Machine learning-guided directed evolution for protein engineering

April 23, 2019

Kevin K. Yang [1] **Zachary Wu** [1] **Frances H. Arnold** [1]

Abstract

Machine learning (ML)-guided directed evolution is a new paradigm for biological design that enables optimization of complex functions. ML methods use data to predict how sequence maps to function without requiring a detailed model of the underlying physics or biological pathways. To demonstrate ML-guided directed evolution, we introduce the steps required to build ML sequence-function models and use them to guide engineering, making recommendations at each stage. This review covers basic concepts relevant to using ML for protein engineering as well as the current literature and applications of this new engineering paradigm. ML methods accelerate directed evolution by learning from information contained in all measured variants and using that information to select sequences that are likely to be improved. We then provide two case studies that demonstrate the ML-guided directed evolution process. We also look to future opportunities where ML will enable discovery of new protein functions and uncover the relationship between protein sequence and function.

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. A protein's function, such as its expression level, catalytic activity, or other properties of interest to the protein engineer, is specified by its amino-acid sequence. Protein engineering inverts this relationship in order to find a sequence that performs a specified function. Unfortunately, current biophysical prediction methods cannot distinguish between the functional level of closely-related proteins, and we cannot reliably map sequence to function.[1,2] Furthermore, the space of possible protein sequences is too large to be searched exhaustively naturally, in the laboratory, or computationally.[3] The problem of finding optimal sequences is NP-hard, meaning there is no known polynomial-time method for searching this space.[4] Functional proteins are also extremely scarce in this vast space of sequences. Moreover, as the threshold level of function increases, the number of sequences having that function decreases exponentially.[5,6] As a result, highly functional sequences are vanishingly rare and overwhelmed by nonfunctional and mediocre sequences.

Directed evolution has been successful because it sidesteps our inability to map protein sequence to function. Inspired by natural evolution, directed evolution climbs a fitness landscape by accumulating beneficial mutations in an iterative protocol of mutation and selection. As shown in Figure 1, 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). In the second step, screening or selection identifies variants with improved properties for the next round of diversification. These steps are repeated until fitness goals are achieved. Illustrated in Figure 1c, directed evolution finds local optima through repeated local searches, taking advantage of functional promiscuity and the fact that functional sequences are clustered in sequence space.[7,5]

Directed evolution is limited by the fact that even the most high-throughput screening or selection methods only sample a

[1]Division of Chemistry and Chemical Engineering, California Institute of Technology, Pasadena, CA, USA. Correspondence to: Frances H. Arnold <frances@cheme.caltech.edu>.

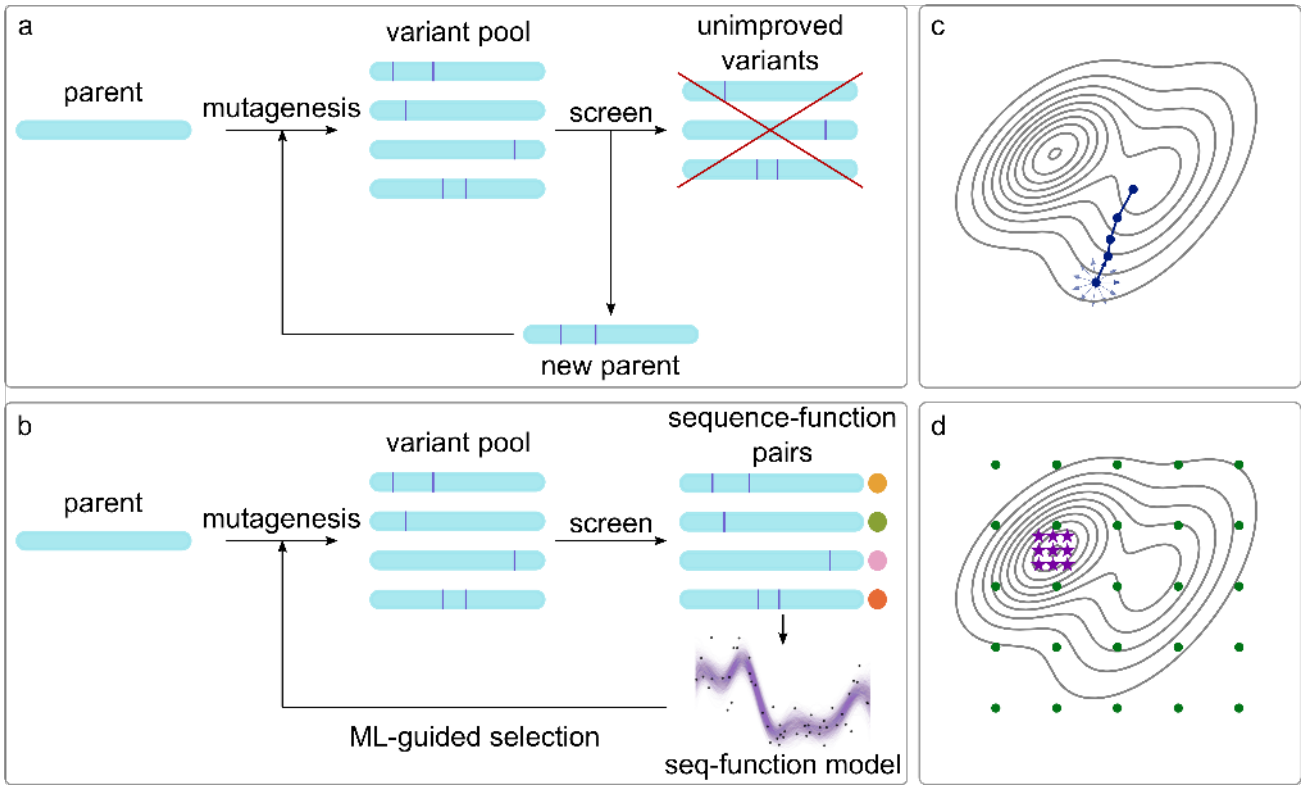

Figure 1. Directed evolution with and without machine learning. (a) Directed evolution uses iterative cycles of diversity generation and screening to find improved variants. Information from unimproved variants is discarded. (b) Machine-learning methods use the data collected in each round of directed 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 function landscape. (d) Machine learning-guided directed evolution often rationally chooses the initial points (green circles) to maximize the information learned from the function landscape, allowing future iterations to quickly converge to improved sequences (violet stars).

fraction of the sequences that can be made using most diversification methods, and developing efficient screens is nontrivial. There are an enormous number of ways to mutate any given protein: for a 300-amino acid protein there are 5,700 possible single amino acid substitutions and 32,381,700 ways to make just two substitutions with the 20 canonical amino acids. Screening exhaustively to find rare beneficial mutations is expensive and time-consuming and sometimes simply impossible. Moreover, directed evolution requires at least one minimally-functional parent and a locally-smooth sequence-function landscape for stepwise optimization.[8] Recombination methods may allow for bigger jumps in sequence space while retaining function,[9] 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 may be required to achieve meaningful performance improvements.

