Forward and Inverse Problems in Machine Learning Approaches to Biological Sequence Design

by

David Henry Brookes

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Biophysics

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Jennifer Listgarten, Co-chair
Professor David F. Savage, Co-chair
Professor Yun S. Song
Professor Sergey Levine

Spring 2021

Forward and Inverse Problems in Machine Learning Approaches to Biological Sequence Design

Abstract
Forward and Inverse Problems in Machine Learning Approaches to Biological Sequence Design
by
David Henry Brookes
Doctor of Philosophy in Biophysics
University of California, Berkeley
Professor Jennifer Listgarten, Co-chair
Professor David F. Savage, Co-chair

Advances in high-throughput experimental technologies now allow for the probing of the fitness of up to millions of biological sequences. Parallel developments in machine learning (ML) techniques set the stage for a new class of ML-based computational approaches to biological sequence engineering. These approaches can largely be regarded as solving one of two related problems: forward and inverse problems. In a forward problem, the aim is to estimate a fitness function from high-throughput experimental data. In an inverse problem, the aim is to use a predictive model of fitness to design a sequence or library of sequences that satisfies a particular engineering goal. In this dissertation, we will discuss various aspects of both of these sets of problems, and propose solutions for certain scenarios.

First, we present a method for solving one instantiation of an inverse problem. This method, Conditioning by Adaptive Sampling (*CbAS*), is designed to optimize models of protein fitness when those models make biased predictions in regions of sequence space far from the set of training data. *CbAS* is an example of a broad class of optimization methods known as Estimation of Distribution Algorithms (EDAs). We show that many EDAs can be viewed as a form of Expectation-Maximization (EM), a popular inference algorithm in probabilistic ML.

Next, we explore the forward problem in more depth. It is an open problem to characterize how much data is needed for accurate estimation of fitness functions. There is a growing body of evidence demonstrating that empirical fitness functions display substantial sparsity when represented in terms of epistatic interactions. Moreover, the theory of Compressed Sensing provides scaling laws for the number of samples required to exactly recover a sparse function. Motivated by these results, we study the sparsity of fitness functions sampled from a generalization of the NK model, a widely-used random field model for fitness functions. In particular, we present theoretical results that allow us to test the effect of the Generalized NK (GNK) model's interpretable parameters—sequence length, alphabet size, and assumed interactions between sequence positions—on the sparsity of fitness functions sampled from the model, and use these results to determine the number of measurements required to exactly recover GNK fitness functions. Further, we show that GNK fitness functions with parameters set according to protein structural contacts accurately approximate the sparsity of empirical fitness functions and can be used to approximate the number of experimental measurements required to effectively estimate a protein fitness function.

Finally, we describe an end-to-end modeling approach for designing a library of insertion sequences to the Adeno-Associated Virus (AAV) capsid with the goal of improving the ability of the capsid to fold and package the viral genome. AAV is a promising vector for delivering gene therapies and packaging is a requirement for effective delivery to a target tissue; a library with improved packaging ability is more likely to infect these tissues in a downstream experiment. Our approach to library design requires solutions to both forward and inverse problems. In particular, we first fit a predictive model to a set of high-throughput experimental data that reports on packaging fitness. We then present a novel method to design libraries that optimally balance average fitness and sequence diversity, and apply this method to the design of AAV insertion sequence libraries.

In summary, this dissertation presents a number of ML-based techniques for biological sequence engineering, as well as theoretical insights that can guide future methodological developments. This work provides further motivation for the biological community to adopt machine learning as a tool to be used alongside experiments in order to effectively engineer sequences.

Contents

List of Figures

Acknowledgments

I can hardly take sole credit for this dissertation. The work presented here is a product of a vast network of personal and professional support that I have been fortunate to enjoy.

First and foremost, I would like to thank my parents, Linc and Judy Brookes. For as long as I can remember, they have encouraged me to vigorously pursue my passions and this dissertation is a result of such a passion. I also need to thank my brother, Andrew Brookes. His friendship and love transcends thousands of miles of physical distance to provide encouragement and comfort in my times of need.

This work would not have been possible without my advisor, Professor Jennifer Listgarten, who has provided me with both the support and direction to succeed, as well as the freedom to pursue my own research interests. I am humbled to call her a mentor, a colleague, and a friend. I have been fortunate to collaborate with a number of inspirational researchers over the past few years. In particular, I thank Professor David V. Schaffer for his enthusiasm and patience during my often tumultuous attempts to incorporate machine learning into his group's experimental pipelines, Dr. Kevin Murphy for his enlightening commentary and encyclopedic knowledge of machine learning, and Dr. Amirali Aghazadeh for introducing me to the wonderful world of sparse recovery. I would also like to thank Professor Frances Arnold for providing many useful critiques of my work. I am grateful to Professors Yun S. Song, David F. Savage, and Sergey Levine for serving on my dissertation committee, and Professors Michael R. DeWeese and Priya Moorjani for serving on my qualifying exam committee. Additionally, I am indebted to my undergraduate research advisor, Professor Teresa Head-Gordon, for her mentorship and generous support.

Every idea presented in this dissertation has, at one point or another, been enthusiastically supported, criticized or otherwise tinkered with by my wonderful colleagues in the ScienceML research group; I am eternally grateful to Akosua Busia, Clara Wong-Fannjiang, Hunter Nisonoff and Chloe Hsu for their brilliance and kindness. I am also thankful to many members of the broader computational biology and machine learning student communities at UC Berkeley; I particularly thank Nick Bhattacharya, Neil Thomas, and Nilesh Tripuraneni for many useful and enjoyable conversations. I also thank my former colleagues in the Head-Gordon lab, Professor Luis Ruiz Pestana, Dr. Alex Albaugh and Dr. Lisa Felberg for their mentorship and friendship of a younger version of myself.

I am fortunate to have the support and love of a large extended family. I am grateful to my grandmother, Paula Freilich, whose lifelong dedication to creating a kinder world is a constant source of inspiration to me, and whose brownies are a consistent source of comfort. I am also thankful to my late uncle, Dr. Michael Freilich, who not only kindly secured me my first research job in high school, but whose passionate pursuit of scientific excellence and positive social impact has, and always will, embolden my own research pursuits.

Last, but certainly not least, I would like to thank my close friends, Sam Stowe, Caitlin Boise and Jacob Miller, for a life filled with laughter and love.

Chapter 1

Introduction

Biological sequence engineering broadly refers to the practice of constructing nucleotide or amino acid sequences that perform a particular function of interest, or improve the function of a wild type sequence. Targets for sequence engineering include naturally-occurring enzymes [95], enzymes that catalyze novel reactions [24, 147], fluorescent proteins [42], RNA and DNA catalysts [170] and aptamers [50], as well as certain viral [102, 97] and bacterial [59] genomes.

Both experimental and computational approaches to sequence engineering suffer from low-throughput in the face of combinatorial explosions in the size of sequence space. For example, the total number of possible amino acid sequences of length 100 (a relatively small protein) is $20^{100} \approx 10^{130}$, while biochemical assays in which the fitness of protein variants are tested individually can only probe at most hundreds of sequences. Similarly, attempts at *ab initio* computational screening of sequence fitness, such as energy minimization techniques [101] or Molecular Dynamics (MD) simulations [100, 58], are too computationally intensive to apply to more than a handful of variants and have historically suffered from poor accuracy [173].

Experimental techniques that are capable of screening and selecting sequences in bulk such as Directed Evolution [38, 12] and SELEX [178] substantially improve on the throughput of sequence engineering pursuits, with a modern instantiation of these methods claiming to screen 10^{14} RNA sequences [83]. One drawback of classical bulk screening methods is that the fitness of individual sequences is hidden, with typically only a handful of variants individually characterized after multiple rounds of screening and selection. However, this weakness has largely been overcome in recent years due to the advent and rapidly diminishing costs of Next Generation Sequencing (NGS) technology. NGS allows researchers to determine the number of individual sequence variants that are present in libraries before and after a bulk screen and selection occurs; the relative change in abundance of a variant between the two libraries then serves as a proxy to the fitness of that variant. Methods that employ bulk screening and selection along with NGS have broadly been termed Multiplex Assays of Variant Effects (MAVEs) [94, 52] and include Deep Mutational Scanning (DMS) [56], Massively Parallel Reporter Assays (MPRAs) [79], and modified versions of SELEX [156]. These methods have

produced data sets containing fitness measurements for up to 10^7 nucleotide sequences [134, 180] and nearly one million protein variants [137].

Parallel to these advances in data acquisition techniques has been the astonishingly rapid emergence of Machine Learning (ML) as a nearly-ubiquitous computational tool to model large data sets in scientific contexts and otherwise. This emergence has largely been due to the re-introduction of Automatic Differentiation as an efficient method for training Artificial Neural Network (ANN) models using the backpropagation algorithm [19], and the incorporation of these tools into robust software packages such as Tensorflow [1]. These developments enable rapid testing and deployment of ML models and algorithms, which are now regularly used in pursuits ranging from the processing of data at the Large Hadron Collider [138] to the creation and detection of fake news articles [196].

Unsurprisingly, ML has been applied to biological sequence engineering in a variety of ways, including to develop new knowledge-based force fields for MD simulations [194], predict protein structure directly from amino acid sequence [157], and predict the fitness effects of single mutations in protein sequences from evolutionary data alone [142]. A particularly promising application of ML to sequence engineering, and the one that motivates this dissertation, is to model the data generated by MAVEs, and use the resulting models to design sequences and libraries of sequences. There are a number of goals that motivate this type of modeling, including to find a single sequence that performs a function more effectively than any of those observed in the experimental data, or to build a library that can be used as a starting point for another iteration of bulk screening and selection.

We can broadly classify the potential uses of ML in this context as solving either a "forward" or an "inverse" problem. In a forward problem, the goal is to use data from a MAVE, and perhaps auxiliary information, to build a model (a "forward model") that can predict the fitness of sequences variants from sequence information alone. In an inverse problem, the goal is to use a forward predictive model to design sequences or a sequence library that satisfy a given engineering goal. In this dissertation, we will discuss various aspects of both of these sets of problems, and propose solutions for certain scenarios. In particular, in Chapter 2, we present a method for solving one instantiation of an inverse problem; in Chapter 3, we discuss connections between a useful class of algorithms for solving inverse problems known as Estimation of Distribution Algorithms (EDAs) and Expectation Maximization (EM), a popular inference algorithm in probabilistic machine learning; in Chapter 4, we explore the forward problem in more depth, and present results for calculating the number of experimental measurements required to estimate simulated fitness functions; in Chapter 5, we describe an end-to-end modeling approach for designing a library of Adeno-Associated Virus (AAV) capsid sequences that requires solutions to both the forward and inverse problems. In the next two sections of this chapter, we will motivate these specific explorations by describing more general aspects of the forward and inverse problems.

1.1 Forward problems

The central aim of a forward problem is to use a data set of fitness measurements, and potentially additional information, to build a model that can predict fitness from sequence. The "fitness" of a sequence refers to a numerical quantity that describes a property of sequences that we wish to maximize. The function that maps a space of sequences to fitness values is known as a fitness function, or fitness landscape. When fitness landscapes were first introduced by Sewall Wright [188], fitness referred to the ability of an organism to survive in its environment, and the fitness landscape was used as a tool to explore how an organism's genome adapted over the course of evolution. Now fitness may refer to any property related to any space of sequences; for instance, the brightness of a fluorescent protein [154], the reactivity of catalytic RNA [134], or the expression level corresponding to a promoter sequence [180].

One way to pose the forward problem is as estimating a fitness function from a set of data. There are two major types of data that are used for this purpose: so-called "labeled" data sets that contain a collection of $(\mathbf{s}, y)$ tuples, where $\mathbf{s}$ is a sequence and y is a fitness measurement associated with $\mathbf{s}$, and "unlabeled" data sets that contain only a set of sequences that have been deemed fit (e.g., all natural protein sequences that perform a given function). The common ML terminology is to say that models trained with labeled data are "supervised" while those trained with unlabeled data are "unsupervised". The forward problems approached in this dissertation exclusively use labeled data, though there is a rich area of research that we do not discuss that explores estimating fitness functions from unlabeled data [76, 142, 143, 107] and from mixtures of labeled and unlabeled data [22, 77].

Approaches to modeling of labeled fitness data

There exist a wide variety of methods for modeling labeled fitness data that have been introduced variously in the machine learning, computational biology and biophysics communities. Among these methods, the distinction between ML and non-ML methods is not exactly clear; for instance, certain biophysical models involve approximately solving for Maximum Likelihood parameter estimates with non-convex optimization methods [125], which is a core feature of many ML methods. Perhaps a better distinction is between "top-down" approaches, where the aim is to build the most powerful predictive model possible, without regard for the interpretation of the model, and "bottom-up" approaches, where a model is constructed based on assumptions about the form of the fitness function (which may be, for instance, based on biophysical models) and free parameters in the model are fit to the data.

The top-down modeling of fitness functions represents a straightforward application of supervised ML engineering procedures [114]. In a typical application of these methods, one first determines one or a few modeling metrics one wishes to optimize for (e.g., the mean-squared error between predicted and experimental fitness values), determines the model architecture (i.e., the assumed functional form of the fitness function) that optimizes these metrics using the cross-validation procedure, and then fits parameters of the ultimately chosen model to

the complete data set. An excellent review of the many examples of this top-down paradigm in fitness function estimation is given in [193]. The representational power of modern deep learning model architectures, and the empirical success of these methods in varied disciplines, suggests that top-down modeling approaches will ultimately prove to produce the most powerful models in terms of predictive performance. However, these techniques provide little, if any, meaningful biological interpretation, and modeling insights are rarely applicable across multiple studies. In contrast, bottom-up models generally have interpretable parameters that allow for biological insight and theoretical explorations.

Bottom-up modeling of fitness data involves two aspects: defining a model for the noise in the data and determining a suitable parametrized functional form for the fitness function. In the most common modeling strategy, where parameters are fit with exact or approximate Maximum Likelihood Estimation, the noise model leads to a loss function for the parameters that is minimized to find the parameter estimate that best fits the data [114]. Noise models are specific to each type of experiment, and are often inextricably linked with how one processes raw experimental data to assign fitness values to sequences. This is particularly true for MAVEs, where the raw data are sequencing reads that must be converted to scalar fitness measurements. A number of sound statistical procedures for performing this conversion in certain MAVEs have been presented that include models for the noise in the resulting fitness measurements [110, 149, 49]; more generally applicable noise models have also been considered [13]. Unlike the noise model, the functional form of the fitness function model does not depend on particular experiment used to acquire data, but rather on assumptions about how sequence features relate to fitness. These models are generally based on a few core principles, which we describe next.

Bottom-up models of fitness functions

In order to describe the various assumptions and interpretations of bottom-up models of fitness functions, for the remainder of this section we will consider fitness functions of binary sequences where each sequence position is either equal to -1 or 1. The simplest practical model of fitness for binary sequences of length L is the independent site model, where the evaluation of the model fitness function, f on a sequence, $\mathbf{s} = [s_1, s_2, ..., s_L]$, is given by:

$$f(\mathbf{s}) = \beta_0 + \beta_1 s_1 + \beta_2 s_2 + ... + \beta_L s_L \tag{1.1}$$

where β_0 is a constant term, and each β_i is a parameter that represents the effect position i has on the fitness. This model encodes the assumption that sequence positions independently contribute to fitness, with no interactions between positions. Despite its simplicity, this model can be surprisingly effective in modeling certain fitness functions [154, 77] (another example is shown in Chapter 5). However, models that either implicitly or explicitly encode interactions between sequence positions nearly always outperform the independent site model. Direct interactions between sequence positions are known as 'epistatic' interactions. A fitness function model that contains all possible epistatic interactions between pairs

of positions is given by

$$f(\mathbf{s}) = \beta_0 + \sum_{i=1}^{L} \beta_i s_i + \sum_{i=1}^{L} \sum_{j=1}^{L} \beta_{ij} s_i s_j \tag{1.2}$$

where each β_{ij} is a parameter that represents effect of the epistatic interaction between positions i and j on fitness.[1] Similarly, 'higher-order' epistasis between three or more positions can be encoded by including polynomial terms of the form $\beta_U \prod_{i \in U} \mathbf{s}_i$, where $U \subseteq \{1, 2, ..., L\}$ is a set of unique position indices and β_U is a parameter representing the effect of interaction between sequence positions in U on fitness. Although more expressive than the independent site model, models that contain pairwise and third-order interactions have been shown to make biased predictions when fit to both simulated and experimental fitness data [127]. This suggests that either higher-order epistatic interaction terms, or an alternative solution, are required to effectively model fitness functions.

One alternative solution to including many higher-order epistatic terms is to apply a monotonic nonlinear function to either the independent site model in Equation (1.1), or the pairwise interaction model in Equation (1.2). These models reflect the assumption that certain fitness effects are associated with global trends rather than sequence-specific traits, and thus are often called 'global epistasis' models [126, 94]. An example of a global trend is 'diminishing returns' in fitness functions, where fitness improvements to a sequence tend to decrease when successive beneficial mutations are made to the sequence, regardless of the identity of the specific mutations [177]. In certain cases, the nonlinearities associated with global epistasis are based on biophysical models and do not contain additional parameters [190, 125]. In the other cases, the nonlinearities are themselves parameterized and fit to data [126, 175]. These models can often dramatically improve fitness predictions compared to the underlying models that the nonlinearities are applied to [126], and yet are still highly interpretable (in contrast to other nonlinear models such as neural networks). Still, there is a growing body of evidence demonstrating that natural fitness functions contain higher-order epistatic that cannot be explained by global epistasis [152, 136], and thus a more general model is required in many scenarios.

A fitness function model that encompasses all others is one that includes all possible epistatic interactions of all orders. More specifically, any fitness function of binary sequences of length L can be represented exactly as [5]:

$$f(\mathbf{s}) = \sum_{U \in \mathcal{U}} \beta_U \prod_{i \in U} s_i \tag{1.3}$$

where $\mathcal{U} \coloneqq \mathscr{P}(\{1, ..., L\})$ is the power set of position indices in the sequence. This representation of a fitness function is known variously as the Walsh [73], Walsh-Hadamard (WH) [135, 5], or Fourier [183, 168] expansion of the fitness function. From a practical point of

[1] There are a number of alternative definitions of epistasis that are not captured by Equation (1.2). However, it has been shown that all definitions of epistasis are related by simple linear transformations [135].

view, the advantage of this model is that it can not be misspecified, meaning that if the parameters in Equation (1.3) can be estimated exactly, then there will be no bias in the fitness predictions from the model. Further, it can be shown that Equation (1.3) is an expansion in terms of an orthogonal basis, which allows one to apply powerful estimation techniques from the field of signal processing that carry strong guarantees in terms of the amount of data that is required to estimate the function with a certain amount of error (this perspective is explored in depth Chapter 4). The major practical disadvantage of modeling fitness functions in terms of a WH expansion is that the model contains 2^L parameters, whose estimation quickly becomes infeasible for longer sequences. Despite this, the WH expansion provides a convenient and intuitive exact representation of fitness functions that can be used to gain a greater understanding of fitness function estimation, either through empirical or theoretical explorations.

Our approaches for tackling forward problems in this dissertation draw from many aspects of those discussed so far in this section. In Chapter 5 we use a mixture of top-down and bottom-up strategies to model MAVE data relating the sequencs of 7 amino acid insertions to the AAV capsid to the ability of the resulting capsids to fold properly and package the viral genome. More specifically, we define a bottom-up model for the noise in the MAVE data, and use the corresponding loss function to fit a neural network predictive model, the form of which was chosen based on a top-down approach. We also fit three bottom-up models to the data, including the independent site and pairwise epistasis models analogous to those of Equations (1.1) and (1.2), respectively, and show that the neural network outperforms these models in terms of certain metrics.

Despite the plethora of available methods for estimating fitness functions from labeled data, many questions about fitness function estimation remain unanswered. In Chapter 4 we attempt to elucidate one such question, namely how many labeled experimental measurements are required to effectively estimate a given fitness function. We do so by exploring the sparsity of fitness functions represented in the WH expansion, and analogous expansions for sequences with non-binary alphabets. It has been observed that natural fitness functions can be quite sparse in the WH representation and further, theoretical results from the sub-field of signal processing known as Compressed Sensing [34] provide scaling laws for the number of samples required to exactly recover a sparse function when the function is represented in terms of an orthogonal basis. These results suggest that by studying the sparsity of fitness functions in more depth we may be able to predict the sample complexity of fitness function estimation. Although an increasing number of nearly combinatorially-complete empirical fitness functions are available that could allow us to investigate sparsity in particular example systems, these data necessarily only report on short sequences in limited environments. To explore sparsity in more generality, we study simulated fitness functions sampled from "random field" models of fitness [168], which are often used in evolutionary biology to overcome the lack of empirical fitness functions. We present theoretical results that allow us to test the effect a particular random field's interpretable parameters—sequence length,

alphabet size, and assumed interactions between sequence positions—on the sparsity of the simulated fitness functions and use these results to determine the number of measurements required to exactly recover these fitness functions. Further, we show that if the parameters of the random field are set according to protein structural contacts then the simulated fitness functions accurately approximate the sparsity of empirical fitness functions and can be used to approximate the number of experimental measurements required to effectively estimate a protein fitness function. This work provides new insights on forward problems that should be useful in guiding future directions.

Regardless of the approach, solutions to forward problems do not themselves accomplish the goals of sequence engineering. Additional techniques are required to solve inverse problems, where one aims to construct sequences or a sequence library that satisfy a desired engineering goal. An overview of these problems, and some available solutions, is given in the next section.

1.2 Inverse problems

Inverse problems are more challenging to define than forward problems, as they require one to concretely specify the goal of using computational modeling techniques within a sequence engineering pipeline. We consider two scenarios that may occur after one has collected an experimental data set of fitness measurements from a MAVE (for example). In the first case, one may not want to perform any further high-throughput experiments after this point, and would like to use the data to identify one or a few novel sequences that are more fit than those observed in the experiment. In another case, one may wish to perform further bulk screening and selection steps, and would like to construct a library of sequences that will be used as the starting point for the next iteration of experiments. We refer to these two cases as sequence and library design problems, respectively. Sequence and library design share many conceptual and technical challenges, but also differ in important respects that impact solutions to the problems. We first describe aspects of the sequence design problem, which is also the focus of Chapter 2, before moving to library design, which is discussed further in Chapter 5.

Sequence design

At first glance, the problem of identifying a sequences with high fitness appears to be a straightforward optimization problem, where one uses the given data to build a forward model, and then optimizes this model to find the sequence with maximal predicted fitness. However, an overly naive approach of this type is unlikely to be successful for two major reasons. First, the combinatorial optimization problem of trying to find the sequence with the maximal predicted fitness is NP-hard and thus no general solution exists [132]. Additionally, in many scenarios, the forward model has been trained with data that only reports on the fitness of a small restricted portion of sequence space (e.g., protein sequences within a few

mutations of a wild type), and its prediction will be arbitrarily biased far from this region [28]. In order to overcome these challenges, we must use a method that can (i) efficiently search over vast areas of sequence space to find approximate solutions to the optimization problem, and (ii) restrict the optimization procedure to search only over regions where the forward model is expected to be accurate. In Chapter 2, we will propose a method, Conditioning by Adaptive Sampling (*CbAS*) that satisfies these *desiderata* by combining the representational power of modern generative models with a novel formulation of an Estimation of Distribution Algorithm (EDA) that suitably restricts the optimization procedure to regions near the training data of the forward model [28]. In order to motivate this method, we next discuss EDAs, their conceptual predecessors Genetic Algorithms, and generative models.

Genetic Algorithms (GAs) are an historically popular class of methods for approximately solving combinatorial optimization problems [60], and thus could be used to tackle the first of the above challenges in sequence design. Roughly, GAs are algorithms that iteratively perturb a set of candidate solutions and select for perturbations with improved function evaluations. When GAs are applied to the sequence design problem, the candidate solutions are a set of sequences, the perturbations are mutations to those sequences, and the function evaluations are fitness predictions from the forward model [164]. GAs can be generalized by considering the hypothetical limit where an infinite number of candidate solutions are proposed at each iteration. Then, the set of candidate solutions can be represented as a probability distribution over the search space. This viewpoint leads to the class of methods known as Estimation of Distribution Algorithms [113, 99], where one parameterizes a probability distribution over the search space (the "search model"), samples candidate solutions from this distribution, and adjusts the distribution parameters to assign higher probability mass to regions of the search space with fit candidate solutions. EDAs have been effectively applied to a range of difficult optimization problems in biological contexts [11] and more generally [67].

EDAs have a general formulation that can be adapted to encode constraints on the optimization. *CbAS* presents one possibility for doing so, where the objective function is modified to encourage the search model to only explore regions of sequence space near the training data of the forward model (where 'near' is measured by the probability mass assigned to regions of sequence space in a probabilistic model fit to the training data). The general formulation of EDAs also connects it to a number of other methods, and these connections can be used to further adapt EDAs to new scenarios. In Chapter 3 we explore one such connection, and show that many EDAs can be viewed as performing a type of Expectation-Maximization (EM), which is a popular algorithm for inferring the parameters of latent variable models in ML.

Another intriguing possibility presented by EDAs is to supplement them with machine learning techniques by using modern generative models as search models. Generative modeling is an area of machine learning research where the aim is to fit parameterized probability distributions to a set of unlabeled data such that samples from the resulting distribution have similar characteristics to the training data. Many powerful generative models have been introduced in recent years, most notably Generative Adversarial Networks (GANs) [62] and

Variational Autoencoders (VAEs) [92, 140]. Generative models necessarily learn complex patterns in a set of unsupervised data. Therefore, by using a generative model as a search model, we can hope to learn the patterns that are present in fit candidate solutions, and generate similar solutions. We explore this possibility by using VAEs as the search model in *CbAS*.

Alternative uses of generative models in methods for sequence design have also been explored in the literature. One such use is within model-based Reinforcement Learning (RL) algorithms. Model-based RL is a sub-field of RL that attempts to infer optimal control policies for an artificial agent based on predictions from a model that maps potential actions by the agent to rewards [111]. EDAs are conceptually similar to many methods in model-based RL, though differing motivations and language in the two fields often makes direct comparison difficult. The use of model-based RL for sequence design is explored in [9]. Additionally, in Chapter 2 we compare *CbAS* to a simple model-based RL baseline.

When generative models are used in both EDAs and RL, the parameters of the model are altered at every iteration of the optimization. Another set of methods for sequence design fits a generative model to a data set of initial candidate solutions, and then leaves the model parameters fixed. In these methods, the generative model must be one that provides a low-dimensional latent representation of the data (VAEs and GANs are examples of models with such representations). Then, instead of modifying the parameters of the model to search over the space of sequences, these methods optimize over the latent space and the latent representations resulting from the optimization are then converted to sequences that represent solutions to the design problem. In some instances, the optimization over the latent space is guided by a predictive model of fitness than maps directly from the latent representations to fitness [61]; in others every candidate latent representation is converted to a sequence that is passed through a usual forward model [90]. In Chapter 2 we compare *CbAS* to a number of these techniques, as well as other baselines.

Since *CbAS* was introduced, a number of competing methods for sequence design have been proposed, including the aforementioned model-based RL method [9], a GA method [164] and a method with a fixed generative model [104]. The lack of clear consensus on the appropriate framework for tackling sequence design highlights the difficulty of this problem and the need for further exploration.

