nature methods

REVIEW ARTICLE

https://doi.org/10.1038/s41592-019-0496-6

Machine-learning-guided directed evolution for protein engineering

Kevin K. Yang, Zachary Wu and Frances H. Arnold*

Protein engineering through machine-learning-guided directed evolution enables the optimization of protein functions. Machine-learning approaches predict how sequence maps to function in a data-driven manner without requiring a detailed model of the underlying physics or biological pathways. Such methods accelerate directed evolution by learning from the properties of characterized variants and using that information to select sequences that are likely to exhibit improved properties. Here we introduce the steps required to build machine-learning sequence–function models and to use those models to guide engineering, making recommendations at each stage. This review covers basic concepts relevant to the use of machine learning for protein engineering, as well as the current literature and applications of this engineering paradigm. We illustrate the process with two case studies. Finally, we look to future opportunities for machine learning to enable the discovery of unknown protein functions and uncover the relationship between protein sequence and function.

Protein engineering seeks to design or discover proteins with properties useful for technological, scientific, or medical applications. Properties related to a protein's function, such as its expression level and catalytic activity, are determined by its amino acid sequence. Protein engineering inverts this relationship in order to find a sequence that performs a specified function. However, current biophysical prediction methods cannot distinguish the functional levels of closely related proteins[1,2]. Furthermore, the space of possible proteins is too large to search exhaustively naturally, in the laboratory, or computationally[3]. The problem of finding optimal sequences is NP-hard: there is no known polynomial-time method for searching this space[4]. Functional proteins are scarce in this vast space of sequences, and as the desired level of function increases, the number of sequences having that function decreases exponentially[5,6]. As a result, functional sequences are rare and overwhelmed by nonfunctional and mediocre sequences.

Directed evolution has been successful because it sidesteps scientists' inability to map protein sequence to function. Inspired by natural evolution, directed evolution leads to an accumulation of beneficial mutations via an iterative protocol of mutation and selection. The approach entails sequence diversification to generate a library of modified sequences followed by screening to identify variants with improved properties, with further rounds of diversification until fitness goals are achieved (Fig. 1a). Directed evolution finds local optima through repeated local searches, taking advantage of functional promiscuity and the clustering of functional sequences in sequence space[5,7] (Fig. 1b).

Directed evolution is limited by the fact that even the most high-throughput screening or selection methods sample only a fraction of the sequences that can be made by most diversification methods, and the development of 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. Exhaustive screening to find rare beneficial mutations is expensive and time-consuming, and sometimes impossible. Moreover, directed evolution requires at least one minimally functional parent and a locally smooth sequence–function landscape[8]. Recombination methods may allow for bigger jumps in sequence space while retaining function[9], but these methods are restricted to combinations of previously explored mutations.

Whereas directed evolution discards information from unimproved sequences, machine-learning methods can use this information to expedite evolution and expand the number of properties that can be optimized by intelligently selecting new variants to screen, thereby reaching higher fitness levels than are possible through directed evolution alone[10–14] (Fig. 1c). Machine-learning methods learn functional relationships from data[15,16]—the only added costs compared with those of directed evolution are in computation and DNA sequencing, the costs of which are decreasing rapidly. Machine-learning models can be predictive even when the underlying biophysical mechanisms are not well understood. Furthermore, machine-learning-guided directed evolution is able to escape local optima by learning efficiently about the entire function landscape (Fig. 1d).

Machine learning is not necessarily useful for all protein-engineering applications. Although machine learning will never reduce the expected improvement per iteration, the added costs of sequencing every variant screened and synthesizing desired variants may increase the experimental burden. In cases where the screen is expensive or slow enough to outweigh the cost and time of sequencing and synthesis, machine learning is beneficial. Alternatively, if the library design necessitates gene synthesis (instead of mutagenesis to generate variation), then machine learning should be used to choose which sequences to synthesize. However, it is impossible to predict a priori how much machine learning will speed up an optimization. The decision to use machine learning should consider prior knowledge about the system (the difficulty of the screen, the smoothness of the fitness landscape, etc.).

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 guidance for these steps, as well as two case studies that illustrate the machine-learning-guided directed evolution process. Finally, we consider developments that would allow wider application of machine learning for protein engineering.

Division of Chemistry and Chemical Engineering, California Institute of Technology, Pasadena, CA, USA. *e-mail: fha@cheme.caltech.edu

