proteomics and genomics. *PloS One* **10**(11) (2015), e0141287. doi: 10.1371/journal.pone.0141287 (cit. on p. 100).

14. Carlo Mazzaferro: Predicting Protein Binding Affinity With Word Embeddings And Recurrent Neural Networks (2017), 128223. doi: 10.1101/128223. eprint: https://www.biorxiv.org/content/early/2017/04/18/128223 (cit. on p. 100).

15. Patrick Ng: dna2vec: Consistent vector representations of variable-length k-mers (2017). arXiv: https://arxiv.org/abs/1701.06279 `[q-bio]` (cit. on p. 100).

16. Dhananjay Kimothi, Akshay Soni, Pravesh Biyani, and James M Hogan: Distributed Representations for Biological Sequence Analysis (2016). eprint: https://arxiv.org/abs/1608.05949 (cit. on p. 100).

17. Christina S Leslie, Eleazar Eskin, Adiel Cohen, Jason Weston, and William Stafford Noble: Mismatch string kernels for discriminative protein classification. *Bioinformatics* **20**(4) (2004), 467–476 (cit. on p. 101).

18. UniProt Consortium: UniProt: the universal protein knowledgebase. *Nucleic Acids Res* **45**(D1) (2017), D158–D169. doi: 10.1093/nar/gkw1099 (cit. on p. 102).

19. C. E. Rasmussen and C. K. I. Williams: *Gaussian Processes for Machine Learning*. MIT Press, 2006 (cit. on pp. 102, 103).

20. Radim Řehůřek and Petr Sojka: Software Framework for Topic Modelling with Large Corpora. English. *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. http://is.muni.cz/publication/884893/en. Valletta, Malta: ELRA, May 2010, 45–50 (cit. on p. 102).

21. Wajid Arshad Abbasi and Fayyaz Ul Amir Afsar Minhas: Issues in performance evaluation for host–pathogen protein interaction prediction. *J Bioinform Comput Biol* **14**(03) (2016), 1650011. doi: 10.1142/S0219720016500116 (cit. on p. 103).

22. Shuichi Kawashima et al.: AAindex: amino acid index database, progress report 2008. *Nucleic Acids Res* **36**(suppl_1) (2007), D202–D205 (cit. on p. 103).

23. Yougen Li et al.: A diverse family of thermostable cytochrome P450s created by recombination of stabilizing fragments. *Nat Biotechnol* **25**(9) (2007), 1051 (cit. on p. 104).

24. Martin KM Engqvist et al.: Directed evolution of Gloeobacter violaceus rhodopsin spectral properties. *J Mol Biol* **427**(1) (2015), 205–220. doi: 10.1016/j.jmb.2014.06.015 (cit. on p. 104).

25. Julian Zaugg, Yosephine Gumulya, Alpeshkumar K Malde, and Mikael Bodén: Learning epistatic interactions from sequence-activity data to predict enantioselectivity. *J Comput Aided Mol Des* **31**(12) (2017), 1085–1096. doi: 10.1007/s10822-017-0090-x (cit. on p. 104).

26. Laurens van der Maaten and Geoffrey Hinton: Visualizing data using t-SNE. *J Mach Learn Res* **9**(Nov) (2008), 2579–2605 (cit. on p. 109).

Supplemental Information

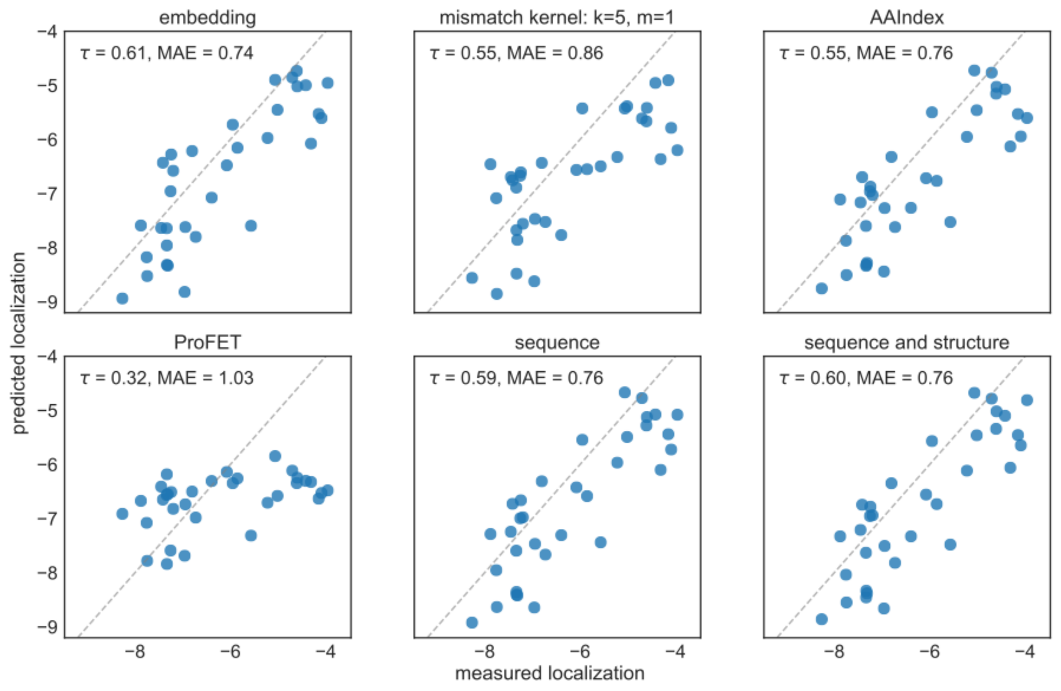


Figure 4.S1: Test predictions for ChR localization. Predicted vs measured values on test sequences using GP regression models trained using each encoding method.

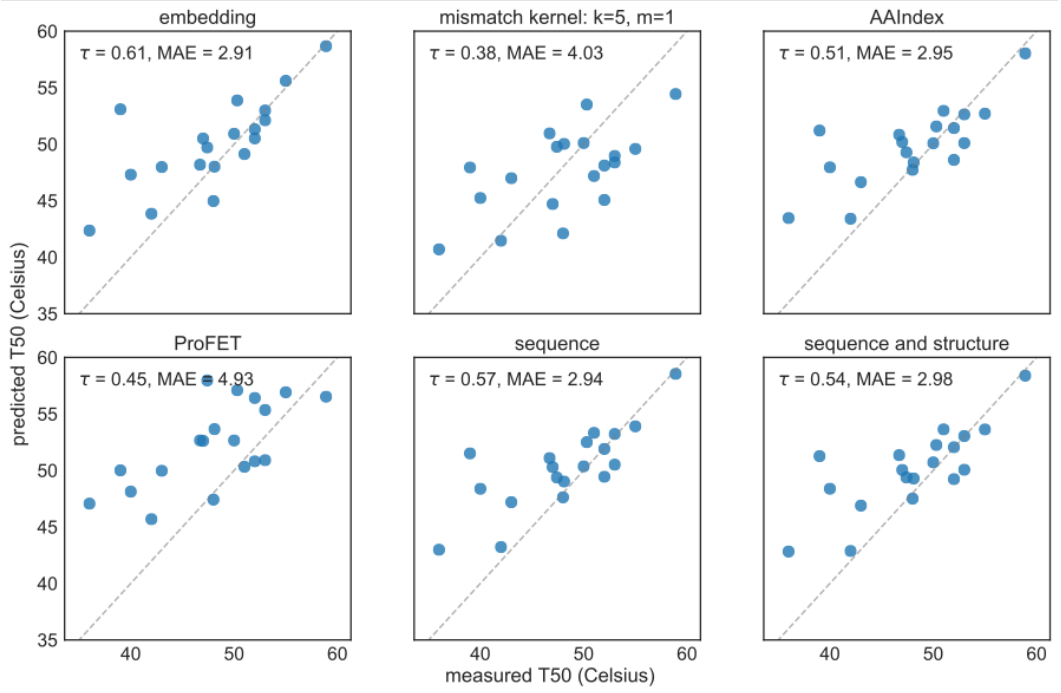


Figure 4.S2: Test predictions for P450 T50. Predicted vs measured values on test sequences using GP regression models trained using each encoding method.

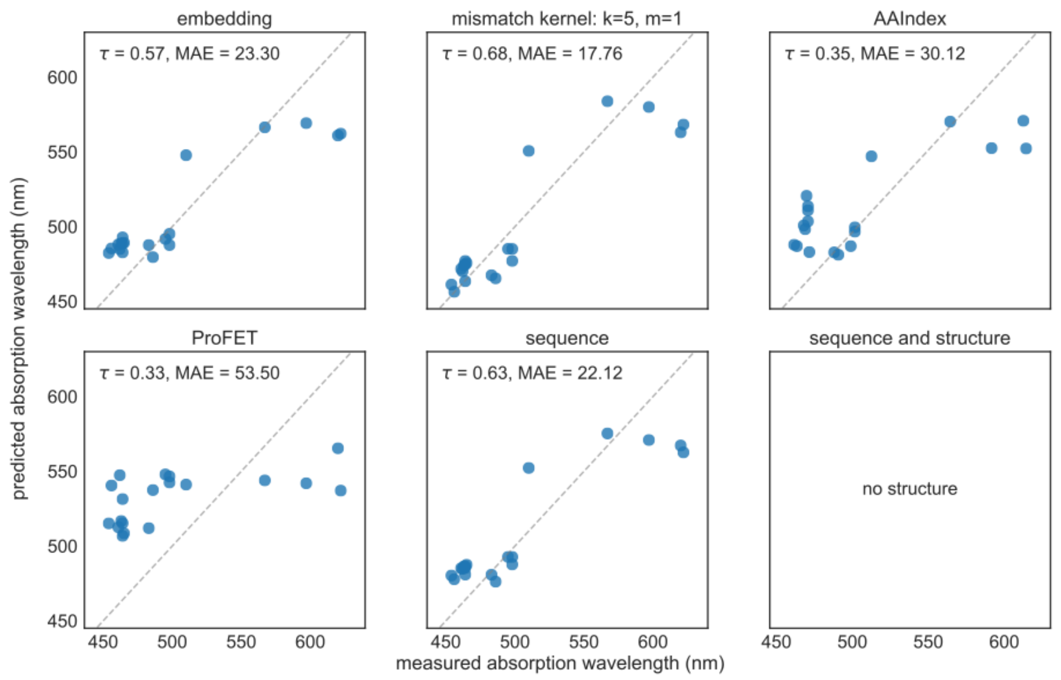


Figure 4.S3: Test predictions for rhodopsin absorption. Predicted vs measured values on test sequences using GP regression models trained using each encoding method.

Dataset 1: localization.txt. This dataset contains the sequences and measurements for the localization task. The columns are name: the name of the sequence; sequence: the amino acid sequence; log_GFP: the localization measurement in arbitrary units; is_train: whether the measurement is part of the training set; and *m*: the number of mutations from the closest parent sequence. The three recombination parents are named cschrimson, c1c2, and cheriff.

Dataset 2: T50.txt. This dataset contains the sequences and measurements for the T50 task. The columns are name: the name of the sequence; sequence: the amino acid sequence; T50: the T50 measurement in °C; is_train: whether the measurement is part of the training set; and *m*: the number of mutations from the closest parent sequence. The three recombination parents are named 00000000, 11111111, and 22222222.

Dataset 3: absorption.txt. This dataset contains the sequences and measurements for the absorption task. The columns are name: the name of the sequence; sequence: the amino acid sequence; peak: the peak absorption wavelength in nanometers; is_train: whether the measurement is part of the training set; and *m*: the number of mutations from the closest parent sequence.

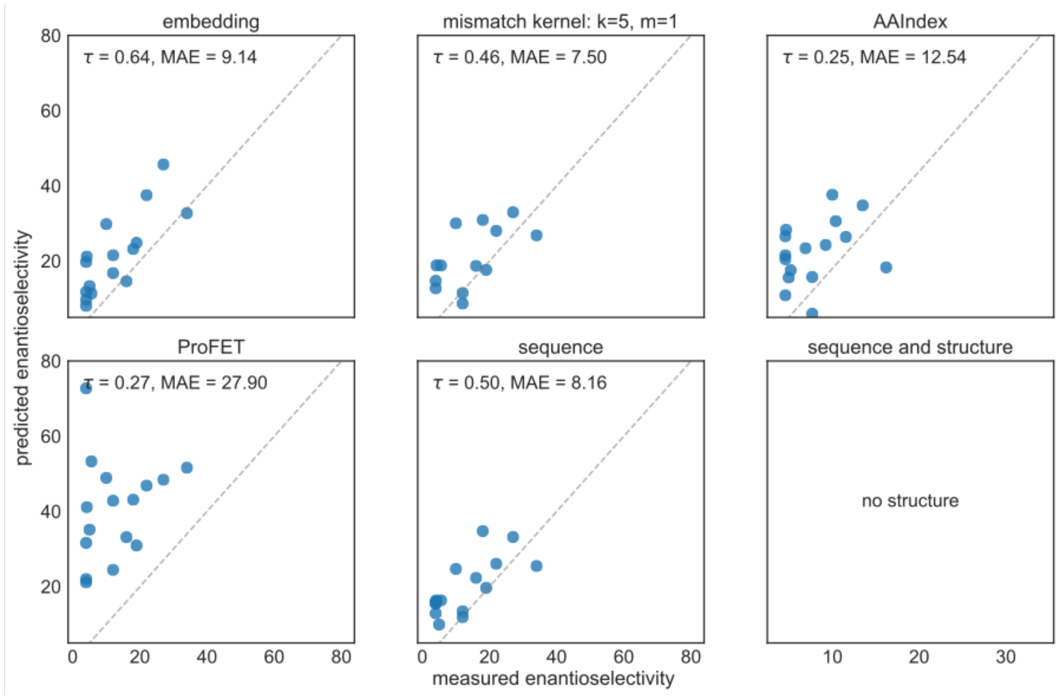


Figure 4.S4: Test predictions for epoxide hydrolase enantioselectivity. Predicted vs measured values on test sequences using GP regression models trained using each encoding method.