While directed evolution discards information from unimproved sequences, 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, reaching higher fitnesses than through directed evolution alone.[10] Figure 1b illustrates this data-augmented cycle. Machine-learning methods learn functional relationships from data[11,12] – the only added costs are in computation and DNA sequencing, the costs of which are decreasing rapidly. Machine-learning models of protein function can be predictive even when the underlying mechanisms are not well-understood. Furthermore, and perhaps most importantly, machine-learning guided directed evolution is able to escape local optima by learning efficiently about the entire function landscape, as illustrated in Figure 1d.

Machine learning is not necessarily useful in all applications. Because one major benefit of machine learning is in reducing the quantity of sequences to test, machine learning will be particularly useful in cases where lack of a high-throughput screen limits or precludes directed evolution. However, when a sufficient number of sequences can be screened or if the fitness landscape is smooth and 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.

Once the decision has been made to use machine learning, there are two key steps: i) building a sequence-function model and ii) using that model to choose sequences to screen. We provide practical guides for these steps as well as two case studies that illustrate the machine learning-guided directed evolution process. Finally, we consider developments that will enable wider applications of machine learning for protein engineering.

2. Building a machine-learning sequence-function model

Machine-learning models require examples of protein sequences and their resulting functional measurement in order to learn. The initial sequences chosen determine what the model can learn. The initial set of variants to screen can be selected in different ways: i) at random from the library,[13] ii) to maximize information about the mutations considered,[14,15,16] or iii) to maximize information about the remainder of the library.[17,18,19] Selecting variants at random is usually the simplest method; however, for low-throughput screens, it can be very important to maximize information obtained from high-cost experiments. Maximizing information about mutations is roughly equal to seeing each mutation considered in as many contexts as possible. Maximizing information about the remainder of the library is roughly equal to maximizing diversity in the training sequences. After collecting the initial training data, the user must decide what type of machine-learning model to use, represent the data in a form amenable to the model, and train the model.

2.1. Choosing a model

A wide range of machine-learning algorithms exist, and no single algorithm is optimal across all learning tasks.[20] For machine learning-guided directed evolution, we are most interested in methods that take input sequences and their associated output values and learn to predict the outputs of unseen sequences.

The simplest of these 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,[13] or blocks of sequence in a library of chimeric proteins made by recombination.[21] Linear models are commonly used as baseline predictors before more powerful models are tried.

Classification and regression trees[22] 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[23] or boosted trees,[24] 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; these have been used to predict enzyme thermostability.[25,26,27]

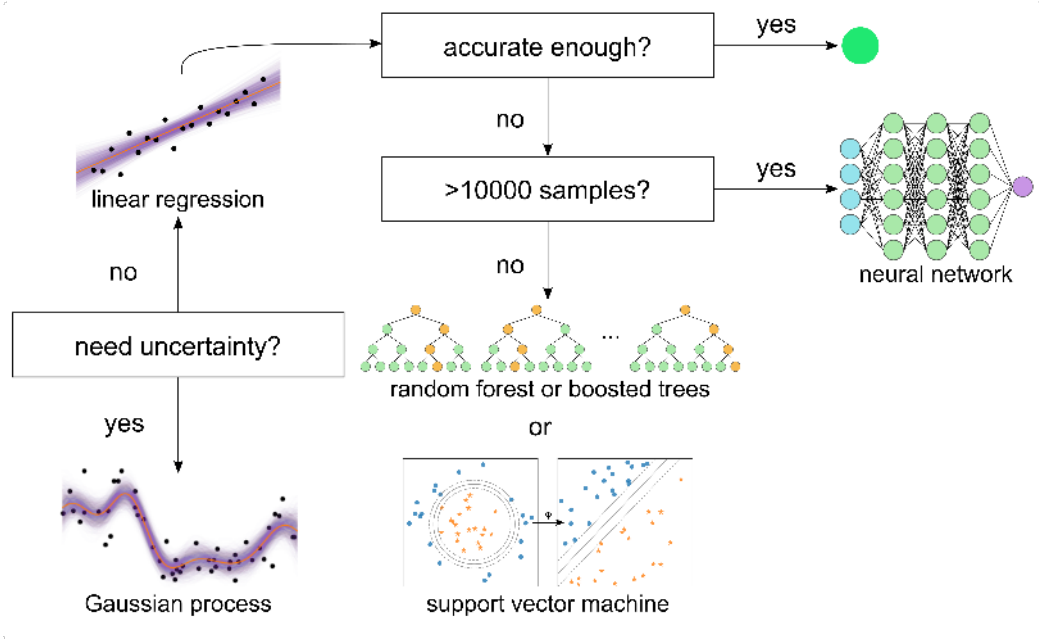

Figure 2. A general heuristic for choosing a machine-learning sequence-function model for proteins.

Kernel methods, such as support vector machines[28] and kernel ridge regression,[29] 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. 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,[30,31] which count the number of shared subsequences between two proteins, and weighted decomposition kernels,[32] which account for three-dimensional protein structure. Support vector machines have been used to predict protein thermostability,[33,34,35,36,25,27,26,37] enzyme enantioselectivity,[38] and membrane protein expression and localization.[39]

Gaussian process models combine kernel methods with Bayesian learning to produce probabilistic predictions.[40] These models rigorously capture uncertainty, which 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.[41,42] Gaussian processes have been used to predict thermostability,[43,17,32] substrates for enzymatic reactions,[44] fluorescence,[45] membrane localization,[18] and channelrhodopsin photo-properties.[19]

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,[46,47,48] protein-MHC binding,[49] binding site prediction,[50] protein-ligand binding,[51,52] solubility,[53] thermostability,[54,55] subcellular localization,[56] secondary structure,[57] functional class,[58,59] and even 3D structure.[60]

Figure 2 shows a general heuristic for choosing an algorithm for modeling protein sequence-function relationships. If estimates of model uncertainty are required, Gaussian processes are the simplest off-the-shelf model family. Otherwise, linear models provide a simple baseline to more complex models. If a linear model is insufficiently accurate, random forests, boosted trees, or support vector machines are fast and efficient for datasets with fewer than 10,000 examples, while neural networks generally provide the best performance on larger datasets.