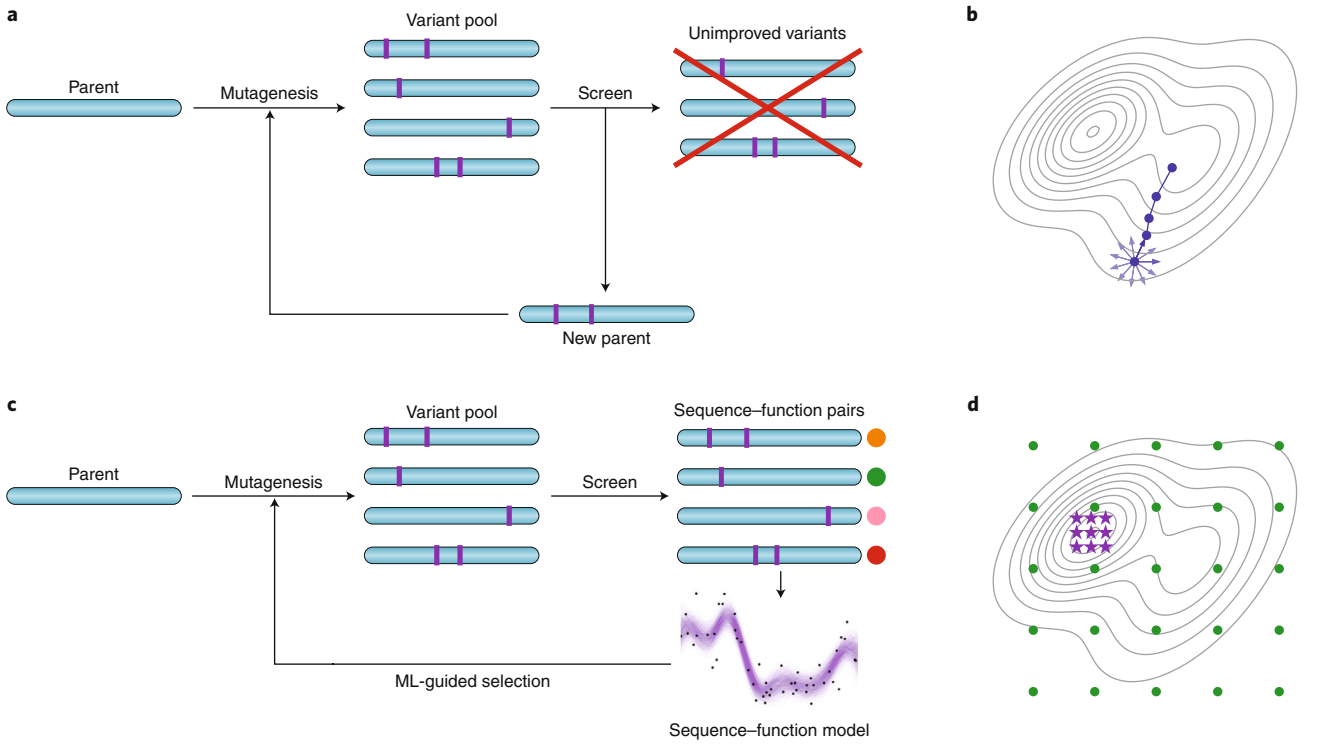

Fig. 1 | Directed evolution with and without machine learning. a, Directed evolution uses iterative cycles of diversity generation and screening to find improved variants. Information from unimproved variants is discarded. **b**, Directed evolution is a series of local searches on the function landscape. **c**, Machine-learning (ML) methods use the data collected in each round of directed evolution to choose which mutations to test in the next round. Careful choice of mutations to test decreases the screening burden and improves outcomes. **d**, Machine-learning-guided directed evolution often rationally chooses the initial points (green circles) to maximize the information learned from the function landscape, thereby allowing future iterations to quickly converge to improved sequences (violet stars).

Building a machine-learning sequence–function model

Machine-learning models learn from examples of protein sequences and the respective functional measurements of the proteins. The examples chosen for building the model determine what the model can learn. The initial set of variants to screen can be selected at random from a library[10] or to maximize information about the mutations considered[11–13,17–19]. The selection of variants at random is usually the simplest method; however, for low-throughput screens, it can be important to maximize information obtained from high-cost experiments, as this will maximize model accuracy on unseen sequences. Maximizing information about the remainder of the library is roughly equivalent 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.

Choosing a model

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

The simplest machine-learning models apply a linear transformation to the input features, such as the amino acid at each position, the presence or absence of a mutation[10], 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] and 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 and have been used to predict enzyme thermostability[25–27].

Kernel methods, such as support vector machines[28] and kernel ridge regression[29], use 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[25–27,33–37], enzyme enantioselectivity[38], and membrane protein expression and localization[39].

Gaussian process (GP) models combine kernel methods with Bayesian learning to produce probabilistic predictions[40]. These models capture uncertainty, providing principled ways to guide experimental design. The run time for exact GP regression scales as the cube of the number of training examples, which makes it unsuitable for large ($>10^3$) datasets, but there are now fast and accurate approximations[41,42]. GPs have been used to predict thermostability[11,32,43], substrates for enzymatic reactions[44], fluorescence[45], membrane localization[12], and channelrhodopsin photoproperties[13].