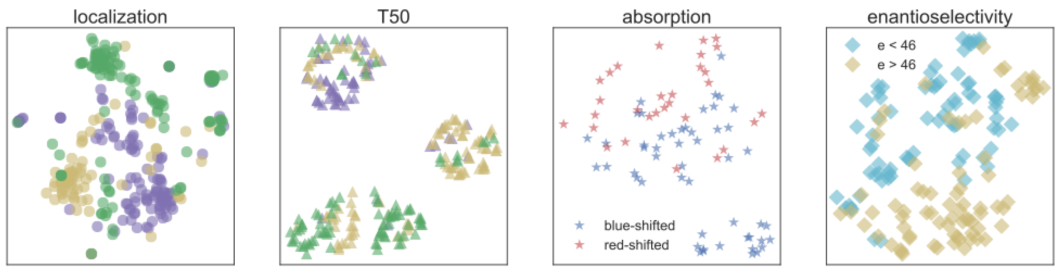


Figure 4.S5: t-SNE visualizations calculated for all embeddings. Visualizations for localization and T50 are colored according to the closest recombination parent. Visualizations for absorption are colored according to whether the peak absorption wavelength is blue- or red-shifted compared to the parent. Visualizations for enantioselectivity are colored by whether each sequence is above or below the median (e = 46).

Dataset 4: enantioselectivity.txt. This dataset contains the sequences and measurements for the enantioselectivity task. The columns are name: the name of the sequence; sequence: the amino acid sequence; e-value: the enantiomeric ratio; is_train: whether the measurement is part of the training set; and *m*: the number of mutations from the closest parent sequence.

Dataset 5: cv_localization.txt. This dataset contains the 20-fold cross-validation results for all embedding hyperparameters on the localization task. The columns are task: the measurement predicted; embedding: the embedding used; kernel: the Gaussian process kernel used; R: the Pearson correlation; R^2: the coefficient of determination; kendalltau: Kendall's τ; log_loss: the log likelihood loss; and SE: the mean squared error. The embedding name always begins with 'X_' followed by the name of the unlabeled sequences used in embedding model training, then the k-mer size k and the window width w. *T*he possible unlabeled sequences are 'original' (the full unrandomized UniProt dataset), 'random' (resample-UniProt), 'scrambled' (shuffled), 'uniform' (resample-uniform), 'small' (1000 sequences randomly sampled from the full UniProt dataset), 'P450_data' (the labeled P450 sequences enumerated in T50.txt), 'P450_all' (all possible chimeras in the recombination library from which the sequences in T50.txt are sampled), 'peak' (the sequences enumerated in absorption.txt), 'ChR_data' (the sequences enumerated in localization.txt), 'ChR_all' (all possible chimeras in the recombination library from which the sequences in localization.txt are sampled), and 'aneh' (the sequences enumerated in enantioselectivity.txt).

Dataset 6: cv_T50.txt. This dataset contains the 20-fold cross-validation results for all embedding hyperparameters on the T50 task. The columns are task: the measurement predicted; embedding: the embedding used; kernel: the Gaussian process kernel used; R: the Pearson correlation; R^2: the coefficient of determination; kendalltau: Kendall's τ; log_loss: the log likelihood loss; and SE: the mean squared error. The embedding name always begins with 'X_' followed by the name of the unlabeled sequences used in embedding model training, then the k-mer size k and the window width w. The possible unlabeled sequences are 'original' (the full unrandomized UniProt dataset), 'random' (resample-UniProt), 'scrambled' (shuffled), 'uniform' (resample-uniform), 'small' (1000 sequences randomly sampled from the full UniProt dataset), 'P450_data' (the labeled P450 sequences enumerated in T50.txt), 'P450_all' (all possible chimeras in the recombination library from which the sequences in T50.txt are sampled), 'peak' (the sequences enumerated in absorption.txt), 'ChR_data' (the sequences enumerated in localization.txt), 'ChR_all' (all possible chimeras in the recombination library from which the sequences in localization.txt are sampled), and 'aneh' (the sequences enumerated in enantioselectivity.txt).

Dataset 7: cv_absorption.txt. This dataset contains the 20-fold cross-validation results for all embedding hyperparameters on the absorption task. The columns are task: the measurement predicted; embedding: the embedding used; kernel: the Gaussian process kernel used; R: the Pearson correlation; R^2: the coefficient of determination; kendalltau: Kendall's τ; log_loss: the log likelihood loss; and SE: the mean squared error. The embedding name always begins with 'X_' followed by the name of the unlabeled sequences used in embedding model training, then the k-mer size k and the window width w. The possible unlabeled sequences are 'original' (the full unrandomized UniProt dataset), 'random' (resample-UniProt), 'scrambled' (shuffled), 'uniform' (resample-uniform), 'small' (1000 sequences randomly sampled from the full UniProt dataset), 'P450_data' (the labeled P450 sequences enumerated in T50.txt), 'P450_all' (all possible chimeras in the recombination library from which the sequences in T50.txt are sampled), 'peak' (the sequences enumerated in absorption.txt), 'ChR_data' (the sequences enumerated in localization.txt), 'ChR_all' (all possible chimeras in the recombination library from which the sequences in localization.txt are sampled), and 'aneh' (the sequences enumerated in enantioselectivity.txt).

Dataset 8: cv_enantioselectivity.txt. This dataset contains the 20-fold cross-validation results for all embedding hyperparameters on the enantioselectivity task. The columns are task: the measurement predicted; embedding: the embedding used; kernel: the Gaussian process kernel used; R: the Pearson correlation; R^2: the coefficient of determination; kendalltau: Kendall's τ; log_loss: the log likelihood loss; and SE: the mean squared error. The embedding name always begins with 'X_' followed by the name of the unlabeled sequences used in embedding model training, then the k-mer size k and the window width w. The possible unlabeled sequences are 'original' (the full unrandomized UniProt dataset), 'random' (resample-UniProt), 'scrambled' (shuffled), 'uniform' (resample-uniform), 'small' (1000 sequences randomly sampled from the full UniProt dataset), 'P450_data' (the labeled P450 sequences enumerated in T50.txt), 'P450_all' (all possible chimeras in the recombination library from which the sequences in T50.txt are sampled), 'peak' (the sequences enumerated in absorption.txt), 'ChR_data' (the sequences enumerated in localization.txt), 'ChR_all' (all possible chimeras in the recombination library from which the sequences in localization.txt are sampled), and 'aneh' (the sequences enumerated in enantioselectivity.txt).

Chapter 5

BATCHED STOCHASTIC BAYESIAN OPTIMIZATION VIA COMBINATORIAL CONSTRAINTS DESIGN

5.1 Introduction

Bayesian optimization techniques leverage regularity assumptions, such as smoothness and continuity, to sequentially optimize unknown utility functions. Bayesian optimization offers efficient solutions across high-dimensional problem settings including experimental design and recommender systems. Bayesian optimization techniques typically assume that items can be directly queried at each iteration. However, in many applications, this is not true: instead, a library of items is specified, and then batches of items from the library are stochastically queried.

As a prototypical example, let us consider *site-saturation mutagenesis* (SSM) [1], a protein-engineering strategy that mutates a small number of critical sites in a protein sequence (cf. Fig. 5.1). At each round, a combinatorial library is designed by specifying the allowed amino acids at the specified sites (step (1-3)), and then a batch of sequences from the library is sampled with replacement (step 4). The sampled sequences are evaluated for their ability to perform a desired function (step 5), such as a chemical reaction. To uncover non-linear effects, it is desirable to simultaneously mutate multiple sites in each round. Ideally, at each iteration, the amino acids to be considered at each site should be chosen to maximize the number of improved sequences expected in the stochastic batch sample from the resulting library.

Finding the such libraries is highly non-trivial: it requires solving a combinatorial optimization problem over an exponential number of items. Libraries are designed by choosing the allowed amino acids at each site (‘constraints’) from the set of all amino acids at all sites. Adding allowed constraints results in an exponential number of items in the library. Furthermore, due to the uncertainty in the predictions, and the fact that the queries will be randomly selected with replacement from the resulting library, one must devise a new optimization scheme as well as new theoretical and algorithmic tools for addressing such problem.

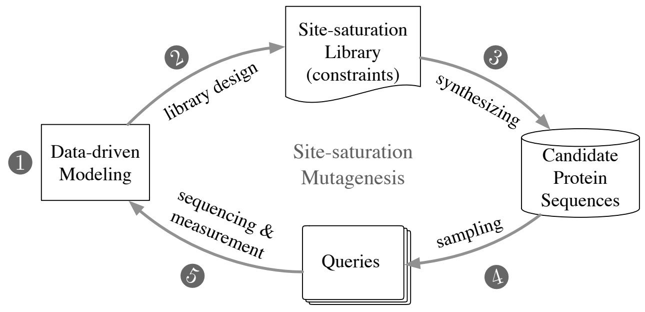

Site-saturation Library (constraints)
Data-driven Modeling
Site-saturation Mutagenesis
Candidate Protein Sequences
Queries
1
2
library design
3
synthesizing
4
sampling
5
sequencing & measurement

Figure 5.1: Data-driven site-saturation mutagenesis. (1) Machine learning model for predicting certain protein properties ; (2) site-saturation library design; (3) synthesize protein sequences according to the site-saturation libraries; (4) randomly sample proteins for sequencing; (5) sequence and measure the properties of the sampled proteins.

Our contribution In this paper, we investigate *Batched Stochastic Bayesian Optimization* (BSBO), a novel Bayesian optimization scheme for choosing a library design in order to guide exploration towards items with greater utility. This scheme is unique in that we choose a library design instead of directly querying items, and the items are queried in stochastic batches (e.g. 10s – 1000s at a time). In particular, we focus on library design for site-saturation mutagenesis, and identify a natural objective function that evaluates the quality of a library design given the current information about the system. We propose Online-DSOpt, an efficient online algorithm for optimization over stochastic batches. In a nutshell, Online-DSOpt assembles each batch by decomposing the objective function into the difference of two submodular functions (DS) [2]. This allows us to employ DS optimization tools to greedily identify sets of constraints that increase the likelihood of finding items with high utility. We demonstrate the performance of Online-DSOpt on both synthetic and two experimentally-generated protein datasets, and show that our algorithm in general outperforms conventional greedy heuristics and efficiently finds rare, highly-improved, sequences.

5.2 Related Work

Gaussian process Bayesian optimization Our work addresses a specific setting for Gaussian process (GP) Bayesian optimization. GPs are infinite collections of random variables such that every finite subset of random variables has a multivariate Gaussian distribution. A key advantage of GPs is that it is very efficient to perform inference, which makes it one of the most popular theoretical tools for Bayesian optimization [3–5]. Notably, [4] introduce the Gaussian Process Upper Confidence

Bound (GP-UCB) algorithm for Bayesian optimization, which provides bounds on the cumulative regret when sequentially querying items. [6] generalize this to batch queries. In contrast to our setting, these algorithms require the ability to *directly* query items, either sequentially or in batches.

GP optimization for protein engineering GP-UCB has been used to find improved protein sequences when sequences can be queried directly [7, 8]. A GP has been used to select constraints for an SSM library [9]. However, although this work empirically finds improved sequences by constraining the amino acids at each site, it do not provide a general procedure for selecting their constraints. Instead, they take advantage of a fortuitous observation, which will not generalize to other systems or even replicate experiments on their system.

Information-parallel learning In addition to the bandit setting [6], there is a large body of literature on various machine learning settings that exploit information-parallelism. For example, in large-scale optimization, mini-batch/parallel training has been extensively explored to reduce the training time of stochastic gradient descent [10, 11]. In batch-mode active learning [12–14], an active learner selects a set of examples to be labeled simultaneously. The motivation behind batch active learning is that in some cases it is more cost-effective to request labels in large batches, rather than one-at-a-time. This setting is also referred to as buy-in-bulk learning [15]. In addition to the simpler modeling assumption of being able to directly issue queries, these approaches also differ from our setting in terms of the objective: the batch-mode active learning algorithms aim to find a set of items that are maximally informative about some target hypothesis (hence to maximally explore), whereas we want to identify the best item (i.e., to both explore and exploit).