2.2. Model training and evaluation

Training a machine learning model refers to tuning its parameters in order to maximize predictive accuracy. The key test for a machine-learning model is the ability to accurately predict labels for inputs it has not seen during training. Therefore, when training models, it is necessary to estimate the model's performance on data *not* in the training set. Thus it is essential to

remove a portion of the data, called the test set, until the absolute end for model evaluation. Typically, the test set comprises approximately 20% of the data.

In addition to parameters, all model families have hyperparameters that determine the form of the model. In fact, the family of model chosen is itself a hyperparameter. 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.[61] 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.

In order to compare models and select hyperparameters during a study, the data that remain after the test set has been removed 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 by providing an estimate of the test error. 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. Cross-validation provides a better estimate of the test error than using a constant validation set but requires more training time.

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.

2.3. Vector representations of proteins

Machine-learning models act on vectors or matrices of numbers, so protein sequences must be vectorized before model training. How each protein sequence is represented determines what can be learned.[62,63] 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.[17,18] 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.[64] 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. 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. However, there are a large number of physical properties that could be used to describe each amino acid or protein, and the molecular properties that dictate each functional property are unknown. AAIndex[65] and ProFET[66] attempt to systematically collect descriptors of protein sequences. There have also been attempts to describe each amino acid using two reduced dimensions based on volume and hydrophobicity[67] and to combine physical properties with structural information[68,36] by encoding each position in the sequence as a combination of the properties of amino acids in its geometric neighborhood.

While a large number of protein sequences have been deposited in databases, only a tiny fraction are labeled with measured properties relevant to any specific prediction task. The 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 incorporating evolutionary information are BLOSUM[69] 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[70,52,71,72,73,74,75] or from structural information.[76] These representations learn to place similar sequences close together in the continuous space of proteins. Learned encodings are low-dimensional 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.[20] 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,[73] 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.

3. Using sequence-function predictions to guide exploration

Once a sequence-function model has been trained, the next set of sequences to screen can be chosen either by collecting mutations believed to be beneficial or by directly optimizing over sequences. 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.[13] Alternately, the model can be used to select combinations of mutations that have a high probability of improving function[45] or to directly predict highly improved variants.[10]

Learning and selection can also be performed directly over sequences. This can be as simple as enumerating all the sequences considered, using the trained model to predict their function, and then synthesizing the best predicted variants. However, if multiple rounds of engineering are to be performed and the sequence-function model provides probabilistic predictions, Bayesian optimization provides a principled way to trade off between exploiting the information learned from previous iterations and exploring unseen regions of sequence space at each iteration.[61] Probabilistic predictions provide a well-calibrated measure of uncertainty in addition to predicting an expected value: the model knows what it does not know. For example, the Gaussian Process Upper Confidence Bound (GP-UCB) algorithm balances exploration and exploitation by selecting variants that maximize a weighted sum of the predictive mean and standard deviation,[77] and is guaranteed to asymptotically minimize the cumulative regret (difference between sampled variants and best variant) over infinite iterations. Figure 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 structure-guided recombination to optimize cytochrome P450 thermostability,[17] channelrhodopsin localization to mammalian cell membranes,[18] and channelrhodopsin light-activated conductance.[19] Because there is no high-throughput screen for the channelrhodopsin properties, it would not have been possible to optimize conductance by traditional directed evolution.

4. Case studies

4.1. Using partial least squares regression to maximize enzyme productivity

An early large-scale evolution campaign guided by machine learning was performed by Fox *et al.* in 2007 to improve the volumetric productivity of a halohydrin dehalogenase in a cyanation reaction by roughly 4000-fold.[13] In each round of evolution, summarized in Figure 4, 10-30 mutations of interest were first identified through random mutagenesis and site-saturation mutagenesis (including sites identified by tertiary structure analysis or sequence homology) and screening. These mutations were then randomly recombined, from which a number of variants equal to three times the number of positions mutated were sequenced and represented as one-hot vectors to form the input sequences used to train their model of choice, partial least squares (PLS; also known as projection to latent structures) regression.[78] The PLS algorithm projects both sequences and fitnesses to a space with reduced dimension to fit the linear model. Thus it is able to fit data where the number of variables exceeds the number of observations and potentially avoids indirect correlations in the model.[79] The resulting linear model can be expressed as

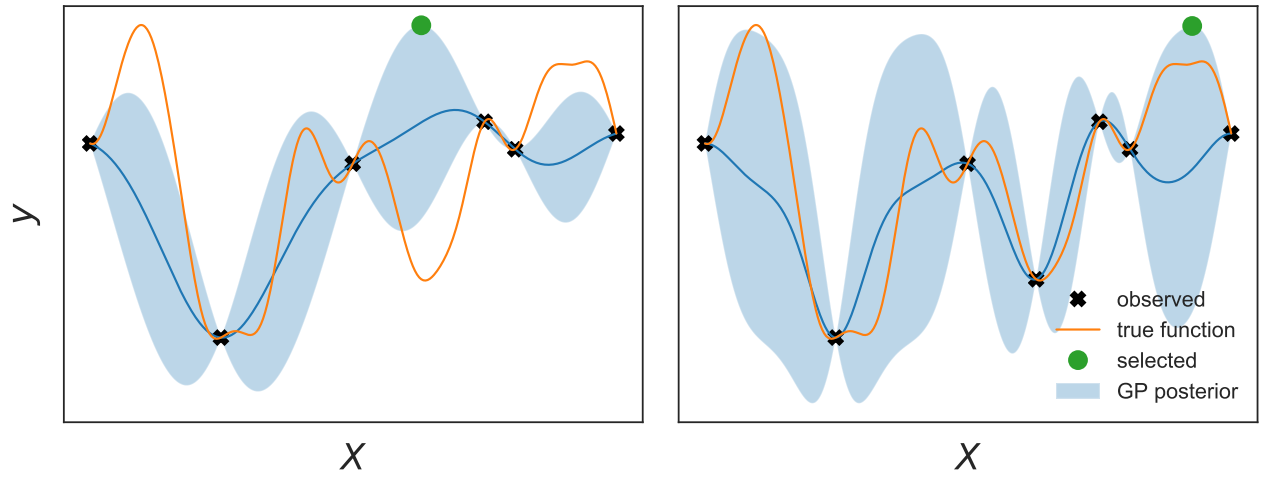

Figure 3. Gaussian Process Upper Confidence Bound algorithm. At each iteration, the next point to be sampled is chosen by maximizing the weighted sum of the posterior mean and standard deviation. This balances exploration and exploitation by exploring points that are both uncertain and have a high posterior mean. The right panel shows the posterior mean and standard deviation after observing the selected point (orange) in the left panel.