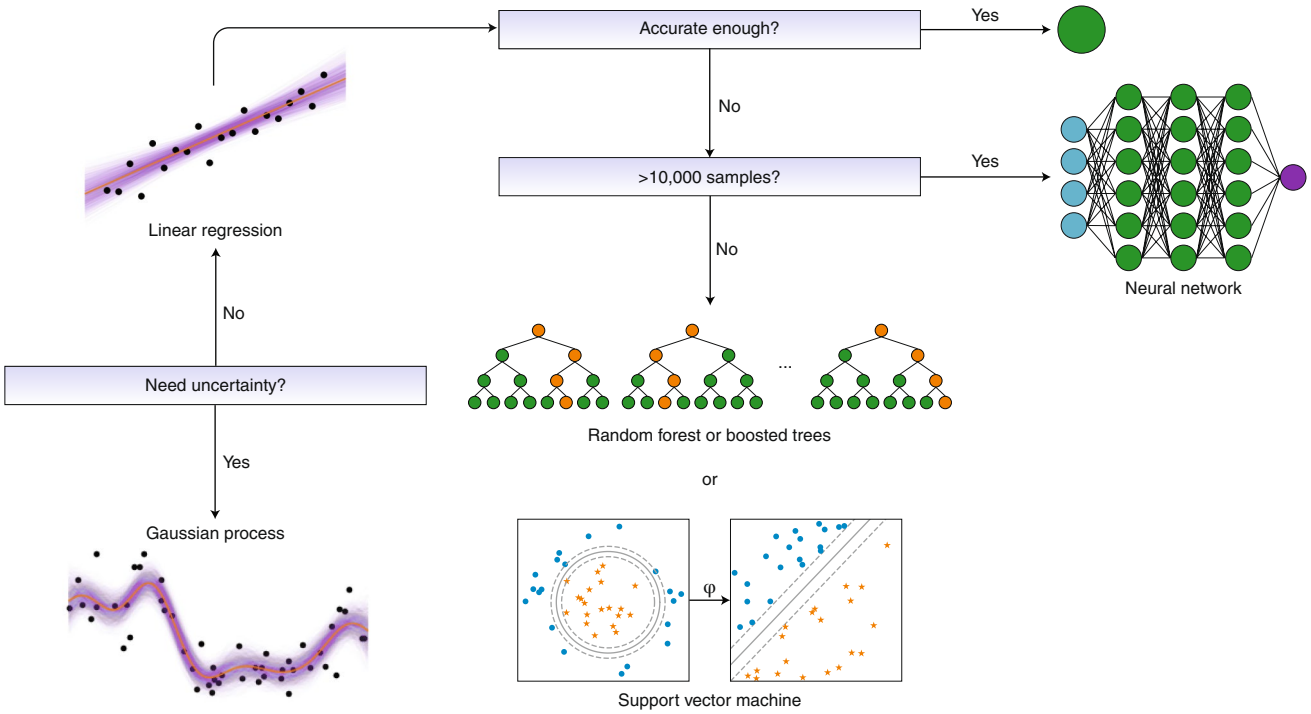

Fig. 2 | A general heuristic for choosing a machine-learning sequence–function model for proteins.

Deep-learning models, also known as neural networks, stack multiple linear layers connected by nonlinear activation functions, which allows them to extract high-level features from structured inputs. Neural networks are well suited for tasks with large labeled datasets. They have been applied to protein–nucleic acid binding[46–48], protein–major histocompatibility complex binding[49], binding-site prediction[50], protein–ligand binding[51,52], solubility[53], thermostability[54,55], subcellular localization[56], secondary structure[57], functional class[58–60], and even three-dimensional structure[61].

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

Model training and evaluation

Training a machine-learning model refers to tuning its parameters to maximize its predictive accuracy. The primary goal of training is to accurately predict labels for inputs not seen during training. Therefore, during model training, performance should be estimated on data not in the training set. It is thus essential to withhold approximately 20% of the data, called the test set, for model evaluation.

In addition to parameters, all model families have hyperparameters that determine the form of the model. In fact, the model family is itself a hyperparameter. Unlike parameters, hyperparameters cannot be learned directly from the data. They may be set manually by the practitioner or determined via a procedure such as grid search, random search, or Bayesian optimization[62]. For example, the hyperparameters for a neural network include the number, size, and connectivity of each layer. The vectorization method is also a hyperparameter. Even modest changes in the values of hyperparameters can considerably affect accuracy, and the selection of optimal values is often computationally intensive, as each set of hyperparameters requires training of a new model.

For model comparison and the selection of hyperparameters, the data that remain after removal of the test set should be further split into a training set and a validation set. The training set is used to learn parameters, and the validation set is used to choose 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 other subsets. The average accuracy across the withheld subsets provides an estimate of test accuracy. Cross-validation provides a better estimate of the test error than the use of a constant validation set, but it requires more training time.

The datasets should be split into training, validation, and test sets to allow an accurate estimate of the model's performance under the conditions in which it will be used. For datasets from mutagenesis studies, which tend to be small and accumulative, the best practice is to train on variants characterized in earlier rounds of mutagenesis and to evaluate model performance on later rounds (to recapitulate the iterative engineering process). When dealing with large, diverse datasets, 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 to test the model's ability to generalize to unrelated sequences.