Library design

The problem of designing sequence libraries for use in downstream bulk screening and selection experiments poses added challenges to those of the sequence design problem. Before going into details about the library design problem, it is important to concretely define what we mean by a 'library' and various related notions. A library is a physical object; in particular, a pool of biological molecules, each of which is a physical manifestation of a sequence. Computational library design requires a library construction technique with tunable parameters that can be specified by a computational method. Examples include site-directed recombination, where one specifies parent sequences and the breakpoints in

those sequences at which to recombine the parents [197]; as well as methods where one specifies the desired marginal statistics of each nucleotide or amino acid at each position in the sequence; and methods where one specifies a list of particular sequences that one wishes to observe in the library. These latter two possibilities are commercially available from, e.g., Twist Biosciences.

The library design problem approached in this dissertation involves specifying the parameters of a library construction technique based on the predictions of a forward model. Similar to the sequence design problem, these types of problems involves searching over sequence space to find sequences with high predicted fitness, and ensuring that this search does not stray into regions of sequence space where the forward model is not expected to be accurate. The added difficulty in designing libraries is that the resulting library must be *diverse*, meaning that it contains sequences that are themselves distant in sequence space. Diversity in a library ensures that that the downstream selection is allowed to explore a broad area of sequence space, which makes it more likely to succeed in identifying fit sequences.

The immediate question in designing diverse libraries is how we will modify the optimization procedure to propose diverse solutions. One option that has been explored is to train a generative model in such a way that diversity is encouraged in the generated sequences [104]. Another option is to perform a multi-objective optimization, where one objective encourages the library to include sequences with high predicted fitness, and the other encourages diversity. This latter option makes clear that diversity and predicted fitness are competing goals in library design: the library with maximal average predicted fitness contains only one sequence, while the maximally diverse library contains sequences uniformly spaced over sequence space and is unlikely to be useful. The most useful library likely lies somewhere between these two extremes.

In Chapter 5 we propose a library design method that explicitly considers the tradeoff between predicted fitness and diversity. Briefly, we represent libraries as probability distributions over sequence space, and find the distribution that optimally balances expected predicted fitness and entropy, which is a natural measure of diversity for probability distributions. We then present methods for converting these abstract distributions into parameters that can be supplied to library construction techniques.

Chapter 2

A robust adaptive sampling technique for solving an inverse problem

Note: This chapter is reproduced from *David H. Brookes, Hahnbeom Park, and Jennifer Listgarten. "Conditioning by adaptive sampling for robust design". In: Proceedings of the International Conference of Machine Learning. 2019, pp. 773–782.* with permission.

2.1 Introduction

In the previous chapter, we discussed a number of different instantiations of inverse problems. In this chapter, we present a technique to solve perhaps a form of the 'sequence design' problem: given a model that predicts the probability that a given sequence satisfies a desired property, find the sequence(s) that maximize this probability. This problem reflects a goal where the resulting solutions are meant to perform a certain function, and are not necessarily meant to be used as starting points for future experimental optimization procedures. The probabilistic formulation of the problem allows it apply to a variety of circumstances. For example, the desired property may be that a sequence optimizes a particular predicted measurement of fitness (e.g., that a protein sequence maximizes predicted fluorescent brightness) or that it falls within a certain range of specified values (e.g., that a protein sequence fluoresces at a certain predicted wavelength).

This problem in its most basic form can be formally stated as follows: given an "oracle", $P(S|\mathbf{x})$, a predictive model which specifies the probability that a desired condition, S, is satisfied by a sequence $\mathbf{x}$, solve for $\mathbf{x}^* = \text{argmax}_{\mathbf{x}} P(S|\mathbf{x})$. As stated, this problem could be solved with any number of combinatorial optimization techniques; for example, the Cross Entropy Method [150] which performs well in the context of biological sequence design [27].

However, in the framing of the problem just described, there is an implicit assumption that the regression oracle is well-behaved, in the sense that it is trustworthy not only in and near the regime of inputs where it was trained, but also beyond. However, it is now well-known that many state-of-the-art predictive models suffer from pathological behaviour,

especially in regimes far from the training data [174, 119]. Methods that optimize these predictive functions directly may be led astray into areas of state space where the predictions are unreliable and the corresponding suggested sequences are unrealistic (e.g., correspond to proteins that will not fold). Therefore, we must modulate the optimization with prior information to avoid this problem. There are two related viewpoints on what this prior information represents: either as encoding knowledge about the regions where the oracle is expected to be accurate (e.g., by representing information about the distribution of training inputs), or as encoding knowledge about what constitutes a 'realistic' input (e.g., by representing proteins known to stably fold). Herein we focus on the former viewpoint, and thus assume that we have access to the input distribution of our oracle training data. However, the latter viewpoint is required when such data is not available.

How then should one use such prior knowledge? Formally, for inputs, $\mathbf{x}$ and property of interest y, we should model the joint probability $p(\mathbf{x}, y) = p(y|\mathbf{x})p(\mathbf{x})$, and then perform design (i.e., obtain one or more sequences with the desired properties) by sampling from the conditional distribution. For example, in the case of maximizing one property oracle, we should sample from $p(\mathbf{x}|y \geq y_{\max})$ to obtain our desired designed sequences. More generally, one conditions on the appropriate desired event $p(\mathbf{x}|S)$, where S is the conditioning event). To achieve this conditioning, we will assume that we have access to a property oracle, $p(y|\mathbf{x})$, which may be a black box function. We also assume that our prior knowledge has been encoded in $p(\mathbf{x})$. The prior density, $p(\mathbf{x})$, can be modelled by training an appropriate generative model, such as a Variational Auto-Encoder (VAE) [92], Real NVP [46], an HMM [18] or a Transformer [181, 143], on the chosen set of 'realistic' examples.

Outside of the above concerns, another *desideratum* for a solution approach to this inverse problem is that the oracle need not be differentiable. In other words, the oracle need only be a black box that provides an input to output mapping. Such a constraint arises from the fact that in many scientific domains, excellent predictive models already exist which may not be readily differentiable. Additionally, we may choose to use wet lab experiments themselves as the oracle. Consequently, we seek to avoid any solution that relies on differentiating the oracle; although for completeness, we compare performance to such approaches in our experiments.

In what follows, we will present a method, which we call Conditioning by Adaptive Sampling (*CbAS*), that solves the aforementioned inverse problem when oracles are black box that are assumed to be untrustworthy outside of their training domain. In particular, *CbAS* is a technique that samples from the conditional distribution discussed above in order to produce solutions to the problem that are robust to oracle pathologies. This chapter is organized as follows: we first discuss a number of methods that are closely related to *CbAS*, then we describe the *CbAS* algorithm and finally demonstrate its effectiveness with two examples: a one-dimensional toy problem and a simulated problem where the goal is to optimize the fluorescent brightness of the Green Fluorescent Protein (GFP).

2.2 Related Work

The problem set-up just described has a strong similarity to that of *activation-maximization* (AM) with a Generative Adversarial Network (GAN) prior [62, 163, 118], which is typically used to visualize what a neural network has learned, or to generate new images. Nguyen et al. [117] use approximate Langevin sampling from the conditional $p(\mathbf{x}|y = c)$, where c is be the event that $\mathbf{x}$ is an image of a particular class, such as a penguin. There are two main differences between our problem setting and that of AM. The first is that AM is conditioning on a discrete class being achieved, whereas our oracles are typically regression models. The second is that our design space is discrete, whereas in AM it is real-valued. Aside from the fact that in any case AM requires a differentiable oracle, these two differences pose significant challenges. The first difference makes it unclear how to specify the desired event to condition on, since the event may be that involving an unknown maximum. Moreover, even if one knew or could approximate this maximum, such an event would be by definition rare or never-seen, causing the relevant conditional probability to yield vanishing gradients from the start. Second, the issue of back-propagating through to a discrete input space is inherently difficult, although attempts have been made to use annealed relaxations [81]. In fact, Killoran et al. [90] adapt the AM-GAN procedure to side-step these two issues for protein design. Thus in our experiments, we compare to their variation of the AM approach.

Gómez-Bombarelli et al. [61] tackle a chemistry design problem much like our own. Their approach is to (i) learn a neural-network-supervised variational auto-encoder (VAE) latent space so as to order the latent space by the property of interest, (ii) build a (Gaussian Process) GP regression model from the latent space to the supervised property labels, (iii) perform gradient-based maximisation of the GP over the latent space, (iv) decode the optimal solution point(s) using the VAE decoder. Effectively they are approximately modelling the joint probability $p(\mathbf{x}, \mathbf{z}, y) = p(y|\mathbf{z})p(\mathbf{x}|\mathbf{z})p(\mathbf{z})$, for latent representation $\mathbf{z}$, and then finding the argument, $\mathbf{z}$, that maximizes $E[p(y|\mathbf{z})]$ using gradient descent. Similarly to AM, this approach does not satisfy our black box oracle *desideratum*. This approach in turn has a resemblance to Engel et al. [51], wherein the goal is image generation, and a GAN objective is placed on top of a VAE in order to learn latent constraints. Because this latter approach was designed for (real-valued) images and for classification labels, we compare only to Gómez-Bombarelli et al. [61].

Gupta and Zou [64] offer a solution to our problem, including the ability to use a non-differentiable oracle. They propose to first train a GAN on the initial set of realistic examples and then iterate the following procedure: (i) create a sample set by generating samples from the GAN, (ii) use an oracle regression model to make a point prediction for sample as whether or not a protein achieved some desired property, (iii) update the sample set by replacing the oldest n samples with the n samples from step 2 that exceed a user-specified threshold that remains fixed throughout the algorithm (and where n at each iteration is determined by this threshold), (iv) retrain the GAN on the updated set of samples from step 3 for one epoch. The intended goal is that as iterations proceed, the GAN will tend to produce samples which better maximize the property. They argue that because they only replace n samples at a

time, the shift in the GAN distribution is slow, enabling them to implicitly stay near the training data if not run for too long. Their procedure does not arise from any particular formalism. As such, it is not clear what objective function is being optimized, nor how to choose an appropriate stopping point to balance progress and staying near the original realistic examples.

Our approach, *CbAS*, offers several advantages over these methods. Our approach is grounded in a coherent statistical framework, with a clear objective, and explicit use of prior information. It does not require a differentiable oracle, which has the added benefit of side-stepping the need to back-propagate through to discrete inputs, or of needing to anneal an approximate representation of the inputs. Our approach is based on parametric conditional density estimation. As such, it resembles a number of model-based optimization schemes, such as Evolutionary Distribution Algorithm (EDA) and Information Geometric Optimization (IGO) approaches [66, 123], both of which have shown to have good practical performance on a wide range of optimization problems. We additionally make use of ideas from Cross Entropy Methods (CEM) for estimating rare events, and their optimization counterparts [151, 150], which allows us to robustly condition on rare events, such as maximization events.

2.3 Conditioning by Adaptive Sampling

Preamble

Our problem can be described as follows. We seek to find settings of the L-dimensional random vector, X (e.g., representative of DNA sequences), that have high probability of satisfying some property *desideratum*. For example, we may want to design a protein that is maximally fluorescent (the *maximization* problem), or that emits light at a specific wavelength (the *specification* problem). We assume that X is discrete, with realizations, $\mathbf{x} \in \mathbb{N}^L$, because we are particularly interested in problems of sequence design. However, our method is immediately applicable to $\mathbf{x} \in \mathbb{R}^L$, such as images.

We assume that we are given a scalar property predictor "oracle", $p(y|\mathbf{x})$, which provides a distribution over a property random variable, Y (herein, typically a real-valued property), given a particular input $\mathbf{x}$. From this oracle model we will want to compute the probability of various events, S, occurring. For maximization design problems, S will be the set of values y such that $y \geq y_{\max}$ (where $y_{\max} \equiv \max_{\mathbf{x}} \mathbb{E}_{p(y|\mathbf{x})}[y]$). In specification design problems, S will be the event that the property takes on a particular value, $y = y_{target}$ (strictly speaking, an infinitesimal range around it). In our development, we will also want to consider sets that correspond to less stringent criteria, such as S corresponding to the set of all y for which $y \geq \gamma$, with $\gamma \leq y_{\max}$. From our oracle model, we can calculate the conditional probability of these sets—that is, the probability that our property *desideratum* is satisfied for a given input—as $P(S|\mathbf{x}) \equiv P(Y \in S|\mathbf{x}) = \int p(y|\mathbf{x}) \mathbb{1}_S(y)\, dy$ (where $\mathbb{1}_S(y) = 1$ when $y \in S$ and 0 otherwise). For the case of thresholding a property value, this turns into a cumulative

density evaluation, $P(S|\mathbf{x}) = p(y \geq \gamma|\mathbf{x}) = 1 - CDF(\mathbf{x}, \gamma)$.

We additionally assume that we have access to a set of realizations of X drawn from some underlying data distribution, $p_d(\mathbf{x})$, which, as discussed above, either represents the training distribution of the oracle inputs, or a distribution of 'realistic' examples. Along with these data, we assume we have a class of generative models, $p(\mathbf{x}|\boldsymbol{\theta})$, that can be trained with these samples and can approximate p_d well. We denote by $\boldsymbol{\theta}^{(0)}$ the parameters of this generative model after it has been fit to the data, yielding our prior density, $p(\mathbf{x}|\boldsymbol{\theta}^{(0)})$.

Our ultimate aim is to condition this prior on our desired property values, S, and sample from the resulting distribution, thereby generating realizations of $\mathbf{x}$ that are likely to have low error in the predictive model or are realistic (i.e., are drawn from the underlying data distribution); and have high probability of satisfying our *desideratum* encoded in S. Toward this end, we will assume that the desired conditional can be well-approximated with a sufficiently rich generative model, $q(\mathbf{x}|\boldsymbol{\phi})$, which need not be of the same parametric form as the prior. As our algorithm iterates, the parameter, $\boldsymbol{\phi}$, will slowly move toward a value that best approximates our desired conditional density. Our approach has a similar flavor to that of variational inference [23], but with a critical difference of needing to handle conditioning on rare events.

Below we outline our approach in the case of maximization of a single property. Details of how to readily generalize this to the specification problem and to more than one property, including a mix of maximization and specification, are in the Supplementary Information of [28].

Our approach

Our design goal can be formalized as one of estimating the density of our prior model, conditioned on our set of desired property values, S,

$$p(\mathbf{x}|S, \boldsymbol{\theta}^{(0)}) = \frac{P(S|\mathbf{x})p(\mathbf{x}|\boldsymbol{\theta}^{(0)})}{P(S|\boldsymbol{\theta}^{(0)})}, \tag{2.1}$$

where $P(S|\boldsymbol{\theta}^{(0)}) = \int d\mathbf{x}\ P(S|\mathbf{x})p(\mathbf{x}|\boldsymbol{\theta}^{(0)})$. In general, there will be no closed-form solution to this conditional density; hence we require a technique with which to approximate it, developed herein. The problem is also harder than it may seem at first because S is in general a rare event (e.g., the occurrence of large property value we have never seen).

To find the parameters of the search model, $q(\mathbf{x}|\boldsymbol{\phi})$, we will minimize the KL divergence

between the target conditional, Eq. (2.1), and the search model,

$$\boldsymbol{\phi}^* = \underset{\boldsymbol{\phi}}{\operatorname{argmin}}\, D_{KL}\left(p(\mathbf{x}|S,\boldsymbol{\theta}^{(0)})||q(\mathbf{x}|\boldsymbol{\phi})\right) \tag{2.2}$$

$$= \underset{\boldsymbol{\phi}}{\operatorname{argmin}}\, -\mathbb{E}_{p(\mathbf{x}|S,\boldsymbol{\theta}^{(0)})}[\log q(\mathbf{x}|\boldsymbol{\phi})] - H_0 \tag{2.3}$$

$$= \underset{\boldsymbol{\phi}}{\operatorname{argmax}}\, \frac{1}{P(S|\boldsymbol{\theta}^{(0)})}\mathbb{E}_{p(\mathbf{x}|\boldsymbol{\theta}^{(0)})}[P(S|\mathbf{x})\log q(\mathbf{x}|\boldsymbol{\phi})] \tag{2.4}$$

$$= \underset{\boldsymbol{\phi}}{\operatorname{argmax}}\, \mathbb{E}_{p(\mathbf{x}|\boldsymbol{\theta}^{(0)})}[P(S|\mathbf{x})\log q(\mathbf{x}|\boldsymbol{\phi})], \tag{2.5}$$

where $H_0 \equiv -\mathbb{E}_{p(\mathbf{x}|S,\boldsymbol{\theta}^{(0)})}[\log p(\mathbf{x}|S,\boldsymbol{\theta}^{(0)})]$ is the entropy of the target conditional distribution. Neither H_0 nor $P(S|\boldsymbol{\theta}^{(0)})$ rely on $\boldsymbol{\phi}$ and thus drop out of the objective.

The objective in Equation (2.5) may seem readily solvable by drawing samples, $\mathbf{x}_i$, from the prior, $p(\mathbf{x}|\boldsymbol{\theta}^{(0)})$, to obtain a Monte Carlo (MC) approximation to the expectation in Equation (2.5); this results in a simple weighted maximum likelihood problem. However, for most interesting design problems, the desired set S will be exceedingly rare,[1] and consequently $P(S|\mathbf{x})$ will be vanishingly small for most $\mathbf{x}$ sampled from the prior. Thus in such cases, an MC approximation to the expectation in Equation (2.5) will exhibit high variance and require an arbitrarily large number of samples to calculate accurately. Any gradient-based approach, such as reparameterization or the log-derivative trick [92, 140], to try to solve the program in Equation (2.5) directly will suffer from a similar problem. To overcome this problem of rare events, we draw inspiration from CEM [151, 150, 27] to propose an iterative, adaptive, importance sampling-based estimation method.

First we introduce an importance sampling distribution, $r(\mathbf{x})$, and rewrite the objective function in Equation (2.5) as

$$\mathbb{E}_{p(\mathbf{x}|\boldsymbol{\theta}^{(0)})}[P(S|\mathbf{x})\log q(\mathbf{x}|\boldsymbol{\phi})] \tag{2.6}$$

$$= \mathbb{E}_{r(\mathbf{x})}\left[\frac{p(\mathbf{x}|\boldsymbol{\theta}^{(0)})}{r(\mathbf{x})}P(S|\mathbf{x})\log q(\mathbf{x}|\boldsymbol{\phi})\right], \tag{2.7}$$

to mitigate the problem that the expectation of $P(S|\mathbf{x})$ over our prior in Equation (2.5) is likely to be vanishingly small. Now the question remains of how to find a good proposal distribution, $r(\mathbf{x})$. Rather than finding a single proposal distribution, we will construct a series relaxed conditions, $S^{(t)}$, and corresponding importance sampling distributions, $r^{(t)}(\mathbf{x})$, such that (a) $E_{r^{(t)}(\mathbf{x})}[P(S^{(t)}|\mathbf{x})]$ is non-vanishing, and (b) $S^{(t)} \supset S^{(t+1)} \supset S$, for all t. The first condition implies that we can draw samples, $\mathbf{x}$, from $r^{(t)}(\mathbf{x})$ that have reasonably high values of $P(S^{(t)}|\mathbf{x})$. The second condition ensures that we slowly move toward our desired property condition; that is, it ensures that $S^{(t)}$ approaches S as t grows large. (In practice, we choose to use the less stringent condition $S^{(t)} \supseteq S^{(t+1)} \supseteq S$, but the stricter condition can trivially be achieved with a minor change to our algorithm, in which case one should be careful to ensure that the variance of the expectation argument does not grow too large).

[1] If not, then the design problem was an easy one for which we did not need specialized methods.

The only remaining issue is how to construct these sequences of relaxed events and importance sampling proposal distributions. Next we outline how to do so when the conditioning event is a maximization of a property.

Consider the first condition, which is that $E_{r^{(t)}(\mathbf{x})}[P(S^{(t)}|\mathbf{x})]$ is non-vanishing. To achieve this, we could (1) set $S^{(t)}$ to be the relaxed condition that $p(y \geq \gamma^{(0)}|\mathbf{x})$ where $\gamma^{(0)}$ is the Q^{th} percentile of property values predicted for those samples used to construct the prior, and (2) set the proposal distribution to the prior, $r^{(t)}(\mathbf{x}) = p(\mathbf{x}|\boldsymbol{\theta}^{(0)})$. For Q small enough, $E_{q(\mathbf{x}|\boldsymbol{\phi}^{(t)})}[P(S^{(t)}|\mathbf{x})]$ will be non-vanishing by definition and condition (a) will be satisfied. Thus, by construction, we can now reasonably perform maximization of the objective in Equation (2.7) instantiated with $S^{(t)}$ because the rare event is no longer rare. In fact, this is how we set the first tuple of the required sequence, $(S^{(0)}, r^{(t)})$. After that first iteration, we will then have solved for our approximate conditional, under a relaxed version of our property *desideratum* to obtain $q(\mathbf{x}|\boldsymbol{\phi}^{(0)})$. Then, at each iteration, we set $r^{(t)} = q(\mathbf{x}|\boldsymbol{\phi}^{(t-1)})$, and $\gamma^{(t)}$ to the Q^{th} percentile of property values predicted from the samples obtained in the $(t-1)^{\text{th}}$ iteration. By the same arguments made for the initial tuple of the sequence, we will have achieved condition (a), and condition (b) can trivially be achieved by disallowing $\gamma^{(t)}$ from decreasing from the previous iteration (i.e., , set it to the same value as the previous iteration if necessary).

Altogether now, Equation (2.7) becomes

$$\mathbb{E}_{q(\mathbf{x}|\boldsymbol{\phi}^{(t)})}\left[\frac{p(\mathbf{x}|\boldsymbol{\theta}^{(0)})}{q(\mathbf{x}|\boldsymbol{\phi}^{(t)})}P(S^{(t)}|\mathbf{x})\log q(\mathbf{x}|\boldsymbol{\phi})\right], \tag{2.8}$$

which we can approximate using MC with samples drawn from the search model, $\mathbf{x}_i^{(t)} \sim q(\mathbf{x}|\boldsymbol{\phi}^{(t)})$ for $i = 1, ..., M$, and where M is the number of samples taken at each iteration,

$$\boldsymbol{\phi}^{(t+1)} = \underset{\boldsymbol{\phi}}{\text{argmax}} \sum_{i=1}^{M} \frac{p(\mathbf{x}_i^{(t)}|\boldsymbol{\theta}^{(0)})}{q(\mathbf{x}_i^{(t)}|\boldsymbol{\phi}^{(t)})}P(S^{(t)}|\mathbf{x}_i^{(t)})\log q(\mathbf{x}_i^{(t)}|\boldsymbol{\phi}). \tag{2.9}$$

At last, we now have a low-variance estimate of our objective (or rather, a relaxed version of it that will get annealed), which can be viewed as a weighted maximum likelihood with weights given by $\frac{p(\mathbf{x}_i^{(t)}|\boldsymbol{\theta}^{(0)})}{q(\mathbf{x}_i^{(t)}|\boldsymbol{\phi}^{(t)})}P(S^{(t)}|\mathbf{x}_i^{(t)})$. Our objective function can now be optimized using any number of standard techniques for training generative models. (In the Supplementary Information of [28] we show how to extend our method to models that can only be trained with variational approximations to the maximum likelihood objective).

It is clear from Equation (2.9) that the variance of the MC estimate not only relies on $P(S^{(t)}|\mathbf{x})$ being non-vanishing for many of the samples but also that the density ratio, $\frac{p(\mathbf{x}|\boldsymbol{\theta}^{(0)})}{q(\mathbf{x}|\boldsymbol{\phi}^{(t)})}$ is similarly well-behaved. Fortunately, this is enforced by the weighting scheme itself, which encourages the density of the search model to remain close to the prior distribution. This is intuitively satisfying, as it shows that minimizing the KL divergence between the search

distribution and the target conditional requires balancing the maximization the probability of $S^{(t)}$ with adherence to the prior distribution.

Extension to intractable latent variable models

Our final objective in Equation (2.9) requires us to reliably evaluate the densities of the prior and search models for a given input $\mathbf{x}$. This is often not possible, particularly for latent variable models where the marginalization over the latent space is intractable. Although one might consider using an Evidence Lower Bound (ELBO) [23] approximation to the required densities, one can exploit the structure of our objective to exactly derive the needed quantity. In particular, we can maintain exactness of our objective for prior and search model densities where one can only calculate the joint densities of the observed and latent variables, namely $p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta}^{(0)})$ and $q(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta}^{(t)})$, which can be achieved only if both model's densities are defined on the same latent variable space, $\mathcal{Z}$.

This extension relies on the fact that an expectation over a marginal distribution is equal to the same expectation over an augmented joint distribution, for instance $\mathbb{E}_{p(x)}[f(x)] = \mathbb{E}_{p(x,y)}[f(x)]$. Starting with Equation (2.6) and using this fact, we arrive at an equivalent objective function,

$$\mathbb{E}_{p(\mathbf{x}|\boldsymbol{\theta}^{(0)})}[P(S^{(t)}|\mathbf{x}) \log q(\mathbf{x}|\boldsymbol{\phi})] \tag{2.10}$$

$$= \mathbb{E}_{p(\mathbf{x},\mathbf{z}|\boldsymbol{\theta}^{(0)})}[P(S^{(t)}|\mathbf{x}) \log q(\mathbf{x}|\boldsymbol{\phi})] \tag{2.11}$$

$$= \mathbb{E}_{q(\mathbf{x},\mathbf{z}|\boldsymbol{\phi}^{(t)})}\left[\frac{p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta}^{(0)})}{q(\mathbf{x}, \mathbf{z}|\boldsymbol{\phi}^{(t)})} P(S^{(t)}|\mathbf{x}) \log q(\mathbf{x}|\boldsymbol{\phi})\right]. \tag{2.12}$$

This objective can then be optimized in a similar manner to the originally presented case, only now using an MC approximation with joint samples $\mathbf{x}_i^{(t)}, \mathbf{z}_i^{(t)} \sim q(\mathbf{x}, \mathbf{z}|\boldsymbol{\phi}^{(t)})$.

Note that in the common case where both models are constructed such that they have the same density over $\mathcal{Z}$, $p(\mathbf{z})$, then (2.12) can be further simplified using $\frac{p(\mathbf{x}|\mathbf{z},\boldsymbol{\theta}^{(0)})p(\mathbf{z})}{q(\mathbf{x}|\mathbf{z},\boldsymbol{\phi}^{(t)})p(\mathbf{z})} = \frac{p(\mathbf{x}|\mathbf{z},\boldsymbol{\theta}^{(0)})}{q(\mathbf{x}|\mathbf{z},\boldsymbol{\phi}^{(t)})}$.