$$y = \sum_{m=1}^{q} c_m x_m \tag{1}$$

where c_m is the additive contribution of each mutation to fitness, and x_m indicates the presence ($x_m = 1$) or absence ($x_m = 0$) of the mutation. In this form, the PLS model has the distinct advantage of being readily interpreted. The authors classified each mutation as either beneficial, deleterious, or neutral based on the magnitude and sign of each coefficient. The classification of the mutation determines whether the mutation is retained, discarded, or tested again in the next round of evolution. However, when the Pearson's r from leave-one-out cross-validation was low, the authors did not have much confidence in the model and would be more biased toward keeping mutations to test in future rounds. Finally, the best variant identified in the library with randomly recombined mutations was fixed as the starting sequence for the next round.

This case study was one of the first large protein engineering campaigns guided by statistical analysis. A total of 519,045 variants were tested, 268,624 from the initial diversity generation used to identify mutations to model with PLS and 250,421 from libraries designed by their approach. In 18 rounds of evolution, no variant showed more than a threefold improvement in any individual round. The evolution likely could have been accelerated by testing different vectorization methods and machine-learning algorithms to improve the accuracy at each round. The linear form of the model was used based on the assumption that local regions of the sequence-function landscape display predominantly additive effects. For fitness landscapes where this is not the case, an alternative model must be used. This work, which followed validation of the approach on a theoretical fitness landscape,[78] remains a landmark effort in the application of statistical modeling to protein engineering.

4.2. Using Bayesian optimization to maximize thermostability of a cytochrome P450

Romero *et al.*[17] demonstrate the utility of a machine-learning method that is particularly suitable in cases where it is expensive or difficult to screen in a directed evolution experiment. These authors sought to increase the thermostability, as measured by the T_{50}, of cytochrome P450s generated by recombining sequence fragments from the heme domains of the bacterial cytochrome P450 enzymes CYP102A1, CYP102A2, and CYP102A3. T_{50} is defined as the temperature at which an enzyme loses half its activity after a 10-minute incubation. The sequence fragments were chosen to minimize the number of contacts broken, where contacts are amino acids within a certain distance (4.5 Å) of each other. Chimeric genes can be made by directly synthesizing the DNA sequence for each construct, but this is expensive for large numbers of sequences.

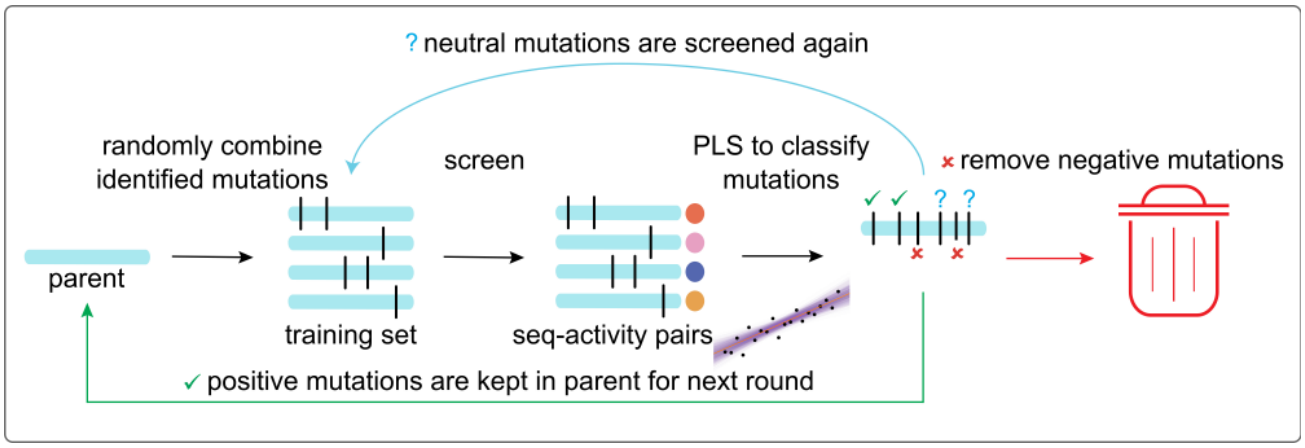

Figure 4. Directed evolution using partial least squares (PLS) regression. In this approach, Fox *et al.* randomly recombine mutations previously identified through classical techniques such as random or site-directed mutagenesis.[13] Variants with these mutations are screened and sequenced, and the data are used to fit a linear model with the PLS algorithm. Based on the magnitude and sign of the contributions of the linear model, mutations are classified as beneficial, neutral, or deleterious, after which mutations are fixed, retested, or removed, respectively. With this approach, Fox *et al.* were able to improve the volumetric productivity of a protein-catalyzed cyanation reaction roughly 4000-fold in 18 rounds of evolution.

Furthermore, the T_{50} measurement requires multiple incubations and measurements for each variant and is not particularly high-throughput. Therefore, this was a case where engineering without machine learning would have been difficult.

The authors trained initial Gaussian process models for T_{50} and the presence or absence of function on 242 chimeric P450s, and then validated model performance on a test set of chimeric P450s generated with different boundaries between sequence fragments. The Gaussian process model used a one-hot representation of the protein's 3-dimensional structure, which was demonstrated to be more predictive than a one-hot representation of the primary sequence. Gaussian process models are a good fit for this problem setting where only a small amount of data were available. These models also provide probabilistic predictions, which can be used to guide data-efficient exploration and optimization, but the computational requirements scale poorly with the number of training examples.