Vector representations of proteins

Machine-learning models act on vectors of numbers, not directly on protein sequences. How each protein sequence is vectorized determines what can be learned[63,64]. A protein sequence is a string of length L in which each residue is sampled 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, this 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 series of $A - 1$ zeros and one 1, with the position of the 1 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

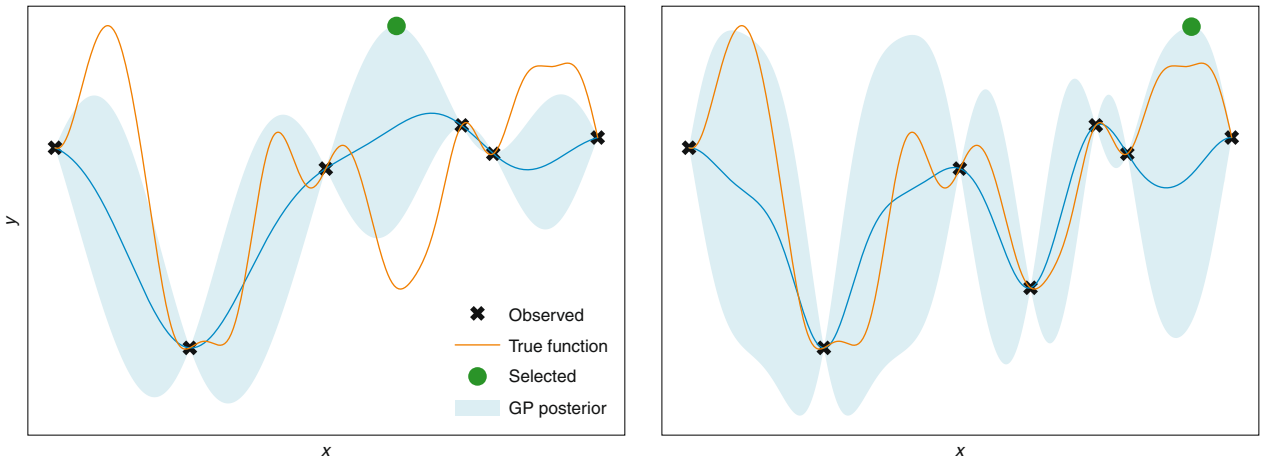

Fig. 3 | GP-UCB algorithm. At each iteration, the next point to be sampled is chosen on the basis of the maximized weighted sum of the posterior mean and s.d. This balances exploration and exploitation by exploring points that are uncertain and have a high posterior mean. The right-hand panel shows the posterior mean and s.d. after observation of the selected point (green) in the left-hand panel.

also be one-hot encoded[11,12]. One-hot encodings are inherently sparse, memory inefficient, and high-dimensional, and presuppose no notion of similarity between sequence or structural elements. Nevertheless, one-hot encodings can be considered a good baseline encoding.

Proteins can also be encoded on the basis of physical properties. For example, each amino acid can be represented by its charge, volume, or hydrophobicity, and each protein can be represented by a combination of these properties. Higher-level properties, such as predicted secondary structure, can also be used. AAIndex[65] and ProFET[66] have large collections of physical descriptors for protein sequences. There have also been attempts to reduce each amino acid to two dimensions on the basis of volume and hydrophobicity[67] or to combine physical properties with structural information[36,68] by encoding each position as a combination of the amino acids in its geometric neighborhood. However, the molecular properties that dictate each functional property are typically unknown a priori.

While many protein sequences have been deposited in databases, most are unlabeled. These unlabeled sequences contain information about the distribution of amino acids selected by evolution to compose proteins, which may be helpful across prediction tasks. The simplest encodings incorporating evolutionary information are BLOSUM[69] and 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[52,70–76] or from structural information[77]. These representations learn to place similar sequences 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 for all 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 for comparison. 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[74,75]. The encoding should ultimately be chosen empirically to maximize predictive performance.

Using sequence–function predictions to guide exploration

Once a sequence–function model has been trained, the next set of sequences to be screened can be chosen via the collection of beneficial mutations or direct optimization over sequences. For the former, linear models of the mutational effects can be learned and directly interpreted in order to classify mutations as beneficial, neutral, or deleterious. Mutations can then be fixed, eliminated, or reconsidered in future rounds of optimization[10]. Alternatively, the model can be used to select combinations of mutations with a high probability of improving function[14,45,78].

Optimization can also be carried out directly over sequences. This can be as simple as enumerating all the sequences considered, predicting their function, and synthesizing the best predicted variants. However, if multiple rounds of optimization are to be performed and the sequence–function model provides probabilistic predictions, Bayesian optimization provides a way to balance the use of the information learned and the exploration of unseen regions of sequence space[62]. Probabilistic predictions provide a well-calibrated measure of uncertainty: the model knows what it does not know. For example, the GP upper confidence bound (GP-UCB) algorithm balances exploration and exploitation by selecting variants that maximize a weighted sum of the predictive mean and s.d.[79] (Fig. 3). Alternatively, the model and data can be fully exploited with the GP lower-confidence-bound algorithm, which selects variants that maximize the weighted difference between the predictive mean and s.d. These approaches have been combined with structure-guided recombination to optimize cytochrome P450 thermostability[11], channelrhodopsin localization to mammalian cell membranes[12], and channelrhodopsin light-activated conductance[13]. With no high-throughput screen for channelrhodopsin properties, it would not have been possible to optimize conductance by traditional directed evolution.