Submodularity and DS optimization The importance of submodularity [16] has been widely recognized in recent years in theoretical computer science and machine learning and is key to solving many discrete problems. While there has been a growing number of problems that can be expressed as submodular minimization [17] or maximization [18, 19] problems, it still only captures a small subset of discrete optimization problems. [2] show that any set function q can be decomposed as the difference of two submodular functions h and g. Replacing h with its modular upper bound, g with its modular lower bound, or both reduces the problem of minimizing q to a series of submodular minimizations, submodular maximizations, or modular minimizations, respectively, that are guaranteed to reduce q at every iteration and to arrive at a local minimum of q [20]. In general, DS decomposition requires exponential time. We prove two polynomial-time decompositions of our objective function.

5.3 Problem Statement

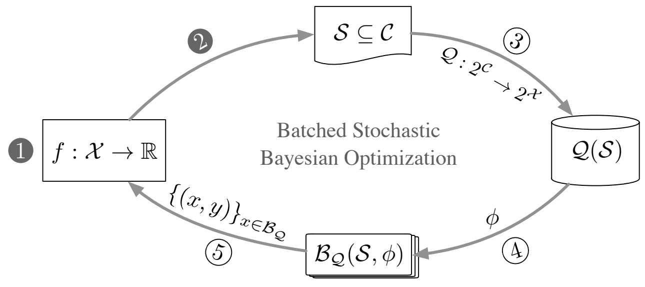
We propose the batched stochastic Bayesian optimization (BSBO) problem, in which an algorithm iteratively optimizes an unknown utility function $f : \mathcal{X} \to \mathbb{R}$ (Fig. 5.2, step (1)).

- The algorithm designs a library by choosing a set of constraints $\mathcal{S} \subseteq C$ based on its current knowledge about f (Fig. 5.2, step (2)). We will use a GP to model f. This generates a library: $Q(\mathcal{S})$, where $Q : 2^C \to 2^\mathcal{X}$ denotes the physical process that produces items under these constraints (Fig. 5.2, step (3)).
- A batch of n queries: $\mathcal{B}_Q(\mathcal{S}, \phi) \subseteq \mathcal{X}$ where ϕ represents the random state of the sampling procedure, is randomly selected from the library $Q(\mathcal{S})$ via a stochastic sampling procedure (Fig. 5.2, step (4)). Each query x achieves utility $f(x)$ and incurs some cost $c(\{x\})$, where $c : 2^\mathcal{X} \to \mathbb{R}$ denotes the cost function of a set of items (Fig. 5.2, step (5)).
- The results of the query are used to update our GP posterior for f.

This setting presents the dual challenges of needing to optimize over the space of constraints (which generate a library) instead of directly over items and of stochastic sampling instead of exact queries. If at round t we pick constraints $\mathcal{S}_t$, then the expected utility of the t^{th} batch is $F(\mathcal{S}_t) \triangleq \mathbb{E}_\phi\left[\sum_{x\in\mathcal{B}_Q(\mathcal{S}_t,\phi)} f(x)\right]$.

The goal in BSBO is to identify improved items as efficiently as possible. Assume that we have a budget of querying n items for each batch of experiments and that each batch $\mathcal{B}_Q$ is selected by sampling uniformly from the library. At each iteration, we wish to select the constraints $\mathcal{S}$ that will maximize the (expected) number of improved items observed in the next stochastic batched query. If the current best item has a value τ, then this objective is

$$F(\mathcal{S}) = \mathbb{E}_\phi\left[\sum_{x\in\mathcal{B}_Q(\mathcal{S})} \mathbb{1}(f(x) > \tau)\right] \tag{5.1}$$



1
$f : \mathcal{X} \to \mathbb{R}$
2
$\mathcal{S} \subseteq \mathcal{C}$
3
$\mathcal{Q} : 2^{\mathcal{C}} \to 2^{\mathcal{X}}$
$\mathcal{Q}(\mathcal{S})$
Batched Stochastic Bayesian Optimization
ϕ
4
$\mathcal{B}_{\mathcal{Q}}(\mathcal{S}, \phi)$
5
$\{(x, y)\}_{x \in \mathcal{B}_{\mathcal{Q}}}$

Figure 5.2: The batched stochastic Bayesian optimization setting. (1) Bayesian modeling (2) combinatorial constraints design; (3) candidate query generation; (4) random sampling; (5) batched queries.

Here, $\mathbb{1}$ is the indicator function. This objective is intractable under the GP posterior, as the dependencies between $f(x)$ preclude a closed form. We ignore the dependencies between the utilities to arrive at the following surrogate function

$$\hat{F}(\mathcal{S}) = \sum_{x \in Q(\mathcal{S})} \rho(x) \left[1 - \left(1 - \frac{1}{|Q(\mathcal{S})|} \right)^{n} \right] \tag{5.2}$$

The rewards $\rho(x) = P(f(x) > \tau)$ can be computed for all $x \in Q(C)$ from the GP posterior for each item using the Gaussian survival function by ignoring off-diagonal entries in the predictive posterior covariance. Note that the surrogate objective $\hat{F}$ captures the expected reward under an *independence* assumption. As we will demonstrate later in §5.5, despite such an assumption, we observe a strong correlation between $\hat{F}$ and F on the experimental datasets we study, which are known to have high dependencies between the rewards $\rho(x)$ of different items.

We now consider the site-saturation library design problem as a special case of batched stochastic Bayesian optimization. In SSM, the utility function $f(x)$ specifies the utility of a protein sequence x, and the constraint set $C = \bigcup_{\ell=1}^{L} C^{(\ell)}$ specifies the set of amino acids allowed at each site of the protein sequence. Here, L denotes the number of sites, and $C^{(\ell)}$ denotes the set of all possible amino acids[1] at site ℓ. We denote the set of amino acids selected for site ℓ by $\mathcal{S}^{(\ell)}$; hence $\mathcal{S} = \bigcup_{\ell=1}^{L} \mathcal{S}^{(\ell)}$. The candidate query pool (library) Q consists of all possible protein sequences that can be generated w.r.t. the constraints $\mathcal{S}$:

$$Q(\mathcal{S}) = \prod_{\ell=1}^{L} \mathcal{S}^{(\ell)} \tag{5.3}$$

Note that adding constraints generally *increases* the number of allowed items.

[1] In SSM, $C^{(\ell)}$ corresponds to the 20 canonical amino acids.

Algorithm 1: Online Batched Constraints Design via DS Optimization (Online-DSOpt)

```
1 Input: Constraints set $C = \bigcup_{\ell=1}^{L} C^{(\ell)}$; number of rounds $T$; budget on each batch $n$; GP prior on $f$
  begin
2     $\mathcal{A} \leftarrow \emptyset$
      /* iteratively select the next batch */
      for $t$ in $1, \ldots, T$ do
          /* compute the reward matrix $M = \{f(x)\}$ */
3         $M \leftarrow$ ComputeReward(posterior on $f, \mathcal{A}$)
4         $\mathcal{S} \leftarrow$ DSOpt$(C, M, n)$
          /* posterior update */
5         $\mathcal{A} \leftarrow \mathcal{A} \cup \mathcal{B}_Q(\mathcal{S}, \phi)$
6     Output: Optimizer of $f$
```

5.4 Algorithms

We now present Online-DSOpt (Algorithm 1), an effective online learning framework for (online) batched stochastic Bayesian optimization. Our framework relies on a novel discrete optimization subroutine, namely, DSOpt, which aims to maximize the expected reward for each batched experiment. At each iteration, Online-DSOpt uses a GP trained on previously-observed items to compute the reward for each item $x \in Q(C)$ (cf., Line 3) and then invokes DSOpt to select constraints. Pseudocode for DSOpt is presented in Algorithm 2. A batch of items is then sampled stochastically from the resulting library and used to update the GP.

A key component of the DS optimization subroutine DSOpt is a DS decomposition of the objective. Note that in general, finding a DS decomposition of an arbitrary set function requires searching through a combinatorial space and can be computationally prohibitive. As one of our main contributions, we present two polynomial time algorithms, DSConstruct-SA (Algorithm 3) and DSConstruct-DC (Algorithm 4), for decomposing our surrogate objective. Both algorithms exploit the structure of the objective function: DSConstruct-SA decomposes the objective via submodular augmentation [2]; DSConstruct-DC decomposes the objective via a difference of convex functions (DC) decomposition.

DS Optimization

After it obtains the submodular decomposition $\hat{F} = -(h - g)$ from either of the DS construction procedures, DSOpt (Algorithm 2 proceeds to optimize the DS function via an iterative greedy algorithm. For example, let us consider running the

Algorithm 2: DS Optimization (DSOpt)

```
1  Input: Constraints set $C = \bigcup_{\ell=1}^{L} C^{(\ell)}$; reward matrix $M = \{f(x)\}$; budget on each
     batch $n$
   begin
2      Set up $\hat{F}$ from the inputs $(M, n)$
       /* Decompose $\hat{F}$ into diff of submod funcs.                        */
3      $h, g \leftarrow$ DSConstruct-SA$(\hat{F}, C)$
       (or $h, g \leftarrow$ DSConstruct-DC$(\hat{F}, C)$)
       /* Initiliaze the starting position                                      */
4      $\mathcal{S}_{\text{cand}} \leftarrow \emptyset$
5      $\mathcal{S} \leftarrow$ Init$(C)$
       /* Optimize $h - g$ using ModMod or SupSub.                              */
       while $\mathcal{S}$ not converged do
           /* Keep track of local search solutions                              */
6          $\mathcal{S}_{\text{cand}} \leftarrow \mathcal{S}_{\text{cand}} \cup$ LocalSearch$(\hat{F}, \mathcal{S}, C)$
           /* Make a greedy move from $\mathcal{S}$                             */
7          $\mathcal{S} \leftarrow$ ModMod$(h - g, \mathcal{S}, C)$
           (or $\mathcal{S} \leftarrow$ SupSub$(h - g, \mathcal{S}, C)$)
8      $\mathcal{S}_{\text{cand}} \leftarrow \mathcal{S}_{\text{cand}} \cup \{\mathcal{S}\}$
       /* pick the best among candidate solutions                               */
9      $\mathcal{S}^* \leftarrow \arg\min_{\mathcal{S} \in \mathcal{S}_{\text{cand}}} \{\hat{F}(\mathcal{S})\}$
10     Output: Set of selected constraints $\mathcal{S}^*$
```

Modular-modular procedure (ModMod) [20] for making a greedy move at Line 7 of Algorithm 2. Since our goal is to maximize $-(h-g)$ (i.e., to minimize $h-g$), we will seek to minimize the upper bound on $h-g$. The ModMod procedure constructs a modular upper bound on the first submodular component, denoted by $\mathsf{ub}^h \geq h$, and a modular lower bound on the second submodular component, denoted by $\mathsf{lb}^g \leq g$. Both modular bounds are tight at the current solution $\mathcal{S}$: $\mathsf{ub}^h(\mathcal{S}) = h(\mathcal{S})$, $\mathsf{lb}^g(\mathcal{S}) = g(\mathcal{S})$. ModMod then tries to solve the following optimization problem, starting from $\mathcal{S}$:

$$\mathcal{S}^* \in \arg\min_{\mathcal{S}} \left(\mathsf{ub}^h(\mathcal{S}) - \mathsf{lb}^g(\mathcal{S})\right).$$

To ensure that we find a better solution, we augment the ModMod procedure with a sequence of additional local search solutions, and in the end pick the best among all. The local search procedure, LocalSearch (cf. Line 6 of Algorithm 2), sequentially makes greedy steps (by adding or removing a constraint from the current solution) until no further action is improving the current solution. The following theorem states that our DS optimization subroutine DSOpt is guaranteed to find a "good" solution:

Algorithm 3: DS Construction via Submodular Augmentation (DSConstruct-SA)

```
1 Input: Constraints set $C = \bigcup_{\ell=1}^{L} C^{(\ell)}$; surrogate objective function $\hat{F}$, budget on
    each batch $n$; selected constraints $\mathcal{S}$
  begin
2     $v(x) \leftarrow \sqrt{x}$ for $x \in \{1, \ldots, |C|\}$
3     $\alpha \leftarrow v(n-2) + v(n) - 2v(n-1)$
      /* compute $\beta'$ of Eq.(5.5)                                  */
      foreach $x \in \{1, \ldots, |Q(C)|\}$ do
4         $r_1(x) \leftarrow \left(1 - \frac{1}{s}\right)^n - \left(1 - \frac{1}{2s}\right)^n$
5         $r_2(x) \leftarrow \max_{\mathcal{T}:|\mathcal{T}|\leq s} \sum_{x\in\mathcal{T}} f(x)$
6     $\beta' \leftarrow -\max_x r_1(x) r_2(x)$
7     $h_1(\mathcal{S}) \leftarrow \frac{|\beta'|}{\alpha} v(|\mathcal{S}|)$
8     $g_1(\mathcal{S}) \leftarrow \hat{F}(\mathcal{S}) + \frac{|\beta'|}{\alpha} v(|\mathcal{S}|)$
9     Output: DS decomposition $\hat{F} = -(h_1 - g_1)$
```

Theorem 1 (Adapted from[20]) *Algorithm 2 is guaranteed to find a set of constraints that achieves a local maximum of* $\hat{F}$*.*

DS Construction via Submodular Augmentation

It is well-established that every set function can be expressed as the sum of a submodular and a supermodular function [2]. In particular,[20] provide the following constructive procedure for decomposing a set function into the DS form: Given a set function q, one can define $\beta = \min_{\mathcal{S}\subseteq\mathcal{S}'\subseteq C\setminus j} \Delta_q(j \mid \mathcal{S}) - \Delta_q(j \mid \mathcal{S}')$, where $\Delta_q(j \mid \mathcal{S}) := q(\mathcal{S} \cup \{j\}) - q(\mathcal{S})$ denotes the gain of adding j to $\mathcal{S}$. When q is not submodular, we know that $\beta < 0$. Now consider any strictly submodular function p, with $\alpha = \min_{\mathcal{S}\subseteq\mathcal{S}'\subseteq C\setminus j} \Delta_p(j \mid \mathcal{S}) - \Delta_p(j \mid \mathcal{S}') > 0$. Define $h(\mathcal{S}) = q(\mathcal{S}) + \frac{|\beta'|}{\alpha} p(\mathcal{S})$ for any $\beta' < \beta$. It is easy to verify that h is submodular since $\min_{\mathcal{S}\subseteq\mathcal{S}'\subseteq C\setminus j} \Delta_h(j \mid \mathcal{S}) - \Delta_h(j \mid \mathcal{S}') \geq \beta + |\beta'| \geq 0$. Hence $q(\mathcal{S}) = h(\mathcal{S}) - \frac{|\beta|}{\alpha} p(\mathcal{S})$ is a difference between two submodular functions.