After validating the accuracy of their models, the authors wanted to select a small set of additional sequences to bolster their models' knowledge of the recombination landscape. In general, this can be done by selecting those sequences that most reduce uncertainty in the predictions. This is measured by the mutual information between the measured sequences and the remaining sequences. Typically, these sequences will be very diverse. However, because many variants are non-functional and therefore provide no information about T_{50}, Romero *et al.* used their classification model to select 26 additional sequences by maximizing the expected mutual information.[17] Informally, these 30 sequences combined a high probability of being functional with high sequence diversity, and in fact, 26 of these sequences were functional despite being on average 106 mutations from the closest parent. Thus, this demonstrates the ability of a machine learning model to efficiently explore very diverse chimeric sequences while minimizing the resources wasted on screening non-functional proteins.

With sufficient training data collected, Bayesian optimization was then used to search for more thermostable variants. First, four rounds of the batch Gaussian process upper-bound algorithm yielded a diverse sampling of thermostable P450s. However, because none of these variants increased the maximum observed T_{50}, the authors checked their sequence-function model by screening a sequence predicted to be stabilized with high certainty. Two additional iterations of GP-UCB were then followed by a pure exploitation round of five sequences, two of which were more thermostable than any previously observed P450s.

By using previously collected data, an accurate sequence-function model, and Bayesian optimization, the authors thus demonstrated a framework for data-efficient protein engineering, which has since been transferred to other protein systems and properties.[80,18] The framework and results are summarized in Figure 5.

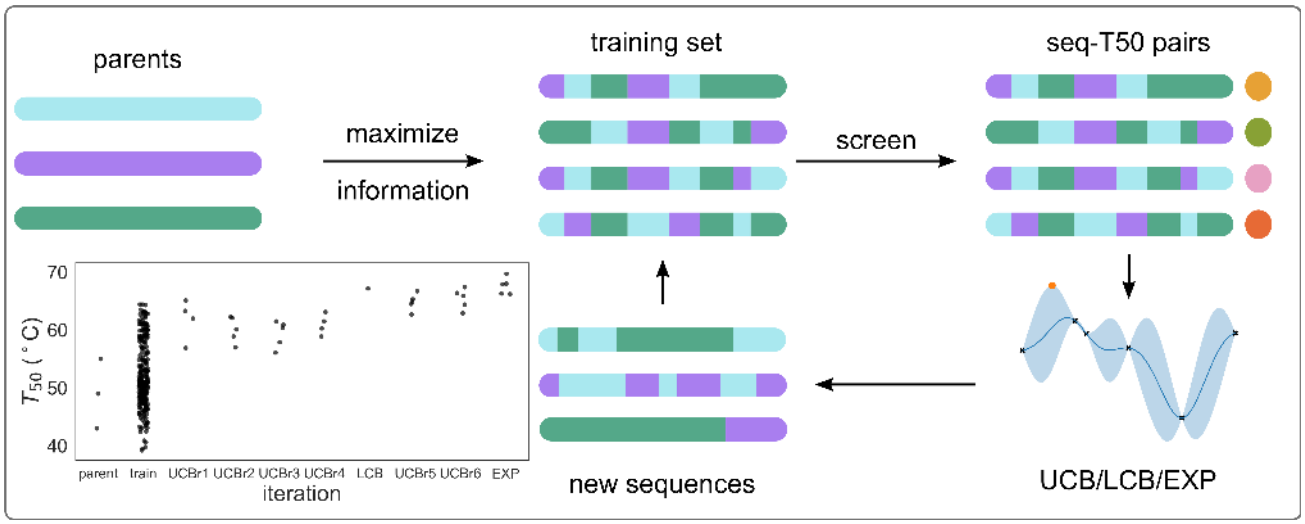

Figure 5. Directed evolution using Gaussian processes and Bayesian optimization. After an initial training set chosen to be maximally informative, subsequent batches of sequences are chosen using the Gaussian process upper confidence bound (UCB) or lower confidence bound (LCB) algorithms, or to fully exploit the model (EXP). The inset shows the T_{50} for variants found in each round.

5. Conclusions and future directions

Supervised machine-learning methods have demonstrated their utility in directed protein evolution. However, broader applications of machine learning will require taking advantage of unlabeled protein sequences or sequences labeled for properties other than those of specific interest to the protein engineer. Databases such as UniProt[81] contain hundreds of millions of protein sequences, some of which are annotated with structural or functional information. These sequences contain information about the sequence motifs and patterns that result in a functional protein, and the structural and functional annotations provide clues as to how structure and function arise from sequence. These annotations can be learned from sequences,[74] and embeddings trained on these annotations may be able to transfer knowledge from UniProt to specific problems of interest.[82]

These large quantities of unlabeled or partially-labeled 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 functional proteins, and the complete landscape is plastered 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, although these samples are biased by nature's evolutionary methods for generating proteins. A method for selectively sampling from the distribution of functional 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 provide such a tool, and are an attractive alternative to *de novo* design methods.[83]

Unlike discriminative models that learn $p(y|x)$ in order to predict labels y given inputs x, generative models learn to generate examples that are not in the training set by learning the generating distribution $p(x)$ for the training data. Tantalizingly, generative models in other fields have been trained to generate new faces,[84] sketches,[85] and even music.[86] 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.[87,88] An autoencoder is a neural network that learns to encode an input as a vector (encoding) and then reconstructs the input from the vector (decoding) (Figure 6). By learning an encoding with smaller dimensionality than the original input, the model extracts the most important information from the input in an information-dense format. The encoding can then be used as an input to other learning algorithms. In a variational autoencoder, the learned encoding is further constrained to encourage the encodings to be densely packed to allow interpolation between examples and the ability to mix and match properties.[89] Applied to protein sequences, variational autoencoders can learn complex epistatic relationships among variants, allowing semi-supervised predictions of variant functionality based only on existing sequences without a need for individual measurements.