Case study 1: using partial least-squares regression to maximize enzyme productivity

An early large-scale evolution campaign guided by machine learning improved the volumetric productivity of a halohydrin dehalogenase in a cyanation reaction by roughly 4,000-fold[10]. In each round of evolution, 10–30 mutations of interest were identified through traditional directed evolution methods (Fig. 4). These mutations were then randomly recombined, and from the resulting pool, a number of variants (three times the number of positions mutated) were sequenced and represented as one-hot vectors to train a partial

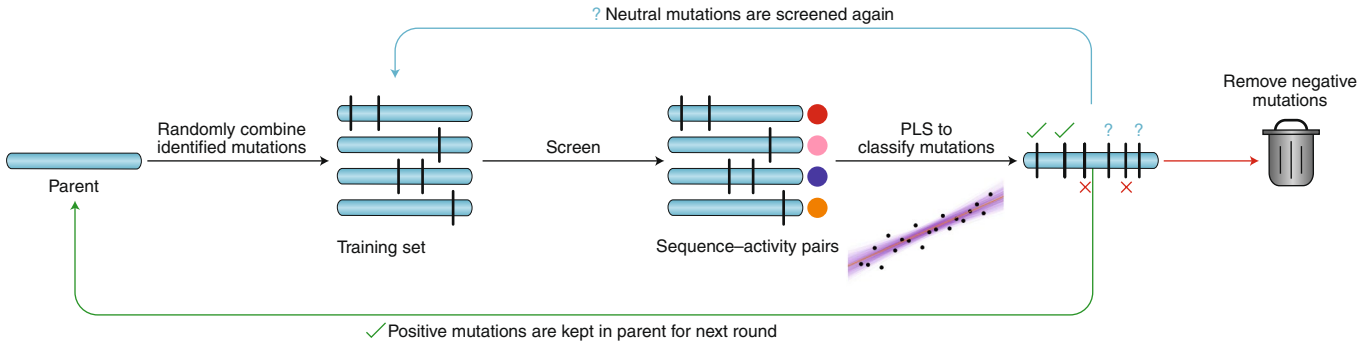

Fig. 4 | Directed evolution using PLS regression. In this approach, Fox et al. randomly recombine mutations previously identified through a classical technique such as random or site-directed mutagenesis[13]. Variants with these mutations are screened and sequenced, and the data are used to fit a linear model with the PLS algorithm. On the basis of 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. This approach improved the volumetric productivity of a protein-catalyzed cyanation reaction roughly 4,000-fold in 18 rounds of evolution.

least-squares (PLS) regression model[80]. The PLS algorithm projects both sequences and fitnesses to a space with reduced dimensions to fit the linear model. Thus, PLS is able to fit data for which the number of variables exceeds the number of observations and potentially avoid indirect correlations in the model[81]. The resulting linear model can be expressed as

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

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. Mutations were classified as beneficial, deleterious, or neutral on the basis of their PLS coefficients to determine whether each was retained, discarded, or retested. However, when the model's accuracy was low, the authors were more biased toward retesting mutations 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 protein-engineering campaigns guided by machine learning. In it, 519,045 variants were tested, of which 268,624 were used to identify mutations to model with PLS and 250,421 were chosen using PLS. In 18 rounds of optimization, none of the rounds achieved more than a threefold improvement. Machine learning sped up optimization by finding beneficial mutations that would otherwise have been obscured by their co-occurrence with deleterious mutations. However, optimization could have been accelerated by tests of different vectorization methods and machine-learning algorithms to improve model accuracy. The final linear model assumed that local regions of the sequence–function landscape display predominantly additive effects. For fitness landscapes where this is not the case, an alternative model should be used. Nevertheless, this work, which followed an in silico demonstration of the approach on a theoretical fitness landscape[81], remains a landmark effort in the application of statistical modeling to protein engineering.

Case study 2: using Bayesian optimization to maximize the thermostability of cytochrome P450

Machine learning is particularly suitable when it is expensive or difficult to screen for the property of interest. Romero et al. increased the thermostability of cytochrome P450s, as measured by T_{50} (the temperature at which an enzyme loses half its activity after a 10-min incubation), by recombining sequence fragments from the heme domains of the bacterial cytochrome P450 proteins CYP102A1, CYP102A2, and CYP102A3[11] (Fig. 5). The sequence fragments were chosen to minimize the number of contacts broken, where contacts are amino acids within 4.5 Å of each other. Machine learning provides a benefit in such a situation because the chimeric genes must be made via direct synthesis of the DNA sequence for each construct, and the T_{50} measurement requires multiple incubations and measurements for each variant and is relatively low throughput.

Romero et al. trained initial GP models for T_{50} and the presence or absence of function on 242 chimeric P450s, and then evaluated model performance on a test set of chimeric P450s generated with different boundaries between sequence fragments. The GP model used a one-hot representation of the protein's three-dimensional structure, which was more predictive than a one-hot representation of the primary sequence. GP models are a good fit for this problem setting in which only a small amount of data is available. These models also provide probabilistic predictions, which can be used to guide data-efficient exploration and optimization.

After validating the accuracy of their models, Romero et al. selected a small set of additional sequences to augment their models' knowledge of the recombination landscape. In general, one does this by selecting the sequences that most reduce uncertainty in the predictions, as 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 nonfunctional and therefore provide no information about T_{50}, Romero et al. used their classification model to select 30 additional sequences by maximizing the expected mutual information[11]. Informally, these 30 sequences combined a high probability of being functional with high sequence diversity, and 26 of these sequences were functional despite being on average 106 mutations from the closest parent. This demonstrates the ability of a machine-learning model to efficiently explore diverse sequences while minimizing the resources wasted on screening of nonfunctional proteins.