We refer to the above decomposition strategy as DSConstruct-SA (where SA stands for "submodular augmentation"), and present the pseudo code in Algorithm 3. As suggested in [20], we choose the submodular augmentation function $p(\mathcal{S}) = v(|\mathcal{S}|)$, where $v(x)$ is a concave function, and therefore $\alpha = \min_{x\leq x';x,x'\subseteq\mathbb{Z}} v(x+1) - v(x) - v(x'+1) - v(x')$. This leads to the following decomposition of our surrogate objective

Algorithm 4: DS Construction via DC Decomposition (DSConstruct-DC)

```
1 Input: Constraints set $\prod_{\ell=1}^{L} C_\ell$; budget on each batch $n$; selected constraints $\mathcal{S}$
  begin
2     $u(x) \leftarrow \frac{x^2}{2}$, $\alpha \leftarrow 1$
3     $r(x) \leftarrow \left(1 - \frac{1}{x}\right)^n$,
4     $\beta \leftarrow |\min_x r''(x)|$ for $x \in \{1, \dots, |Q(C)|\}$.
5     $h_2(\mathcal{S}) \leftarrow -\left(1 + \frac{\beta}{\alpha} u(Q(|\mathcal{S}|))\right) \sum_{x \in Q(\mathcal{S})} f(x)$
6     $g_2(\mathcal{S}) \leftarrow -\left(r(|Q(\mathcal{S})|) + \frac{\beta}{\alpha} u(Q(|\mathcal{S}|))\right) \sum_{x \in Q(\mathcal{S})} f(x)$
7     Output: DS decomposition $\hat{F} = -(h_2 - g_2)$
```

$\hat{F}$:

$$\hat{F}(\mathcal{S}) = \underbrace{\left(\hat{F}(\mathcal{S}) + \frac{|\beta'|}{\alpha} v(|\mathcal{S}|)\right)}_{g_1(\mathcal{S})} - \underbrace{\frac{|\beta'|}{\alpha} v(|\mathcal{S}|)}_{h_1(\mathcal{S})} \tag{5.4}$$

where h_1, g_1 by construction are submodular functions, and β' is a lower bound on β:

$$\beta' \le \beta = \min_{\mathcal{S} \subseteq \mathcal{S}' \subseteq C \setminus j} \Delta_{\hat{F}}(j \mid \mathcal{S}) - \Delta_{\hat{F}}(j \mid \mathcal{S}'). \tag{5.5}$$

The key step of the DSConstruct-SA algorithm is to construct such a lower bound β'. The following lemma, which is proved in the Appendix, shows that one can compute β' in polynomial time, and hence can efficiently express $\hat{F}$ as a DS as defined in Eq. (5.4).

Lemma 2 *Algorithm 3 returns a DS-decomposition of $\hat{F}$ in polynomial time.*

DS Construction via DC Decomposition

We now consider an alternative strategy for decomposing the surrogate function $\hat{F}$, based on a novel construction procedure that reduces to expressing a continuous function as the difference of convex (DC) functions. Concretely, we note that $\hat{F}(\mathcal{S})$ consists of two (multiplicative) terms: (i) a supermodular set function $\sum_{x \in Q(\mathcal{S})} f(x)$, and (ii) a set function that only depends on the cardinality of the input, i.e., $\sum_{x \in Q(\mathcal{S})} f(x) \left(1 - \left(1 - \frac{1}{|Q(\mathcal{S})|}\right)^n\right)$. As is further discussed in the Appendix, we show that one can exploit this structure, and focus on the DC decomposition of term (ii). We provide the detailed algorithm in Algorithm 4, and refer to it as DSConstruct-DC (where DC stands for "difference of convex decomposition").

It is easy to check from Algorithm 3 that DSConstruct-SA runs in quadratic time w.r.t. $|Q(C)|$. In contrast, DSConstruct-DC only requires finding the minimum of an array of size $|Q(C)|$, which, in the best case, runs in linear time w.r.t. $|Q(C)|$. At the end of the algorithm, DSConstruct-DC outputs the following DS function:

$$\hat{F}(\mathcal{S}) = \underbrace{\sum_{x \in Q(\mathcal{S})} (-\rho(x)) \cdot \left(r(|Q(\mathcal{S})|) + \frac{\beta}{\alpha} u(Q(|\mathcal{S}|)) \right)}_{g_2(\mathcal{S})} - \underbrace{\sum_{x \in Q(\mathcal{S})} (-\rho(x)) \left(1 + \frac{\beta}{\alpha} u(Q(|\mathcal{S}|)) \right)}_{h_2(\mathcal{S})} \tag{5.6}$$

where $u(x)$ is a non-negative, monotone convex function[2], $\alpha = \min_x u''(x)$, $r(x) = \left(1 - \frac{1}{x}\right)^n$, and $\beta = |\min_x r''(x)|$. We then prove the following results:

Lemma 3 *With the decomposition as defined in Eq.* (5.6)*, both functions h, g are submodular and hence we obtain a DS-decomposition of* $\hat{F}$.

5.5 Experiments

In this section, we empirically evaluate our algorithm on both synthetic and real protein datasets. First, we compare DSOpt against two intuitive greedy heuristics on synthetic datasets (§5.5). We then justify the choice of our surrogate objective (Eq. (5.2)) with numerical simulations on real protein datasets in §5.5, and demonstrate the performance of our batched online optimization algorithm Online-DSOpt in §5.5.

Synthetic Examples for Batched Optimization

Baseline Algorithms

For simplicity, we refer to the two versions of DSOpt as DSOpt-SA (i.e., DSOpt with subroutine DSConstruct-SA) and DSOpt-DC (i.e., DSOpt with subroutine DSConstruct-DC), respectively. We compare against two intuitive greedy search heuristics: Greedy-Add, which greedily adds constraints and Greedy-Rem, which greedily removes constraints until the objective stops improving. Because the empty set is a local optimum (adding any single constraint still results in no valid queries), it is necessary to begin the optimization at a set of constraints that yields a non-empty set of queries.

[2]For example, in practice we set $u(x) = \frac{x^2}{2}$ and thus $\alpha = 1$.

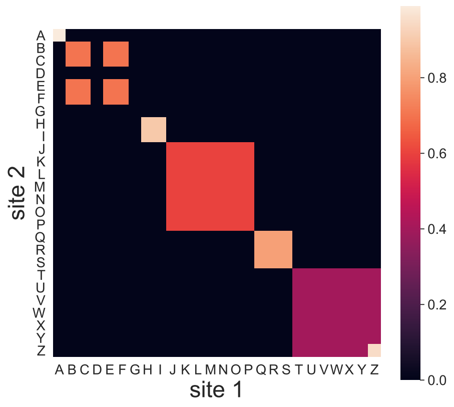


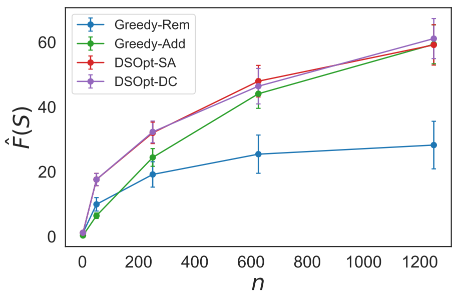
Figure 5.3: The cell values for the synthetic dataset with $L = 2$ and $|C^{(\ell)}| = 26\ \forall \ell \in \{1, 2\}$.

Synthetic Datasets

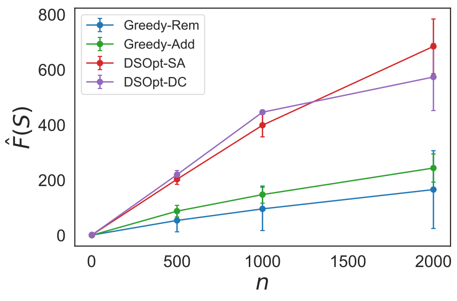
To demonstrate the performance of DSOpt under various configurations, we evaluate the algorithms on two synthetic datasets designed to have multiple local minima. As shown in Fig. 5.3, our first synthetic dataset consists of two sites with $|C^{(1)}| = |C^{(2)}| = 26$. Values for the items in the library are constructed such that there are disjoint blocks of items with non-zero $\rho(x)$ separated by regions where $\rho(x) = 0$. This guarantees that there are multiple local optima in the constraint space. Similarly, we create a second synthetic dataset with $L = 15$ and $|C^{(\ell)}| = 2\ \forall \ell \in [L]$. The library is structured such that only cells representing subsequences containing specific substrings have non-zero values. Therefore, it is likely that the dataset contains many local optima, which makes it challenging for finding the optimal set of constraints.

Results on Synthetic Datasets

We compare the algorithms across a range of batch sizes n. At each batch size, we initialize each algorithm at C, the constraints that result in the single best query, and 18 randomly selected sets of constraints. Fig. 5.4a shows the results for the compared algorithms on the first dataset ($L = 2, |C^{(\ell)}| = 26$), when we vary $n \in [0, 1200]$, and Fig. 5.4b shows the results on the second dataset ($L = 15, |C^{(\ell)}| = 2$) for $n \in [0, 2000]$. We observe that DSOpt-SA and DSOpt-DC consistently outperform Greedy-Add and Greedy-Rem across all values of n, under both dataset configurations. At small n, the optima tend to have few constraints, so Greedy-Add performs particularly poorly. As n approaches infinity, the optimum



(a) $L = 2$, $|C^{(\ell)}| = 26\ \forall \ell$



(b) $L = 15$, $|C^{(\ell)}| = 2\ \forall \ell$

Figure 5.4: Performance of each algorithm on finding constraints on synthetic datasets. Error bars are standard errors. $\hat{F}$ is the approximate objective, and n is the batch size.

approaches the ground set C, and so Greedy-Rem performs particularly poorly. DSOpt-SA and DSOpt-DC perform very similarly across all values of n. In theory, DSOpt-SA and DSOpt-DC can escape local optima to find better solutions than the local search algorithm (although this appears to be rare on our synthetic datasets).

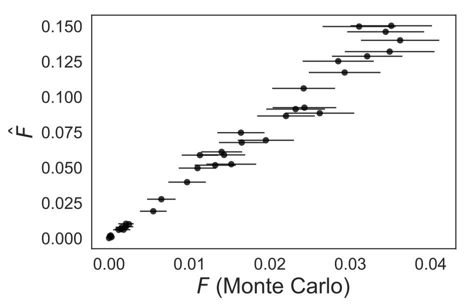
Real Experiments for Batched Online Optimization

Protein Datasets

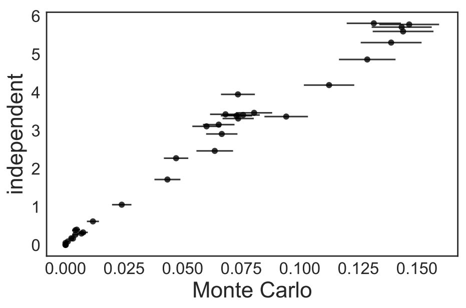
We further evaluate our algorithms on two experimental protein-engineering datasets. The experimental datasets consist of measured fitness values for every sequence in four-site SSM libraries for protein G domain B1 (GB1) [21], an immunoglobulin binding protein, and the protein kinase PhoQ [22]. These fitness landscapes are known to be highly non-additive (mutations have different effects in combination than individually). Having measurements for every fitness value in each library allows us to simulate engineering via multiple rounds of SSM.