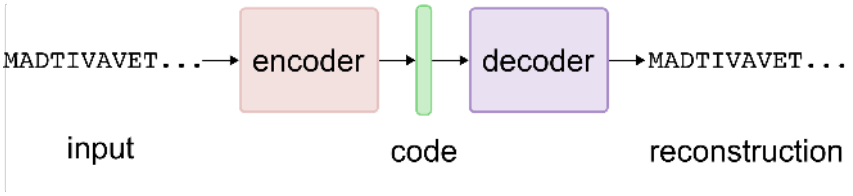

Figure 6. Autoencoder. An autoencoder consists of an encoder model and a decoder model. The encoder converts the input to a low-dimensional vector (code). The decoder reconstructs the input from this code. Typically, the encoder and decoder are both neural network models, and the entire autoencoder model 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.

In addition, the protein variants generated by a variational autoencoder or other generative model can be highly sequence-divergent from known sequences but potentially still functional.[90] These can be starting points for further engineering, or the generative model itself can be directly tuned *in silico* to produce sequences optimized for a desired property. Recently, recurrent neural networks and generative adversarial networks[91] have been used to generate novel antimicrobial peptides[92,93] and protein structures,[94] and there has been an effort to develop a mathematical framework for adapting a generative model to sample sequences with one or more specified properties.[95,96] While 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. However, 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. 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 and engineer proteins for novel functions.

Acknowledgments

The authors wish to thank Yuxin Chen, Kadina Johnston, Bruce Wittmann, and Hopen Yang for comments on early versions of the manuscript and members of the Arnold lab, Justin Bois, and Yisong Yue for general advice and discussions on protein engineering and machine learning.

This work was supported by the U.S. Army Research Office Institute for Collaborative Biotechnologies [W911F-09-0001 to F.H.A.], the Donna and Benjamin M. Rosen Bioengineering Center [to K.K.Y.], and the National Science Foundation [GRF2017227007 to Z.W.].

The authors declare that they have no competing financial interests.

Correspondence and requests for materials should be addressed to F.H.A.
(email: frances@cheme.caltech.edu).

References

[1] Dou, J. *et al.* Sampling and energy evaluation challenges in ligand binding protein design. *Protein Sci* **26**, 2426–2437 (2017).

[2] Garcia-Borràs, M., Houk, K. N. & Jiménez-Osés, G. Computational design of protein function. *Computational Tools for Chemical Biology* **3**, 87 (2017).

[3] Mandecki, W. The game of chess and searches in protein sequence space. *Trends Biotech* **16**, 200–202 (1998).

[4] Pierce, N. A. & Winfree, E. Protein design is NP-hard. *Protein Eng* **15**, 779–782 (2002).

[5] Smith, J. M. Natural selection and the concept of a protein space. *Nature* **225**, 563 (1970).

[6] Orr, H. A. The distribution of fitness effects among beneficial mutations in Fisher's geometric model of adaptation. *J Theor Biol* **238**, 279–285 (2006).

[7] Khersonsky, O. & Tawfik, D. S. Enzyme promiscuity: a mechanistic and evolutionary perspective. *Annu Rev Biochem* **79**, 471–505 (2010).

[8] Romero, P. A. & Arnold, F. H. Exploring protein fitness landscapes by directed evolution. *Nat Rev Mol Cell Biol* **10**, 866 (2009).

[9] Drummond, D. A., Silberg, J. J., Meyer, M. M., Wilke, C. O. & Arnold, F. H. On the conservative nature of intragenic recombination. *Proc Natl Acad Sci USA* **102**, 5380–5385 (2005).

[10] Wu, Z., Kan, S. B. J., Lewis, R. D., Wittmann, B. J. & Arnold, F. H. Machine learning-assisted directed protein evolution with combinatorial libraries. *Proc Natl Acad Sci USA* (2019). https://www.pnas.org/content/early/2019/04/11/1901979116.full.pdf.

[11] Hastie, T. & Tibshirani, R. *The Elements of Statistical Learning; Data Mining, Inference and Prediction* (Springer, New York, 2008).

[12] Murphy, K. *Machine learning, a probabilistic perspective* (MIT Press, 2012).
Murphy's textbook provides a thorough introduction to modern machine learning.

[13] Fox, R. J. *et al.* Improving catalytic function by ProSAR-driven enzyme evolution. *Nat Biotechnol* **25**, 338 (2007).

[14] Liao, J. *et al.* Engineering proteinase K using machine learning and synthetic genes. *BMC Biotechnol* **7**, 16 (2007).

[15] Govindarajan, S. *et al.* Mapping of amino acid substitutions conferring herbicide resistance in wheat glutathione transferase. *ACS Synth Biol* **4**, 221–227 (2014).

[16] Musdal, Y., Govindarajan, S. & Mannervik, B. Exploring sequence-function space of a poplar glutathione transferase using designed information-rich gene variants. *Protein Eng Des Sel* **30**, 543–549 (2017).

[17] Romero, P. A., Krause, A. & Arnold, F. H. Navigating the protein fitness landscape with Gaussian processes. *Proc Natl Acad Sci USA* **110**, E193–E201 (2013).
This is the first study to combine SCHEMA recombination with the GP-UCB algorithm to optimize a protein property.

[18] Bedbrook, C. N., Yang, K. K., Rice, A. J., Gradinaru, V. & Arnold, F. H. Machine learning to design integral membrane channelrhodopsins for efficient eukaryotic expression and plasma membrane localization. *PLoS Comput Biol* **13**, e1005786 (2017).

[19] Bedbrook, C. N., Yang, K. K., J, R. E., Viviana, G. & Arnold, F. H. Machine learning-guided channelrhodopsin engineering enables minimally-invasive optogenetics. *In preparation* (2018).

[20] Wolpert, D. H. The lack of a priori distinctions between learning algorithms. *Neural Comput* **8**, 1341–1390 (1996).

[21] Li, Y. *et al.* A diverse family of thermostable cytochrome P450s created by recombination of stabilizing fragments. *Nat Biotechnol* **25**, 1051 (2007).

[22] Breiman, L. *Classification and Regression Trees* (Routledge, 2017).

[23] Breiman, L. Random forests. *Machine Learning* **45**, 5–32 (2001).

[24] Friedman, J. H. Stochastic gradient boosting. *Comput Stat Data An* **38**, 367–378 (2002).

[25] Tian, J., Wu, N., Chu, X. & Fan, Y. Predicting changes in protein thermostability brought about by single- or multi-site mutations. *BMC Bioinformatics* **11**, 370 (2010).

[26] Li, Y. & Fang, J. PROTS-RF: a robust model for predicting mutation-induced protein stability changes. *PloS One* **7**, e47247 (2012).