With sufficient training data collected, the authors then used Bayesian optimization to search for more thermostable variants. First, four rounds of the batch GP upper-bound algorithm yielded a diverse sampling of thermostable P450s. However, because none of these variants increased the maximum observed T_{50}, Romero et al. checked their sequence–function model by screening a sequence predicted to be stabilized with high certainty. Two additional iterations

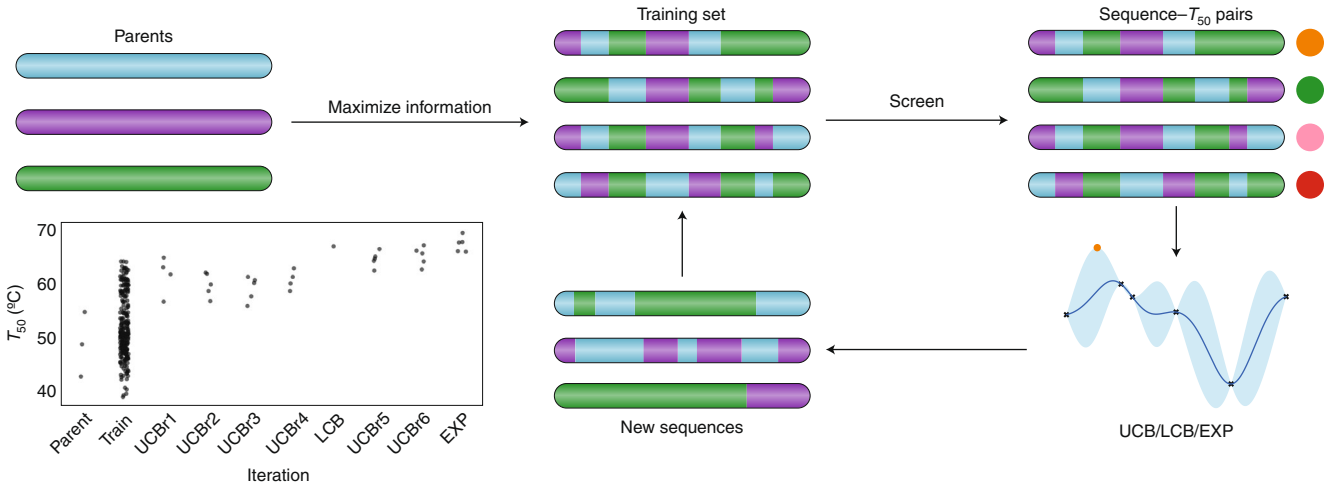

Fig. 5 | Directed evolution using GPs and Bayesian optimization. After the selection of an initial training set chosen to be maximally informative, subsequent batches of sequences are chosen using the GP upper confidence bound (UCB) or lower confidence bound (LCB) algorithm, or to fully exploit the model (EXP). The plot shows the T_{50} for variants found in each round.

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 demonstrated a framework for data-efficient protein engineering that has since been transferred to other protein systems and properties[12,13].

Conclusions and future directions

Machine-learning methods have demonstrated utility in directed protein evolution. However, for broader applications of machine learning, scientists will have to take advantage of unlabeled protein sequences or sequences labeled for properties other than those of specific interest to the protein engineer. Databases such as UniProt[82] 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 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[83].

These large quantities of sequence data may also enable machine-learning models to generate artificial protein diversity leading to novel functions. Only a tiny fraction of the amino acid landscape encodes functional proteins, and the complete landscape contains cliffs and holes where small changes in sequence result in a complete loss of function. Natural and designed proteins are samples from the distribution of functional proteins, although these samples are biased by evolutionary constraints. A method for sampling from the distribution of functional proteins would enable large jumps to previously unexplored sections of sequence space. Generative models of the distribution of functional proteins provide such a tool, and are an attractive alternative to de novo design methods[84].

Unlike discriminative models that learn the probability $p(y|x)$ 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. Generative models in other fields have been trained to generate faces[85], sketches[86], and even music[87]. Instead of using neural network models to directly learn the mapping from protein sequence to function, variational autoencoders can be trained to learn the distribution of allowed

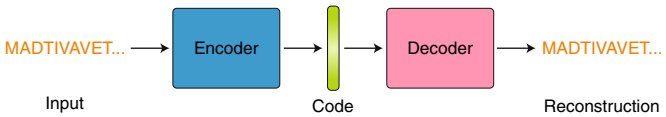

Fig. 6 | Autoencoder. An autoencoder consists of an encoder model and a decoder model. The encoder converts the input to a low-dimensional vector (code). The decoder reconstructs the input from this code. Typically, the encoder and decoder are both neural network models, and the entire autoencoder model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

mutations within functional protein families[88,89]. 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) (Fig. 6). By learning an encoding with smaller dimensionality than the original input, the model extracts the most important information from the input. 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[90]. Applied to protein sequences, variational autoencoders can learn complex epistatic relationships among variants, thus allowing predictions of variant functionality based only on existing sequences, without a need for individual measurements[88,89].

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[91]. These can be starting points for further engineering, or the generative model itself can be tuned in silico to produce sequences with a desired property. Recently, recurrent neural networks have generated novel antimicrobial peptides[92,93] and protein structures[94], and there has been an effort to develop a mathematical framework for shifting 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 desired functions, this remains a promising and largely unexplored field.