Suitability of the Surrogate Objective

Online-DSOpt uses a GP posterior to model the unobserved utilities. However, there is no closed form for the true batch constraint design objective F as defined in Eq. (5.1), which is to choose constraints $\mathcal{S}$ that maximize the expected number of improved observations found by querying $Q(\mathcal{S})$. The surrogate objective function $\hat{F}$ (Eq. (5.2)) ignores dependencies between items, and thus will overestimate the true objective. To test the suitability of the surrogate objective function, we selected an initial batch of sequences from each of the protein datasets consisting of all the single mutants plus 100 randomly-selected sequences, trained a GP regression model, and



(a) $\hat{F}$ vs. F for GB1



(b) $\hat{F}$ vs. F for PhoQ

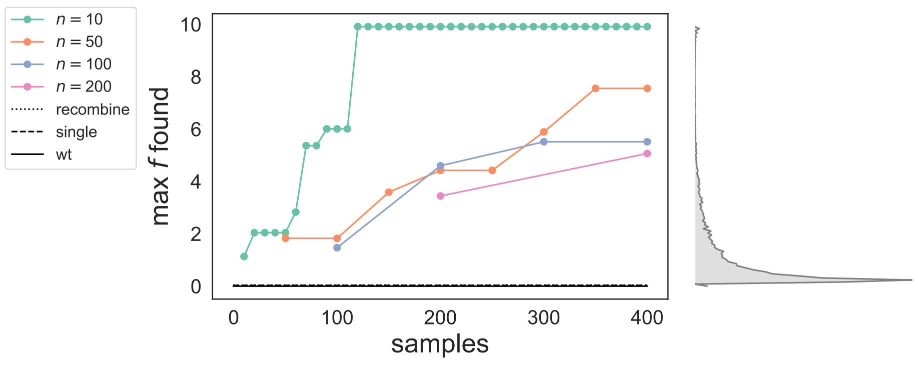
Figure 5.5: Comparing $\hat{F}$ (Eq. (5.2)) against the Monte Carlo estimates of F (Eq. (5.1)). Error bars are standard errors for the Monte Carlo estimates. The approximate objective correlates well with Monte Carlo estimates of the exact objective.

used the posterior to compute the rewards $\rho(x)$. In Fig. 5.5a and Fig. 5.5b, we plot the values of the surrogate objective function $\hat{F}$ (using $\rho(x)$ computed from the previous step) against the Monte Carlo estimates of the true objective values F for the two protein datasets. As can be observed from both plots, even though the independence assumption leads to overestimating the number of improved sequences that will be found, the values for the surrogate are well-correlated with the true objective, and hence one can use $\hat{F}$ as a proxy to identify protein sequences of high fitness values.

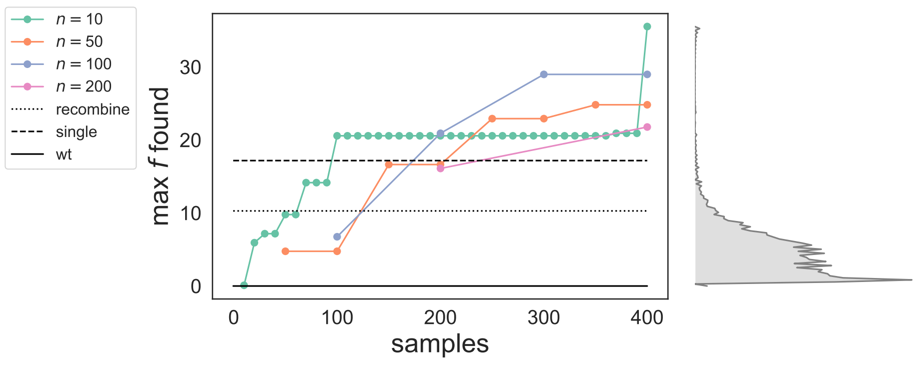
Results on Real Protein Datasets

We test Online-DSOpt on the PhoQ and GB1 datasets to demonstrate its ability to select constraints that result in libraries enriched in improved sequences. For both PhoQ and GB1, we evaluate Online-DSOpt by varying the batch sizes $n \in \{10, 50, 100, 200\}$. For each batch size n, we initiated the experiments by selecting an initial batch of n randomly-selected sequences. We then ran $k = 400/n - 1$ iterations of the algorithm with batch size n, resulting in k more batched queries. This simulates an SSM experiment with k rounds of diversification, screening, and selection. At each iteration, we train a GP regression model using a Matérn kernel with $\nu = \frac{5}{2}$ in order to compute the rewards $\rho(x)$. Results for each experiment are shown in Fig. 5.6.

For both GB1 and PhoQ, Online-DSOpt finds improved sequences. Importantly, it finds much better sequences than combining the best mutation at each site in the wild-type background or the best sequence with a single mutation from the wild-type, as shown in Fig. 5.6a and Fig. 5.6b. These are common experimental heuristics



(a) GB1

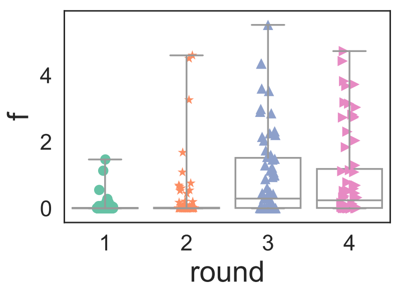


(b) PhoQ

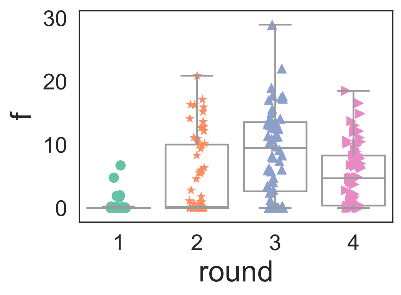
Figure 5.6: Results on (a) GB1 and (b) PhoQ when running Online-DSOpt with different batch sizes, with a total budget of 400 samples. In the left plots, each colored line corresponds to a different batch size, and every marker corresponds to the maximal value of f found at the end of that batch. The solid horizontal lines show the fitnesses for the wild-type sequences (wt) as the baseline of the batched experiments. The dashed horizontal lines show the fitnesses for the best sequences with exactly one mutation from the wild type (single). The dotted horizontal lines show the fitnesses for the sequences that combine the best amino acid at each site determined in the wild-type background (recombine). The plots on the right show the probability mass of each fitness value in the corresponding dataset.

for dealing with multi-site SSM libraries where the library is too large to reasonably screen.

Our results imply that in comples landscapes such as for GB1 and PhoQ, considering multiple sites simultaneously is necessary to escape local optima in the sequence-function landscape. In GB1, only looking at single mutations or recombining the best mutations at each site results in very poor fitnesses with no improvement over the wild-type. In PhoQ, the best single mutant (the dashed line) has a higher fitness than recombining the best mutations at each site (the dotted line).



(a) GB1



(b) PhoQ

Figure 5.7: Fitness values f for the sampled protein sequences when running Online-DSOpt over four rounds. Each round contains 100 samples ($n = 100$). Different rounds are represented with different colors, and each point represents a protein sequence sampled in the corresponding round.

In contrast, the probability mass function plots in Fig. 5.6a and Fig. 5.6b show that Online-DSOpt finds extremely-rare ($> 99.9^{\text{th}}$ percentile) sequences that would be extremely difficult to find by randomly sampling < 500 sequences from the entire library. Perhaps surprisingly, Online-DSOpt is even able to find the *optimal* protein sequence on both the datasets for small batch size $n = 10$ over 40 rounds batched queries. Note that in practice it is desirable to keep a low number of rounds. Fig. 5.7 illustrates the sampled protein sequences when running Online-DSOpt under a more practical setting (with batch size $n = 100$, which is approximately the number of samples that fit on a 96-well plate). Online-DSOpt significantly reduces the library size for a multi-site SSM library in order to increase the probability of finding sequences with improved fitness values.

5.6 Conclusion

In this paper, we investigated a novel Bayesian optimization problem: batched stochastic Bayesian optimization. This problem setting poses two unique challenges: optimizing over the space of constraints instead of directly over items and stochastic sampling. We proposed an effective online optimization framework for searching through the combinatorial design space of constraints in order to maximize the expected number of improved items sampled at each iteration. In particular, we proposed a novel approximate objective function that links a model trained on the individual items to the constraint space and derived two efficient DS decompositions for this objective. Our method efficiently finds sequences with improved fitnesses in fully-characterized SSM libraries for the proteins GB1 and PhoQ, demonstrating its potential to enable engineering via simultaneous SSM even in cases where it is not feasible to measure more than a tiny fraction of the sequences in the library.

Acknowledgments

This work was supported in part by the Donna and Benjamin M. Rosen Bioengineering Center, NSF Award #1645832, Northrop Grumman, Bloomberg, and a Swiss NSF Early Mobility Postdoctoral Fellowship.

References

1. Manfred T Reetz, Li-Wen Wang, and Marco Bocola: Directed evolution of enantioselective enzymes: iterative cycles of CASTing for probing protein-sequence space. *Angew Chem Int Ed Engl* **118**(8) (2006), 1258–1263. doi: 10.1002/anie.200502746 (cit. on p. 119).

2. Mukund Narasimhan and Jeff Bilmes: A submodular-supermodular procedure with applications to discriminative structure learning. *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*. AUAI Press. 2005, 404–412 (cit. on pp. 120, 121, 124, 126).

3. C. E. Rasmussen and C. K. I. Williams: *Gaussian Processes for Machine Learning*. MIT Press, 2006 (cit. on p. 120).

4. Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger: Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design. *Proc. International Conference on Machine Learning (ICML)*. 2010 (cit. on p. 120).

5. Zi Wang, Bolei Zhou, and Stefanie Jegelka: Optimization as estimation with Gaussian processes in bandit settings. *Artificial Intelligence and Statistics*. 2016, 1022–1031 (cit. on p. 120).

6. Thomas Desautels, Andreas Krause, and Joel W Burdick: Parallelizing exploration-exploitation tradeoffs in gaussian process bandit optimization. *The Journal of Machine Learning Research* **15**(1) (2014), 3873–3923 (cit. on p. 121).

7. Philip A Romero, Andreas Krause, and Frances H Arnold: Navigating the protein fitness landscape with Gaussian processes. *Proc Natl Acad Sci USA* **110**(3) (2013), E193–E201. doi: 10.1073/pnas.1215251110 (cit. on p. 121).

8. Claire N Bedbrook, Kevin K Yang, Austin J Rice, Viviana Gradinaru, and Frances H Arnold: Machine learning to design integral membrane channelrhodopsins for efficient eukaryotic expression and plasma membrane localization. *PLoS Comput Biol* **13**(10) (2017), e1005786. doi: 10.1371/journal.pcbi.1005786 (cit. on p. 121).

9. Yutaka Saito et al.: Machine-Learning-Guided Mutagenesis for Directed Evolution of Fluorescent Proteins. *ACS synthetic biology* (2018). doi: 10.1021/acssynbio.8b00155 (cit. on p. 121).

10. Mu Li, Tong Zhang, Yuqiang Chen, and Alexander J Smola: Efficient mini-batch training for stochastic optimization. *Proceedings of the 20th ACM SIGKDD*

international conference on Knowledge discovery and data mining. ACM. 2014, 661–670 (cit. on p. 121).

11. Martin Zinkevich, Markus Weimer, Lihong Li, and Alex J Smola: Parallelized stochastic gradient descent. *Advances in neural information processing systems*. 2010, 2595–2603 (cit. on p. 121).

12. Steven C. H. Hoi, Rong Jin, Jianke Zhu, and Michael R. Lyu: Batch mode active learning and its application to medical image classification. *Proceedings of the 23rd international conference on Machine learning*. ICML ’06. Pittsburgh, Pennsylvania: ACM, 2006, 417–424 (cit. on p. 121).

13. Andrew Guillory and Jeff A Bilmes: Interactive Submodular Set Cover. *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*. 2010, 415–422 (cit. on p. 121).

14. Yuxin Chen and Andreas Krause: Near-optimal Batch Mode Active Learning and Adaptive Submodular Optimization. *International Conference on Machine Learning (ICML)*. 2013 (cit. on p. 121).

15. Liu Yang and Jaime Carbonell: Buy-in-bulk active learning. *Advances in Neural Information Processing Systems*. 2013, 2229–2237 (cit. on p. 121).

16. George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher: An analysis of approximations for maximizing submodular set functions—I. *Mathematical programming* **14**(1) (1978), 265–294 (cit. on p. 121).

17. S Jegelka and J Bilmes: Submodularity beyond submodular energies: coupling edges in graph cuts. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2011)*. IEEE. 2011, 1897–1904 (cit. on p. 121).

18. David Kempe, Jon Kleinberg, and Éva Tardos: Maximizing the spread of influence through a social network. *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2003, 137–146 (cit. on p. 121).

19. Andreas Krause and Carlos Guestrin: Near-optimal observation selection using submodular functions. *AAAI*. **7**. 2007, 1650–1654 (cit. on p. 121).

20. Rishabh Iyer and Jeff Bilmes: Algorithms for approximate minimization of the difference between submodular functions, with applications. *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*. AUAI Press. 2012, 407–417 (cit. on pp. 122, 125, 126).

21. Nicholas C Wu, Lei Dai, C Anders Olson, James O Lloyd-Smith, and Ren Sun: Adaptation in protein fitness landscapes is facilitated by indirect paths. *Elife* **5** (2016), e16965. doi: 10.7554/eLife.16965 (cit. on p. 130).

22. Anna I Podgornaia and Michael T Laub: Pervasive degeneracy and epistasis in a protein-protein interface. *Science* **347**(6222) (2015), 673–677. doi: 10.1126/science.1257360 (cit. on p. 130).