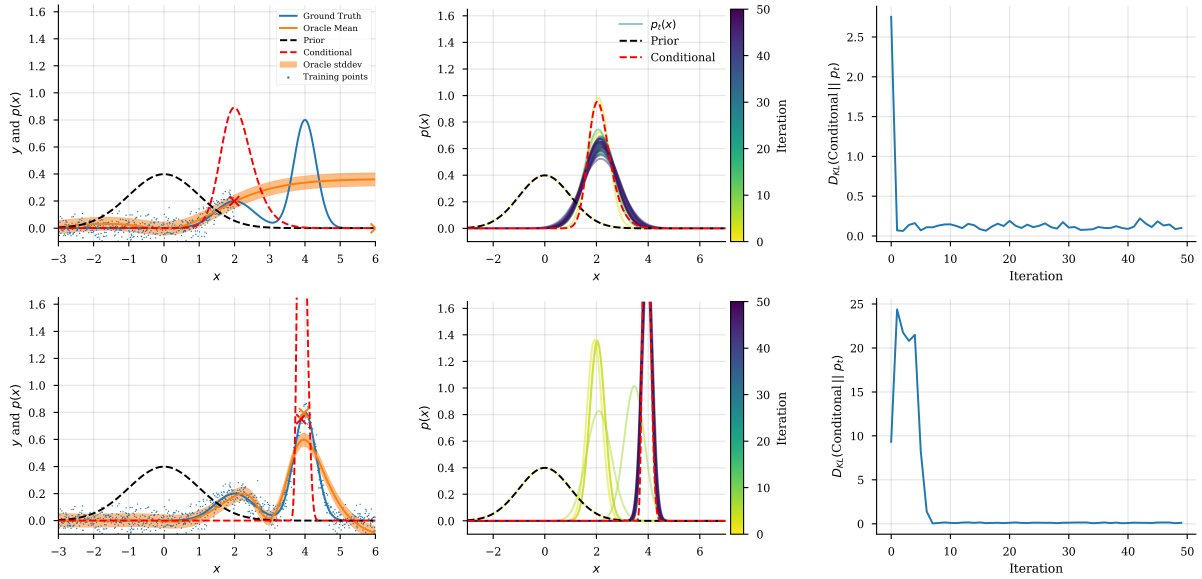
Practical Considerations

In practice, we often use the same parametric form for the search model and the prior density. This allows us to simply initialize the search distribution parameters as $\boldsymbol{\phi}^{(1)} = \boldsymbol{\theta}^{(0)}$.

Additionally, in practice we cache the search model parameters $\boldsymbol{\phi}^{(t)}$ and use these to initialize the parameters at the next time step in order to reduce the computational cost of the training procedure at each step.

2.4 Experiments

We perform two main sets of experiments. In the first, we use simple, one-dimensional, toy examples to demonstrate some of the main points about our approach. In the second



(a) **(b)** **(c)**

Figure 2.1: An illustrative example. (a) Relevant distributions and functions for two oracles (mean and standard deviation shown in orange). The oracle in the top plot was given training data corresponding to only half the domain, while the bottom one was given training data covering the whole domain. The prior conditioned on the property that the oracle is greater than its maximum is in red. The value of the ground truth at the mode of the conditional and the maximum of the oracle are shown as red and orange X's, demonstrating that the mode of the conditional distribution corresponds to a higher value of the ground truth than the maximum of the oracle (b) Evolution ('static animation') of the estimated conditional distribution as *CbAS* iterates; the exact distribution is shown in red in panel a. (c) KL divergence between the conditional and search distributions shown in (b), showing that our final approximate conditional is close to the real one.

set of experiments, we ground ourselves in a real protein fluorescence data set, conducting extensive simulations to compare methods.

An illustrative example

We first wanted to investigate the properties of our approach with a simple, visual example to understand how the prior influences the solutions found. In particular we wanted to see how an oracle could readily become untrustworthy away from the training data, and how use of the prior might alleviate this. We also wanted to see that even when the oracle is trustworthy, that our approach still yields sensible results. Finally, we wanted to ensure that our approach does indeed approximate the desired conditional distribution satisfactorily for simple cases that we can see. The summary results of this experimentation are shown in

Figure 2.1.

To run this set of experiments, we first constructed a ground truth property function, comprising the superposition of two unnormalized Gaussian bumps. The goal was to find an input, x, that maximizes the ground truth, when only the oracle and a prior are given.

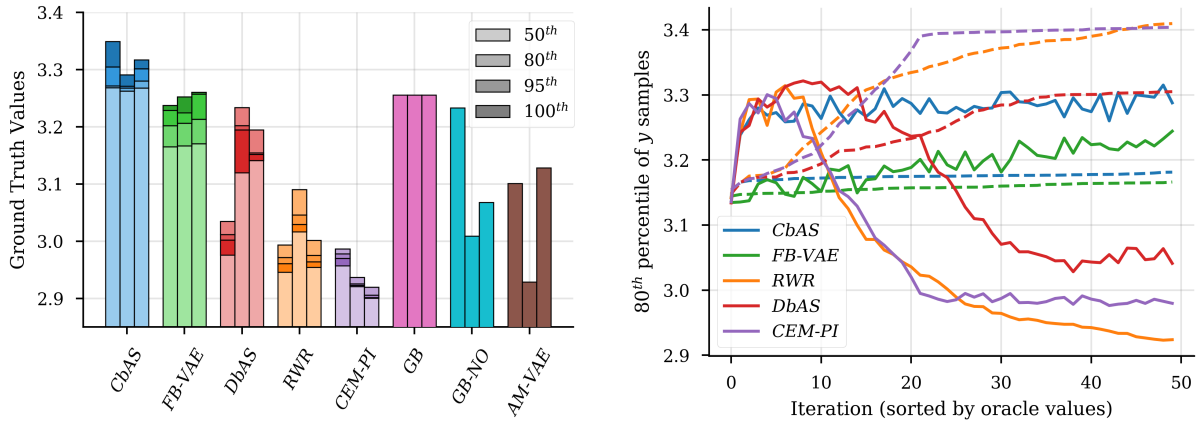
Next we created two different oracles, by giving one access only to data that covered half of the domain, and the other by giving it data that covered the entire domain (including all those in the first data set). These training data were ground truth observations corrupted with zero-mean Gaussian noise with variance of 0.05. Then two oracles were trained, one for each data set, and of the form, $p(y|x) = \mathcal{N}(\mu(x), \sigma^2)$ where $\mu(x)$ is a two hidden-layer neural network fit to one of the training sets and σ^2 was set to the mean squared error between the ground truth and $\mu(x)$ on a hold-out set.

The resulting oracles, and the underlying ground truth are shown in Figure 2.1a. The oracle trained with the smaller data set suffers from a serious pathology—it continues to increase in regions where the ground truth function rapidly decreases to zero. This is exactly the type of pathology that *CbAS* aims to overcome by estimating the conditional density of the prior rather than directly optimizing the objective. The second oracle does not suffer from as serious a pathology, and serves to show that *CbAS* can still perform well in the case that the oracle is rather accurate (i.e., the prior does not overly constrain the search). As with any Bayesian method with a prior, there may be settings where the prior could lead one astray, and this example is simply meant to convey some intuition. The next set of experiments, grounded in a protein design problem, suggest that the prior we constructed, used within *CbAS*, works well in practice.

We construct our target set S as being the set of values for which Y is greater than the maximum of the oracle's expectation, for x values between minus three and six. In this simple 1D case, we can evaluate the target conditional density very accurately by calculating $P(S|x)p_0(x)$ for many values of $\mathbf{x}$ and using numerical quadrature to estimate the normalizing constant. The target conditional is shown in red in 2.1a. We can see that in both cases, the mode of the conditional lies near a local maximum and the conditional assigns little density to regions where the oracle is highly biased.

Finally, we test the effectiveness of *CbAS* in estimating the desired conditional densities (Figure 2.1b,c). We use a search distribution that is the same parametric form as the prior, that is, a Gaussian distribution and run our method for 50 iterations, with the quantile update parameter, $Q = 1$ (meaning that $\gamma^{(t)}$ will be set to the maximum over the sampled mean oracle values at iteration t) and $M = 100$ samples taken at each iteration. Figure 2.1b shows the search distributions that result from our scheme at each iteration of the algorithm, overlayed with target conditional distribution. Figure 2.1c) shows the corresponding KL divergences between the search and target distributions as the method proceeds. We can see qualitatively and quantitatively (in terms of the KL divergence) that in both cases the distributions converge to a close approximation of the target distribution.

Application to protein fluorescence optimization



(a) **(b)**

Figure 2.2: Design for maximization of protein fluorescence. (a) For sampling-based methods, namely *CbAS*, *FB-VAE*, *DbAS*, *RWR* and *CEM-PI*, shown are the mean values of the ground truth evaluated at samples coming from different percentiles (50^{th}, 80^{th}, 95^{th}, and 100^{th}) of oracle predictions over all iterations. The gradient-based methods, *GB*, *GB-NO* and *AM-VAE*, yield only a single protein at each iteration and converge rapidly; thus only the ground truth value of the final single protein for each of these methods is used. For sampling-based methods, an optimal method has darkest bars at the top, followed by progressively less dark bars, indicating that the method has successfully avoided untrustworthy regions of the space. Methods that have the darkness out of this order are being led astray into untrustworthy oracle regions. The height of a bar shows how well the ground truth fluorescence is maximized, where higher is better. For *CbAS*, the height is highest, and the darkness ordering, correct—unsurprisingly, as avoiding oracle pathologies should help achieve higher ground truth during maximization. This trend of better darkness ordering yielding high property values holds for all sampling based methods. The sets of three bars indicate the three different oracles (each bar was averaged over three random runs). (b) For one representative run in panel a), a trajectory is shown for each method. Each point on a dashed lines shows the 80^{th} percentile of oracle evaluations of the samples at that iteration. The corresponding point on a solid line shows the mean ground truth value of those same samples. The last point on a curve shows what final protein sequence would be used from each method (i.e., that with the highest oracle value seen, as the ground truth would be unknown)—high dashed values and low solid ones are methods that have been led astray into pathological regions of the oracle.

Fluorescent proteins are a workhouse of modern molecular biology. Improving them has a long history, but also serves as a good test bed for protein optimization. Thus we here focus on the problem of maximizing protein fluorescence. In particular, we perform a systematic

comparison of our method, *CbAS*, to seven other approaches, described below. We anchored our experiments on a real protein fluorescence data set [154].

Methods considered We compared our approach to other competing methods, including, for completeness, those which can only work with differentiable oracles. For models that originally used a GAN prior, we instead use a VAE prior so as to make the comparisons meaningful.[2] We compare our method, *CbAS* , against the following methods: (i) *AM-VAE*—the activation-maximization method of Killoran et al. [90]. This method requires a differentiable oracle. (ii) *FB-VAE*—the method of Gupta and Zou [64]. This method does not require a differentiable oracle. (iii) *GB-NO*—the approach described by Gómez-Bombarelli et al. [61]. This method requires a differentiable oracle. (iv) *GB*—the approach implemented by Gómez-Bombarelli et al. [61] which has some additional optimization constraints placed on the latent space that were not reported in the paper but were used in their code. This method requires a differentiable oracle. (v) *DbAS*—a method similar to *CbAS*, but which assumes an unbiased oracle and hence has no need for or ability to incorporate a prior on the input design space. This method is an instantiation of the Cross Entropy Method designed to optimize $P(S|\mathbf{x})$ in the context of biological sequence design. (vi) *RWR*—Reward Weighted Regression [131], which is similar to *DbAS*, but without taking into account the oracle uncertainty. This method does not require a differentiable oracle. (vii) *CEM-PI*—use of the Cross Entropy Method to maximize the Probability of Improvement [165], an acquisition function often used in Bayesian Optimization; this approach does not make use of a prior on the input design space. This method does not require a differentiable oracle.

In these experiments we set the quantile update parameter of *CbAS* to $Q = 1$. However, we show in the Supplementary Information of [28] that results are relatively insensitive to the setting of Q.

Simulations In order to compare methods it was necessary to first simulate the real data with a known ground truth because evaluating hold out data from a real data set is not feasible in our problem setting. To obtain a ground truth model we trained a GP regression (GPR) model on the protein fluorescence data, with a protein-specific kernel [159], whose feature space was augmented by adding a bias feature and exponentiating to obtain a second order polynomial kernel. Our ground truth is the mean of this GPR model, which, notably, is a different model class than the oracle, as we expect in practice.

Next, for each protein sequence in the original data set we compute its ground truth fluorescence. Then we take those sequences that lie in the bottom 20^{th} percentile of ground truth fluorescence, choose 5,000 of them at random, and use these to train our oracles using maximum likelihood. We wanted to investigate different kinds of oracles with different properties, with a particular focus on investigating different uncertainty estimates. Specifically, we considered three types of oracle. The first was a single-layer neural network model with homoscedastic, Gaussian noise in its predictions. The second was an ensemble of five of these

[2] As shown in [27], the GAN and VAE appear to yield roughly similar results in this problem setting.

models, each with heteroscedastic noise, as in Lakshminarayanan et al. [98]. The third was the same as the second, but with an ensemble of 20 neural networks.

Next we train a standard VAE prior on those same 5,000 samples. This VAE gets used by *CbAS* and *AM-VAE* (as the prior) and in *FB-VAE* (for initialization). Because *GB* and *GB-NO* require training a supervised VAE, this is done separately, but with the same 5,000 samples, and corresponding ground truth values.

To fairly compare the methods we keep the total number of samples considered during each run, N, constant. We call this the sequence budget. This sequence budget corresponds to limiting the total number samples drawn from the generative model in *CbAS*, *DbAS*, *FB-VAE*, *RWR* and *CEM-PI*; and limiting the number of total gradient step updates performed in the *AM-VAE*, *GB* and *GB-NO* methods. For the latter class of methods, if the method converged but had not used up its sequence budget, then the method was re-started with a new, random initialization and executed until the sequence budget was exhausted, or a new initialization was needed. For convergence we used the built-in method that comes with *GB* and *GB-NO*, and for *AM-VAE*, convergence is defined as that the maximum value did not improve in 100 iterations.

For each of the three oracle models, we ran three randomly initialized runs of each method with a sequence budget of 10,000. Figure 2.2a shows the results from this experiment, averaged over the three separate runs. Methods that have no notion of a prior (*DbAS*, *RWR*, *CEM-PI*) are clearly led astray, as they only optimize the oracle, finding themselves in untrustworthy regions of oracle space. This suggests that our simulation settings are in the regime of the illustrative example shown in Figure 2.1a (top).

2.5 Conclusion

In this Chapter we have (i) introduced a new method that enables one to solve design problems in discrete space, and with a potentially non-differentiable forward model, (ii) we introduced a new way to perform approximate, conditional density modelling for rich classes of models, and importantly, in the presence of rare events, (iii) showed how to leverage the structure of latent variable models to achieve exact importance sampling in the presence of models whose density cannot be computed exactly. Additionally, we showed that compared to other alternative solutions to this problem, even those which require the oracle to be differential, our approach yields competitive results.

Our method, Conditioning by Adaptive Sampling, is a type of model-based optimization (MBO), where one constructs a search model and attempts to optimize the expectation of the target function over the density of the search model. A major class of model-based optimization techniques are known as Estimation of Distribution Algorithms, or EDAs. We analyze EDAs in depth in the next chapter, and in particular connect them to the Expectation-Maximization algorithm, which is well studied in probabilistic machine learning.

The core assumption underlying *CbAS* is that the given forward model is biased far from the training set. This assumption is based on the empirical observation that models trained

on experimental fitness function data tend to exhibit pathologies far from the regime in which they were trained. In Chapter 4 we will explore the estimation of fitness functions in more depth. In particular, we will attempt to determine how many experimental measurements are required to perfectly estimate fitness functions that exhibit substantial sparsity. This type of analysis could be used to refine the assumptions underlying a method such as *CbAS*.

CbAS is designed to solve a particular instantiation of the inverse problem. In Chapter 5 we will describe another inverse problem where one wishes to design an entire library of sequences that balance a trade-off between diversity and optimizing a given forward model. We will use our solution to this problem to design a library of Adeno-associated viral capsids that are more capable of properly folding and packaging the viral genome over a base library.

Chapter 3

Estimation of Distribution algorithms and connections to the Expectation-Maximization algorithm

Note: This chapter is reproduced from *David Brookes, Akosua Busia, Clara Fannjiang, Kevin Murphy, and Jennifer Listgarten. "A View of Estimation of Distribution Algorithms through the Lens of Expectation-Maximization". In: GECCO. 2020, pp. 189–190.* with permission.

3.1 Introduction

In the previous chapter, we introduced a model-based optimization (MBO) method for solving a particular instantiation of the inverse problem. In that case the search model of the MBO is used to modulate the assumed error in the given forward model. In Chapter 5 we will present an MBO technique to solve another inverse problem where we wish to design a library of amino acid sequences that are both balance the competing requirements of diversity and high average predictions in a given forward model. In this case, the search model of the MBO will represent the density of amino acid sequences in the library, which we will enforce to have high entropy (so it represents a diverse set of sequences) while also optimizing the expected predictions of the forward model. From these two examples, it is clear that MBOs represent a diverse set of methods that can be tailored to solve different inverse problems with varying assumptions and goals.

In this chapter, we will explore some fundamental properties of a particular class of MBOs in more depth, known as Estimation of Distribution Algorithms, or EDAs. In particular, we will show that EDAs can be written as a form of the famous Expectation-Maximization algorithm often used in probabilistic machine learning. Because EM sits on a rigorous statistical foundation and has been thoroughly analyzed, this connection provides a new coherent framework with which to reason about EDAs.

EDAs are a widely used class of algorithms designed to solve optimization problems of the form $\mathbf{z}^* = \text{argmax}_{\mathbf{z}\in\mathcal{Z}} f(\mathbf{z})$, where $f : \mathcal{Z} \to \mathbb{R}$ is a function over a space of discrete or continuous inputs, $\mathcal{Z}$. Instead of solving this objective directly, EDAs solve the related objective

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\text{argmax}}\, \mathbb{E}_{p(\mathbf{z}|\boldsymbol{\theta})}[f(\mathbf{z})], \tag{3.1}$$

where $p(\mathbf{z}|\boldsymbol{\theta})$ is a probability density over $\mathcal{Z}$, parameterized by D parameters $\boldsymbol{\theta} \in \mathbb{R}^D$. When $p(\mathbf{z}|\boldsymbol{\theta})$ has the capacity to represent point masses on the maxima of $f(\mathbf{z})$, then these two formulations have the same optimal values. Reasons for using the latter formulation of Equation 3.1 include convenience for derivative-free optimization [69], enabling of rigorous analysis of associated optimization algorithms [198], and the leveraging of a probabilistic formulation to incorporate auxiliary information [28]. In many cases this objective is further modified by applying a monotonic *shaping* function, $W(\cdot)$ to $f(\mathbf{z})$ [187, 186, 195], which alters convergence properties of algorithms to solve it, but does not change the optima.

The general algorithmic template for EDAs is as follows. Beginning with an initial parameter setting, $\boldsymbol{\theta}^{(0)}$, of the parametrized density, $p(\mathbf{z}|\boldsymbol{\theta})$, each iteration $t \in \{0, 1, ...T\}$ of an EDA generally consists of three steps:

1. Draw N samples, $\{\mathbf{z}_i\}_{i=1}^N$, from $p(\mathbf{z}|\boldsymbol{\theta}^{(t)})$.
2. Evaluate $W(f(\mathbf{z}_i))$ for each $\mathbf{z}_i$.
3. Find a $\boldsymbol{\theta}^{(t+1)}$ that uses the weighted samples and corresponding function evaluations to move $p(\mathbf{z}|\boldsymbol{\theta})$ towards regions of $\mathcal{Z}$ that have large function values.

The last step is generally accomplished by attempting to solve

$$\boldsymbol{\theta}^{(t+1)} = \text{argmax}_{\boldsymbol{\theta}} \sum_{i=1}^N W(f(\mathbf{z}_i)) \log p(\mathbf{z}_i|\boldsymbol{\theta}), \tag{3.2}$$

which can be seen as a weighted maximum likelihood problem with weights $W(f(\mathbf{z}_i))$. We refer to this set of steps as a "core" EDA because it ties together most EDAs.

In the case of Covariance Matrix Adaptation (CMA-ES), a particularly popular EDA [69], this core algorithm is often modified in a variety of ways to improve performance. For example, samples from previous iterations may be used, directly or indirectly, in the last step, resulting in smoothing of the parameter estimates across iterations. Adaptive setting of parameter-specific step sizes, and "path evolution" heuristics [69, 123, 28] are also common. These layers on top of the core EDA have generally been derived in a manner specific only to CMA-ES, and are not readily generalizable to other EDAs. For this reason, we restrict ourselves to the core EDA algorithm just described. However, as we shall see, our formalism also allows for a large class of EDA parameter smoothing variations which encompass many, but not all, variations of CMA-ES.

EDAs are also distinguished by the choice of parametrized density, $p(\mathbf{z}|\boldsymbol{\theta})$, which we refer to here as the "search model".[1] Many EDAs use exponential family models for the search model, most commonly the multivariate Gaussian as in CMA-ES. However, Bayesian networks [121], Boltzmann machines [78, 161], Markov Random Fields [158], Variational Autoencoder [28], and others [89] have also been used. Unless otherwise specified, our analysis in the following is quite general and makes no assumptions on the choice of search model.

In this chapter, we show that a large class of EDAs—including, but not limited to, variants of CMA-ES—can be written as Monte Carlo (MC) Expectation-Maximization (EM) [20], and in the limit of infinite samples, as exact EM [44]. Because EM sits on a rigorous statistical foundation and has been thoroughly analyzed, this connection provides a new framework with which to reason about EDAs. Within this framework, we also show how many parameter-smoothed variants of CMA-ES can be written as Monte Carlo maximum *a posteriori*-EM (MAP-EM), which suggests a possible avenue to develop similar smoothing algorithms for EDAs with other search models.

3.2 Related Work

The Information Geometry Optimization (IGO) framework [123] unifies a large class of approaches—including EDAs such as CEM [150] and CMA-ES [69]—for solving the objective in Equation (3.1) by discretizing a natural gradient flow. Instantiations of IGO are most readily seen as tantamount to using the score function estimator (sometimes referred to as the "log derivative trick") on Equation (3.1), combined with natural gradient, as in Natural Evolution Strategies [187, 186]. The IGO framework does not connect to EM; therefore, IGO provides a complementary viewpoint to that presented herein.

Akimoto et al. [7] show how CMA-ES with rank-μ updates, but without global step size and evolution paths, can be viewed as a natural evolution strategy. In this context, they briefly remark on how a simplified CMA-ES can be viewed as performing generalized EM with a partial-M step using a natural gradient. The technical underpinnings of this result are restricted specifically to CMA-ES because of the dependence on a Gaussian search model, and do not readily generalize to the broader EDA setting considered herein.

Staines *et al.*introduce the notion of variational optimization, by explicitly considering the fact that the optimum of the objective function in Equation (3.1) is a lower bound on the optimum of f. They also clearly delineate conditions under which the bound can be satiated, and the objective function is convex and has derivatives that exist [169]; no connections to EDAs are made.

[1]This object is often simply referred to as the "probability distribution" [89], however; we use "search model" to distinguish it from other probability distributions used in our formulations.

3.3 Background: Free Energy view of EM

Before deriving the connection between EM and EDAs, we first review some needed background on EM. EM is a well-known method for performing maximum likelihood parameter estimation in latent variable models that exploits the structure of the joint likelihood between latent and observed variables [44]. Intuitively, each E-step imputes the latent variables, and the subsequent M-step then uses these "filled in" data to do standard maximum likelihood estimation, typically in closed form. EM iterates between these E- and M- steps until convergence. We use the Free Energy interpretation of EM and its accompanying generalizations [115] in order to reveal the connection between EDAs and EM.

Let $\mathbf{x}$ and $\mathbf{z}$ be observed and latent variables, respectively. The task of maximum likelihood estimation in a latent variable model is to find $\hat{\boldsymbol{\phi}} = \text{argmax}_{\boldsymbol{\phi}} \mathcal{L}(\boldsymbol{\phi})$ where

$$\mathcal{L}(\boldsymbol{\phi}) = \log p(\mathbf{x}|\boldsymbol{\phi}) = \log \int p(\mathbf{x}, \mathbf{z}|\boldsymbol{\phi}) d\mathbf{z}, \tag{3.3}$$

for some model density p parameterized by $\boldsymbol{\phi}$. In [115], the authors define a function known as the free energy, given by

$$F(q, \boldsymbol{\phi}) = \mathbb{E}_{q(\mathbf{z})}[\log p(\mathbf{x}, \mathbf{z}|\boldsymbol{\phi})] + H[q], \tag{3.4}$$

where $q(\mathbf{z})$ is any probability density over the latent variables, $H[q]$ is the entropy of q, and the term preceding the entropy is known as the *expected complete log-likelihood*. The free energy lower bounds the log-likelihood, $\mathcal{L}(\boldsymbol{\phi}) \geq F(q, \boldsymbol{\phi})$, and this bound is satiated only when $q(\mathbf{z})$ is equal to the true posterior, $q(\mathbf{z}) = p(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi})$. If the true posterior cannot be calculated exactly, one may approximate it in one of several ways, two of which are described next.

In the first, the posterior is approximated by restricting it to a parameterized family of distributions, $q(\mathbf{z}|\boldsymbol{\psi})$, where both the parameters of the likelihood and the variational family must now be estimated. This is known as variational EM. Unless the true posterior lies in the parameterized class, which is not typically the case, the bound $\mathcal{L}(\boldsymbol{\phi}) \geq F(q, \boldsymbol{\phi})$ cannot be satiated, leading to a *variational gap* given by $D_{KL}(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi}))$, where D_{KL} denotes the KL divergence.

In another approximation, one draws samples from the true posterior, $\mathbf{z}_i \sim p(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi}^{(t)})$, to estimate the expected complete log likelihood. This is known as Monte Carlo EM (MC-EM), and can be seen as approximating the posterior with a mixture of weighted particles, $q(\mathbf{z}) = \frac{1}{N} \sum_i \delta_{\mathbf{z}_i}(\mathbf{z})$. This also induces a "gap" between the true and approximate posterior (because of the use of finitely many samples), which can be similarly computed as $D_{KL}(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi}))$, and hence we will refer to it also as a variational gap. One major distinction between variational and MC-EM is that variational EM explicitly attempts to minimize the variational gap at each iteration through optimization, while MC-EM only implicitly closes the gap as the number of samples goes to infinity.

All of EM, MC-EM, and variational EM can be viewed as alternating coordinate descent on the free energy function [115]. In particular, the alternating updates at iteration t are given by:

- E-step: $q^{(t+1)} \leftarrow \text{argmax}_q \, F(q, \boldsymbol{\phi}^{(t)})$, that is, compute or estimate the posterior over the hidden variables. It can be shown that this is equivalent to minimizing the variational gap by solving $q^{(t+1)}(\mathbf{z}) = \text{argmin}_q \, D_{KL}(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi}^{(t)})$.

- M-step: $\boldsymbol{\phi}^{(t+1)} \leftarrow \text{argmax}_{\boldsymbol{\phi}} \, F(q^{(t+1)}, \boldsymbol{\phi})$, which is equivalent to maximizing the expected complete log likelihood with respect to $\boldsymbol{\phi}$, $\boldsymbol{\phi}^{(t+1)} \leftarrow \text{argmax}_{\boldsymbol{\phi}} \, \mathbb{E}_{q^{(t+1)}(\mathbf{z})}[\log p(\mathbf{x}, \mathbf{z}|\boldsymbol{\phi})]$. This can also be seen as solving a weighted maximum likelihood problem with data $(\mathbf{x}_i, \mathbf{z}_i)$, and weights given by by the approximate posterior, $q^{(t+1)}$.

When the variational gap can be driven to zero, as in exact EM, this procedure is guaranteed to never decrease the likelihood in Equation (3.3). Moreover, the convergence properties of exact EM to both local and global minima have been carefully studied (e.g., [44, 115, 14]). Rigorous generalizations of EM to partial E- and M-steps also emerge naturally from this viewpoint [115].