Machine-learning methods have already expanded the number of proteins and properties that can be engineered by directed evo-

lution. However, advances in both computational and experimental techniques, including generative models and deep mutational scanning[97], will 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 with which to extract knowledge from protein data and engineer proteins for novel functions.

Received: 25 October 2018; Accepted: 17 June 2019;
Published online: 15 July 2019

References

1. Dou, J. et al. Sampling and energy evaluation challenges in ligand binding protein design. *Protein Sci.* **26**, 2426–2437 (2017).
2. Garcia-Borras, M., Houk, K. N. & Jiménez-Osés, G. Computational design of protein function. In *Computational Tools for Chemical Biology* (ed. Martín-Santamaría, S.) 87–107 (Royal Society of Chemistry, 2017).
3. Mandecki, W. The game of chess and searches in protein sequence space. *Trends Biotechnol.* **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**, 563–564 (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**, 866–876 (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. Fox, R. J. et al. Improving catalytic function by ProSAR-driven enzyme evolution. *Nat. Biotechnol.* **25**, 338–344 (2007).
11. 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.
12. 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).
13. Bedbrook, C. N., Yang, K. K., Robinson, J. E., Gradinaru, V. & Arnold, F. H. Machine learning-guided channelrhodopsin engineering enables minimally-invasive optogenetics. Preprint at https://www.biorxiv.org/content/10.1101/565606v1 (2019).
This paper demonstrates the utility of machine learning for optimizing a property that would not be possible to engineer with directed evolution alone.
14. 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* **116**, 8852–8858 (2019).
15. Hastie, T. & Tibshirani, R. *The Elements of Statistical Learning: Data Mining, Inference and Prediction* (Springer, 2008).
16. Murphy, K. *Machine Learning, a Probabilistic Perspective* (MIT Press, 2012).
Murphy's textbook provides a thorough introduction to modern machine learning.
17. Liao, J. et al. Engineering proteinase K using machine learning and synthetic genes. *BMC Biotechnol.* **7**, 16 (2007).
18. Govindarajan, S. et al. Mapping of amino acid substitutions conferring herbicide resistance in wheat glutathione transferase. *ACS Synth. Biol.* **4**, 221–227 (2015).
19. 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).
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**, 1051–1056 (2007).
22. Breiman, L. *Classification and Regression Trees* (Routledge, 2017).
23. Breiman, L. Random forests. *Mach. Learn.* **45**, 5–32 (2001).
24. Friedman, J. H. Stochastic gradient boosting. *Comput. Stat. Data Anal.* **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 Bioinforma.* **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. *Mach. Learn.* **20**, 273–297 (1995).
29. Nadaraya, E. On estimating regression. *Theory Probab. Its 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.* **2002**, 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 Bioinforma.* **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. Jr. 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 *Proc. 32nd International Conference on Machine Learning* (eds. Bach, F. & Blei, D.) 1775–1784 (JMLR, 2015).
42. Wang, K. A. et al. Exact Gaussian processes on a million data points. Preprint at https://arxiv.org/abs/1903.08114 (2019).
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 (2014).
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.* **7**, 2014–2022 (2018).
46. Zhang, S. et al. A deep learning framework for modeling structural features of RNA-binding protein targets. *Nucleic Acids Res.* **44**, e32 (2016).
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**, 831–838 (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. Preprint at https://www.biorxiv.org/content/early/2017/12/24/239236 (2017).
50. Jiménez, J., Doerr, S., Martínez-Rosell, G., Rose, A. S. & 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. Preprint at https://arxiv.org/abs/1703.10603 (2017).
52. Mazzaferro, C. Predicting protein binding affinity with word embeddings and recurrent neural networks. Preprint at https://www.biorxiv.org/content/early/2017/04/18/128223 (2017).
53. Khurana, S. et al. DeepSol: a deep learning framework for sequence-based protein solubility prediction. *Bioinformatics* **34**, 2605–2613 (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 Genom.* **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. Preprint at https://arxiv.org/abs/1412.7828 (2014).
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. Bileschi, M. L. et al. Using deep learning to annotate the protein universe. Preprint at https://www.biorxiv.org/content/10.1101/626507v3 (2019).
61. Hopf, T. A. et al. Three-dimensional structures of membrane proteins from genomic sequencing. *Cell* **149**, 1607–1621 (2012).
62. Snoek, J., Larochelle, H. & Adams, R. P. Practical Bayesian optimization of machine learning algorithms. In *NIPS '12: Proceedings of the 25th International Conference on Neural Information Processing Systems* (eds. Pereira, F. et al.) 2951–2959 (Curran Associates, 2012).
63. Domingos, P. A few useful things to know about machine learning. *Commun. ACM* **55**, 78–87 (2012).