Proofs

Lemma 4 *Let $g(\mathcal{S}) = \sum_{x \in Q(\mathcal{S})} f(x)$, where $Q(\mathcal{S})$ is defined in Eq.* (5.3). *If $\forall x,\ f(x) \geq 0$, then g is monotone supermodular.*

Proof Let $\ell \in [L]$, and $j \in \mathcal{C}^{(\ell)}$ be any constraint at site ℓ. For $\mathcal{S} \subseteq \mathcal{C} \setminus \{j\}$, define $\Delta_g(j \mid \mathcal{S}) = \sum_{x \in Q(\mathcal{S} \cup \{j\})} f(x) - \sum_{x \in Q(\mathcal{S})} f(x)$ to be the gain of adding j to the set $\mathcal{S}$. By definition of $Q(\mathcal{S})$, we have $Q(\mathcal{S}) = \prod_{k=1}^{L} \mathcal{S}^{(k)}$, and

$$\begin{aligned} Q(\mathcal{S} \cup \{j\}) &= \left(\mathcal{S}^{(\ell)} \cup \{j\}\right) \times \prod_{k \neq \ell} \mathcal{S}^{(k)} \\ &= \left(\{j\} \times \prod_{k \neq \ell} \mathcal{S}^{(k)}\right) \bigcup \left(\mathcal{S}^{(\ell)} \times \prod_{k \neq \ell} \mathcal{S}^{(k)}\right) \\ &= \left(\{j\} \times \prod_{k \neq \ell} \mathcal{S}^{(k)}\right) \bigcup \left(\prod_{k=1}^{L} \mathcal{S}^{(k)}\right) \end{aligned} \tag{5.7}$$

Then,

$$\Delta_g(j \mid \mathcal{S}) = \sum_{x \in Q(\mathcal{S} \cup \{j\})} f(x) - \sum_{x \in Q(\mathcal{S})} f(x) \overset{Eq.\ (5.7)}{=} \sum_{x \in \{j\} \times \prod_{k \neq \ell} \mathcal{S}^{(k)}} f(x)$$

Now let us consider $\mathcal{S}'$ such that $\mathcal{S} \subseteq \mathcal{S}' \subseteq \mathcal{C} \setminus \{j\}$. Clearly $\forall k \in [L],\ \mathcal{S}^{(k)} \subseteq \mathcal{S}'^{(k)}$. Therefore, $\Delta_g(j \mid \mathcal{S}') - \Delta_g(j \mid \mathcal{S}) = \sum_{x \in \{j\} \times \prod_{k \neq \ell}(\mathcal{S}'^{(k)} \setminus \mathcal{S}^{(k)})} f(x) \geq 0$ and hence g is supermodular. ■

Proof of Lemma 2

We now show that Algorithm 3 leads to a polynomial algorithm for constructing a lower bound on Eq. (5.5), and hence on constructing a DS-decomposition of the surrogate objective function $\hat{F}$ (Eq. (5.2)).

Proof [Proof of Lemma 2] Let $g(\mathcal{S}) = \sum_{x \in Q(\mathcal{S})} f(x)$. By definition we have

$$\hat{F}(\mathcal{S}) = g(\mathcal{S}) \left(1 - \left(1 - \frac{1}{|Q(\mathcal{S})|}\right)^n\right) = \underbrace{g(\mathcal{S})}_{\hat{F}_1(\mathcal{S})} - \underbrace{g(\mathcal{S}) \left(1 - \frac{1}{|Q(\mathcal{S})|}\right)^n}_{\hat{F}_2(\mathcal{S})} = \hat{F}_1(\mathcal{S}) - \hat{F}_2(\mathcal{S})$$

We know from Lemma 4 that $\hat{F}_1$ is supermodular. Let $j \in \mathcal{C}$ and $\mathcal{S} \subseteq \mathcal{C} \setminus \{j\}$. The gain of j on $\hat{F}_1$, denote by $\Delta_1(j \mid \mathcal{S})$, is monotone decreasing.

Let $\Delta_2(j \mid \mathcal{S}) = \hat{F}_2(\mathcal{S} \cup \{j\}) - \hat{F}_2(\mathcal{S})$. Our goal is to find a lower bound on

$$\begin{aligned}
\beta &= \min_{\mathcal{S} \subseteq \mathcal{S}' \subseteq C \setminus j} \left(\Delta_{\hat{F}}(j \mid \mathcal{S}) - \Delta_{\hat{F}}(j \mid \mathcal{S}')\right) \\
&= \min_{\mathcal{S} \subseteq \mathcal{S}' \subseteq C \setminus j} \left(\underbrace{\Delta_1(j \mid \mathcal{S}) - \Delta_1(j \mid \mathcal{S}')}_{\geq 0} + \Delta_2(j \mid \mathcal{S}) - \Delta_2(j \mid \mathcal{S}') \right) \qquad (5.8)
\end{aligned}$$

Therefore, it suffices to find a lower bound $\Delta_2(j \mid \mathcal{S}) - \Delta_2(j \mid \mathcal{S}')$. The gain of j on $\hat{F}_2$ is

$$\begin{aligned}
\Delta_2(j \mid \mathcal{S}) &= \hat{F}_2(\mathcal{S} \cup \{j\}) - \hat{F}_2(\mathcal{S}) \\
&= \sum_{x \in Q(\mathcal{S} \cup \{j\})} f(x) \left(1 - \frac{1}{|Q(\mathcal{S} \cup \{j\})|}\right)^n - \sum_{x \in Q(\mathcal{S})} f(x) \left(1 - \frac{1}{|Q(\mathcal{S})|}\right)^n \\
&= \sum_{x \in Q(\mathcal{S} \cup \{j\}) \setminus Q(\mathcal{S})} f(x) \left(1 - \frac{1}{|Q(\mathcal{S} \cup \{j\})|}\right)^n + \\
&\qquad \sum_{x \in Q(\mathcal{S})} f(x) \left(\left(1 - \frac{1}{|Q(\mathcal{S} \cup \{j\})|}\right)^n - \left(1 - \frac{1}{|Q(\mathcal{S})|}\right)^n \right)
\end{aligned}$$

Let $r(\mathcal{S}) = \left(1 - \frac{1}{|Q(\mathcal{S})|}\right)^n$. Then, the above equation can be simplified as

$$\begin{aligned}
\Delta_2(j \mid \mathcal{S}) &= \hat{F}_2(\mathcal{S} \cup \{j\}) - \hat{F}_2(\mathcal{S}) \\
&= \underbrace{\sum_{x \in Q(\mathcal{S} \cup \{j\}) \setminus Q(\mathcal{S})} f(x) r(\mathcal{S} \cup \{j\})}_{T_1(\mathcal{S})} + \underbrace{\sum_{x \in Q(\mathcal{S})} f(x) \left(r(\mathcal{S} \cup \{j\}) - r(\mathcal{S})\right)}_{T_2(\mathcal{S})}
\end{aligned}$$

It is easy to verify that $T_1(\mathcal{S})$ is monotone increasing function of $\mathcal{S}$. Let us consider $\mathcal{S}'$ such that $\mathcal{S} \subseteq \mathcal{S}' \subseteq C \setminus \{j\}$. We have

$$\begin{aligned}
\Delta_2(j \mid \mathcal{S}') - \Delta_2(j \mid \mathcal{S}) &\geq T_2(\mathcal{S}') - T_2(\mathcal{S}) \\
&\overset{T_2 \geq 0}{\geq} -g(\mathcal{S})(r(\mathcal{S} \cup \{j\}) - r(\mathcal{S}))
\end{aligned}$$

Therefore, it suffices to find a lower bound on $-g(\mathcal{S})(r(\mathcal{S} \cup \{j\}) - r(\mathcal{S}))$. Further notice that

$$0 \leq g(\mathcal{S}) \leq \max_{\mathcal{T} : |\mathcal{T}| \leq |Q(\mathcal{S})|} \sum_{x \in \mathcal{T}} f(x) \qquad (5.9)$$

and it is not hard to verify that

$$0 \leq r(\mathcal{S} \cup \{j\}) - r(\mathcal{S}) \leq \left(1 - \frac{1}{|Q(\mathcal{S})|}\right)^n - \left(1 - \frac{1}{2|Q(\mathcal{S})|}\right)^n \qquad (5.10)$$

Therefore, combining term (5.9) with (5.10), we get a lower bound on β:

$$\beta \geq -\max_{s\in\{1,\ldots,|Q(C)|\}} \left(\left(\left(1-\frac{1}{s}\right)^n - \left(1-\frac{1}{2s}\right)^n \right) \underbrace{\max_{\mathcal{T}:|\mathcal{T}|\leq s} \sum_{x\in\mathcal{T}} f(x)}_{\text{Term 2}} \right) \tag{5.11}$$

Note that term 2 is a modular function and can be optimized greedily. Therefore, computing the RHS of Eq. 5.11 can be efficiently done in polynomial time w.r.t. $|Q(C)|$. ■

Proof of Lemma 3: Difference of Convex Construction of DS Decomposition

Lemma 5 *Let $g : 2^C \to \mathbb{R}_{\geq 0}$ be a non-negative, non-decreasing supermodular function, and $u : \mathbb{R} \to \mathbb{R}$ be a non-decreasing convex function. For $\mathcal{S} \subseteq C$, define $h(\mathcal{S}) = g(\mathcal{S}) \cdot u(|\mathcal{S}|)$. Then h is supermodular.*

Proof Let $j \in C$ and $\mathcal{S} \subseteq C \setminus \{j\}$. The gain of j is

$$\begin{aligned}\Delta_h(j \mid \mathcal{S}) &= h(\mathcal{S} \cup \{j\}) - h(\mathcal{S}) \\ &= g(\mathcal{S} \cup \{j\}) \cdot u(|\mathcal{S} \cup \{j\}|) - g(\mathcal{S}) \cdot u(|\mathcal{S}|) \\ &= (g(\mathcal{S} \cup \{j\}) - g(\mathcal{S})) \cdot u(|\mathcal{S} \cup \{j\}|) + g(\mathcal{S})\,(u(|\mathcal{S} \cup \{j\}|) - u(|\mathcal{S}|))\end{aligned}$$

Let us consider $\mathcal{S}'$ such that $\mathcal{S} \subseteq \mathcal{S}' \subseteq C \setminus \{j\}$. We have

$$\begin{aligned}\Delta_h(j \mid \mathcal{S}) &= (g(\mathcal{S} \cup \{j\}) - g(\mathcal{S})) \cdot u(|\mathcal{S} \cup \{j\}|) + g(\mathcal{S})\,(u(|\mathcal{S} \cup \{j\}|) - u(|\mathcal{S}|)) \\ &\overset{(a)}{\leq} (g(\mathcal{S}' \cup \{j\}) - g(\mathcal{S}')) \cdot u(|\mathcal{S}' \cup \{j\}|) + g(\mathcal{S})\,(u(|\mathcal{S} \cup \{j\}|) - u(|\mathcal{S}|)) \\ &\overset{(b)}{\leq} (g(\mathcal{S}' \cup \{j\}) - g(\mathcal{S}')) \cdot u(|\mathcal{S}' \cup \{j\}|) + g(\mathcal{S}')\,(u(|\mathcal{S}' \cup \{j\}|) - u(|\mathcal{S}'|)) \\ &= \Delta_h(j \mid \mathcal{S}')\end{aligned}$$

where step (a) is due to g being monotone supermodular (i.e., $g(\mathcal{S}' \cup \{j\}) - g(\mathcal{S}') \geq g(\mathcal{S} \cup \{j\}) - g(\mathcal{S}) \geq 0$) and u being monotone (i.e., $u(|\mathcal{S}' \cup \{j\}|) \geq u(|\mathcal{S} \cup \{j\}|)$); step (b) is due to g being non-negative monotone (i.e., $g(\mathcal{S}') \geq g(\mathcal{S}) \geq 0$) and u being convex (i.e., $u(|\mathcal{S}' \cup \{j\}|) - u(|\mathcal{S}'|) \geq u(|\mathcal{S} \cup \{j\}|) - u(|\mathcal{S}|)$). Therefore h is supermodular. ■

Lemma 6 *Let $w : \mathbb{R} \to \mathbb{R}$ be a convex function and $u : \mathbb{R} \to \mathbb{R}$ a convex non-decreasing function, then $u \circ w$ is convex. Furthermore, if w is non-decreasing, then the composition is also non-decreasing.*

Proof By convexity of w:

$$w(\alpha x + (1-\alpha)y) \le \alpha w(x) + (1-\alpha)w(y).$$

Therefore, we get

$$\begin{aligned} u(w(\alpha x + (1-\alpha)y)) &\overset{(a)}{\le} u\left(\alpha w(x) + (1-\alpha)w(y)\right) \\ &\overset{(b)}{\le} \alpha u(w(x)) + (1-\alpha)u(w(y)). \end{aligned}$$

Here, step (a) is due to the fact that u is non-decreasing, and step (b) is due to the convexity of u. Therefore $u \circ w$ is convex. If w is non-decreasing, it is clear that $u \circ w$ is also non-decreasing, hence completes the proof. ■

Lemma 7 (Horst and Thoai [1]) *Let $r : \mathbb{R} \to \mathbb{R}$ be a non-decreasing, twice continuously differentiable function. Then r can be represented as the difference between two non-decreasing convex functions.*

Proof Let $u : \mathbb{R} \to \mathbb{R}$ be a non-decreasing, strictly convex function, and $\alpha = \min_x u''(x)$; clearly, $\alpha > 0$.