One can also use the free energy viewpoint of EM to optimize a maximum *a posteriori* (MAP) objective given by $\mathcal{L}_{MAP}(\boldsymbol{\phi}) \equiv \log p(\mathbf{x}|\boldsymbol{\phi}) + \log p_0(\boldsymbol{\phi})$, where p_0 is some prior distribution over parameters. This yields a corresponding free energy,

$$F_{MAP}(q, \boldsymbol{\phi}) \equiv \mathbb{E}_{q(\mathbf{z})}[\log p(\mathbf{x}, \mathbf{z}|\boldsymbol{\phi}) + \log p_0(\boldsymbol{\phi})] + H[q], \tag{3.5}$$

upon which one can perform the same coordinate descent algorithm just described. This formulation is referred to as MAP-EM. We will draw connections between MAP-EM and EDAs with smoothed parameter updates such as those commonly used in CMA-ES.

3.4 Formal connection between EM and EDAs

We are now ready to use the EM framework presented in Section 3.3, to show that EDAs using the update rule in Equation (3.2) can be written as MC-EM, and as exact EM in the infinite sample limit. We then show that generalizations of Equation (3.2) that allow for parameter smoothing—such as CMA-ES's smoothing with the previous estimate and use of an evolutionary path—can be readily incorporated into the EM-EDA connection by using the MAP-EM framework. Finally, we highlight a connection between EM and standard gradient-based optimization.

Derivation of EDAs as MC-EM

As described in the Introduction to this chapter, EDAs seek to solve the objective defined in Equation (3.1),

$$\hat{\boldsymbol{\theta}} \coloneqq \operatorname*{argmax}_{\boldsymbol{\theta}} \mathbb{E}_{p(\mathbf{z}|\boldsymbol{\theta})}[f(\mathbf{z})] \tag{3.6}$$

$$= \operatorname*{argmax}_{\boldsymbol{\theta}} \log \mathbb{E}_{p(\mathbf{z}|\boldsymbol{\theta})}[f(\mathbf{z})] \tag{3.7}$$

$$= \operatorname*{argmax}_{\boldsymbol{\theta}} \mathcal{L}_{EDA}(\boldsymbol{\theta}), \tag{3.8}$$

where $f(\mathbf{z})$ is the black-box function to be optimized, $p(\mathbf{z}|\boldsymbol{\theta})$ is what we refer to as the *search model*, parameterized by $\boldsymbol{\theta}$, and we define $\mathcal{L}_{EDA}(\boldsymbol{\theta}) \equiv \log \mathbb{E}_{p(\mathbf{z}|\boldsymbol{\theta})}[f(\mathbf{z})]$. This expression can be thought of as an EDA equivalent to the 'log marginal likelihood' in EM, only without any observed data, $\mathbf{x}$.

Some EDAs monotonically transform $f(\mathbf{z})$ with a shaping function, $W(\cdot)$, which may be, for example, an exponential [131], a quantile-based transformation [123, 150], or a cumulative density function (CDF) [28]. Although this transformation does not change the optima, it may alter the optimization dynamics. Often this shaping function is changed at each iteration in a sample-dependent, adaptive manner (which links these methods to annealed versions of EM and VI [75, 108]). In such a setting, the connection that we will show between EDAs and EM holds within each full iteration. For notational simplicity, we drop the $W(\cdot)$ and assume that $f(\mathbf{z})$ has already been transformed. We additionally assume that the transformation is such that $f(\mathbf{z}) \geq 0$ for all $\mathbf{z} \in \mathcal{Z}$.

To link Equation (3.8) to EM, we introduce a probability density, $q(\mathbf{z})$, which allows us to derive a lower bound on $\mathcal{L}_{EDA}$ using Jensen's inequality:

$$\mathcal{L}_{EDA}(\boldsymbol{\theta}) = \log \mathbb{E}_{p(\mathbf{z}|\boldsymbol{\theta})}[f(\mathbf{z})] \tag{3.9}$$

$$= \log \mathbb{E}_{q(\mathbf{z})}\left[\frac{p(\mathbf{z}|\boldsymbol{\theta})f(\mathbf{z})}{q(\mathbf{z})}\right] \tag{3.10}$$

$$\geq \mathbb{E}_{q(\mathbf{z})}\left[\log(p(\mathbf{z}|\boldsymbol{\theta})f(\mathbf{z}))\right] + H[q] \tag{3.11}$$

$$= F(q, \boldsymbol{\theta}), \tag{3.12}$$

where F is the same free energy function appearing in the previous section on EM, except that the complete likelihood is replaced with the term $f(\mathbf{z})p(\mathbf{z}|\boldsymbol{\theta})$. When $f(\mathbf{z})p(\mathbf{z}|\boldsymbol{\theta})$ is normalizable, then it can be shown that there is an EDA "variational gap" given by $F(q, \boldsymbol{\theta}) - \mathcal{L}_{EDA} = -D_{KL}(q(\mathbf{z})||\tilde{p}(\mathbf{z}|\boldsymbol{\theta}))$ where we define the tilted density,

$$\tilde{p}(\mathbf{z}|\boldsymbol{\theta}) = \frac{p(\mathbf{z}|\boldsymbol{\theta})f(\mathbf{z})}{\int_{\mathcal{Z}} p(\mathbf{z}|\boldsymbol{\theta})f(\mathbf{z})d\mathbf{z}}, \tag{3.13}$$

which is the EDA counterpart to the exact posterior in EM. We can now construct a coordinate ascent algorithm on the free energy defined in Equation (3.12) that mirrors the EM

algorithm. In particular, this algorithm iterates between E-steps that solve

$$q^{(t+1)} \leftarrow \operatorname*{argmin}_{q} D_{KL}(q(\mathbf{z})||\tilde{p}(\mathbf{z}|\boldsymbol{\theta}^{(t)})),$$

and M-steps that solve

$$\theta^{(t+1)} \leftarrow \operatorname*{argmax}_{\theta} \mathbb{E}_{q^{(t+1)}(\mathbf{z})} \left[\log(p(\mathbf{z}|\boldsymbol{\theta})f(\mathbf{z}))\right].$$

To make the precise connection between practically implemented EDA and EM, we introduce a particular approximate 'posterior' for the E-step that is given by a mixture of weighted particles:

$$q^{(t+1)}(\mathbf{z}) = \frac{\sum_{i=1}^{N} f(\mathbf{z}_i)\delta_{\mathbf{z}_i}(\mathbf{z})}{\sum_{i=1}^{N} f(\mathbf{z}_i)}, \tag{3.14}$$

where $\{\mathbf{z}_i\}_{i=1}^{N}$ are samples drawn from $p(\mathbf{z}|\boldsymbol{\theta}^{(t)})$, as in EDAs. Using this posterior approximation, the M-step amounts to solving the objective:

$$\begin{aligned}
\boldsymbol{\theta}^{(t+1)} &= \operatorname*{argmax}_{\boldsymbol{\theta}} \mathbb{E}_{q^{(t+1)}(\mathbf{z})} \left[\log(p(\mathbf{z}|\boldsymbol{\theta})f(\mathbf{z}))\right] && (3.15)\\
&= \operatorname*{argmax}_{\boldsymbol{\theta}} \frac{\sum_{i=1}^{N} \int_{\mathcal{Z}} f(\mathbf{z}_i)\delta_{\mathbf{z}_i}(\mathbf{z}) \log p(\mathbf{z}|\boldsymbol{\theta})\, d\mathbf{z}}{\sum_{i=1}^{N} f(\mathbf{z}_i)} && (3.16)\\
&= \operatorname*{argmax}_{\boldsymbol{\theta}} \sum_{i=1}^{N} f(\mathbf{z}_i) \log p(\mathbf{z}_i|\boldsymbol{\theta}), && (3.17)
\end{aligned}$$

which is exactly the generic EDA update step defined in Equation (3.2).

From this exposition it also becomes clear that EDAs can be thought of as performing a type of MC-EM that uses Importance Sampling [106] rather than direct sampling from the posterior. In particular, the EDA sampling procedure uses proposal distribution, $p(\mathbf{z}|\boldsymbol{\theta}^{(t)})$, (sampled in the E-step), and importance weights $p(\mathbf{z}|\boldsymbol{\theta})f(\mathbf{z})/p(\mathbf{z}|\boldsymbol{\theta}) = f(\mathbf{z})$ to estimate the expected "complete log likelihood", $\mathbb{E}_{q^{(t+1)}(\mathbf{z})} \left[\log(p(\mathbf{z}|\boldsymbol{\theta})f(\mathbf{z}))\right]$, in the M-step.

This is our main result, as it shows that many EDAs can be viewed as an EM algorithm that uses the particle-based posterior approximation given by Equation (3.14). For any $\mathbf{z}$, we have
$q^{(t+1)}(\mathbf{z}) \xrightarrow{p} \tilde{p}(\mathbf{z}|\boldsymbol{\theta}^{(t)})$ as $n \to \infty$ by the law of large numbers and Slutsky's theorem [63]. In this limit of infinite particles, the approximate posterior matches the tilted distribution—the EDA equivalent to the "exact posterior"—and our algorithm inherits the same guarantees as exact EM, such as guaranteed improvement of the objective function at each iteration, as well as local and global convergence properties [115, 14, 153].

In many cases, EDAs use a member of an exponential family for the search model. Letting $\boldsymbol{\theta}$ denote the expectation parameters, then the search model has the density:

$$p(\mathbf{z}|\boldsymbol{\theta}) = h(\mathbf{z}) \exp\left(\boldsymbol{\eta}(\boldsymbol{\theta})^T T(\mathbf{z}) - A(\boldsymbol{\theta})\right), \tag{3.18}$$

where h is the base measure, $\boldsymbol{\eta}(\boldsymbol{\theta})$ are the natural parameters, $T(\mathbf{z})$ are sufficient statistics and $A(\boldsymbol{\theta})$ is the log-partition function. Then the update of Equation (3.17) takes the simple form of a weighted maximum likelihood estimate for the exponential family:

$$\boldsymbol{\theta}^{(t+1)} = \frac{\sum_{i=1}^{N} f(\mathbf{z}_i) T(\mathbf{z}_i)}{\sum_{i=1}^{N} f(\mathbf{z}_i)}. \tag{3.19}$$

Next we show how exponential family search models can be used to connect parameter-smoothed EDAs to MAP-EM.

3.5 Smoothed EDAs as MAP-EM

In CMA-ES, the parameter updates are smoothed between iterations in a number of ways [68]. For example, the covariance estimate is typically smoothed with the previous estimate. Additionally, it may be further smoothed using a rank one covariance matrix obtained from the "evolution path" that the algorithm has recently taken.[2] Next, we consider these types of smoothing in detail. However, any smoothing update, or combination thereof, can be similarly derived by adjusting the form of the prior distribution. A benefit of viewing CMA-ES updates in this general form is that it becomes more straightforward to determine how to do similar types of smoothing for EDAs without Gaussian search models.

When smoothing with the previous parameter estimate, the updates can be written as

$$\boldsymbol{\theta}^{(t+1)} = (1-\gamma)\boldsymbol{\theta}^{(t)} + \gamma\tilde{\boldsymbol{\theta}}^{(t+1)}, \tag{3.20}$$

where $\tilde{\boldsymbol{\theta}}^{(t+1)}$ is the solution to Equation (3.17) and γ is a hyperparameter that controls the amount of smoothing. In the case where the search model is a member of an exponential family, as is defined in Equation (3.18), we will show that the smoothed update of Equation (3.20) is equivalent to a particular MAP-EM update that uses the tilted density of Equation (3.13) as the approximate posterior. To see this, consider the conjugate prior to the exponential family,

$$p_0(\boldsymbol{\theta}|\boldsymbol{\lambda}) = \exp\left(\boldsymbol{\lambda}_1^T \boldsymbol{\eta}(\boldsymbol{\theta}) - \lambda_2 A(\boldsymbol{\theta}) - B(\boldsymbol{\lambda})\right), \tag{3.21}$$

where $\boldsymbol{\lambda} = (\boldsymbol{\lambda}_1, \lambda_2)$, $B(\boldsymbol{\lambda})$ is the log-partition function of the prior, and we have assumed that the base measure is constant. We now consider the modified EDA objective:

$$\hat{\mathcal{L}}_{EDA}(\boldsymbol{\theta}) = \log \mathbb{E}_{p(\mathbf{z}|\boldsymbol{\theta})}[f(\mathbf{z}) p_0(\boldsymbol{\theta}|\boldsymbol{\lambda})], \tag{3.22}$$

which is analogous to the MAP objective defined in Section 3.3. It can be shown that by replacing Equation (3.9) with this modified objective, performing analogous steps as those in Equations (3.10)-(3.12) and (3.15)-(3.17)), and using the same definition of the tilted density

[2] See the two terms in Equation 11 in [68]).

and approximate posterior as in Equations (3.13) and (3.14), respectively, we arrive at the update equation:

$$\boldsymbol{\theta}^{(t+1)} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_{i=1}^{N} f(\mathbf{z}_i) \log p(\mathbf{z}_i|\boldsymbol{\theta}) p_0(\boldsymbol{\theta}|\boldsymbol{\lambda}). \tag{3.23}$$

If we now let $\lambda_2 = 1/\gamma - 1$ and allow $\boldsymbol{\lambda}_1$ to change every iteration as $\boldsymbol{\lambda}_1 = (1/\gamma - 1)\boldsymbol{\theta}^{(t)}$, it can then be shown that the solution to this objective is given by:

$$\begin{aligned} \boldsymbol{\theta}^{(t+1)} &= \frac{\boldsymbol{\lambda}_1}{1+\lambda_2}\boldsymbol{\theta}^{(t)} + \frac{1}{1+\lambda_2}\tilde{\boldsymbol{\theta}}^{(t+1)} && (3.24) \\ &= (1-\gamma)\boldsymbol{\theta}^{(t)} + \gamma\tilde{\boldsymbol{\theta}}^{(t+1)}, && (3.25) \end{aligned}$$

where

$$\tilde{\boldsymbol{\theta}}^{(t+1)} = \frac{\sum_{i=1}^{N} f(\mathbf{z}_i) T(\mathbf{z}_i)}{\sum_{i=1}^{N} f(\mathbf{z}_i)}$$

is equivalent to the RHS of Equation (3.19), the solution of original EDA objective. Since the update in Equation 3.25 is identical to the smoothed update of Equation 3.20, we can see that smoothing can be viewed as a consequence of performing MAP-EM in the context of EDAs, with a particular choice of prior parameters.

3.6 Conclusion

Estimation of Distribution algorithms represent a powerful class of methods for solving model-based optimization objectives. As we have seen in the previous chapter (and will again see in Chapter 5), such MBO objectives can be tailored to represent various inverse problems in applying machine learning to biological sequence. Therefore, developing a broader understanding of MBO objectives and methods to solve them can empower further development *CbAS*-like methods to tackle biological sequence design problems. In this chapter we have made one such contribution by showing a novel connection between a broad class of EDAs and EM. Additionally, we have presented an illustrative example of insight from EM that can be applied to EDAs by way of this connection.

In the next Chapter, we will shift our focus towards the forward problem, and discuss conditions under which one can perfectly estimate a biological fitness function.

Chapter 4

On the sparsity of fitness functions and implications for learning

4.1 Introduction

In the previous two chapters we discussed methods for solving inverse problems given a suitable predictive model of fitness, and in the next chapter we will present another such method. For the most part, we have been agnostic to the properties of this predictive model, as well as the specific nature of the underlying fitness function that the model attempts to estimate. In other words, we have mostly ignored the details of the forward problem when attempting to solve the inverse problem. In this chapter, we will turn our focus to the forward problem, and attempt to understand how certain properties of fitness functions of biological sequences effect our ability to accurately model such functions. In particular, there is a growing body of evidence demonstrating that fitness functions display substantial sparsity when represented in a particular basis; we will use this observation to try to understand the number of experimental measurements required to model a fitness function.

As discussed in Chapter 1, the forward problem in applying machine learning to sequence engineering is to build models of fitness functions given experimental fitness measurements. These measurements generally represent only a tiny fraction of those required to comprehensively characterize a fitness function. Here we will attempt to elucidate an open question in forward modeling, namely the number of training samples that are required to effectively estimate a given fitness function. This knowledge is crucial in the initial design of experiments as it allows researchers to assess which of a variety of experimental techniques should be used to probe a particular fitness function such that it can be recovered with a high fidelity, or similarly, how to restrict the scope of an experimental probe of a fitness function (e.g., by mutating fewer positions in a sequence, or allowing positions to only mutate to a certain number of alternative elements) such that the resulting data allows one to accurately recover the function under study. We approach this problem by combining tools from evolutionary biology and modern signal processing, which allows us to investigate the sample complexity

of fitness function estimation for sequences of any length and alphabet size (e.g., binary, nucleotide or amino acid alphabets), and test how different assumptions on simple, intuitive properties of the functions affect these results.

We posit that the question of the sample complexity of fitness function estimation likely can be better understood by investigating the sparsity of fitness functions, which is defined as the number of non-zero coefficients in a given expansion of the function[1]. It has recently been observed that some natural fitness functions are sparse when represented in terms of epistatic interactions [152, 191, 15] and that this property can be exploited to improve estimators of fitness [136, 4, 5]. It is well known in the field of signal processing that the sparsity of a signal is intimately related to the properties of estimators of that signal, and in particular to the sample complexity of certain algorithms designed to reconstruct signals from incomplete measurements. More specifically, results from the sub-discipline of signal processing known as Compressed Sensing (CS) provide scaling laws for the number of measurements required to reconstruct a signal in terms of its sparsity [34]. Additionally, there are results that describe the error induced by running these algorithms for signals that are only approximately sparse [39, 33].

Compressed Sensing is an appropriate framework for our purposes partly because there exist natural bases for representing fitness functions. The sparsity of a function depends crucially on the basis in which that function is represented, and discovering bases in which certain types of functions are sparse is a crucial task in CS [103]. The most popular basis for representing fitness functions of binary sequences is the Walsh-Hadamard (WH) basis; indeed, all of the aforementioned work studying the sparsity of fitness functions uses the WH basis and it is used in a number of theoretical studies of fitness functions [73, 168, 116]. The WH basis has a number of convenient properties that make it suitable for representing fitness functions, including that it intuitively encodes a fitness function in terms of epistatic interactions [135]. The WH basis can only be naturally applied in the case of binary sequences; however, herein we will show how to construct bases analogous to the WH basis that can represent fitness functions of sequences with non-binary alphabets (e.g., nucleotides or amino acids). This construction is based on the theory of Graph Fourier transforms [141], and thus we will refer to these as 'Fourier' bases [183, 168]. In what follows, we will only consider the sparsity of fitness functions in the Fourier basis, and thus will simply refer to this as the 'sparsity of the fitness function'.

Equipped with these bases, we aim to more comprehensively probe the sparsity of fitness functions than has been done previously, and use these results to develop general predictions for the sample complexity of fitness function estimation. It is not immediately clear how one could generalize previous work to make such predictions in cases where experimental data is not already available. One possibility could be to analyze more experimental data and attempt to make generalizable insights from this analysis. There exist an increasing number

[1]In a quirk of common terminology, a signal is called 'sparse' when it contains many zero coefficients, but the 'sparsity' is formally defined as the number of nonzero coefficients. Thus, a 'sparse' signal has *low* 'sparsity'.

of combinatorially-complete empirical fitness landscapes [43, 189, 16] and the sparsity of some of these have been probed to a certain extent [152, 185, 136]. However, these landscapes necessarily only report on short sequences (up to 4 amino acids, 10 nucleotides, or 13 binary-encoded positions) in limited environments, and thus may not be representative of fitness functions more broadly. A traditional approach in evolutionary biology to overcome the lack of sufficient empirical fitness landscapes is to instead study 'random field' models of fitness, which assign fitness to sequences based on stochastic processes constructed to mimic the statistical properties of natural fitness functions (see, e.g., [85] for a canonical example of this paradigm and [2] for more recent work). We will follow a similar line of reasoning and study the sparsity of random field models of fitness, allowing us to probe properties of a much broader class of fitness functions than the available empirical landscapes. A note on terminology: the stochastic process associated with a particular random field model specifies (often implicitly) a joint probability distribution over the fitness of all sequences; we will refer to a single sample from this joint distribution as being a fitness function sampled from the random field model.

In this chapter, we will restrict our attention to a specific random field model, namely a generalization of the NK model [86], and calculate the sparsity of fitness functions sampled from this model. The NK model is a particularly useful and well-studied random field model of fitness that can represent a rich variety of fitness functions using only two parameters: L, the length of sequences,[2] and K, the degree of epistatic interactions between positions in the sequence. As a consequence, many properties of the NK model have been extensively studied, with a particular focus on the number and heights of local optima [48, 30], the correlation between the fitness of sequences as a function of the Hamming distance between the sequences [168, 32], and the properties of adaptive walks on these landscapes [86, 120]. Importantly, this model and simple extensions are able to capture many of the characteristics of experimental fitness landscapes, including the properties of local fitness optima [72, 6] and correlation functions [148, 116]. Additionally, simulated data sampled from NK models have been used as a benchmark to test machine learning models, and the results from these tests compare favorably to analogous tests on experimental fitness datasets [57, 109].

In the original NK model defined by Kauffman [86], each position in the sequence interacts with exactly K other positions that are either adjacent to the position or are chosen uniformly at random; the set of interacting positions is known as a "neighborhood". In the generalized NK (GNK) model [30], these neighborhoods can be specified to be any size and include specific positions. Thus, in the GNK model the choice of neighborhoods is a tunable 'parameter' that generalizes the usual K parameter, and together with with the sequence length, L, and alphabet size, q, fully specifies the model. We will show that we can exactly calculate the sparsity of fitness functions sampled from the GNK model represented in the Fourier basis, as a function of L, q, and the choice of neighborhoods. We then use these results within CS theory to determine the minimum number of samples required to recover

[2] In the original definition of the model, N is used for the sequence length, but here we reserve N for the number of observed measurements.

these fitness functions. To gain intuition for the theory, we will calculate the sparsity results for a number of 'standard' neighborhood schemes, namely the random and adjacent position schemes used in the original NK model, as well as an oft-used scheme known as the Block Model that separates a sequence into non-interacting blocks of densely interacting positions [130, 124].

The choice of neighborhoods can encode certain biological intuitions about interactions between positions in a sequence. Thus, our results allow one to predict the sample complexity of fitness function estimation under a variety of biologically-relevant assumptions. To test the practical relevance of these predictions, we will describe a scheme to construct neighborhoods based on contacts in a given atomistic protein structure and compare the Fourier coefficients of the resulting GNK fitness functions to those of empirical protein fitness functions. In particular, we show that GNK models with neighborhoods constructed based on protein structure accurately approximate the sparsity of the empirical fitness functions of the mTagBFP2 fluorescent protein [136] and the protein encoded by the His3 gene in yeast [137], and are even able to correctly identify many of the important higher-order epistatic interactions in these fitness functions. We further predict the number of samples required to recover these structure-based GNK fitness functions, and show that estimators trained with this number of empirical fitness values are able to estimate the empirical fitness function with a relatively small amount of error. These results suggest that our theory can be used as a practical tool to predict the sample complexity of protein fitness function estimation.

The structure of this chapter is as follows: in Section 4.2 we describe the myriad of work related to ours; in 4.4 we formally define fitness functions and the estimation problem; in Section 4.5 we review the key concepts in Compressed Sensing that are relevant to our results; in Section 4.6 we discuss the properties of the Walsh-Hadamard basis and show how to construct analogous Fourier bases for sequences with larger alphabets; in Section 4.7 we formally define the GNK model and the standard neighborhood schemes; in Section 4.8 we present our main results on the sparsity of fitness functions sampled from the GNK model; in Section 4.9 we use these results to predict how many measurements are required to exactly recover fitness functions sampled from the GNK model; in 4.10 we describe a simple procedure for constructing GNK neighborhoods based on protein structure and apply our results to make sparsity and sample complexity predictions for the mTagBFP2 and His3 fitness functions; we conclude in 4.11 with a discussion of the broader relevancy of our results and potential future directions. Proofs for the mathematical results presented in this chapter are provided in Appendix A.

4.2 Related Work

In [30], the authors define the GNK model for binary sequences and calculate the 'rank' of these models, which is equal to the number of non-zero WH coefficients of fitness functions sampled from the model, and thus the sparsity of the functions in the WH basis. They additionally calculate statistics of the WH coefficients for sampled fitness functions. Our

main theoretical results are extensions of these calculations to the GNK model defined for sequences with any size alphabet. The proofs of these results (shown in Appendix A) provide a new spectral graph-theoretic perspective to the results of [30]. It should also be noted that the goal of [30] is quite different from our own, as they mainly attempt to correlate the rank of a GNK model with the number of local fitness optima in fitness functions sampled from the model, whereas we focus on the estimation of these functions. The authors of [120] calculate the ranks of GNK models that use the 'standard' neighborhood construction schemes that we will discuss here. Our results for these models again extends these calculations to the case of sequences with larger alphabets.

There is quite a large body of work that studies various properties of the WH coefficients of fitness functions. A number of papers calculate the WH coefficients of individual empirical fitness landscapes [189, 191] and characterize the sparsity of these coefficients [136, 15]. Some others have done *post-hoc* analysis of a number of experimental studies to calculate the WH coefficients [152]; a particularly notable example is [185], which tests the hypothesis that the influence of a coefficient decays with the degree of epistatic interaction that it represents in many empirical landscapes. All of this work provides motivation for our own and suggests that further study of the WH representation of of fitness functions is warranted. Most directly relevant to our work are studies of the WH coefficients of random field models, and in particular of the NK model. In [116], the authors calculate the 'amplitude spectrum' of a number of random field models, including the NK model. These calculations are based on the results of Stadler [168], who define the amplitude spectrum and provide a number of useful tools for studying random field models of fitness. The amplitude spectrum is, roughly, the expected normalized sum of squared WH coefficients corresponding to each order of epistatic interactions. Our results (and those in [30]) provide more refined information about the coefficients than the amplitude spectrum, in the sense that they describe the statistics of individual coefficients rather than averaged properties of sets of coefficients. In [73], the authors use the WH representation to show that the possible functions sampled from the NK model form an extremely restricted subset of all possible functions of binary sequences of length L. Similarly to the amplitude spectrum, these results report on global properties of WH coefficients, while our results provide more local information about individual coefficients. In [136], the authors construct a combinatorially complete empirical fitness function for the mTagBFP2 fluorescent protein, calculate the WH coefficients of this fitness function, recognize the sparsity of these coefficients, and test a Compressed Sensing algorithm's ability to recover this fitness function with a small number of samples. Not only did [136] serve as a major inspiration for this chapter, but we will make use of their data in Section 4.10.

In addition to the above work, our results rely heavily on the theory of Compressed Sensing, which we will review in more detail in Section 4.5. Before moving to that discussion, we will first briefly discuss modern experimental capabilities for probing fitness functions, and then formally define fitness functions discuss the estimation problem.

4.3 Experimental capabilities

The central aim of this chapter is to determine the number of experimental observations that are required to effectively estimate a fitness function. This warrants a discussion about what is currently possible with modern experimental technologies for probing fitness. High-throughput technologies for probing fitness generally fall under the umbrella of multiplex assays of variant effect (MAVEs) [52, 94]. Roughly, a typical MAVE experiment has three major steps: (i) creating a library of mutated sequences (ii) subjecting that library to an environment where sequences with high fitness will be amplified (or alternatively, those with low fitness will be diminished) and (iii) use Next-Generation Sequencing (NGS) technology to determine the contents of the library resulting from the selection. The data resulting from this procedure can then be processed to determine a scalar value of the fitness for every sequence that appeared in the initial library. Examples of experimental methods that fit this paradigm are Deep Mutational Scanning (DMS) for probing protein fitness [56], Massively Parallel Reporter Assays for determining the fitness of regulatory sequences [79], and SELEX experiments that use NGS to find functional aptamer sequences [156].