[27] Jia, L., Yarlagadda, R. & Reed, C. C. Structure based thermostability prediction models for protein single point mutations with machine learning tools. *PloS One* **10**, e0138022 (2015).

[28] Cortes, C. & Vapnik, V. Support-vector networks. *Machine learning* **20**, 273–297 (1995).

[29] Nadaraya, E. On estimating regression. *Theor Probab Appl* **9**, 141–142 (1964).

[30] Leslie, C., Eskin, E. & Noble, W. S. The spectrum kernel: A string kernel for SVM protein classification. *Pac Symp Biocomput* 564–575 (2002).

[31] Leslie, C. S., Eskin, E., Cohen, A., Weston, J. & Noble, W. S. Mismatch string kernels for discriminative protein classification. *Bioinformatics* **20**, 467–476 (2004).

[32] Jokinen, E., Heinonen, M. & Lähdesmäki, H. mGPfusion: Predicting protein stability changes with Gaussian process kernel learning and data fusion. *Bioinformatics* **34**, i274–i283 (2018).

[33] Capriotti, E., Fariselli, P. & Casadio, R. I-Mutant2.0: predicting stability changes upon mutation from the protein sequence or structure. *Nucleic Acids Res* **33**, W306–W310 (2005).

[34] Capriotti, E., Fariselli, P., Calabrese, R. & Casadio, R. Predicting protein stability changes from sequences using support vector machines. *Bioinformatics* **21**, ii54–ii58 (2005).

[35] Cheng, J., Randall, A. & Baldi, P. Prediction of protein stability changes for single-site mutations using support vector machines. *Proteins* **62**, 1125–1132 (2006).

[36] Buske, F. A., Their, R., Gillam, E. M. & Bodén, M. In silico characterization of protein chimeras: relating sequence and function within the same fold. *Proteins* **77**, 111–120 (2009).

[37] Liu, J. & Kang, X. Grading amino acid properties increased accuracies of single point mutation on protein stability prediction. *BMC Bioinformatics* **13**, 44 (2012).

[38] Zaugg, J., Gumulya, Y., Malde, A. K. & Bodén, M. Learning epistatic interactions from sequence-activity data to predict enantioselectivity. *J Comput Aided Mol Des* **31**, 1085–1096 (2017).

[39] Saladi, S. M., Javed, N., Müller, A. & Clemons, W. M. A statistical model for improved membrane protein expression using sequence-derived features. *J Biol Chem* **293**, 4913–4927 (2018).

[40] Rasmussen, C. E. & Williams, C. K. I. *Gaussian Processes for Machine Learning* (MIT Press, 2006).

[41] Wilson, A. G. & Nickisch, H. Kernel interpolation for scalable structured Gaussian processes (KISS-GP). In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, 1775–1784 (2015).

[42] Wang, K. A. *et al.* Exact gaussian processes on a million data points (2019). https://arxiv.org/abs/1903.08114.

[43] Pires, D. E., Ascher, D. B. & Blundell, T. L. mCSM: predicting the effects of mutations in proteins using graph-based signatures. *Bioinformatics* **30**, 335–342 (2013).

[44] Mellor, J., Grigoras, I., Carbonell, P. & Faulon, J.-L. Semisupervised Gaussian process for automated enzyme search. *ACS Synth Biol* **5**, 518–528 (2016).

[45] Saito, Y. *et al.* Machine-learning-guided mutagenesis for directed evolution of fluorescent proteins. *ACS Synth Biol* (2018).

[46] Zhang, S. *et al.* A deep learning framework for modeling structural features of RNA-binding protein targets. *Nucleic Acids Res* **44**, e32–e32 (2015).

[47] Alipanahi, B., Delong, A., Weirauch, M. T. & Frey, B. J. Predicting the sequence specificities of DNA-and RNA-binding proteins by deep learning. *Nat Biotechnol* **33**, 831 (2015).

[48] Zeng, H., Edwards, M. D., Liu, G. & Gifford, D. K. Convolutional neural network architectures for predicting DNA-protein binding. *Bioinformatics* **32**, i121–i127 (2016).

[49] Hu, J. & Liu, Z. DeepMHC: Deep convolutional neural networks for high-performance peptide-MHC binding affinity prediction 239236 (2017). https://www.biorxiv.org/content/early/2017/12/24/239236.

[50] Jiménez, J., Doerr, S., Martínez-Rosell, G., Rose, A. & De Fabritiis, G. DeepSite: protein-binding site predictor using 3D-convolutional neural networks. *Bioinformatics* **33**, 3036–3042 (2017).

[51] Gomes, J., Ramsundar, B., Feinberg, E. N. & Pande, V. S. Atomic convolutional networks for predicting protein-ligand binding affinity (2017). https://arxiv.org/abs/1703.10603.

[52] Mazzaferro, C. Predicting protein binding affinity with word embeddings and recurrent neural networks 128223 (2017). https://www.biorxiv.org/content/early/2017/04/18/128223.

[53] Khurana, S. *et al.* DeepSol: A deep learning framework for sequence-based protein solubility prediction. *Bioinformatics* **1**, 9 (2018).

[54] Dehouck, Y. *et al.* Fast and accurate predictions of protein stability changes upon mutations using statistical potentials and neural networks: PoPMuSiC-2.0. *Bioinformatics* **25**, 2537–2543 (2009).

[55] Giollo, M., Martin, A. J., Walsh, I., Ferrari, C. & Tosatto, S. C. NeEMO: a method using residue interaction networks to improve prediction of protein stability upon mutation. *BMC Genomics* **15**, S7 (2014).

[56] Almagro Armenteros, J. J., Sønderby, C. K., Sønderby, S. K., Nielsen, H. & Winther, O. DeepLoc: prediction of protein subcellular localization using deep learning. *Bioinformatics* **33**, 3387–3395 (2017).

[57] Sønderby, S. K. & Winther, O. Protein secondary structure prediction with long short term memory networks (2014). https://arxiv.org/abs/1412.7828.

[58] Szalkai, B. & Grolmusz, V. Near perfect protein multi-label classification with deep neural networks. *Methods* **132**, 50–56 (2018).

[59] Cao, R. *et al.* ProLanGO: Protein function prediction using neural machine translation based on a recurrent neural network. *Molecules* **22**, 1732 (2017).