Let $\beta = |\min_x r''(x)|$. Define

$$v(x) = r(x) + \frac{\beta}{\alpha} u(x) \tag{5.12}$$

It is easy to verify that

$$v''(x) = r''(x) + \frac{\beta}{\alpha} u''(x) \ge r''(x) + \beta \ge 0.$$

Hence, $v(x)$ is convex. Furthermore, since both r and u are non-decreasing, v is also non-decreasing. Therefore, $r(x) = v(x) - \frac{\beta}{\alpha} u(x)$ is the difference between two non-decreasing convex functions. ■

Lemma 8 *Let $r : \mathbb{R} \to \mathbb{R}$ be a non-decreasing, twice continuously differentiable function, and $w : \mathbb{R} \to \mathbb{R}$ a convex non-decreasing function, then $r \circ w$ can be represented as the difference between two non-decreasing convex functions.*

Proof By Lemma 7, we can represent $r(x) = v(x) - \frac{\beta}{\alpha} u(x)$, where u, v are non-decreasing convex functions, and α, β are as defined in Eq. (5.12). Therefore,

$$r \circ w(x) = v \circ w(x) - \frac{\beta}{\alpha} \cdot u \circ w(x)$$

By Lemma 6, $v \circ w$ and $u \circ w$ are both non-decreasing convex, which completes the proof. ■

Now we are ready to prove Lemma 3.

Proof [Proof of Lemma 3] Let $g(\mathcal{S}) = \sum_{x \in Q(\mathcal{S})} f(x)$. By definition we have

$$\hat{F}(\mathcal{S}) = g(\mathcal{S}) \left(1 - \left(1 - \frac{1}{|Q(\mathcal{S})|}\right)^n\right) = g(\mathcal{S}) - g(\mathcal{S}) \left(1 - \frac{1}{|Q(\mathcal{S})|}\right)^n$$

Let $r(x) = \left(1 - \frac{1}{x}\right)^n$, and $w : \mathbb{R} \to \mathbb{R}$ be a convex function, such that $w(|\mathcal{S}|) = |Q(\mathcal{S})|$. Note that such function w exists, because the set function $h(\mathcal{S}) := |Q(\mathcal{S})|$ is supermodular. Therefore, we have

$$\hat{F}(\mathcal{S}) = g(\mathcal{S}) - g(\mathcal{S}) \cdot r \circ w(|\mathcal{S}|)$$

Furthermore, note that r is non-decreasing, twice continuously differentiable at $[1, \infty)$. By Lemma 8, we get

$$\begin{aligned} \hat{F}(\mathcal{S}) &= g(\mathcal{S}) - g(\mathcal{S}) \cdot \left(v \circ w(|\mathcal{S}|) - \frac{\beta}{\alpha} \cdot u \circ w(|\mathcal{S}|)\right) \\ &= g(\mathcal{S}) \left(1 + \frac{\beta}{\alpha} \cdot u \circ w(|\mathcal{S}|)\right) - g(\mathcal{S}) \cdot (v \circ w(|\mathcal{S}|)), \qquad (5.13) \end{aligned}$$

where $u : \mathbb{R} \to \mathbb{R}$ can be any non-decreasing, strictly convex function, $\alpha = \min_x u''(x)$, $\beta = |\min_{x \geq 1} r''(x)|$, and $v(x) = r(x) + \frac{\beta}{\alpha} u(x)$.

We know from Lemma 4 that g is supermodular. Since both $1 + \frac{\beta}{\alpha} \cdot u \circ w(x)$ and $v \circ w(x)$ are convex, then by Lemma 5, we know that both terms on the R.H.S. of Eq. (5.13) are supermodular, and hence we obtain a DS decomposition of function $\hat{F}$. ■

Chapter 6

LEARNED DECOMPOSITION KERNELS FOR SEQUENCE-FUNCTION PREDICTION

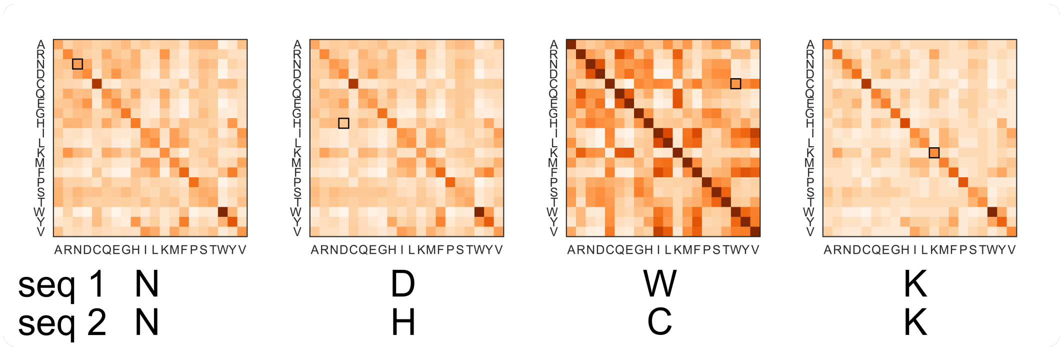
6.1 Introduction

Gaussian processes (GPs) are Bayesian non-parametric models widely applied across problem domains because of their flexibility and probabilistic interpretation. The central component of a GP is the kernel (covariance) function, which determines the similarity between inputs and projects the inputs into a high-dimensional feature space without explicitly calculating the coordinates in this new space. The choice of kernel profoundly affects the accuracy of Gaussian process models.

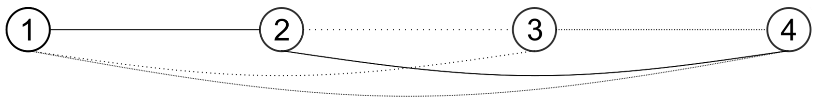
Recently, there has been much interest in using GPs to model biological sequences, such as the relationship between protein sequence and function[1–3]. Generic kernels, such as polynomial, squared exponential, or Matérn kernels, are most natural in continuous input spaces, but do not exploit the discrete string or graph properties present in structured inputs. Kernels tailored to exploit these properties should lead to predictive and interpretable GPs.

In many applications, it is also important to be able to interrogate and interpret the model, as the model itself can be a source of knowledge about the underlying physical or biological processes. The non-parametric nature of GPs, which allows them to grow in expressiveness with the data available, can make them very difficult to interpret unless the kernel is tailored to the input structure.

Variants of a sequence can be considered as strings where one or more positions have been substituted. It is therefore natural for a kernel to evaluate at each position the effect of the substitutions (if any) along with the effect of substitutions at contacting positions. The effects of substitutions can be encoded in one or more (positive definite) substitution matrices S^p, where $S_p^{a,b}$ denotes a penalty for substituting token a for token b. Substitution matrices can be compiled from empirical observations of substitution frequency across a corpus of related sequences, but these may not be well-suited to the desired prediction task. Likewise, the contact map between positions can be computed from 3-dimensional structural data[4]. However, structural data is not always available, and effects between positions in the sequence are not limited to positions in close physical proximity. We propose a learned decomposition



(a) Position-specific substitution matrices



(b) Soft pairwise contacts between positions

Figure 6.1: Example of a learned decomposition kernel between two sequences of length 4. (a) The position-specific substitution matrices at each position. The entry in each matrix corresponding to the tokens present is boxed. (b) Each position is fractionally in contact with each other position. The solidity of the edges denotes the strength of the contact.

kernel (LDK) which learns a substitution matrix S^p at each position p in the sequence. In addition, the LDK learns position-wise contact weights directly from data. These weights range from 0 to 1 and can be interpreted as describing how much two positions effect each other. The LDK is illustrated in Figure 6.1.

First, we review GP regression and discuss related work. We then describe the learned decomposition kernel. Finally, we show that LDKs can achieve comparable accuracy to deep kernel learning[5] on two protein mutation datasets and demonstrate that the learned kernel parameters can be visualized and interpreted to draw insights into which substitutions and positions the model considers to be important when comparing sequences.

6.2 Gaussian processes

We briefly review the posterior predictive equations and marginal likelihood for GP regression. A full treatment can be found in [6]. A Gaussian process is a distribution over functions. If $f(\mathbf{x}) \sim GP(\mu, k_\gamma)$, then any finite subset collection of function values has a joint Gaussian distribution

$$\mathbf{f}(X) = \{f(\mathbf{x}_1, ..., f(\mathbf{x}_n\}^T \sim \mathcal{N}\left(\mu, K_{X,X}\right) \tag{6.1}$$

with a mean vector $\mu_i = \mu(\mathbf{x}_i)$ and a covariance matrix $(K_{X,X})_{i,j} = k_\gamma\left(\mathbf{x}_i, \mathbf{x}_j\right)$ determined by the mean function and covariance kernel of the Gaussian process. The covariance, or kernel function, k_γ is parametrized by γ. Without loss of generality, we take the μ to be uniformly 0.

The goal in GP regression is to learn an unknown function $f : \mathbb{R}^D \mapsto \mathbb{R}$ given a dataset $\mathcal{D}$ consisting of n input vectors $X = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$ of dimension D corresponding to a $n \times 1$ vector $\mathbf{y}$ of targets. If we assume a Gaussian process prior and additive homoscedatic Gaussian noise, $y(\mathbf{x}) \sim \mathcal{N}(f(\mathbf{x}, \sigma_n))$, then the posterior predictive distribution at n_* text points X_* has mean

$$\mathbb{E}(\mathbf{f}_*) = K_{X_*,X}\left(K_{X,X} + \sigma_n^2 I\right)^{-1}\mathbf{y} \tag{6.2}$$

and covariance

$$\text{cov}(\mathbf{f}_*) = K_{X_*,X_*} - K_{X_*,X}\left(K_{X,X} + \sigma_n^2 I\right)^{-1} K_{X,X_*} \tag{6.3}$$

$K_{X,X}$ is the $n \times n$ covariance matrix calculated between the training inputs X, while $K_{X_*,X}$, for example, is the $n_* \times n$ covariance matrix between the test inputs X_* and the training inputs X. All the covariance matrices depend implicitly on the kernel hyperparameters γ. The structure of the data is modeled by choosing the kernel and learning γ. The marginal likelihood of the data $\mathcal{D}$, the probability of generating the data from a GP with the kernel specified by hyperparameters γ, provides a principled probabilistic framework for choosing hyperparameters:

$$\log p\,(\mathbf{y}|\gamma, X) \propto -\mathbf{y}^T(K_\gamma + \sigma_n^2 I)^{-1}\mathbf{y} - \log|K_\gamma + \sigma_n^2 I| \tag{6.4}$$

K_γ is shorthand for $K_{X,X}$ evaluated at γ. Learning is achieved by maximizing the marginal likelihood over γ and the noise hyperparameter σ_n.

6.3 Related work

Kernels for biological sequences

A variety of GP kernels have been previously used to predict functional properties from sequence, including linear[1], Matérn[2,7], and squared exponential[3]. These kernels make no effort to account for the string and graph structure of proteins. String spectrum kernels[8] count the number of matching substrings of length k between pairs of sequences, while mismatch string kernels[9] count the number of matching substrings of length k allowing for up to m mismatches. These take advantage of the string structure of biological sequences, but not their graph structure nor the fact that not all substitutions have equal effects.

Weighted decomposition kernel The weighted decomposition kernel compares two sequences by considering, at each position, the substitution (if any) at that position and at all positions considered to be its contacts. For two sequences of length L, the weighted decomposition (WDK)[10] is:

$$k(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{L} \left[S_{x_i, x_i'} \sum_{j \in \mathrm{nbs}(i)} S_{x_j, x_j'} \right] \tag{6.5}$$

Where x_i is the i^{th} token in sequence $\mathbf{x}$, S is a positive semi-definite substitution matrix such that $S_{a,b}$ is the penalty for replacing token a with token b, and $\mathrm{nbs}(i)$ denotes the set of positions in contact with position i. The WDK defines the similarity between two variants as the average position and neighborhood similarity over all positions. The kernel matrix is normalized such that

$$\hat{k}(\mathbf{x}, \mathbf{x}') = \frac{k(\mathbf{x}, \mathbf{x}')}{\sqrt{k(\mathbf{x}, \mathbf{x}) k(\mathbf{x}', \mathbf{x}')}} \tag{6.6}$$

Multiple substitution matrices can be combined using multiple kernel learning. For M kernels computed using M substitution matrices, the combined kernel matrix is

$$K_\phi = \sum_{m=1}^{M} w_m K_m \tag{6.7}$$

where w_m are learnable hyper-parameters.

WDKs take into account the 3-D structure of the inputs while also incorporating a substitution matrix that intuitively encodes the notion of similarity between pairs of tokens. However, they require a contact map for the sequences. Furthermore, their performance depends strongly on the choice of substitution matrices. Jokinen *et al.* use 21 substitution matrices from AAIndex2[11]. In contrast, the learned decomposition kernel does not require pre-set substitution matrices or structural information.