So how many sequences can these methods probe? The limiting factor is the number and length of sequence reads that can be processed by NGS. The Illumina NovaSeq 6000 sequencer can process up to 20 billion (2×10^{10}) reads with at most 150 base pairs reported in each read. At a sequencing depth of 1000 reads per unique sequence, this could be used to probe the fitness of up to $O(10^7)$ unique sequences. This limit has been approached in experiments probing the fitness of certain short nucleotide sequences; for instance Pitt and Ferré-D'Amare [134] report the catalytic fitness of ~10^7 unique 54 nucleotide RNA sequences and recently Vaishnav et al. [180] measured the effects on expression of over 2×10^7 unique 80 bp yeast promoter sequences. The situation for proteins is slightly bleaker, with most data sets containing 10^4 to 10^5 unique amino acid sequences. For instance, Sarkisyan et al. [154] report the fluorescence of 5×10^4 variants of the Green Fluorescent Protein, Rocklin et al. [144] tests the stability of 5×10^4 miniproteins, and Hinkley et al. [74] probe the fitness of 7×10^4 variants of the HIV protease and reverse transcriptase proteins. However, this picture is rapidly changing, with larger data sets recently becoming available: Bryant et al. [29] report the viability of 2×10^5 variants of the Adeno-Associated virus (AAV) capsid, and the data set of Pokusaeva et al. [137] contains over 8×10^5 unique amino-acid altering variants of the His3 gene in yeast.

Our aim in this chapter is determine the conditions under which fitness functions can be recovered with the data resulting from a high-throughput experimental probe of fitness. The above numbers can serve as a baseline for what constitutes a reasonable sample complexity for a given problem.

4.4 Fitness functions and estimation

A fitness function maps a space of sequences to a scalar property of interest. In particular, let $\mathcal{S}^{(L,q)}$ be the set of all length L sequences of an alphabet of size q, where $|\mathcal{S}^{(L,q)}| = q^L$ is the size of the sequence space. A fitness function is any function that maps the sequence space to scalar values, $f : \mathcal{S}^{(L,q)} \to \mathbb{R}$.

Some care should be taken in deciding appropriate values for L and q for a given situation. For our purposes, fitness functions are best understood in the context of hypothetical mutagenesis experiments, where L positions are allowed to mutate, and each position may be assigned one of q elements from an alphabet. Some of the subtleties in these definitions can be seen by considering the different possibilities for defining fitness functions of a protein. Suppose there is some protein made up of M amino acids and we would like to understand how mutations effect the fitness of this protein. Further suppose we choose $L < M$ positions that we are particularly interested in (e.g., a set of positions that are part of or close to a binding pocket). In certain cases, we may only be interested in how the fitness is affected when each of these positions mutates to one specific other amino acid (for instance, the amino acid present in that position in a homologous protein, as is the case in [136] and [8]); in this case, we have a binary alphabet (i.e., $q = 2$) because there are only two options at each of the L mutated positions. In other cases, we may be interested in determining the effects of positions mutating to *any* other amino acid, in which case $q = 20$. Wu et al. [189] probe this latter type of fitness function, testing the fitness of all combinations of the 20 amino acids in each of 4 positions in protein GB1, for a total of $20^4 = 160,000$ measurements. In some other cases, positions are allowed to mutate to a restricted subset of all amino acids [16], and thus $2 \leq q < 20$, which may represent a the set of amino acids observed in known homologs [137], or the set of single-site amino acid mutations that have neutral or positive fitness compared to a wild type in a saturation mutagenesis experiment [29].

Our primary focus in this chapter is the estimation of fitness functions. In order to understand the estimation problem formally, first note that any fitness function can be represented exactly as $\mathbf{f} = \mathbf{\Phi}\boldsymbol{\beta}$, where $\mathbf{f} = (f(\mathbf{s}))_{\mathbf{s}\in\mathcal{S}^{(L,q)}}$ is the vector of all q^L fitness evaluations, $\mathbf{\Phi}$ is a $q^L \times q^L$ orthogonal basis, and $\boldsymbol{\beta}$ is a vector of q^L coefficients. In this formulation, each row of $\mathbf{\Phi}$ represents an encoding of a particular sequence in $\mathcal{S}^{(L,q)}$. Although any $q^L \times q^L$ orthogonal basis could be used to represent the fitness function, in what follows we will always use the Fourier basis for sequences of length L and alphabet size q, which will be defined in Section 4.6, below.

The estimation problem is to recover $\boldsymbol{\beta}$ from an incomplete set of measurements. In other words, the aim is to estimate $\boldsymbol{\beta}$ given a set of sequences and measured fitness values corresponding to those sequences (i.e., rows of $\mathbf{\Phi}$ and corresponding elements of $\mathbf{f}$). More concretely, suppose we observe (possibly) noisy fitness measurements, $\mathbf{y}$, for N sequences whose indices are given in a set $I \subset \{1, 2, ..., q^L\}$. Assume a Gaussian noise model, $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\eta}$, where $\boldsymbol{\eta} \sim \mathcal{N}(0, \sigma^2\mathbf{I})$ corresponds to Gaussian measurement noise, σ^2 is the variance of the measurement noise, and $\mathbf{X} \coloneqq [\mathbf{\Phi}_{I_1}^T, ..., \mathbf{\Phi}_{I_N}^T]^T$ is the $N \times q^L$ matrix containing each row of $\mathbf{\Phi}$ whose index is in I. Our goal in the fitness function estimation problem is

to recover a good approximation to $\boldsymbol{\beta}$ given the incomplete set of noisy measurements. In all but the trivial edge case where $N = q^L$, this is an underdetermined linear system that requires additional information to be solved. In the next section, we briefly describe some aspects of the theory of Compressed Sensing, which provides conditions under which these types of underdetermined linear systems can be solved with certain simple algorithms.

4.5 Compressed Sensing

The field of Compressed Sensing is primarily concerned with studying algorithms that can solve underdetermined systems such as the fitness function estimation problem, and determining the conditions under which recovery with a specified amount of error can be guaranteed. One such algorithm is the LASSO, which is also well-studied in machine learning as a form of regularized linear regression [70]. Using the fitness function estimation problem as an example, LASSO solves the following convex optimization program:

$$\min_{\hat{\boldsymbol{\beta}}} \|\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}\|_2^2 + \nu\|\hat{\boldsymbol{\beta}}\|_1 \tag{4.1}$$

where ν is a parameter determining the strength of regularization.

The key determinant of the success of algorithms such as LASSO in recovering a particular signal is how well the coefficients that represent the signal in a given basis can be approximated by a sparse vector. Specifically, Candes and Plan [33] prove that when the subset of indices I are sampled uniformly at random from $\{1, 2, ..., q^L\}$, and the number of samples satisfies

$$N \geq C_0 \cdot S \log q^L \tag{4.2}$$

then the solution to the program in Equation (4.1), denoted $\boldsymbol{\beta}^*$, satisfies with high probability

$$\|\boldsymbol{\beta} - \boldsymbol{\beta}^*\|_2 \leq C_1 \frac{\|\boldsymbol{\beta} - \boldsymbol{\beta}_S\|_1}{\sqrt{S}} + C_2\sigma \tag{4.3}$$

where C_0, C_1, and C_2 are constants and $\boldsymbol{\beta}_S$ is the best S-sparse approximation to $\boldsymbol{\beta}$, (i.e., the vector that contains the S elements of $\boldsymbol{\beta}$ with the largest magnitude and sets all others elements to zero), and σ^2 is the variance of the measurement noise. Equation (4.3) has a number of important implications. First, it tells us that if $\boldsymbol{\beta}$ is itself S-sparse, then, in a noiseless setting, it can be recovered *exactly* with $\mathcal{O}(S \log M)$ measurements. Otherwise, if $\boldsymbol{\beta}$ is not exactly sparse but is well approximated by a sparse vector, then it can be approximately recovered with error on the order of $\frac{1}{\sqrt{S}}\|\boldsymbol{\beta} - \boldsymbol{\beta}_S\|_1$, which is proportional to the sum of the magnitudes of the $q^L - S$ elements of $\boldsymbol{\beta}$ with the smallest magnitudes.

The above results provide a straightforward path towards predicting the sample complexity of fitness function estimation: if we can calculate the sparsity of fitness functions, then we can predict the number of samples required for recovery up to a certain amount of error using Equations (4.2) and (4.3). We will primarily focus on cases where a fitness

function is exactly sparse in the Fourier basis and we can calculate the sparsity. In this case, we need only to determine an appropriate value of the constant C_0 in order to determine an N such that exact recovery of the fitness function is guaranteed. Although natural fitness functions are unlikely to be exactly sparse, they may be well approximated by sparse vectors, and Equation (4.3) tells us that the error of the fitness function estimator will be controlled in this case. In particular, imagine we predict that a fitness function will require N measurements to be recovered by assuming that it is exactly S-sparse, but in reality the fitness function has S large coefficients and many small coefficients. Then the resulting error from estimating the fitness function N samples will be small (proportional to the sum of the magnitudes of all of the small coefficients). This justifies our focus on exact sparsity, despite the fact that natural fitness functions are likely rarely exactly sparse. We will also focus on recovery in the noiseless setting, where $\sigma = 0$ and thus exact recovery can be guaranteed for sparse functions. Equation (4.3) shows that if a function is exactly S-sparse and (4.2) is satisfied, then the remaining error is due only to the measurement noise and not any properties of the function. Since here we are primarily concerned with understanding how assumed properties of fitness functions affect the sample complexity of estimating those functions, it is thus most appropriate to consider the noiseless setting and leave the estimation of error due to measurement noise to future work.

4.6 Fourier bases for fitness functions

The sparsity of a class of signals depends crucially on the basis with which they are represented. Indeed, determining bases that sparsify particular classes of signals is an important component of practical CS research as it greatly effects the behavior of recovery algorithms [34, 103]. Therefore the immediate concern in applying CS to the estimation of fitness functions is deciding with which basis we should represent fitness functions. The most popular bases for this purpose is the Walsh-Hadamard (WH) basis, which in matrix form is a normalized Hadamard matrix [184, 135]. The WH basis has a number of convenient and intuitive properties that make it suitable for representing fitness functions. Not only have many natural fitness functions been shown to be sparse in the WH basis, but Poelwijk et al. [135] showed that the WH basis can be used to unify a number of different definitions of epistasis.

One of the major advantages of the WH basis is that it can be interpreted as representing fitness functions in terms of epistatic interactions of increasing orders. To see this, let $\mathbf{H}_L$ be the WH basis for binary sequences of length L. Then any fitness function of binary sequences of length L can be represented by $\mathbf{f} = \mathbf{H}_L\boldsymbol{\beta}$ and in this representation, the fitness function evaluated on an individual sequence, $\mathbf{s} = [s_1, s_2, ..., s_L] \in \mathcal{S}^{(L,2)}$, has the form of an intuitive multi-linear polynomial:

$$f(\mathbf{s}) = \sum_{U \in \mathcal{U}} \beta_U \prod_{i \in U} s_i \tag{4.4}$$

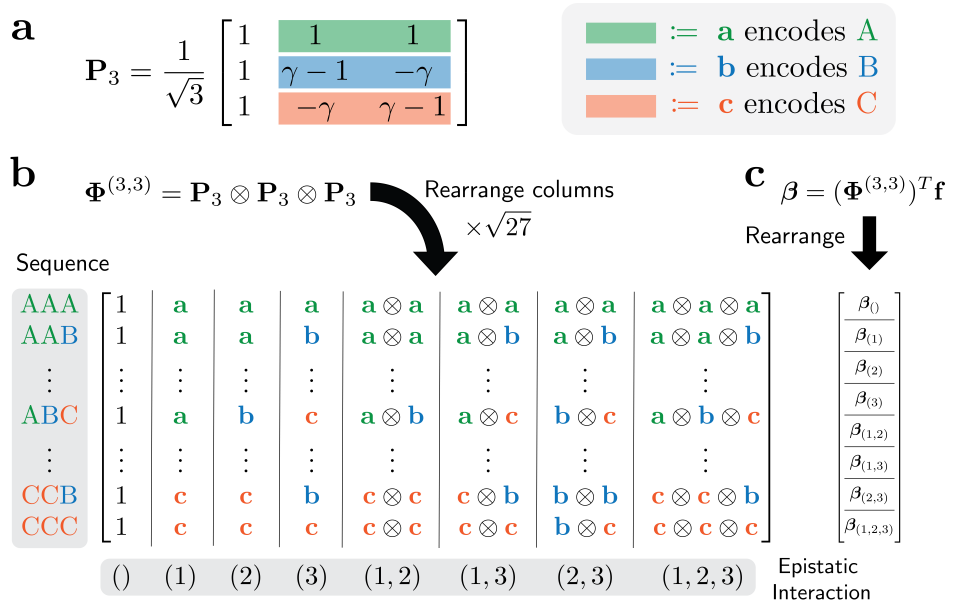


Figure 4.1: Example Fourier basis construction for $q = 3$ and $L = 3$. Consider $L = 3$ sequences constructed from the alphabet elements $\{A, B, C\}$. (a) Alphabet encodings constructed from the final $q - 1 = 2$ eigenvectors of the complete graph $K(3)$, where $\gamma := \frac{1}{\sqrt{3}-1}$. The encoding vectors $\mathbf{a}, \mathbf{b}$ and $\mathbf{c}$ corresponding to alphabet elements A, B and C, respectively, are highlighted with unique colors. (b) Fourier basis constructed according to Equation (4.8) for $L = 3$. The unnormalized basis with columns shuffled to represent epistatic interactions is shown in the enlarged matrix, with vertical bars representing column-wise concatenation. Each row represents a sequence shown on the left hand side of the matrix. Columns within vertical bars represent the epistatic interaction indicated beneath the matrix. (c) Fourier coefficients $\boldsymbol{\beta}$ with elements shuffled so that the sub-vectors, $\boldsymbol{\beta}_U$, represent epistatic interactions, where horizontal bars represent row-wise concatenation.

where $\mathcal{U} := \mathscr{P}(\{1, ..., L\})$ is the power set of position indices in the sequence. Each U with $|U| = r$ is a collection of r unique indices and each term in the polynomial represents an 'epistatic' interaction of order r between positions in the sequence Equation (4.4) could also be written more clearly as:

$$f(\mathbf{s}) = \beta_0 + \sum_{i=1}^{L} \beta_i s_i + \sum_{i \neq j} \beta_{ij} s_i s_j + \sum_{i \neq j \neq k} \beta_{ijk} s_i s_j s_k + ...$$

with the sum continuing up to the L^{th} order interaction term, and where we define β_0 as the coefficient corresponding to the empty set. It is clear that there are 2^L total terms in this expansion, and $\binom{L}{r}$ epistatic interactions of order r. This construction of a fitness function

in terms of epistatic interactions carries a strong biological intuition and allows for clear interpretation of the fitness function's parameters.

Additionally, the WH basis admits a recursive definition that allows it be constructed quickly. Specifically, we have the recursion relation:

$$\mathbf{H}_{L+1} = \begin{bmatrix} \mathbf{H}_L & \mathbf{H}_L \\ \mathbf{H}_L & -\mathbf{H}_L \end{bmatrix} \tag{4.5}$$

where we define $\mathbf{H}_0 \coloneqq [1]$. This can also written without recursion as:

$$\mathbf{H}_L = \bigotimes_{l=1}^{L} \mathbf{H}_1 \tag{4.6}$$

where $\otimes$ represents the Kronecker product and $\mathbf{H}_1 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ can be determined from Equation (4.5).

The WH basis can only be naturally applied to fitness functions of binary sequences (where $q = 2$), which poses a challenge in biological contexts where common alphabets include the nucleotide ($q = 4$) and amino acid ($q = 20$) alphabets. The usual solution to this problem is to encode elements of a larger alphabet as binary sequences, and use the WH basis to represent a fitness function of sequences in this encoding. However, this misrepresents the structure of sequence space for a number of reasons. An easy way to see this is to consider the size of the sequences space corresponding to a binary encoding. For example the "one-hot" encoding scheme of amino acids encodes each amino acid as a length 20 bit string; the number of amino acid sequences of length L is 20^L, while the binary encodings of these sequences are elements of a binary sequence space of size $2^{20L} = 1,048,576^L$. This latter number also corresponds to the number of WH coefficients required to represent the fitness function in the one-hot encoding, and is much too large to be of any practical use. How then might we construct a more appropriate basis for representing fitness of sequences of larger alphabets? Ideally this basis will share analogous convenient properties to those of the WH basis. Fortunately, the WH basis is a member of a broader class of bases that admits straightforward extensions to larger alphabets. In particular, the WH basis is the Graph Fourier basis corresponding to a graph representing the space of sequences. The Graph Fourier basis corresponding to a given graph is a complete set of orthogonal eigenvectors of the Graph Laplacian of the graph. Graph Fourier bases have many useful properties and have been used extensively for processing signals defined on graphs [141].

The WH basis is specifically the Graph Fourier basis corresponding to the Hamming graph $H(L, 2)$ [166]. The vertices of $H(L, 2)$ represent all unique binary sequences of length L; two sequences are adjacent in $H(L, 2)$ if they differ in exactly one position (i.e., the Hamming distance between the two sequences is equal to one). The Hamming graphs $H(L, q)$ are defined in the same way for sequences with alphabet size q. Thus, we can construct an analogous Graph Fourier basis to the WH basis to represent sequences with larger alphabets

by calculating the eigenvectors of the Graph Laplacian of $H(L, q)$. In what follows, we will only consider functions defined on Hamming graphs and we will thus refer to Graph Fourier bases corresponding to Hamming graphs simply as 'Fourier' bases.

An important property of the Hamming graph $H(L, q)$ is that it can be constructed as the L-fold Graph Cartesian product of the "complete graph" of size q [166]. The complete graph of size q, denoted $K(q)$, has q vertices (which represent elements of the alphabet in our case) and edges between all pairs of vertices. Due to the spectral properties of graph products, the eigenvectors of the Hamming graph (i.e., the Fourier basis) can be calculated as a function of the eigenvectors of the complete graph. We have the following result regarding the eigenvectors of the complete graph.

Proposition 4.1. *An orthonormal set of eigenvectors of the Graph Laplacian of the complete graph $K(q)$ are given by the columns of the following Householder matrix:*

$$\mathbf{P}_q \coloneqq \mathbf{I}_q - \frac{2\mathbf{w}\mathbf{w}^T}{\|\mathbf{w}\|_2^2} \tag{4.7}$$

where $\mathbf{w} \coloneqq \mathbf{1}_q - \|\mathbf{1}_q\|_2 \mathbf{e}_1$ and $\mathbf{1}_q$ is a vector of length q whose elements are all equal to one.

The complete graph is equal to the Hamming graph $H(1, q)$, and thus Equation (4.7) constructs the Fourier basis for sequences of length one and alphabet size q. Each row of $\mathbf{P}_q$ corresponds to a sequence of length one; the first columns is constant for all rows while the remaining $q-1$ columns encode the alphabet elements (i.e., the final $q-1$ elements of a row uniquely identify the alphabet element that the row corresponds to). More specifically, let $\tilde{\mathbf{P}}_q$ be the matrix containing the final $q-1$ unnormalized columns of $\mathbf{P}_q$, such that $\mathbf{P}_q = \frac{1}{\sqrt{q}}\left[\mathbf{1}_q \mid \tilde{\mathbf{P}}_q\right]$, where $\mid$ denotes column-wise concatenation. Then the i^{th} row of $\tilde{\mathbf{P}}_q$ encodes the i^{th} element of the alphabet. We will denote each of these encodings as $\mathbf{p}_q(s)$, where s is an element of the alphabet (i.e., each $\mathbf{p}_q(s)$ is a row of $\tilde{\mathbf{P}}_q$). As an example, 4.1a shows $\mathbf{P}_3$, which encodes the three element alphabet $\{A, B, C\}$. The colored rows in this figure together represent $\tilde{\mathbf{P}}_q$.

Then, it can be shown that Fourier basis corresponding to the Hamming graph $H(L, q)$, which can be used to represent fitness functions of sequences of length L and alphabet size q, is given by the L-fold Kronecker product of the eigenvectors of the complete graph, as shown in the following proposition.

Proposition 4.2. *An orthonormal set of eigenvectors of the Graph Laplacian of the Hamming graph $H(L, q)$ are given by the columns of following the $q^L \times q^L$ matrix:*

$$\mathbf{\Phi}^{(L,q)} = \bigotimes_{i=1}^{L} \mathbf{P}_q \tag{4.8}$$

where $\mathbf{P}_q$ is defined in Equation (4.7).

We note that these bases are not entirely novel: an equivalent form for nucleotide alphabets is given in [171]. Additionally, if we used a q-point Discrete Fourier Transform matrix instead of $\mathbf{P}_q$ in (4.8) than we would recover the basis used in [167].

There are a number of interesting points to be made about this basis. First, it can easily be seen that $\mathbf{P}_2 = \mathbf{H}_1$, and thus $\mathbf{\Phi}^{(L,2)} = \mathbf{H}_L$, which confirms that the WH basis is the Fourier basis for $q = 2$. Additionally, the Fourier basis for larger alphabets also shares the intuition of encoding epistatic interactions between positions in a sequence. Any fitness function of sequences of length L and alphabet size q can be represented as $\mathbf{f} = \mathbf{\Phi}^{(L,q)}\boldsymbol{\beta}$ and in this representation, the fitness function evaluated on a specific sequence, $\mathbf{s} \in \mathcal{S}^{(L,q)}$ can again be written in terms of interactions between positions in the sequence:

$$f(\mathbf{s}) = \sum_{U \in \mathcal{U}} (\boldsymbol{\beta}_U)^T \boldsymbol{\phi}_U(\mathbf{s}) \tag{4.9}$$

where $\boldsymbol{\phi}_U(\mathbf{s}) \coloneqq \frac{1}{\sqrt{q^L}} \bigotimes_{i \in U} \mathbf{p}_q(s_i)$ is a length $(q-1)^{|U|}$ vector representing the interaction between the sequence positions in U, and $\boldsymbol{\beta}_U$ is a length $(q-1)^{|U|}$ sub-vector of $\boldsymbol{\beta}$. In other words, the Fourier basis constructed with Equation (4.8) represents an r^{th} order epistatic interaction with a length $(q-1)^r$ vector where the elements are all possible products between elements in the vectors that encode alphabet elements involved in the interactions. Figure 4.1 shows an example construction of the Fourier basis for $q = 3$ and $L = 3$ which illuminates many of these properties.

These Fourier bases provide a powerful and intuitive representation of a fitness function of sequences of any length and alphabet size. In what follows, we will investigate the sparsity of fitness functions represented in these bases.

4.7 The Generalized NK (GNK) model

In order to move beyond the available empirical data and study more fitness functions more generally, we will probe the sparsity of fitness function sampled from random field models. A random field model specifies a stochastic process over sequence space to assign fitness to sequences, which implicitly defines a joint probability distribution over the fitness values of all sequences. Importantly, this also induces a probability distribution for the Fourier coefficients, $\boldsymbol{\beta}$. A wide variety of random field models exist, ranging in intricacy from the House of Cards (HoC) model [93], where fitness values are assigned *i.i.d.* to each sequence, to those modeling complex biophysical processes [190].

We will focus on the Generalized NK (GNK) model, a random field model introduced by Buzas and Dinitz [30]. The parameters of the GNK model can be tuned to represent a variety of biologically relevant assumptions, but it is still simple enough that many of its properties can be calculated exactly. As we will see, the properties that can be calculated exactly include the mean and variance of the Fourier coefficients, and the sparsity of fitness functions sampled from the model.

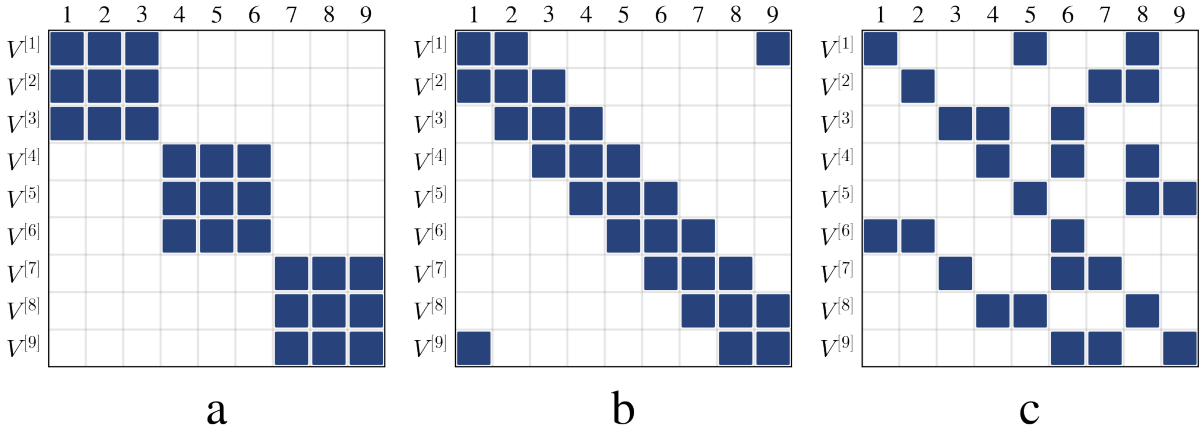


Figure 4.2: Graphical depictions of neighborhood schemes for $L = 9$ and $K = 3$. The vertical axis represents neighborhoods and the horizontal axis represents positions. A square in the $(i, j)^{\text{th}}$ position in the grid denotes that sequence position j is in the neighborhood $V^{[i]}$. (a) Block Neighborhood (b) Adjacent Neighborhood (c) Random Neighborhood.

In order to be defined, the GNK model requires the specification of a 'neighborhood' for each position in a sequence. A neighborhood $V^{[j]} \subset \{1, 2, ..., L\}$ corresponding to position j contains j and $K_j - 1$ other position indices, where we define $K_j := |V^{[j]}|$. These neighborhoods can roughly be thought of as the positions that position j interacts with; we will call interactions due to positions co-occuring in a neighborhood 'positional' interactions, which is in contrast to epistatic interactions which are defined as interactions that have nonzero coefficients in the Fourier expansion of a fitness function. In the classical NK model [86], the neighborhoods are all of the same size (i.e., $K_j = K$ for all j), and are constructed by one of a few standard schemes, which we will discuss shortly. The GNK model allows for complete flexibility in the specification of neighborhoods, and thus can apply to a broader range of biological contexts.

Definition 4.1 (The GNK Model)**.** Given sequence length, L, alphabet size, $q \geq 2$, and set of neighborhoods $\mathcal{V} := \{V^{[j]}\}_{j=1}^{L}$, a fitness function sampled from the GNK model assigns a fitness to every sequence $\mathbf{s} \in \mathcal{S}^{(L,q)}$ with the following two steps:

1. Let $\mathbf{s}^{[j]} := (s_k)_{k \in V^{[j]}}$ be the subsequence of $\mathbf{s}$ corresponding to the indices in the neighborhood $V^{[j]}$. Assign a 'subsequence fitness', $f_j(\mathbf{s}^{[j]})$ to every possible subsequence, $\mathbf{s}^{[j]}$, by drawing a value from the unit normal distribution. In other words, $f_j(\mathbf{s}^{[j]}) \sim \mathcal{N}(0, 1)$ for every $\mathbf{s}^{[j]} \in \mathcal{S}^{(K_j,q)}$, and for every $j = 1, 2, ..., L$.