64. Bengio, Y., Courville, A. & Vincent, P. Representation learning: a review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.* **35**, 1798–1828 (2013).
65. Kawashima, S. et al. AAindex: amino acid index database, progress report 2008. *Nucleic Acids Res.* **36**, D202–D205 (2008).
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. Preprint at https://arxiv.org/abs/1701.06279 (2017).
72. Kimothi, D., Soni, A., Biyani, P. & Hogan, J. M. Distributed representations for biological sequence analysis. Preprint at https://arxiv.org/abs/1608.05949 (2016).
73. Yang, K. K., Wu, Z., Bedbrook, C. N. & Arnold, F. H. Learned protein embeddings for machine learning. *Bioinformatics* **34**, 2642–2648 (2018).
74. Schwartz, A. S. et al. Deep semantic protein representation for annotation, discovery, and engineering. Preprint at https://www.biorxiv.org/content/early/2018/07/10/365965 (2018).
75. Alley, E. C., Khimulya, G., Biswas, S., AlQuraishi, M. & Church, G. M. Unified rational protein engineering with sequence-only deep representation learning. Preprint at https://www.biorxiv.org/content/10.1101/589333v1 (2019).
76. Rives, A. et al. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. Preprint at https://www.biorxiv.org/content/10.1101/622803v2 (2019).
77. Bepler, T. & Berger, B. Learning protein sequence embeddings using information from structure. *Seventh International Conference on Learning Representations* https://openreview.net/forum?id=SygLehCqtm (2019).
78. Yang, K. K., Chen, Y., Lee, A. & Yue, Y. Batched stochastic Bayesian optimization via combinatorial constraints design. *Proc. Mach. Learn. Res.* **89**, 3410–3419 (2019).
79. Srinivas, N., Krause, A., Kakade, S. M. & Seeger, M. Gaussian process optimization in the bandit setting: no regret and experimental design. In *Proc. 27th International Conference on Machine Learning* (eds. Fürnkranz, J. & Joachims, T.) 1015–1022 (Omnipress, 2010).
80. 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.
81. de Jong, S. Simpls: an alternative approach to partial least squares regression. *Chemom. Intell. Lab. Syst.* **18**, 251–263 (1993).
82. The UniProt Consortium. UniProt: the universal protein knowledgebase. *Nucleic Acids Res.* **45**, D158–D169 (2017).
83. Pan, S. J. & Yang, Q. A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.* **22**, 1345–1359 (2010).
84. Baker, D. An exciting but challenging road ahead for computational enzyme design. *Protein Sci.* **19**, 1817–1819 (2010).
85. Radford, A., Metz, L. & Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. Preprint at https://arxiv.org/abs/1511.06434 (2015).
86. Ha, D. & Eck, D. A neural representation of sketch drawings. *Sixth International Conference on Learning Representations* https://openreview.net/forum?id=Hy6GHpkCW (2018).
87. Roberts, A., Engel, J., Raffel, C., Hawthorne, C. & Eck, D. A hierarchical latent vector model for learning long-term structure in music. Preprint at https://arxiv.org/abs/1803.05428 (2018).
88. Sinai, S., Kelsic, E., Church, G. M. & Nowak, M. A. Variational auto-encoding of protein sequences. Preprint at https://arxiv.org/abs/1712.03346 (2017).
89. Riesselman, A. J., Ingraham, J. B. & Marks, D. S. Deep generative models of genetic variation capture the effects of mutations. *Nat. Methods* **15**, 816–822 (2018).
This study predicts the effects of mutations without using any labeled data.
90. Kingma, D. P. & Welling, M. Auto-encoding variational Bayes. Preprint at https://arxiv.org/abs/1312.6114 (2014).
91. Costello, Z. & Garcia Martin, H. How to hallucinate functional proteins. Preprint at https://arxiv.org/abs/1903 (2019).
92. Müller, A. T., Hiss, J. A. & Schneider, G. Recurrent neural network model for constructive peptide design. *J. Chem. Inf. Model.* **58**, 472–479 (2018).
93. Gupta, A. & Zou, J. Feedback GAN (FBGAN) for DNA: a novel feedback-loop architecture for optimizing protein functions. Preprint at https://arxiv.org/abs/1804.01694 (2018).
94. Anand, N. & Huang, P. Generative modeling for protein structures. In *Advances in Neural Information Processing Systems 31* (eds. Bengio, S. et al.) 7504–7515 (Curran Associates, 2018).
95. Brookes, D. H. & Listgarten, J. Design by adaptive sampling. Preprint at https://arxiv.org/abs/1810.03714 (2018).
96. Brookes, D. H., Park, H. & Listgarten, J. Conditioning by adaptive sampling for robust design. *Proc. Mach. Learn. Res.* **97**, 773–782 (2019).
97. Fowler, D. M. & Fields, S. Deep mutational scanning: a new style of protein science. *Nat. Methods* **11**, 801–807 (2014).

Acknowledgements

The authors thank Y. Chen, K. Johnston, B. Wittmann, and H. Yang for comments on early versions of the manuscript, as well as members of the Arnold lab, J. Bois, and Y. Yue for general advice and discussions on protein engineering and machine learning. This work was supported by the US 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.).

Author contributions

K.K.Y., Z.W., and F.H.A. conceptualized the project. K.K.Y. wrote the manuscript with input and editing from all authors.

Competing interests

The authors declare no competing interests.

Additional information

Reprints and permissions information is available at www.nature.com/reprints.

Correspondence should be addressed to F.H.A.

Peer review information: Nina Vogt was the primary editor on this article and managed its editorial process and peer review in collaboration with the rest of the editorial team.

Publisher's note: Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.