Deep kernel learning

Deep kernel learning (DKL) combines the expressive power of neural networks with the probabilistic GP framework. DKL composes a base kernel $k_b(\mathbf{x}_i, \mathbf{x}_j|\theta)$with a neural network embedding $g(\mathbf{x}, \mathbf{w})$:

$$k_{\text{DKL}}(\mathbf{x}_i, \mathbf{x}_j|\theta, \mathbf{w}) = k_b(g(\mathbf{x}_i, \mathbf{w}), g(\mathbf{x}_j, \mathbf{w})|\theta) \tag{6.8}$$

The neural network thus learns an embedded representation for the base kernel. The kernel parameters θ and $\mathbf{w}$ are jointly learned by maximizing the log marginal likelihood of the GP (Eq. 6.4). While DKL learns very expressive kernels that combine the advantages of neural networks and GPs, the kernels can be difficult or impossible to interpret.

6.4 Learned decomposition kernels

Instead of using a pre-determined substitution matrix and contact map derived from the structure, LDKs learn the substitution matrices and an adjacency matrix from the data. Specifically, an LDK has learnable hyperparameters $A \in \mathbb{R}^{L\times m\times d_A}$ and $W \in \mathbb{R}^{L(L-1)/2}$, where m is the number of tokens and L is the length of the sequences. The substitution matrix S_p for the p^{th} position is then calculated as

$$S^p = A^p A^{pT} \tag{6.9}$$

Defining $W_{i,j}$ as the contact weight between positions i and j, the LDK is

$$k(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{L} \left[S^i_{x_i,x'_i} \sum_{j\neq i} \text{sigmoid}(W_{i,j}) S^j_{x_j,x'_j} \right] \tag{6.10}$$

Thus, the LDK learns an embedded representation for each token at each position, which is then used to calculate the substitution matrices for each position. Instead of considering a predetermined set of contacts at each position, the LDK learns the degree to which each pair of positions is in contact. Figure 6.1 illustrates the LDK.

6.5 Experiments

We evaluated the performance of GPs with LDKs on two experimental protein datasets consisting of all the variants made by randomizing 4 positions in the sequence. We compare performance against a DKL model with a similar number of kernel parameters, and then we interpret the LDK parameters.

Table 6.1: Experimental results

Dataset	τ					MAE				
	LDK	C	P	WDK	DKL	LDK	C	P	WDK	DKL
PhoQ	0.625	0.623	0.625	0.623	0.618	0.595	0.620	0.600	0.623	0.582
GB1	0.547	0.543	0.546	0.452	0.550	0.465	0.468	0.467	0.471	0.450

LDK: learned decomposition kernel; C: learned pairwise contacts but only one learned substitution matrix; P: learned substitution matrix for each position but pre-set contacts; WDK: one learned substitution matrix and pre-set contacts; DKL: deep kernel learning

Datasets

The experimental datasets consist of function measurements for every sequence in four-site site-saturation libraries for protein G domain B1 (GB1)[12], an immunoglobulin binding protein, at positions 39, 40, 41, and 54, and the protein kinase PhoQ[13] at sites 284, 285, 288, and 289. These datasets are known to have strong higher-order effects: the effects of substitutions depends strongly on the context in which they occur. Each dataset contains measurements for 160,000 sequences. There were randomly divided into a training set of 96,000 sequence-function pairs, a validation set of 32,000 sequence-function pairs, and a test set of 32,000 sequence-function pairs.

Model comparisons

Table 6.1 compares Kendall's τ and mean absolute error (MAE) for GPs with learned decomposition kernels and deep kernels. All GP models were trained with stochastic batch gradient updates with batchsize 8000. The learning rates and number of epochs were tuned using accuracy on the validation set. The LDKs learn a 64-dimensional embedding for each amino acid at each position along with 6 pairwise contact weights for a total of 1286 parameters. The embedding dimension does not have a strong effect on the accuracy. This is expected, as the substitution matrix is full rank for any d greater than the number of tokens. For numerical stability, all of the embeddings were initialized so that the resulting substitution matrix fit the 'isomorphicity of replacements'[14] substitution matrix from AAIndex2[11]. The model is not sensitive to initialization to other substitution matrices from AAIndex2. The deep kernels consist of a neural network with layer sizes [80-16-8] (1432 parameters) and a squared exponential kernel. Following Wilson *et al.*[5], the neural networks were pretrained on the training data. For both Kendall's τ and MAE, GPs with LDKs are within 5% of GPs with DKL for both datasets.

In addition to comparing to DKL, we also compared to the following variants of LDKs: (1) a weighted decomposition kernel with one learned substitution matrix for all the positions and a pre-determined binary pairwise contact map (WDK); (2) an LDK that learns a contact map at each position but uses a pre-determined binary pairwise contact map (P), and (3) an LDK that learns one learned substitution matrix for all the positions but learned pairwise contact weights (C). For kernels that require a pre-determined binary contact map, cross-validation determined that setting all possible contacts to 'True' resulted in the most accurate predictions. This makes sense because the positions randomized were chosen to be in close proximity to each other. GPs with LDKs are more accurate than any of the control variants, showing that learning pairwise contact weights improves accuracy over using pre-set binary contacts and that having a substitution matrix for each position improves accuracy over having one shared substitution matrix.

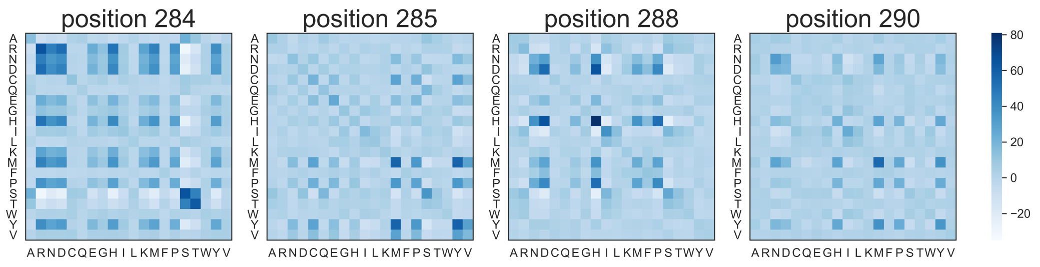
Visualization and interpretation

One advantage of the LDK is that both the learned substitution matrices and the learned pairwise contacts are interpretable. Figure 6.2 shows that although the substitution matrices for every position in PhoQ and GB1 were initialized to the same values (Figure 6.2e), the final learned matrices are qualitatively different from each other and from the initial values. Strikingly, the learned matrices generally penalize substitutions much less than the initial matrix; in these systems, the majority of possible tokens at each position either have small effects or effects that are highly-dependent on context.

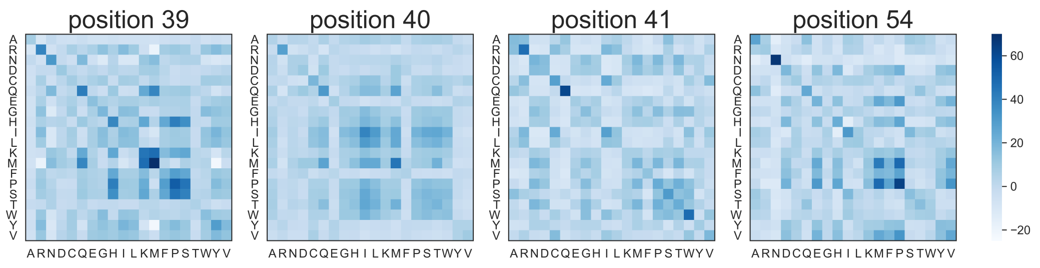
Figures 6.2c and 6.2d show the learned pairwise contacts for PhoQ and GB1, respectively. It has been previously found that the strongest coupling in PhoQ occurs between positions 284 and 285 and positions 284 and 288, likely due to physical constraints on packing. Positions 284 and 288 are separated by three residues in the primary sequence of PhoQ but are adjacent to each other within an α-helix[13]. In GB1, the four randomized positions were chosen because they are in close physical physical proximity in the core of the protein. The learned pairwise contacts reflect the close physical proximity between positions 39 and 40 as well as 41 and 54[12]. The LDK recapitulates this from sequence-function data alone.

6.6 Discussion

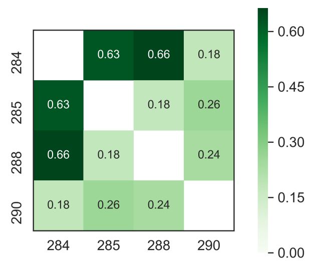
In this paper, we investigated a novel kernel for learning sequence-function relationships. Learned decomposition kernels are well-suited to describe differences



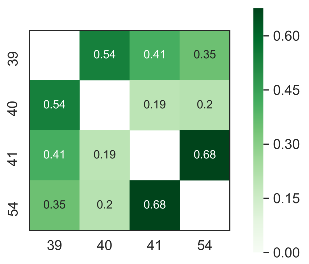
(a) Substitution matrices for PhoQ



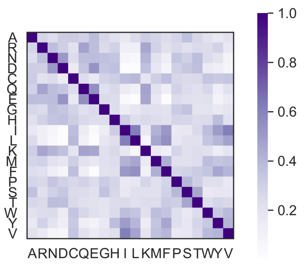
(b) Substitution matrices for GB1



(c) Contacts for PhoQ



(d) Contacts for GB1



(e) Initial substitutions

Figure 6.2: Learned substitution matrices and pairwise contacts for PhoQ and GB1.

between sequences of discrete tokens. By learning a substitution matrix at each position in addition to pairwise substitution weights, they combine expressiveness and interpretability. LDKs achieve comparable accuracy to deep kernel learning with a similar number of kernel parameters over two experimental protein datasets. Furthermore, the learned substitution matrices and pairwise contact weights can be visualized to interpret the resulting model. In theory, LDKs can be composed with neural networks. For example, a neural network can be used to learn position-specific substitution matrices for each pair of sequences from the sequences. Higher order effects could then be learned by the neural network instead of being coded into the base kernel. This approach may be able to combine the ability of neural networks to learn higher-order and hierarchical relationships with the interpretability of a substitution kernel.

References

1. Philip A Romero, Andreas Krause, and Frances H Arnold: Navigating the protein fitness landscape with Gaussian processes. *Proc Natl Acad Sci USA* **110**(3) (2013), E193–E201. doi: 10.1073/pnas.1215251110 (cit. on pp. 141, 143).
2. Claire N Bedbrook, Kevin K Yang, Austin J Rice, Viviana Gradinaru, and Frances H Arnold: Machine learning to design integral membrane channelrhodopsins for efficient eukaryotic expression and plasma membrane localization. *PLoS Comput Biol* **13**(10) (2017), e1005786. doi: 10.1371/journal.pcbi.1005786 (cit. on pp. 141, 143).
3. Yutaka Saito et al.: Machine-Learning-Guided Mutagenesis for Directed Evolution of Fluorescent Proteins. *ACS Synth Biol* (2018). doi: 10.1021/acssynbio.8b00155 (cit. on pp. 141, 143).
4. Philip A Romero, Andreas Krause, and Frances H Arnold: Navigating the protein fitness landscape with Gaussian processes. *Proc Natl Acad Sci USA* **110**(3) (2013). This is the first study to combine SCHEMA recombination with the GP-UCB algorithm to optimize a protein property., E193–E201. doi: 10.1073/pnas.1215251110 (cit. on p. 141).
5. Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing: Deep kernel learning. *Artificial Intelligence and Statistics*. 2016, 370–378 (cit. on pp. 142, 146).
6. C. E. Rasmussen and C. K. I. Williams: *Gaussian Processes for Machine Learning*. MIT Press, 2006 (cit. on p. 142).
7. Claire N Bedbrook, Kevin K Yang, Robinson Eliott J, Gradinaru Viviana, and Frances H Arnold: Machine learning-guided channelrhodopsin engineering enables minimally-invasive optogenetics. *In preparation* (2018) (cit. on p. 143).
8. Christina Leslie, Eleazar Eskin, and William Stafford Noble: The spectrum kernel: A string kernel for SVM protein classification. (2002), 564–575 (cit. on p. 143).
9. Christina S Leslie, Eleazar Eskin, Adiel Cohen, Jason Weston, and William Stafford Noble: Mismatch string kernels for discriminative protein classification. *Bioinformatics* **20**(4) (2004), 467–476 (cit. on p. 143).
10. Emmi Jokinen, Markus Heinonen, and Harri Lähdesmäki: mGPfusion: Predicting protein stability changes with Gaussian process kernel learning and data fusion. *Bioinformatics* **34**(13) (2018), i274–i283. doi: 10.1093/bioinformatics/bty238 (cit. on p. 144).
11. Shuichi Kawashima et al.: AAindex: amino acid index database, progress report 2008. *Nucleic Acids Res* **36**(suppl_1) (2007), D202–D205 (cit. on pp. 144, 146).
12. Nicholas C Wu, Lei Dai, C Anders Olson, James O Lloyd-Smith, and Ren Sun: Adaptation in protein fitness landscapes is facilitated by indirect paths. *Elife* **5** (2016), e16965. doi: 10.7554/eLife.16965 (cit. on pp. 146, 147).
13. Anna I Podgornaia and Michael T Laub: Pervasive degeneracy and epistasis in a protein-protein interface. *Science* **347**(6222) (2015), 673–677. doi: 10.1126/science.1257360 (cit. on pp. 146, 147).

14. E Tüdös, M Cserzö, and I Simon: Predicting isomorphic residue replacements for protein design. *International journal of peptide and protein research* **36**(3) (1990), 236–239 (cit. on p. 146).