2. For every $\mathbf{s} \in \mathcal{S}^{(L,q)}$, the subsequence fitness values are summed to produce the total fitness values $f(\mathbf{s}) = \sum_{j=1}^{L} f_j(\mathbf{s}^{[j]})$.

We define $\text{GNK}(L, q, \mathcal{V})$ as the probability distribution over fitness functions induced by the the above stochastic steps.

There are a number of simple intuitions that can be helpful in understanding the definition of the GNK model. If two sequences share a subsequence corresponding to a neighborhood, then they also share a subsequence fitness value. In this way, sequences that are close in Hamming distance are more likely to share subsequence fitness values and thus have more correlated fitness values. If the neighborhoods are large, then nearby sequences will be less likely to share neighborhood subsequences, and thus fitness values will be less correlated between nearby sequences compared to a model with smaller neighborhoods. In the limiting case where $K_j = L$ for every $j = 1, 2, ..., L$, all fitness values are uncorrelated from one another. In the other limiting case, where $K_j = 1$ for all j, fitness values between sequences are maximally correlated and the landscape is characterized by a single fitness peak (i.e., it has a single unique local optima which is the global optimum).

In the original definition of the GNK model, the subsequence fitness values may be drawn from any distribution with finite variance [30]. For simplicity, here we only consider the case where subsequence fitness values are drawn from the unit normal distribution; however, many of our results could straightforwardly be extended to the more generalized GNK model.

It is clear that the key choice in defining the GNK model is in how the neighborhoods are constructed. For a given problem, neighborhoods may be constructed according to some prior knowledge about the system under consideration. For example, in a later section we will discuss a scheme that constructs neighborhoods based on contacts in a protein structure. In order to gain some intuition about how different neighborhood schemes impact the sparsity of fitness functions, we will analyze three "standard" schemes for constructing neighborhoods, known as the Block Neighborhood (BN), Adjacent Neighborhood (AN), and Random Neighborhood (RN) schemes [116, 120]. All three of these schemes have a uniform neighborhood size at every position (i.e., $K_j = K$ for all $j = 1, 2, .., L$ and for some $1 \leq K \leq L$) and thus can be used to compare how different neighborhood structures impact sparsity at fixed values of L, q and K. The AN and RN schemes are what are used to define the classical NK model of Kauffman and Weinberger [86], and the BN scheme was introduced shortly thereafter by Perelson and Macken [130] as a simplified model for which many properties can be calculated exactly [124]. We describe each of these schemes in more detail below; graphical depictions of each scheme can be seen in Figure 4.2.

Block Neighborhood In the Block Neighborhood scheme, the parameter K must be specified such that L is a multiple of K. In order to construct the neighborhoods, we split the positions into L/K blocks of size K. Each block is "fully connected" in the sense that every neighborhood of a position in the block contains all other positions in the block, but no positions outside of the block. The GNK model using the BN scheme essentially results

in L/K independent fitness functions that are summed together, where within each of the blocks fitness values are uncorrelated. Without loss of generality, we can assume that the blocks contain adjacent positions, so that the neighborhoods are given by

$$V^{[j]} = \left\{ j, K \left\lfloor \frac{j-1}{K} \right\rfloor + 1, K \left\lfloor \frac{j-1}{K} \right\rfloor + 2, ..., K \left\lfloor \frac{j-1}{K} \right\rfloor + K \right\} \tag{4.10}$$

where $\lfloor \rfloor$ is the floor operator.

Adjacent Neighborhood In the Adjacent Neighborhood scheme, each position's neighborhood contains its $K-1$ nearest neighbors in terms of position indices. In order to ensure that all neighborhoods are of the same size, the sequence has periodic boundary conditions, such that, e.g., position 1 interacts with position L. To understand this, imagine arranging the positions in a circle, where position i is adjacent to $i+1$ in the circle and position L is adjacent to position 1. Then, in the AN scheme, a position's neighborhood contains the $\frac{K-1}{2}$ closest positions to it in both the clockwise and counterclockwise directions if K is an odd number. If K is an even number, then we will include the $\frac{K-2}{2}$ counterclockwise positions and the $\frac{K}{2}$ clockwise positions in the neighborhood. Formally, letting $a_j \coloneqq j - \lfloor \frac{K-1}{2} \rfloor - 1$ the neighborhood of position j is given by:

$$V^{[j]} = \{ j, a_j \bmod L + 1, (a_j + 1) \bmod L + 1, ..., (a_j + K) \bmod L + 1 \} \tag{4.11}$$

Random Neighborhood Perhaps the most common neighborhood scheme is the Random Neighborhood construction, where one specifies an integer K (where $1 \leq K \leq L-1$) and each $V^{[j]}$ contains j and $K-1$ other position indices selected uniformly at random from $\{1, 2, ..., L\} \backslash j$.

The RN scheme can be used to study fitness functions in cases where we have no *a priori* knowledge about how the positions in the sequence interact with one another. For instance, SELEX experiments can probe the fitness of vast numbers of aptamer sequences [156], but it is unclear which positions in an aptamer sequence interact with one another to, for instance, bind to a target molecule. However, as we will show, it is possible to calculate the *expected* sparsity of GNK fitness functions whose neighborhoods are randomly constructed with the RN scheme, which can then be used to estimate the sample complexity of fitness function estimation when we know little about how positions in the sequence interact with one another.

There are many possible modifications to these schemes that may encode different biological assumptions and can easily be used within the GNK model. For example, in [116], the authors define a modified RN scheme where each K_j is a random variable drawn from some distribution with support on $\{1, ..., L\}$.

As we will see, the choice of neighborhood scheme has a considerable effect on the sparsity of fitness functions sampled from the GNK model, which we will henceforth refer to as 'GNK

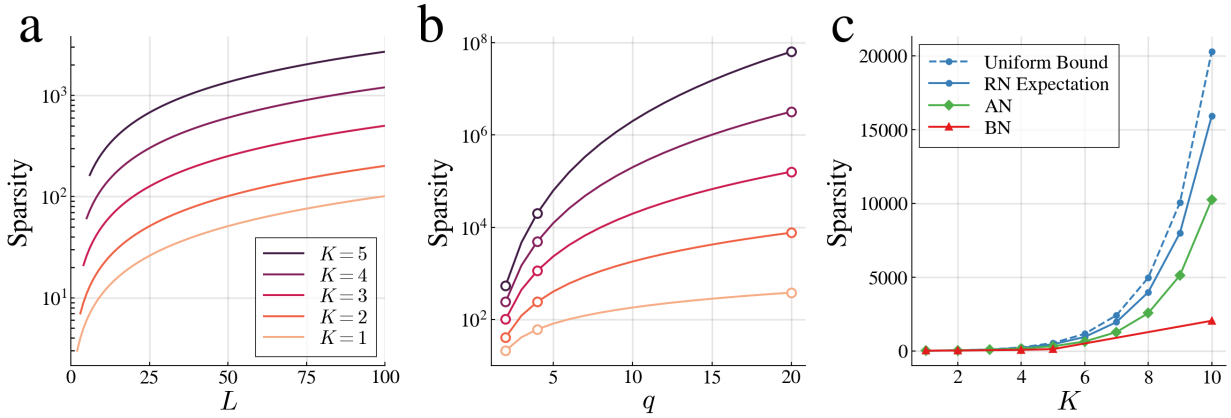


Figure 4.3: The sparsity of fitness functions sampled from the GNK model. (a) Upper bound on sparsity of GNK fitness functions with uniform neighborhood sizes for $q = 2$ and a range of settings of the L and K parameters (b) Upper bound on the sparsity of GNK fitness functions with uniform neighborhood sizes for $L = 20$ and a range of settings of the alphabet size q and the K parameter (colors as in (a)). Alphabet sizes corresponding to binary ($q = 2$), nucleotide ($q = 4$), and amino acid ($q = 20$) alphabets are highlighted with open circles. (c) Sparsity of GNK fitness functions with neighborhoods constructed with each of the standard schemes for $L = 20$ and $q = 2$, and for different settings of the uniform neighborhood size, K.

fitness functions'. In the next section, we will present a result that allows us to calculate the sparsity of fitness functions given the sequence length, alphabet size, and a particular set of neighborhoods. We will use this to result to calculate an upper bound on the sparsity of GNK fitness functions with any set of uniformly-sized neighborhoods, as well as the expected sparsity of GNK fitness functions with Random neighborhoods, and the exact sparsity of GNK fitness functions with Block and Adjacent neighborhoods.

4.8 The sparsity of GNK fitness functions

A somewhat remarkable feature of the GNK model that will allow us to make sparsity calculations is that the Fourier coefficients of GNK fitness functions are independent random variables whose mean and variance can be calculated exactly for a given sequence length, L, alphabet size, q, and neighborhoods, $\mathcal{V}$. Specifically, we have the following theorem (proved in Appendix A, along with all other results in this section.)

Theorem 4.1. *Let* $\mathbf{f} = (f(\mathbf{s}))_{\mathbf{s}\in\mathcal{S}^{(L,q)}}$ *be the complete vector of evaluations of a fitness function* $f \sim GNK(L, q, \mathcal{V})$. *Then the Fourier coefficients of* f, $\boldsymbol{\beta} = (\boldsymbol{\Phi}^{(L,q)})^T\mathbf{f}$, *are distributed according to* $\boldsymbol{\beta} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\lambda}\mathbf{I})$ *(i.e., normally distributed with zero mean and diagonal covari-*

ance). Let β_i be an element of $\boldsymbol{\beta}_{U(i)}$, the sub-vector of $\boldsymbol{\beta}$ representing the epistatic interaction $U(i)$. Then the variance of β_i is given by

$$\lambda_i = \sum_{j=1}^{L} q^{L-K_j} I\left(U(i) \subseteq V^{[j]}\right) \tag{4.12}$$

where $I\left(U \subseteq V^{[j]}\right)$ is an indicator function that is equal to one if $U(i)$ is a subset of or equal to $V^{[j]}$ and zero otherwise.

Theorem 4.1 provides a number of important insights into the Fourier representation of GNK fitness functions. First, since $\boldsymbol{\beta}$ is normally distributed with zero mean and diagonal covariance matrix, the vector of variances, $\boldsymbol{\lambda}$, fully specifies the distribution of $\boldsymbol{\beta}$. Additionally, the variances have a fairly intuitive form. Specifically, each coefficient β_i corresponds to epistatic interaction, $U(i)$, and Equation (4.12) tells us that the more neighborhoods that contain the collection of positions corresponding to the epistatic interaction, the higher the variance of that coefficient will be. Finally, Equation (4.12) makes it clear that a Fourier coefficient has zero variance, and thus is deterministically equal to zero, when the epistatic interaction that it corresponds is not a subset of any neighborhoods. Since each epistatic interaction U is represented by $(q-1)^{|U|}$ Fourier coefficients, it is straightforward to calculate the number of zero Fourier coefficients, and thus the sparsity, of GNK fitness functions. Specifically, we have the following theorem that describes the result of this calculation.

Theorem 4.2. *Let $S(f) \coloneqq \text{supp}(\boldsymbol{\beta})$ be the sparsity of a fitness function f of sequences of length L and alphabet size q with Fourier coefficients $\boldsymbol{\beta}$, where $\text{supp}(\boldsymbol{\beta})$ is equal to the number of nonzero elements of $\boldsymbol{\beta}$. Then for any $f \sim GNK(L, q, \mathcal{V})$,*

$$S(f) = \sum_{U \in \mathcal{T}(\mathcal{V})} (q-1)^{|U|} \tag{4.13}$$

almost surely, where $\mathcal{T}(\mathcal{V}) \coloneqq \bigcup_{j=1}^{L} \mathscr{P}(V^{[j]})$ is the union of the powersets of the neighborhoods.

Theorem 4.2 makes concrete the connection between neighborhoods and epistatic interactions. The GNK model assigns a non-zero Fourier coefficients to any epistatic interaction between positions that co-occur in *any* neighborhood. For example, if positions 3 and 4 in a sequence are both in some neighborhood $V^{[j]}$, then every element of $\boldsymbol{\beta}_{\{3,4\}}$ is nonzero (remember that $\boldsymbol{\beta}_U$ contains the $(q-1)^{|U|}$ elements of $\boldsymbol{\beta}$ corresponding to the epistatic interaction U). Additionally, an important insight from Theorem 4.2 is that for a fixed L, q and $\mathcal{V}$, the sparsity is a deterministic quantity, and the stochasticity of the model only effects the non-zero coefficients. The only case where the sparsity is stochastic is when the neighborhoods are random variables, as with the Random Neighborhood scheme discussed in the previous section.

It is clear from Theorem 4.2 that the sparsity of GNK fitness functions depends only on L, q and the neighborhoods $\mathcal{V}$. In order to demonstrate the usefulness of the above results, and to gain some initial understanding of how the tunable parameters in the GNK model impact the sparsity of sampled fitness functions, we will apply Theorem 4.2 to the standard neighborhood schemes described in the previous section. The common feature between these schemes is that they have uniform neighborhood sizes. The following proposition provides a useful upper bound on the sparsity of fitness functions sampled from any such GNK model with uniform neighborhood sizes.

Proposition 4.3. *Let $\mathcal{V}_K$ be a set of neighborhoods where $K_j = K$ for $j = 1, 2, ..., L$ and $1 \leq K \leq L$. Then, the sparsity of any $f \sim GNK(L, q, \mathcal{V}_K)$ is bounded above by:*

$$S(f) \leq 1 + L(q-1) + L\sum_{r=2}^{K} \binom{K}{r}(q-1)^r \tag{4.14}$$

The result from [30] that this bound is tight for many setting of L and K follows straightforwardly.

Proposition 4.3 is most useful when applied to the RN scheme, as this is the only of the three standard schemes where there is stochasticity in the sparsity. However, it also serves as a useful baseline because it requires very few assumptions on the interactions in the fitness function, and bounds the sparsity of the other neighborhood schemes. In Figures 4.3a and 4.3b we plot the bound of Equation (4.14) for a variety of settings of L, q and K. These figures demonstrate some generally applicable properties of GNK fitness functions. Most importantly, we see that the sequence length, L, has a much smaller relative impact on the sparsity compared to the alphabet size q within the biologically-relevant ranges that are plotted. As we will see, this result carries through to sample complexity predictions, though to a lesser extent.

We additionally have the following results regarding the sparsity of GNK fitness with neighborhoods constructed with the Adjacent and Block Neighborhood schemes, and the expected sparsity resulting from the Random Neighborhood scheme.

Proposition 4.4. *Let $\mathcal{V}_{BN}$ be neighborhoods constructed with Block Neighborhood scheme of Equation 4.10 for a given L, q and K. Then, the sparsity of a fitness function f sampled from $GNK(L, q, \mathcal{V}_{BN})$ is given by:*

$$S(f) = \frac{L}{K}(q^K - 1) + 1 \tag{4.15}$$

Proposition 4.5. *Let $\mathcal{V}_{AN}$ be neighborhoods constructed with the Adjacent Neighborhood scheme of Equation 4.11 for a given L, q and K. Then, the sparsity of any $f \sim GNK(L, q, \mathcal{V}_{AN})$ is given by:*

$$S(f) = L\sum_{r=0}^{K} \binom{K-1}{r-1}(q-1)^r \tag{4.16}$$

Proposition 4.6. *Let $\mathcal{V}_{RN}$ be a set of neighborhoods sampled according to the Random Neighborhood scheme for a given L, q and K. Then, the expected sparsity of a fitness function f sampled from GNK$(L, q, \mathcal{V}_{RN})$, with the expectation taken over the possible realizations of $\mathcal{V}_{RN}$, is given by:*

$$\mathbb{E}_{\mathcal{V}_{RN}}[S(f)] = \sum_{r=0}^{K} \binom{L}{r} p(r)(q-1)^r \tag{4.17}$$

where

$$p(r) = 1 - (1 - \alpha(r))^r \left(1 - \alpha(r)\frac{K-r}{L-r}\right)^{L-r} \tag{4.18}$$

and $\alpha(r) = \frac{(K-1)!}{(L-1)!}\frac{(K-r)!}{(L-r)!}$.

These results allow us to compare the impact of the neighborhood structures on the sparsity of sampled fitness functions. Specifically, in Figure 4.3c we plot the upper bound of Equation (4.14) with the exact or expected sparsity of GNK fitness functions with each of the standard neighborhood schemes as functions of the neighborhood size, K for a fixed setting of sequence length and alphabet size. We can see that even at the same setting of K, different neighborhood schemes result in striking differences in the sparsity of sampled fitness functions. This is perhaps surprising, as certain properties of fitness functions (for instance the amplitude spectrum and autocorrelation functions) depend only on K and not on the specific choice of neighborhoods [116].

Figure 4.3c shows that choice of neighborhoods can have a substantial impact on the sparsity of GNK fitness functions, and thus also on the sample complexity predictions that we will make in the following sections. Therefore, one should attempt to accurately capture the neighborhoods of a given sequence of interest before applying our results. Further on, we will suggest a method for doing so based on contacts in a protein structure.

4.9 Exact recovery of GNK fitness functions

Theorem 4.2 allows us to calculate the sparsity of any GNK fitness function, given the model's tunable parameters. This allows us to straightforwardly apply CS theory to determine the number of samples required to recover these functions exactly. Specifically, we can use Equation (4.2) to determine an N such that exact recovery is guaranteed for an $S(f)$-sparse fitness function f. In order to so, however, we need to determine an appropriate value for the constant C_0.

We determined a reasonable value for C_0 via straightforward numerical experiments. Specifically, we (i) sampled a fitness function from a GNK model, (ii) subsampled N sequence-fitness pairs uniformly at random from the complete fitness function for a range of settings of N, (iii) ran LASSO on each of the subsampled data sets and (iv) determined the smallest N such that the fitness function is exactly recovered by LASSO. Letting $\hat{N}$ be the minumum

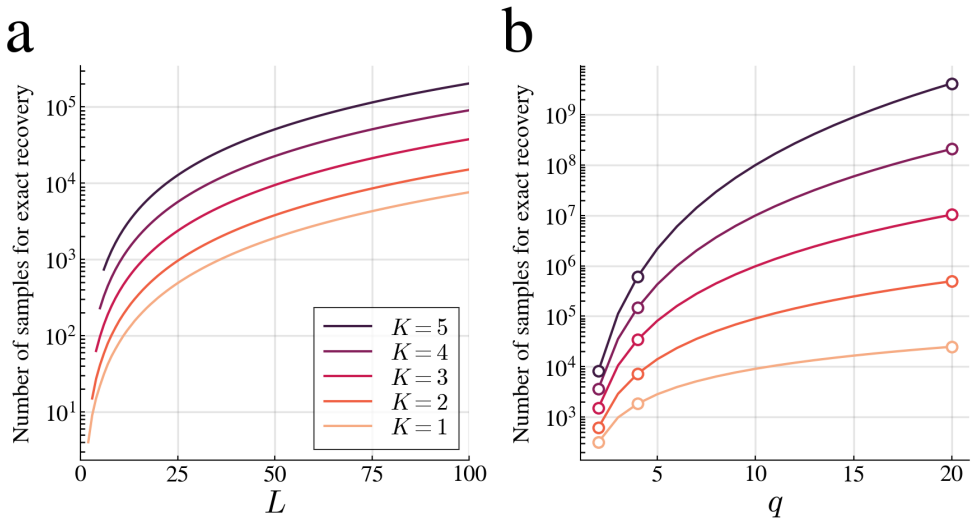


Figure 4.4: Upper bounds on the minimum number of samples required to exactly recover GNK fitness functions with uniform neighborhood sizes. (a) Bound as a function of sequence length, L, for $q = 2$ and a range of settings of K. (b) Bound as a function of alphabet size, q, for $L = 20$ and a range of settings of K.

N for which exact recovery occurs, then

$$\hat{C}_0 = \frac{\hat{N}}{S(f)\log_{10}(q^L)} \tag{4.19}$$

is the minimum value of C_0 that satisfies (4.2), where $S(f)$ is calculated calculated with Equation (4.13). We ran multiple replicates of this experiment for neighborhoods sampled according to the RN scheme, for different settings of L, q and K. In all of these experiments, $\hat{C}_0$ was less than 2.5, and we thus use $C_0 = 2.5$ for all further analysis. This value is in agreement with those often reported in the literature [36, 3]

Equipped with an estimate of C_0, we are now able to calculate the minimum number of samples required to exactly recover a GNK fitness function by using Equation (4.2) along with the sparsity calculations discussed in the previous section. Specifically,

$$N = \lceil C_0 \cdot S(f)\log_{10}(q^L) \rceil \tag{4.20}$$

is the minimum number of samples that guarantees exact recovery, where $\lceil\rceil$ represents the ceiling operator. Examples of these calculations are shown in Figure 4.4, where we calculated upper bounds on the minimum number of samples required to recover GNK fitness functions with uniform neighborhood sizes for a range of settings of L, q, and K. The "bound" portion of this statement is due to the upper bound on the sparsity given in (4.14), and not to any aspects of the CS theory. Note that Figures 4.4a and 4.4b result directly from applying

Equation (4.20) to the sparsity calculations shown in Figures 4.3a and 4.3b, respectively. We do not show an analog to Figure 4.3c because q and L are held constant in this plot, so in the analogous sample complexity plot, each curve will simply be multiplied by a constant factor of $C_0 \log_{10}(q^L)$, but will remain identical relative to on another. The intuition from Figure 4.3c follows directly to the sample complexity predictions.

The results in Figure 4.4 concretely show how many samples are required to perfectly estimate a fitness function under simple assumptions about that function. We only require that (i) the fitness function is sampled from the GNK model, and (ii) that the maximum degree of epistatic interaction is equal to K. There are two important insights that can be derived from Figure 4.4. First, the number of samples required to perfectly estimate these fitness functions is many orders of magnitude smaller than the total size of sequence space. Consider, for instance the point in Figure 4.4 where $L = 50$ and thus the size of sequence space is $2^{50} \approx 10^{15}$, 10 orders of magnitude greater than the largest plotted sample complexity. Additionally, it is clear by comparing Figures 4.4a and 4.4b that increasing the alphabet size within biologically relevant ranges increases the number of samples required to recover fitness functions at a faster rate than increasing the length of the sequence.

The results in Figure 4.4 can be refined when more specific information about a particular fitness function is known. In the next section, we will move beyond the assumption of uniform neighborhood sizes and present a method for constructing neighborhoods based on interactions in a protein structure. We will apply this scheme to the mTagBFP structure and show that the sparsity of the resulting GNK fitness functions is a good approximation to that of the empirical mTagBFP2 fitness function due to Poelwijk et al. [136]. We then use the sparsity of the GNK model to predict the number of samples required to estimate the empirical fitness function, and show that the error of a LASSO estimator trained with this number of samples is indeed quite small.

4.10 Application to protein fitness functions

For a given fitness function of interest, the standard GNK neighborhoods may not sufficiently capture the true interaction structure between the positions in the sequence. However, one major feature of the GNK model, and our results on GNK fitness functions, is that they can be used with any given set of neighborhoods. In certain cases we may be able to construct the neighborhoods based on prior knowledge about the specific fitness function of interest. One such case is protein fitness functions where the 3D protein structure is known. These types of fitness functions are studied often in protein engineering pursuits, where one may wish to determine how mutations to a wild-type protein sequence (whose structure is often known) effect its function [189, 154, 136, 137], or to determine whether novel sequences will stably fold into a target structure [144]. In both of these cases, the fitness function will be intimately related to the protein structure, and thus, an appropriate GNK model for modeling these fitness function should use the information given by the structure.

In order to incorporate structural information, we will construct GNK neighborhoods

based on 'contacts' between amino acids that are inferred from the structure. Following [146], we say two amino acids are in contact if any pair of atoms in the amino acids are within 4.5 Angstroms (Å) of each in other in Euclidean distance. There is now a large body of work demonstrating that structural contacts can be predicted by analyzing the statistical correlations between positions in evolutionary related sequences [112, 128, 53] and also from the data resulting from a Deep Mutational Scanning (DMS) experiment [145, 155]. Further, [10] found that most apparent "long-range" correlations (i.e., correlations that cannot be explained by structural contacts) that are observed in evolutionary data analyses are actually due to structural contacts in homologous structures or homo-oligomeric interfaces. In both of the evolutionary and DMS cases, sequences are selected based on their fitness, either through natural selection or the experimental procedure of DMS, and this selection results in correlations that from which structural contacts can be recovered. Thus the success of these contact prediction algorithms suggests that protein fitness functions encode structural contacts. Here we make this assumption concrete by constructing neighborhoods based on structural contacts. In particular, we will use the "Structural Neighborhood" (SN) scheme for GNK model, defined below.

Structural Neighborhood In the SN scheme, the neighborhood of a position j contains all positions that are in structural contact with position j in a given 3D structure.

The most interesting aspect of this neighborhood construction scheme is how it encodes epistatic interactions through Theorem 4.2. In particular, GNK fitness functions with Structural Neighborhoods contain an r^{th} order epistatic interaction when $r-1$ of the amino acids corresponding to the interaction are in structural contact with a central amino acid. To our knowledge, this is a novel hypothesis to explain how higher order epistasis arises in protein fitness functions.

In order to test this hypothesis, we first applied the SN scheme to the crystal structure of the mTagBFP fluorescent protein (PDB: 3M24 [172]), and compared the resulting GNK fitness functions to the empirical fitness function of the closely-related mTagBFP2 protein (for which no structure is available) resulting from the experiments of [136]. The data from [136] reports the blue fluorescence brightness of all combinations of mutations in 13 positions in mTagBFP2, where each of these positions is allowed to mutate to only one other amino acid (i.e., $L = 13$ and $q = 2$ in this fitness function, and the data reports the fitness of all $2^{13} = 8192$ possible sequences). A graphical depiction of the Structural Neighborhoods corresponding to these 13 positions are shown in the first row of Figure 4.5a.

Letting $\mathcal{V}_{\text{SN}}$ be the set of these Structural neighborhoods, we first used Theorem 4.2 to calculate that $S(f) = 56$ for $f \sim \text{GNK}(13, 2, \mathcal{V}_{\text{SN}})$. We also used Theorem 4.1 to determine the identity and distribution of the 56 non-zero Fourier coefficients of the GNK fitness functions. Next, we solved for the Fourier coefficients of the empirical fitness function. Since this empirical landscape is combinatorially complete, we can solve for its Fourier coefficients with Ordinary Least Squares regression, which results in $\hat{\boldsymbol{\beta}} = (\boldsymbol{\Phi}^{(13,2)})^T \mathbf{y}$. We compared

the magnitudes of each empirical Fourier coefficient with the expected magnitude of the corresponding coefficient in the GNK fitness functions. The results of this comparison for all coefficients corresponding to up to 5^{th} order epistatic are shown in Figure 4.5b. The GNK model identifies many of the largest higher-order epistatic interactions in the empirical landscape as being non-zero, suggesting that the SN scheme is able to approximate the interaction structure of protein fitness functions.