[60] Hopf, T. A. *et al.* Three-dimensional structures of membrane proteins from genomic sequencing. *Cell* **149**, 1607–1621 (2012).

[61] Snoek, J., Larochelle, H. & Adams, R. P. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, 2951–2959 (2012). URL http://papers.nips.cc/paper/4522-practical-bayesian-optimization-of-machine-learning-algorithms.pdf.

[62] Domingos, P. A few useful things to know about machine learning. *Communications of the ACM* **55**, 78–87 (2012).

[63] Bengio, Y., Courville, A. & Vincent, P. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **35**, 1798–1828 (2013).

[64] Capriotti, E., Fariselli, P. & Casadio, R. A neural-network-based method for predicting protein stability changes upon single point mutations. *Bioinformatics* **20**, i63–i68 (2004).

[65] Kawashima, S. *et al.* AAindex: amino acid index database, progress report 2008. *Nucleic Acids Res* **36**, D202–D205 (2007).

[66] Ofer, D. & Linial, M. ProFET: Feature engineering captures high-level protein functions. *Bioinformatics* **31**, 3429–3436 (2015).

[67] Barley, M. H., Turner, N. J. & Goodacre, R. Improved descriptors for the quantitative structure–activity relationship modeling of peptides and proteins. *J Chem Inf Model* **58**, 234–243 (2018).

[68] Qiu, J., Hue, M., Ben-Hur, A., Vert, J.-P. & Noble, W. S. A structural alignment kernel for protein structures. *Bioinformatics* **23**, 1090–1098 (2007).

[69] Henikoff, S. & Henikoff, J. G. Amino acid substitution matrices from protein blocks. *Proc Natl Acad Sci USA* **89**, 10915–10919 (1992).

[70] Asgari, E. & Mofrad, M. R. Continuous distributed representation of biological sequences for deep proteomics and genomics. *PloS One* **10**, e0141287 (2015).

[71] Ng, P. dna2vec: Consistent vector representations of variable-length k-mers (2017). https://arxiv.org/abs/1701.06279.

[72] Kimothi, D., Soni, A., Biyani, P. & Hogan, J. M. Distributed representations for biological sequence analysis (2016). https://arxiv.org/abs/1608.05949.

[73] Yang, K., Wu, Z., Bedbrook, C. & Arnold, F. Learned protein embeddings for machine learning. *Bioinformatics* (2018).

[74] Schwartz, A. S. *et al.* Deep semantic protein representation for annotation, discovery, and engineering. *bioRxiv* 365965 (2018). https://www.biorxiv.org/content/early/2018/07/10/365965.

[75] Alley, E. C., Khimulya, G., Biswas, S., AlQuraishi, M. & Church, G. M. Unified rational protein engineering with sequence-only deep representation learning. *bioRxiv* 589333 (2019). https://www.biorxiv.org/content/10.1101/589333v1.

[76] Bepler, T. & Berger, B. Learning protein sequence embeddings using information from structure (2019). https://arxiv.org/abs/1902.08661.

[77] Srinivas, N., Krause, A., Kakade, S. M. & Seeger, M. Gaussian process optimization in the bandit setting: No regret and experimental design (2009). https://arxiv.org/abs/:0912.3995.

[78] Fox, R. *et al.* Optimizing the search algorithm for protein engineering by directed evolution. *Protein Eng* **16**, 589–597 (2003).
This study is the first to use machine learning to guide directed evolution.

[79] de Jong, S. Simpls: An alternative approach to partial least squares regression. *Chemometrics and Intelligent Laboratory Systems* **18**, 251 – 263 (1993).

[80] Bedbrook, C. N., Yang, K. K., Robinson, J. E., Gradinaru, V. & Arnold, F. H. Machine learning-guided channelrhodopsin engineering enables minimally-invasive optogenetics (2018).

[81] UniProt Consortium. UniProt: the universal protein knowledgebase. *Nucleic Acids Res* **45**, D158–D169 (2017).

[82] Pan, S. J. & Yang, Q. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering* **22**, 1345–1359 (2010).

[83] Baker, D. An exciting but challenging road ahead for computational enzyme design. *Protein Sci* **19**, 1817–1819 (2010).

[84] Radford, A., Metz, L. & Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks (2015). https://arxiv.org/abs/1511.06434.

[85] Ha, D. & Eck, D. A neural representation of sketch drawings (2017). https://arxiv.org/abs/1704.03477.

[86] Roberts, A., Engel, J., Raffel, C., Hawthorne, C. & Eck, D. A hierarchical latent vector model for learning long-term structure in music (2018). https://arxiv.org/abs/1803.05428.

[87] Sinai, S., Kelsic, E., Church, G. M. & Nowak, M. A. Variational auto-encoding of protein sequences (2017). https://arxiv.org/abs/1712.03346.

[88] Riesselman, A. J., Ingraham, J. B. & Marks, D. S. Deep generative models of genetic variation capture mutation effects. *Nat Methods* **15**, 816–822 (2018).
This study predicts the effects of mutations without using any labeled data.

[89] Kingma, D. P. & Welling, M. Auto-encoding variational Bayes (2013). https://arxiv.org/abs/1312.6114.

[90] Costello, Z. & Garcia Martin, H. How to hallucinate functional proteins (2019). https://arxiv.org/abs/1903.00458.

[91] Goodfellow, I. *et al.* Generative adversarial nets. In *Adv Neur In*, 2672–2680 (2014).

[92] Müller, A. T., Hiss, J. A. & Schneider, G. Recurrent neural network model for constructive peptide design. *J Chem Inf Model* (2018).

[93] Gupta, A. & Zou, J. Feedback GAN (FBGAN) for DNA: a novel feedback-loop architecture for optimizing protein functions (2018). https://arxiv.org/abs/1804.01694.

[94] Anand, N. & Huang, P. Generative modeling for protein structures. In Bengio, S. *et al.* (eds.) *Advances in Neural Information Processing Systems 31*, 7504–7515 (Curran Associates, Inc., 2018). URL http://papers.nips.cc/paper/7978-generative-modeling-for-protein-structures.pdf.

[95] Brookes, D. H. & Listgarten, J. Design by adaptive sampling (2018). https://arxiv.org/abs/1810.03714.

[96] Brookes, D. H., Park, H. & Listgarten, J. Conditioning by adaptive sampling for robust design (2019). https://arxiv.org/abs/1901.10060.