We then tested how well the GNK model approximates the sparsity of the empirical fitness function. Although none of the empirical Fourier coefficients are exactly zero, these coefficients display substantial approximate sparsity. In particular, over 95% of the total variance in the coefficients can be explained by the 25 coefficients with the largest magnitude, where the total variance is given by $\|\hat{\boldsymbol{\beta}}\|_2^2$. The blue curve in Figure 4.5c shows the Percent Variance Explained by the empirical Fourier coefficients with the largest magnitudes. We compared this to the percent variance explained by the largest Fourier coefficients of samples of the GNK fitness functions; the mean and standard deviation of these calculations for 10,000 sampled GNK fitness functions are shown as a red curve and shaded region in Figure 4.5c. Considering that this plot shows only the first 75 out of the 8,192 total points that could be included on the horizontal axis, it is clear that the GNK model approximates the sparsity of the empirical fitness function qualitatively well. Of particular importance is the point at which all of the nonzero coefficients of the GNK fitness functions are included in the calculation (i.e., 100% of the variance is explained), which occurs at $S(f) = 56$, and is highlighted by a dashed line in Figure 4.5c; at this point, 97.1% of the empirical variance is explained. This suggests if we use the GNK model with Structural Neighborhoods to predict the number of sample required to exactly recover the mTagBFP2 fitness function, the error resulting from applying a CS algorithm on this number of sampled fitness values will be quite small. We confirmed this by using LASSO to estimate the empirical fitness function.

We repeated these tests for a nearly combinatorial-complete fitness function that is embedded in the data of [137], with results shown in the second row of Figure 4.5. This data reports on the fitness of variants of the protein encoded by the His3 gene in yeast, and the sub-function contains fitness measurements for 2030 out of 2048 possible sequences of 11 positions in the sequence that take on one of two amino acids (i.e., this is an $L = 11$ and $q = 2$ fitness function). We constructed Structural Neighborhoods for these positions based on the I-TASSER [192] predicted structure of this protein reported in [137] and compared the sparsity of the resulting GNK fitness functions to that of the empirical fitness function, with similarly promising results as we have reported for the mTagBFP2 fitness function. In particular, the GNK fitness functions in this case are 76-sparse, and 90.4% of the empirical variance is explained by the 76 coefficients with the largest magnitudes. Additionally, LASSO estimates of the empirical function trained with the predicted number of samples required for recovery have a mean prediction R^2 of 0.865.

4.11 Conclusion

In this chapter, we have described a theory for predicting the number of fitness measurements required to estimate a fitness function under a well-defined set of assumptions. In particular, we have presented results that allow one to calculate the sparsity in the Fourier basis of fitness functions sampled from the Generalized NK model given a sequence length, alphabet size, and set of neighborhoods that encode the assumed interaction structure between positions in the sequence. We used these results within the theory of Compressed Sensing to determine the minimum number of sequences sampled uniformly at random from the space of sequences whose fitness must be measured in order to exactly recover the fitness function. These results allow us to test the effect of sequence length, alphabet size, and interaction structure on the sample complexity of fitness function estimation. Finally, we have described a technique for constructing GNK neighborhoods based on a protein structure, and showed that (i) the resulting GNK model accurately approximates the sparsity of an empirical fitness function, (ii) LASSO estimates of the empirical fitness function that have very little error when trained with the number of samples required to exactly recover the GNK fitness function. This suggests that our results may be able to be used as a practical tool for designing experiments such that one can expect to recover a fitness function with a minimal error.

A few comments should be made about the applicability of these results to practical situations. First, the CS theory applies only when measured sequences are sampled *uniformly* from the space of sequences. Although there can be some leeway in this requirement in practice, the theory, and our results, will only provide lower bounds on the sample complexity of fitness function estimation if the sampling distribution is substantially non-uniform. The most biologically relevant case where the sampling distribution is nonuniform is when a wild-type sequence is randomly mutated with error-prone PCR. In this case, the number of mutations in each mutated sequence is distributed roughly according to the Poisson distribution [47]. Therefore, the likelihood of observing sequences near the wild-type is much larger than observing those with many mutations from the wild type. This type of substantial non-uniformity in the sampling distribution means that many more samples from this distribution may be required to estimate the fitness function than predicted by our theory, and our predictions should be used with caution in this case.

There are a number of practical scenarios that may reduce the predictive power of our results. One is when the protein structure used to construct the GNK Structural Neighborhoods fails to describe the structural contacts that occur due to homo-oligomerization or alternative conformations of the protein, and thus the resulting GNK model will not include structurally-important epistatic interactions. In certain cases, it may be possible to resolve this when the homo-oligomeric structure is available, and when the structures of evolutionary-related proteins display different conformations [10]. In general, care should be taken to construct Structural Neighborhoods that most accurately represent the contacts that occur in the protein. Another case where our theory will lose predictive power is when global non-linearities in the data due to the measurement process or biophysical considera-

tions [190, 125] induce spurious higher-order epistatic [152]. These global nonlinearities may cause a fitness function to appear not to be sparse, and sparse recovery algorithms may fail at estimating such functions from a small number of samples. A number of simple methods exist for removing such nonlinearities [152, 126, 175], which should be applied before comparing the sparsity of an observed fitness function that predicted by our theory or using a sparse recovery algorithm to estimate such a fitness function.

Although we have used LASSO as a motivating example, there exist sample complexity guarantees of the form of Equation (4.2) for a number of CS algorithms [33]. Many of these algorithms (for instance, LASSO) will suffer from time and memory complexity limitations when applied to Fourier bases for long sequences with large alphabets, and thus practical recovery algorithms for these cases should be developed. A promising direction is to use machine learning models such as deep neural networks in conjunction with sparsity-inducing regularization in the Fourier basis [4]; this combines the observation that fitness functions are sparse in the Fourier basis with the representational power of neural networks. Although it may not be possible to provide recovery guarantees for algorithms of this sort, the continued empirical success of machine learning models justifies further exploration in this space.

In Chapters 2 and 3 we explored aspects of inverse problems in applying machine learning to biological sequence design, and in this chapter we have discussed various facets of the corresponding forward problem. In the next chapter, we will describe an end-to-end computational approach for designing a library of Adeno-associated viral capsids to improve packaging ability, which requires solving both forward and inverse problems.

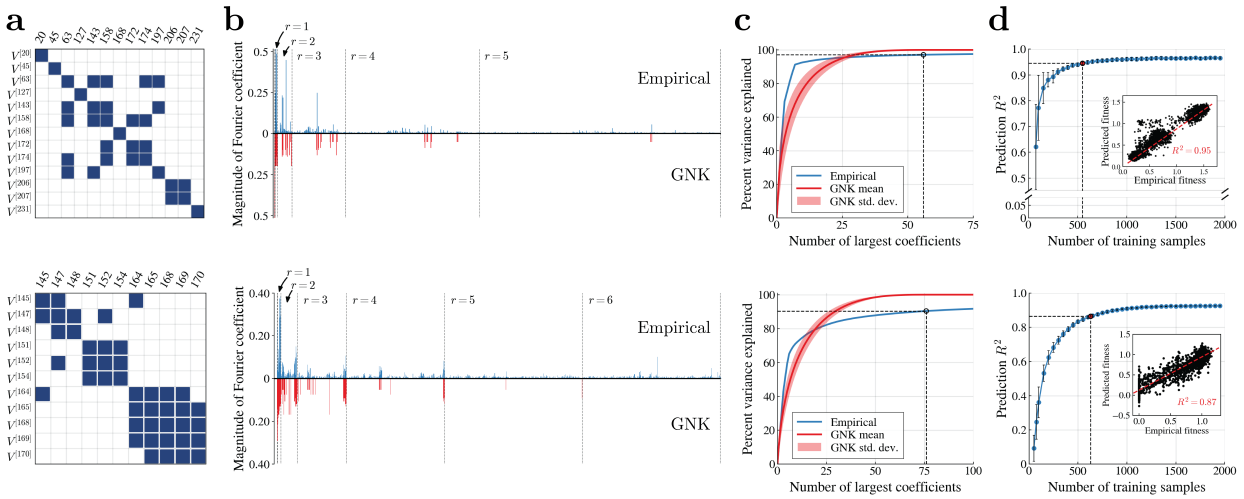


Figure 4.5: Comparison of empirical fitness functions with the GNK fitness with Structural Neighborhoods. *First row*: comparison to mTagBFP2 fitness function from ref. [136]. *Second row*: comparison to yeast His3 fitness function from ref. [137]. (a) Structural Neighborhoods derived from crystal structural of mTagBFP (first row) and I-TASSER predicted structure of protein encoded by His3 gene (second row). (b) Magnitude of empirical Fourier coefficients (upper plot, in blue) compared to the standard deviations of coefficients in the GNK model (reverse plot, in red). Dashed lines separate orders of epistatic interactions, with each group of r^{th} order interactions indicated. (c) Percent of total variance explained by the largest Fourier coefficients in the empirical fitness functions and in fitness functions sampled from the GNK model. The dotted line indicates the exact sparsity of the GNK fitness functions, which is 56 is in the first row and 76 in the second, at which points 97.1% and 90.4% of the empirical variances are explained, respectively. (d) Error of LASSO estimates of empirical fitness functions at a range of training set sizes. Each point on the horizontal axis represents the number of training samples, N, that are used to fit the LASSO estimate of the fitness function. Each point on the blue curve represents the R^2 between the estimated and empirical fitness functions, averaged over 50 randomly sampled training sets of size N. The point at the number of samples required to exactly recover the GNK model with Structural Neighborhoods ($N = 548$ in the first row, and $N = 630$ in the second) is highlighted with a red dot and dashed lines; at this number of samples, the mean prediction R^2 is 0.945 in the first row and 0.865 in the second. Insets show paired plots between the estimated and predicted fitness function for one example training set of size $N = 548$ (first row) and $N = 630$ (second row). Error bars indicate the standard deviation of R^2 over training replicates.

Chapter 5

Designing a library of AAV capsids to improve packaging

5.1 Introduction

In the previous chapters we have considered various aspects of forward and inverse problems in machine learning approaches to biological sequence design independently from one another. Additionally, we have done so in mostly conceptual and theoretical scenarios. In this chapter, we will combine forward and inverse modeling tools to build an end-to-end computational pipeline for (i) processing sequencing data from a high-throughput selection experiment (ii) building a forward model to predict the fitness of individual variants and (iii) using this model to design a libraries that optimally balance expected predicted fitness and diversity. We will apply this pipeline to design a library of insertion sequences to the Adeno-associated virus (AAV) capsid where the aim is improve the ability of the capsids in the library to package the viral genome, while still maintaining a diverse library. The design of a diverse library of sequences represents a distinct inverse problem from that tackled in Chapter 2, and we will present a new set of techniques based on a Maximum Entropy formalism to solve this problem. This work was done in collaboration with the Schaffer Lab at UC Berkeley, who performed the wet-lab experimental portions of the project.

Modified AAV variants are promising vectors for the delivery of gene therapies. Indeed, two AAV-based gene therapies for treating rare inherited diseases are currently FDA-approved [55, 54]. There are two major tasks in developing gene therapies that use AAV as a delivery vector [97]: (i) a genetic "payload" must be developed that will replace the gene implicated in a disease and (ii) the AAV capsid must be modified such that it can effectively deliver the payload to the target tissue. To a certain extent, these problems can be treated separately, and here we focus on the latter problem of designing capsid sequences. Developing generalizable methods to engineer AAV capsids that can deliver therapies to a given tissue promises to greatly enhance the speed with which new gene therapies can be developed, and the ultimate efficacy of these therapies.

The challenges in designing AAV capsids to deliver gene therapies are numerous. First, the majority of the human population have been exposed to one or more variants of AAV, and have developed an immune response to the virus [25]. If an AAV is to deliver a therapy, it must be able to evade the neutralizing antibodies produced by the immune system to prevent infection. Additionally, natural evolution has not resulted in AAVs that infect all tissues or cell types with equal effectiveness. Modifications to the capsid must be made such that the virus can efficiently navigate the intercellular spaces of a tissue of interest, and so that the virus can bind to receptors on the surface of a target cell type and enter those cells [97].

Directed Evolution [38] has emerged as a powerful tool for discovering AAV variants that are able to evade neutralizing antibodies and infect target tissue types [17]. A particularly notable example of this methodology for the purposes of this chapter was carried out by Dalkara et al. [41], who subjected a diversified library of AAV capsid sequences to multiple rounds of *in-vivo* selection with the aim of discovering viral variants that most effectively infect the outer retina in mice. The diversified capsid sequences library was constructed with three independent strategies: random mutagenesis around a particular point mutant, shuffling of extant capsid sequences, and insertion of a random seven amino acid sequence to the heparin binding domain of the capsid, which is implicated in cell surface binding and viral entry [88]. Ultimately, Dalkara et al. [41] found that sequences resulting from the insertion of random 7-mer sequences dominated the pool of sequences after multiple rounds of selection, suggesting that this diversification strategy is a powerful scheme for constructing libraries of capsid sequences that can be engineered experimentally to target specific tissue types. A number of additional studies have demonstrated the effectiveness of adding these 7-mer insertion sequences to the AAV capsid for infecting target tissues [105, 176, 45, 31]. In this chapter, we will describe a joint experimental and computational pipeline for engineering these 7-mer sequences to improve a different, though closely related, property of the resulting AAV variants; namely, the ability of the capsid to properly fold and package the viral genome.

A prerequisite for an AAV variant to deliver a gene therapy is that the capsid protein is able to adopt a stable fold, assemble into the requisite 60-mer capsid structure, and enclose the viral genome within that structure [122]. We will refer to the capacity of a variant to perform these three steps as its "packaging" ability, though this has also been referred to as the "viability" of a variant [122, 29]. Packaging is an inherently probabilistic trait: two separate capsid molecules with the same sequence may differ in the extent to which they successfully fold, assemble and package the genome, simply due to thermodynamic fluctuations and other random biophysical effects. If a particular viral variant has an improved probability of packaging relative to a reference variant, then the viral replication process will result in more viable viruses that can further replicate and infect cells. A reasonable expectation is thus that a capsid with improved packaging ability is more likely to be able to be mutated slightly in order to infect a specific cell type or tissue, as is the overall goal in developing AAV gene therapies.

Based on this presumption, we set out to design a library of 7-mer insertion sequences with

improved packaging ability over a baseline library. The ultimate aim is to use this library as starting point for hypothetical downstream selections like those of Dalkara et al. [41] that seek to engineer AAV capsids to infect specific tissues and cell types. The hypothesis is that these downstream selections are more likely to produce viable gene therapy vectors if the initial library contains more capsid sequences that are able to effectively package. Importantly, in order to increase the success of these downstream selections our designed library should not only contain variants with a high probability of packaging, but should also be as diverse as possible. Encouraging diversity in the library ensures that we do not overly bias towards effective packagers and then do not include variants that are well suited for infecting specific tissues and cell types. In order to design this library, we performed three steps: we (i) experimentally determined the packaging ability of over 5 million unique 7-mer insertion that were randomly sampled from a baseline library, (ii) built a neural network forward model to predict a scalar representation of packaging ability derived from the experimental data and (iii) used this forward model within a novel framework based on a maximum entropy principle [82, 40] to design a library that optimally balances predicted average packaging ability with diversity. We applied our pipeline to design insertion sequences to the AAV5 capsid. AAV5 is an AAV serotype that is particularly promising for delivering gene therapies because there exist relatively low levels of neutralizing antibodies for it in human subjects and it has been shown to infect a broad range of tissues and cell types [133].

The rest of this chapter is structured as follows: in Section 5.2 we briefly describe the selection experiment that is the first step in our pipeline; in Section 5.3 we discuss initial processing and analysis of the sequencing data resulting from the experiment; in Section 5.4 we describe our methods for building forward models to predict the packaging ability of insertion sequences; in Section 5.5 we describe a maximum entropy technique for designing sequence libraries that optimally balance high mean predictions in the forward model and sequence diversity; we conclude in Section 5.6.

5.2 Description of experiments

The baseline insertion sequence library that we sought to improve over was constructed by experimentally sampling sequences from 7 concatenated copies of the NNK degenerate codon; this is the same library construction technique as was used to build the library of insertion sequences in [41]. A degenerate codon specifies the marginal probability of each nucleotide occurring in each of the three positions in a codon. The NNK codon specifies the probability distribution where every nucleotide is equally likely in the first two positions of the codon, and in the final position Adenine and Cytosine have zero probability, while Thymine and Guanine are each equally probable. This distribution is shown in Table 5.1, below. The NNK degenerate codon is useful because it induces a distribution over amino acids where every amino acid has non-zero probability, but the probability of stop codons is minimized.

Insertion sequences sampled from the NNK($\times$7) distribution were inserted into a plasmid

	1	2	3
A	0.25	0.25	0
T	0.25	0.25	0.5
C	0.25	0.25	0
G	0.25	0.25	0.5

Table 5.1: Table of nucleotide probabilities specified by the NNK degenerate codon.

containing the AAV5 genome immediately following the genomic locus representing position 587 in the amino acid sequence of the capsid protein. Position 587 is located in the proximity of the three-fold symmetry axis of the capsid, protruding from the external surface of the virion and has been implicated in receptor binding that is critical for cell-specific entry of the virus. Additionally, Ogden et al. [122] showed that the region around position 587 is one of the few regions of the capsid sequence in which single-site mutations can improve packaging ability. We will refer to the resulting library of capsid variants as the *pre-selection* library. The pre-selection library was then introduced to HEK293T cells and incubated for 72 hours, during which time the processes of viral replication, capsid folding and assembly, and genome packaging occur. The conjecture underlying this experiment is that sequences that have a higher probability of packaging will replicate more effectively during this incubation with the HEK cells, and thus there will be a higher proportion of these sequences in the *post-selection* library, which is constructed by lysing the cells after 72 hours and purifying the resulting viral particles.

Short segments of the viral genomes containing the insertion sequences in both the pre- and post- selection libraries were then PCR amplified and sequenced with the Illumina NovaSeq 6000 platform.

5.3 Data processing and analysis

Our aim was to use the data resulting from the experiment described in the previous section to train a supervised regression model to predict the packaging ability of insertion sequences. A supervised model requires a training set of $(\mathbf{x}, y)$ pairs, where $\mathbf{x}$ represents a set of features that are input into the model (in our case, some representation of an insertion sequence) and y represents the numerical quantity that the model is to predict (in our case, some measure of packaging ability). The sequencing data resulting from the experiment described in the previous section does not immediately report $(\mathbf{x}, y)$ pairs that can be used to train a supervised model, and further processing was required.

In particular, the raw sequencing data consists of 49,619,716 and 55,135,155 sequencing reads corresponding to the pre- and post-selection libraries, respectively. Each read contains (i) a 5 bp unique molecular identifier, (ii) a fixed 21 bp primer sequence, (iii) a 6 bp sequence representing the pre-insertion linker (two fixed amino acids that connect the insertion sequence to the capsid sequence at position 587), (iv) a variable 21 bp sequence containing the

nucleotide insertion sequence, and (v) a 9 bp representing the post-insertion linker (three fixed amino acids that connect the insertion sequence to the capsid sequence at position 588). We filtered the reads, removing those that either contained more than 2 mismatches in the primer sequences or contained ambiguous nucleotides. After this filtering, the pre and post libraries contained 46,049,235 and 45,306,265 reads, respectively. The insertion sequences were then extracted from each read and translated to amino acid sequences. In each library, we then counted the number of reads containing each unique insertion sequence, resulting in pre- and post-selection 'counts' for 8,552,729 unique sequences. Note that only 218,942 of these sequences appear in both libraries, and all others occur only in one of the libraries (and have a count of zero in the other library). This is due to limitations of current sequencing technology, which does not have the capacity to sequence every unique sequence in a library.

In order to convert this data into a numerical quantity that describes the packaging ability corresponding to each insertion sequence, we calculated log "enrichment scores" for each sequence, which is equal to the normalized log ratio of the counts of a sequence in the post- and pre- selection libraries.[1] In particular the log enrichment score for the i^{th} sequence is given by:

$$y_i = \log \frac{n_i^{\text{post}}}{n_i^{\text{pre}}} - \log \frac{N^{\text{post}}}{N^{\text{pre}}}, \tag{5.1}$$

where n_i^{post} and n_i^{pre} are the counts corresponding to sequence i in the pre- and post- selection libraries, respectively, and $N^{\text{post}} = \sum_i n_i^{\text{post}}$ and $N^{\text{pre}} = \sum_i n_i^{\text{pre}}$ are the total number of counts in each library. A pseudocount of 1 was added to each count so that the enrichment score could still be calculated when the sequence only appeared in one of the libraries, and thus had a count of zero in the other library. Enrichment scores are often used in the modeling of sequencing data of the sort produced by our experiment, and many of statistical properties of these quantities have been studied [16, 149].

Intuitively, a log enrichment score reports on how the population of capsids containing a unique insertion sequence changed due to the selection in the HEK cells. A positive enrichment score indicates that the population of an insertion sequence increased, which suggests that the capsid containing this sequence was able to consistently fold, assemble and package the viral genome so that it could continue to replicate. A negative enrichment score, on the other hand, indicates that capsids containing that insertion sequence did not replicate, which is likely due to a failure in one of the steps involved with packaging. The log enrichment scores thus provide a numerical description of an insertion sequence's effect on packaging, which we can use to train a supervised model.

One concern with using enrichment scores to train a supervised model is that they do not provide any information about the raw values of the counts used to compute them. For instance, a log enrichment score $\log(1000/100)$ calculated with $n_i^{\text{post}} = 1000$ and $n_i^{\text{pre}} = 100$ appears identical to $\log(10/1)$ calculated with $n_i^{\text{post}} = 10$ and $n_i^{\text{pre}} = 1$; however, intuitively,

[1] Although not always termed "enrichment scores", this ratio of sequencing counts is commonly used as a measure of fitness [110].

the former score is less likely to have arisen due to noise in the experiment and should be more heavily considered by the model.

We can overcome this problem by estimating a variance associated with each log enrichment score, which takes into account information about the raw counts associated with an enrichment score. To understand how this is calculated, we first recognize that the count associated with a sequence is a random variable. The PCR-amplified libraries contain many orders of magnitudes more molecules than the Illumina platform is able to sequence, and thus each read can be considered a Bernoulli sample from a density of sequences. In this sense, we can model a count as a Binomial random variable, since it represents the number of times that a Bernoulli sample is observed [110]. The log enrichment score is then the log ratio of two Binomial random variables; it can be shown with the Delta Method [87] that, in the limit of infinite samples, the log ratio of two Binomial random variables converges in distribution to a normal random variable [110]. Further, it has been shown that the mean of this normal distribution for count data can be approximated by the log enrichment score of Equation (5.1), and the variance can be approximated by [84, 110]:

$$\sigma_i^2 = \frac{1}{n_i^{\text{post}}}\left(1 - \frac{n_i^{\text{post}}}{N^{\text{post}}}\right) + \frac{1}{n_i^{\text{pre}}}\left(1 - \frac{n_i^{\text{pre}}}{N^{\text{pre}}}\right). \tag{5.2}$$

We can see that these variances are lower for sequences with more raw counts and lower for those with fewer counts, which matches our intuition.

In the next section, we will see how we can use the variances associated with enrichment scores to properly weight data points within a supervised modeling framework. To our knowledge, this is a novel use of these variances.

5.4 Building forward models to predict enrichment scores

The result of the processing described in the previous section was a data set of the form $\{(\mathbf{x}_i, y_i, \sigma_i^2\}_{i=1}^M$ where the $\mathbf{x}_i$ are unique insertion sequences, y_i are log enrichment scores associated with the insertion sequence, σ_i^2 are the estimated variances of the log enrichment scores, and $M = 8,555,729$ is the number of unique insertion sequences in the data. We first randomly split this data into a training set containing 80% of the data and a test set containing the remaining 20% of the data.

As described in the previous section, we can model the log enrichment scores as normal random variables with a particular mean and variance. We further assume that the distribution of an enrichment score given the associated insertion sequence is

$$y_i|\mathbf{x}_i, \sigma_i^2 \sim \mathcal{N}(f_{\boldsymbol{\theta}}(\mathbf{x}_i), \sigma_i^2), \tag{5.3}$$

where $f_{\boldsymbol{\theta}}$ is a function with parameters $\boldsymbol{\theta}$ that parameterizes the mean of the this distribution, and represents a predictive model for enrichment scores. We determined suitable settings of

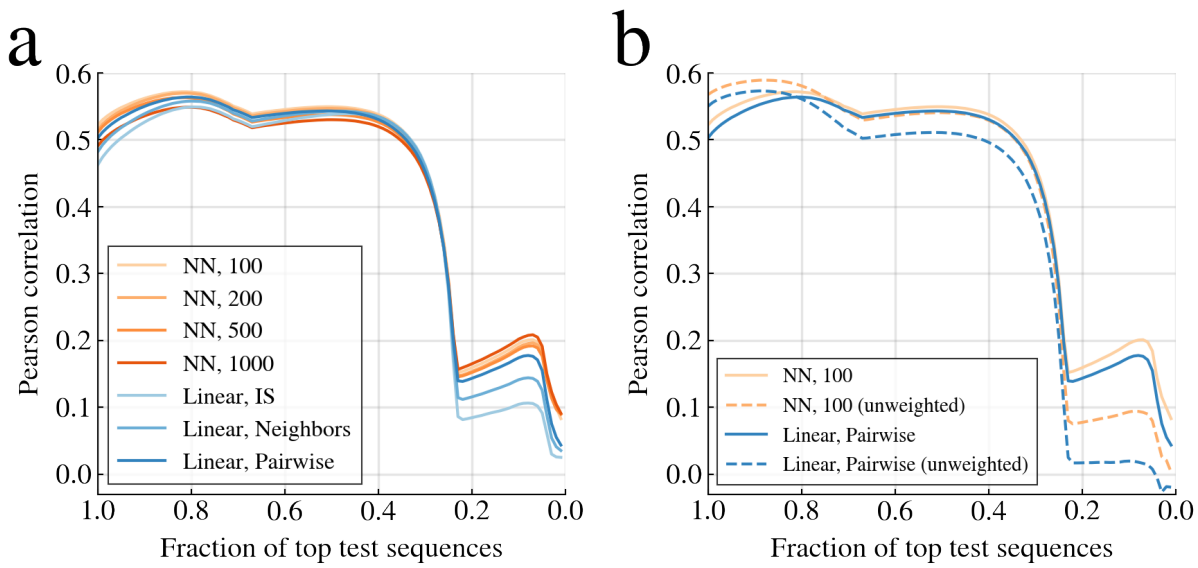


Figure 5.1: Comparison of forward models for predicting AAV5 enrichment scores. In both figures, each point along each curve represents the Pearson correlation between the predicted and true enrichment scores of sequences in a culled subset of the test set. Points along the horizontal axis represent the fraction of most highly enriched sequences in the test set that were included in the subset. Points on all curves are calculated at every 0.01 increment of the horizontal curves. (a) Comparison between all tested models using the weighted loss function. (b) Comparison between models trained with the weighted and unweighted loss functions for two representative models.

the parameters $\boldsymbol{\theta}$ with Maximum Likelihood Estimation (MLE). The log-likelihood of the parameters of this model given the training set of $M' \leq M$ data points is given by

$$\ell\left(\boldsymbol{\theta}; \{\mathbf{x}_i, y_i, \sigma_i^2\}_{i=1}^m\right) = -\frac{m}{2}\log 2\pi - \frac{1}{2}\sum_{i=1}^{m}\left[\log \sigma_i^2 + \frac{1}{\sigma_i^2}\left(y_i - f_{\boldsymbol{\theta}}(\mathbf{x}_i)\right)^2\right]. \tag{5.4}$$

Performing MLE by optimizing this likelihood with respect to the model parameters $\boldsymbol{\theta}$ results in the following optimization objective

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{i=1}^{M'} \frac{1}{\sigma_i^2}\left(y_i - f_{\boldsymbol{\theta}}(\mathbf{x}_i)\right)^2. \tag{5.5}$$

We can see that this is a weighted least-squares loss function, where the weight of each data point is inverse to its variance estimated by Equation (5.2). This has the effect of giving more weight to insertion sequences with more raw counts, as desired. We thus call Equation (5.5)

the "weighted" loss function. To demonstrate the effectiveness of our modeling approach, we also compare to models with an "unweighted" loss where we set $\sigma_i^2 = 1$ for every i, and which represents a standard least-squares loss function.

It is not immediately clear what functional form for $f_{\boldsymbol{\theta}}(\mathbf{x})$ will model the data most effectively. Therefore we evaluated 7 different forms of $f_{\boldsymbol{\theta}}$ which ranged in complexity from a simple linear model to a neural network model with over 1 million parameters. In particular, we tested three linear functions and four feed-forward neural networks (NNs). The three linear models differed in the set of sequence features that were linearly combined. In the first, "Independent Site" (IS) representation, the sequences are one-hot encoded and a parameter is assigned to each position in the one-hot matrix. In the second, "Neighbors" representation, interactions between neighboring positions in the sequence are also one-hot encoded, and in the third, "Pairwise" representation, all possible interactions between positions are one-hot encoded. Let x^j be the amino acid (represented as an integer between 1 and 21) at the j^{th} position in sequence $\mathbf{x}$, for $j = 1, 2, ...L$ where $L = 7$ is the length of the insertion sequences. The linear model with IS features can then be formally expressed as

$$f_{\boldsymbol{\theta}}^{\text{IS}}(\mathbf{x}) = \sum_{j=1}^{L} \sum_{m=1}^{21} \theta_{jm} \delta_m(x^j) \tag{5.6}$$

where $\delta_m(x^j) = 1$ if $x^j = m$ and zero otherwise, and the parameters $\boldsymbol{\theta}$ have been arranged in an $L \times 21$ matrix such that θ_{jm} is a weight corresponding to amino acid m at position j in the sequence. The linear models with Neighbors and Pairwise features expand on that with IS features by adding terms representing interactions between positions:

$$f_{\boldsymbol{\theta}}^{\text{neighbors}}(\mathbf{x}) = f_{\boldsymbol{\theta}}^{\text{IS}}(\mathbf{x}) + \sum_{j=1}^{L-1} \sum_{m,n=1}^{21} \theta_{jmn} \delta_m(x^j) \delta_n(x^{j+1}) \tag{5.7}$$

$$f_{\boldsymbol{\theta}}^{\text{pairwise}}(\mathbf{x}) = f_{\boldsymbol{\theta}}^{\text{IS}}(\mathbf{x}) + \sum_{j=1}^{L} \sum_{k=j}^{L} \sum_{m,n=1}^{21} \theta_{jkmn} \delta_m(x^j) \delta_n(x^k) \tag{5.8}$$

where the interaction parameters have been arranged into appropriate tensors. For all of these linear models, Equation (5.5) is a convex objective function, and the optimal ML parameters can be solved for exactly. In order to stabilize the training of the latter two models, we used a small amount of L2 regularization, with regularization coefficients 0.001 and 0.0025 chosen by cross-validation for the Neighbors and Pairwise representations, respectively.

All of the neural networks that we trained used the IS sequence encoding as inputs, and had two densely connected hidden layers with *tanh* activation functions. The four models differed in the size of each hidden layer, with 100, 200, 500, and 1000 nodes in the hidden layers of each respective model. For these neural network forms of $f_{\boldsymbol{\theta}}$, the objective of Equation (5.5) is non-convex and we use stochastic optimization techniques to solve for suitable parameters. In particular, we implemented these models in Tensorflow [1] and used

the built-in implementation of the Adam algorithm [91] to approximately solve Equation (5.5).

Our aim was to choose one of these models to use within a method to solve the inverse problem of designing a library. In order to assess and compare the prediction quality of each model, we calculated the Pearson correlation between the model predictions and observed enrichment scores for different subsets of sequences in the test set. Our ultimate aim is to use these models to design a library of sequences that package well (i.e., would be highly enriched in the post-selection library), so we wanted to assess how well the models performed for highly enriched sequences. In order to do so, we progressively culled the test set to only include the sequences with the largest observed enrichment scores. The results of these calculations for all of the tested models trained with the weighted loss function are shown in Figure 5.1a.

We can see that neural networks in general perform better than linear models. This is likely due to the fact that the neural network models are able to represent higher-order epistatic interactions that are present in this fitness function, while the tested linear models encode at most second order epistasis. Ultimately we chose the neural network with 100 nodes in each hidden layer, (NN, 100), to use for the next step of designing a library due to its competitive performance with (NN, 1000) and the fact that it has many fewer parameters than (NN, 1000). We next trained two of the models, (NN, 100), and the linear model with pairwise interactions, with the unweighted loss function to compare to the models trained with the weighted loss in order to test the benefits this formulation. The results of this comparison are shown in Figure 5.1b, where we can a clear performance benefit of using the weighted loss function for making predictions on highly enriched sequences.

In the next Section, we will use the NN, 100 model within a maximum entropy framework in order to design insertion sequences libraries that optimally balance model predictions and diversity.

5.5 Maximum entropy techniques for designing optimally balanced libraries

Given the forward model developed in the previous section, the next step in our pipeline was to design a library of insertion sequences to use as a starting point for a downstream selection experiment. In order to do so, we developed a general framework for sequence library design that can be adapted for use with any predictive model of fitness. We will first describe the general framework before applying it to the problem of designing a library of AAV5 insertion sequences.

A general framework

Inherent in the problem of library design is a trade-off between optimizing for the predicted fitness and creating a library with a diverse set of sequences. The library that opti-

mizes the average fitness contains only the single sequence that is most fit, while the most diverse library is uniformly distributed across sequence space irrespective of fitness. The most useful library typically lies somewhere between these two extremes.

Modern library construction techniques allow for different levels of control over the resulting library. For example, one can specify the desired marginal statistics of nucleotides or amino acids at each position in the sequences in the library; a list of specific sequences that are to be included in the library; or a set of parent sequences on which to apply random mutagenesis and/or recombination. Increasingly, the parameters of these library construction techniques are chosen using predictive models of fitness: see, e.g., [129], [29] and [197] for computational tools that use predictive models to specify parameters of each of the three aforementioned library construction techniques, respectively. In some cases, these computational techniques explicitly encourage diversity in the resulting library [104], or diversity may be a natural byproduct of the method that is analyzed *post-hoc* [22]. In this section we present a set of techniques to optimally balance diversity and predicted fitness for a variety of library construction techniques. This enables us to rationally choose library construction parameters, as well as compare different library construction techniques. Notably, our methods are general in that they can be used with any predictive model of fitness, are broadly applicable to different library construction mechanisms, and are simple to implement and extend.

Our approach is based on a maximum entropy formalism, where we represent libraries as probability distributions and aim to find so-called "maximum entropy distributions" that optimize the entropy while also satisfying a constraint on the expected fitness, which is predicted by a user-specified model such as a neural network. Entropy is a measure of diversity for probability distributions which has been used extensively in ecology to describe the diversity in populations [179]. By varying the constraint on the expected fitness, the resulting maximum entropy distributions trace out a Pareto optimal frontier where one cannot improve the diversity without decreasing the expected fitness, and *vice-versa*.

This framework builds on previous work in which multiple optimization objectives are balanced in the design of sequence libraries. Most notably, Parker et al. [129] presents a method for designing libraries constructed by combinatorial mutagenesis using an unlabeled set of natural functional sequence variants. This method is designed to optimally balance a trade-off between so-called "quality" and "novelty" scores; the quality score is a fitness prediction from a sequence potential (i.e., the energy function of a Potts model) whose parameters are fit to the unlabeled set of natural sequences, and the novelty score is a metric that assesses how dissimilar sequences are to any of the natural sequences. Notably, the novelty score encourages sequences to be different from the given set of natural sequences, but not necessarily to be different from one another, and thus does not map directly to our notion of diversity. Verma et al. [182] builds on this work and uses the concept of a Pareto frontier to optimally balance multiple fitness predictors when designing libraries. This latter method again optimizes for novelty scores but do not consider diversity as one of the optimization objectives directly. Our methods improve upon these techniques by (i) allowing for the use of any predictive model of fitness and (ii) explicitly considering a

trade-off with entropy, a natural measure of diversity.

We will show how to apply our formalism to two library construction possibilities that are available through commercial vendors: (i) the "specified" case, where one specifies a list of sequences that are to comprise the library and (ii) the "degenerate" case, where one specifies the marginal probabilities of each nucleotide (or amino acid) at each position in the sequence. We will refer to libraries constructed from the construction techniques corresponding to possibilities (i) and (ii) as specified and degenerate libraries, respectively.

The maximum entropy formulation

Let $\mathcal{X}$ be the space of all sequences may be included in a library (e.g., all amino acid sequences of length 7). We will consider a sequence library to be an abstract quantity represented by a probability distribution with support on $\mathcal{X}$. Let $\mathcal{P}$ represent all such libraries and $p \in \mathcal{P}$ one particular library. The entropy of this library is given by [106]:

$$H[p] = -\sum_{\mathbf{x}\in\mathcal{X}} p(\mathbf{x}) \log p(\mathbf{x}) \tag{5.9}$$

Intuitively, the entropy of a library is smaller when probability mass is concentrated on a small number of sequences, and is larger when the probability mass is spread out among many sequences (i.e., the library is diverse). Now let $f(\mathbf{x})$ be a predictive model of fitness. Our aim is to find a diverse library, $p(\mathbf{x})$, where the expected predicted fitness in the library, $\mathbb{E}_{p(\mathbf{x})}[f(\mathbf{x})]$, is as high as possible. One way to formulate this goal is to say that we want to find the most diverse library (i.e., the library with the largest entropy) such that the expected predicted fitness is above some cutoff. Formally, this objective is written:

$$\begin{aligned} &\max_{p\in\mathcal{P}} H[p] \\ &\text{s.t. } \mathbb{E}_{p(\mathbf{x})}[f(\mathbf{x})] \geq a, \end{aligned} \tag{5.10}$$

where a is the cutoff on the expected predicted fitness. It is straightforward to show that the solution to the objective in Equation (5.10) is given by [82]:

$$p_\lambda(\mathbf{x}) = \frac{1}{Z(\lambda)} \exp\left(\frac{f(\mathbf{x})}{\lambda}\right) \tag{5.11}$$

where $\lambda > 0$ is a Lagrange multiplier that is a monotonic function of the cutoff a and $Z(\lambda) = \sum_{\mathbf{x}\in\mathcal{X}} \exp\left(f(\mathbf{x})/\lambda\right)$ is a normalizing constant. Equation (5.11) gives the probability mass of what is known as a maximum entropy distribution; other examples of maximum entropy distributions include members of the the exponential family of distributions in statistics [87] and the Boltzmann distribution in statistical mechanics [37]. The setting of the parameter λ represents a trade-off between expected predicted fitness and diversity: at a smaller value of λ, p_λ will have high expected predicted fitness, but will be less diverse (i.e., will have probability mass concentrated on a small number of sequences), and as λ is increased,

the diversity will increase but the expected predicted fitness will decrease. In particular, each p_λ represents a point on a Pareto optimal frontier that balances diversity and expected predicted fitness; these distributions can be perturbed to increase either the entropy or the expected fitness, but not both. The entire Pareto frontier could be traced out by calculating the expected predicted fitness and entropy of p_λ for every possible setting of λ.

Of course, the library represented by p_λ is an entirely abstract quantity and cannot be constructed by any experimental technique. Next, we will describe how use the maximum entropy distribution of Equation 5.11 to build specified and degenerate libraries.

Specified libraries

Library construction techniques have advanced such that it is now possible to order a list of thousands of specific oligonucleotide sequences that one wishes to observe in a library and receive a library containing nearly all of those sequences and few additional variants. It is conceptually straightforward to build such a list that approximates the maximum entropy library of Equation 5.11 (though may be practically difficult). In particular, to specify a list of N sequences, we can draw N samples from $p_\lambda(\mathbf{x})$ with, e.g., a Markov Chain Monte Carlo (MCMC) sampling algorithm with $p_\lambda(\mathbf{x})$ as its stationary distribution. If we allow the Markov Chain to equilibrate, then a set of N samples from the chain represent a particle-based approximation to $p_\lambda(\mathbf{x})$ and thus will approximately respect the Pareto optimal property of the maximum entropy library.

Degenerate libraries

The main drawback of specified libraries is their cost: currently about $1 per specified sequence for specified AAV insertion sequence libraries . A cheaper option when less precision is required is a degenerate library, where one specifies the marginal probabilities of each nucleotide (or amino acid) at each position. A degenerate library is a generalization of a degenerate codon library, where one can only specify probabilities according to degenerate nucleotides like 'N' (where $Pr(\text{A}) = Pr(\text{T}) = Pr(\text{C}) = Pr(\text{G}) = 0.25$) or 'K' (where $Pr(\text{T}) = Pr(\text{G}) = 0.5$ and $Pr(\text{A}) = Pr(\text{C}) = 0$) and for which a number of design algorithms exist [80, 162]. The probability mass of a distribution representing a degenerate library of sequences of length L and alphabet size K (i.e., $K = 4$ for nucleotide libraries and $K = 20$ for amino acid libraries) is given by:

$$q_{\boldsymbol{\phi}}(\mathbf{x}) = \prod_{j=1}^{L} \sum_{k=1}^{K} q_{\boldsymbol{\phi}_j}(x^j = k)\delta_k(x^j) \tag{5.12}$$

where $\boldsymbol{\phi} \in \mathbb{R}^{L \times K}$ is a matrix of distribution parameters, $\boldsymbol{\phi}_j$ is the j^{th} row of $\boldsymbol{\phi}$, $\delta_k(x^j) = 1$ if $x_j = k$ and zero otherwise, and

$$q_{\boldsymbol{\phi}_j}(x^j = k) = \frac{e^{\phi_{jk}}}{\sum_{l=1}^{K} e^{\phi_{jl}}}. \tag{5.13}$$

Of course, for an arbitrary predictive model, the maximum entropy distribution of (5.11) will generally not have the form of Equation (5.13). In order to apply the maximum entropy principle to the design of degenerate libraries, we will take a variational approach and find the degenerate library that is the best approximation to the maximum entropy library, where the quality of the approximation is measured by a KL divergence. In particular, we solve for optimal the parameters of the degenerate library by minimizing the KL divergence between $q_{\boldsymbol{\phi}}$ and p_λ:

$$\boldsymbol{\phi}_\lambda = \underset{\boldsymbol{\phi}}{\operatorname{argmin}}\, D_{KL}[q_{\boldsymbol{\phi}}||p_\lambda] \tag{5.14}$$

$$= \underset{\boldsymbol{\phi}}{\operatorname{argmax}}\, \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{x})}[f(\mathbf{x})] + \lambda H[q_{\boldsymbol{\phi}}] \tag{5.15}$$

$$= \underset{\boldsymbol{\phi}}{\operatorname{argmax}}\, F(\boldsymbol{\phi}) \tag{5.16}$$

where we define $F(\boldsymbol{\phi}) \coloneqq \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{x})}[f(\mathbf{x})] + \lambda H[q_{\boldsymbol{\phi}}]$. Interestingly, despite being derived from a maximum entropy principle, this objective represents the usual form of a multi-objective optimization, where λ now controls the balance between the two objectives [182] and the objectives represent the expected predicted fitness and diversity of the degenerate library, respectively. By solving this objective for a variety of settings of λ, our solutions will trace out a Pareto optimal frontier that is distinct from that traced out by the maximum entropy distributions of Equation 5.11 for the same settings of λ. At each point on the frontier, a perturbation to the solution cannot simultaneously improve both objectives. Only globally optimal solutions to the objective lie on the Pareto frontier; however since our objective is a non-convex function of the library parameters, there is no optimization algorithm that can guarantee that we reach the global optimum in finite time. However, the Stochastic Gradient Descent (SGD) algorithm has been shown to consistently find optimal or near-optimal solutions to a variety of non-convex problems, particularly in machine learning [114]. Here we use a variant of SGD based on the score function estimator [96] to solve our objective, and trace out a near-optimal frontier. The score function estimator (sometimes called the 'log derivative trick') is to used to estimate intractable gradients of the form $\nabla_{\boldsymbol{\theta}}\mathbb{E}_{p_{\boldsymbol{\theta}}(x)}[f(x)]$, where $p_{\boldsymbol{\theta}}(x)$ is a probability density parametrized by $\boldsymbol{\theta}$, and $f(x)$ is an arbitrary function. It does so using the equality $\nabla_{\boldsymbol{\theta}}\mathbb{E}_{p_{\boldsymbol{\theta}}(x)}[f(x)] = \mathbb{E}_{p_{\boldsymbol{\theta}}(x)}[f(x)\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(x)]$, which enables one to estimate the RHS using, e.g., a Monte Carlo approximation.

In order to solve the objective in Equation (5.14) we randomly initialize a parameter matrix, $\boldsymbol{\phi}^{(0)}$ with independent normal samples and then update the parameters with

$$\boldsymbol{\phi}^{(t)} = \boldsymbol{\phi}^{(t-1)} + \alpha \nabla_{\boldsymbol{\phi}} F(\boldsymbol{\phi}^{(t-1)}) \tag{5.17}$$

for $t = 1, ..., T$, where we define $F(\boldsymbol{\phi}^{(t-1)}) \coloneqq \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{x})}[f(\mathbf{x})] + \lambda H[q_{\boldsymbol{\phi}}]$ to be the objective function in Equation (5.14). After T iterations, we assume we have reached a near-optimal solution (i.e., $\boldsymbol{\phi}^{(T)}$ can be used as an approximation for $\boldsymbol{\phi}_\lambda$). As we show in Appendix B, the components of the gradient of the objective function with respect to the degenerate

distribution parameters are given by:

$$\begin{aligned}\frac{\partial}{\partial \phi_{jk}} F(\boldsymbol{\phi}) &= \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{x})}\left[w(\mathbf{x})\frac{\partial}{\partial \phi_{jk}}\log q_{\boldsymbol{\phi}_j}(x^j)\right] \\ &= \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{x})}\left[w(\mathbf{x})\left(\delta_k(x^j) - q_{\boldsymbol{\phi}_j}(k)\right)\right] \end{aligned} \tag{5.18}$$

where we define the weights

$$w(\mathbf{x}) \coloneqq f(\mathbf{x}) - \lambda(1 + \log q_{\boldsymbol{\phi}}(\mathbf{x})). \tag{5.19}$$

The expectation in Equation (5.18) cannot be solved exactly, so we use a Monte Carlo approximation:

$$\frac{\partial}{\partial \phi_{jk}} F(\boldsymbol{\phi}) \approx \frac{1}{M}\sum_{i=1}^{M} w(\mathbf{x}_i)\left(\delta_k(x_i^j) - q_{\boldsymbol{\phi}_j}(k)\right) \tag{5.20}$$

with $\mathbf{x}_i \sim q_{\boldsymbol{\phi}}(\mathbf{x})$. In this scheme, the approximate gradients of $F(\boldsymbol{\phi})$ are random variables, which induces stochasticity in the gradient descent algorithm of Equation (5.13). This stochasticity helps the algorithm to escape local optima and achieve near-optimal solutions in a non-convex optimization problem such as that of (5.14).

Application to AAV5 library design

We applied this maximum entropy framework to design libraries of 7-mer insertion sequences to the AAV5 capsid using the (NN, 100) predictive model of fitness described in Section 5.4. We particularly focused on designing degenerate libraries of the 21 nucleotides corresponding to the 7 amino acid insertion. Figure 5.2 demonstrates the results of $2,238$ degenerate library optimizations for these 21 nucleotides with $\alpha = 0.01$, $T = 2000$, and $M = 1000$ and a range of settings of λ. Each point in Figure 5.2a represents a library resulting from one of these optimizations (i.e., a $q_{\boldsymbol{\phi}_\lambda}$). In order to assess the extent to which these libraries trade-off diversity and predicted fitness we compared two quantities corresponding to each library: the mean predicted enrichment (i.e., fitness) of amino acid sequences sampled from the library and the expected hamming distance between any two sequences sampled from the library, which we call the Expected Pairwise Distance (EPD). The EPD is an easily-calculable measure of diversity whose numerical values carry more intuition than entropy. As we show in Appendix B, the EPD of a degenerate library can be calculated exactly as

$$EPD(\boldsymbol{\phi}) = L - \sum_{j=1}^{7}\sum_{k=1}^{21}(\tilde{q}_{\tilde{\boldsymbol{\phi}}_j}(k))^2 \tag{5.21}$$

where $\tilde{q}_{\tilde{\boldsymbol{\phi}}}$ are the amino acid probabilities at 7 positions corresponding to the nucleotide probabilities for 21 positions (these can be calculated easily by appropriately summing over the probabilities of codons). We can see qualitatively how EPD correlates with diversity by

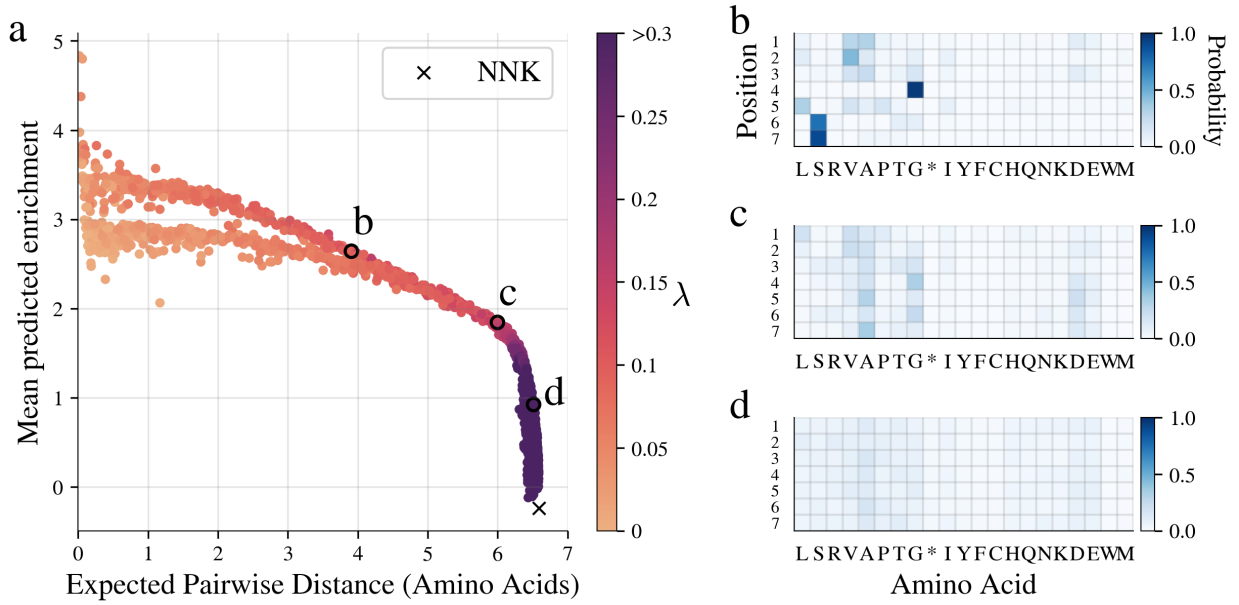


Figure 5.2: Results of maximum entropy degenerate library design for AAV5 insertion sequences. (a) Trade-off curve between mean predicted enrichment and diversity (as measured by Expected Pairwise Distance between pairs of sequences sampled from the library). Each scatter point represents a result of a degenerate library optimization in nucleotide space. (b)(c)(d) Probability mass of each amino acid at each position for the three libraries highlighted in (a).

looking at the probabilities of each amino acid at each position for the designed libraries; three examples are shown in Figures 5.2b, 5.2c, and 5.2d. As we can see, as the EPD increases, the probability mass is spread out over more sequences.

To assess the extent to which libraries that lie on the Pareto frontier balancing diversity and expected fitness can be experimentally realized, we ultimately chose to order Libraries (c) and (d) in Figure 5.2 for further experimental characterization. The results of this characterization will be presented in a future publication.

Although we chose to use degenerate libraries for future experiments, we also computationally compared these to specified libraries built using the maximum entropy technique described above. In particular we used the Metropolis-Hastings algorithm [71] to sample $N = 10,000$ specified sequences from the maximum entropy distribution of Equation 5.11 for 404 different settings of λ. Figure 5.3 compares the fitness-diversity trade-off curves of these Maximum Entropy specified libraries and the Maximum Entropy degenerate libraries described above; the points in orange represent the 404 specified libraries while the points in blue represent the same 2,238 degenerate libraries shown in Figure 5.2a. We can see that specified library construction allows one to build a library with higher expected predicted

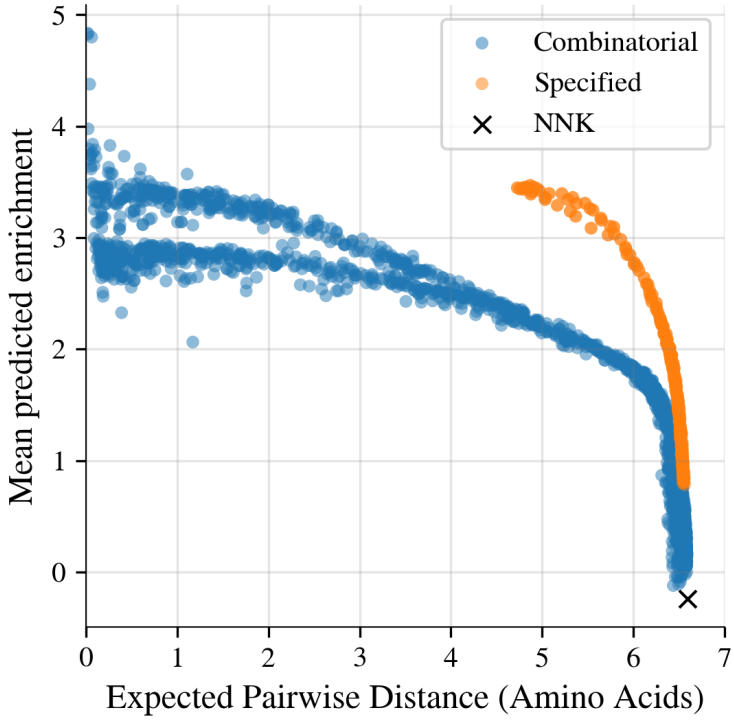


Figure 5.3: Comparison of maximum entropy degenerate and specified insertion sequence libraries. Each point represents a library, with Mean Predicted Enrichment and Expected Pairwise Distance values calculated as in Figure 5.2a.

fitness at the same level of diversity of degenerate libraries. These results suggest that as specified libraries became cheaper to build, they should be the primary choice of researchers wishing to perform data-driven design of sequence libraries.

5.6 Conclusion

In this Chapter we have described an end-to-end experimental and computational pipeline for data-driven design of a library insertion sequences to the AAV capsid, with the aim of improving the packaging ability of the resulting capsids. In order to do so, we have presented solutions for solving the forward problem of predicting the enrichment scores of sequences, and the inverse problem of designing sequence libraries that optimally balance predicted fitness and diversity.

Our solution to the inverse problem is based on a maximum entropy formalism that is broadly applicable to a variety of library construction techniques and predictive models

of fitness. One of the most interesting aspects of this formulation is its connections to the Estimation of Distribution algorithms described in Chapters 2 and 3. In particular, an EDA-style objective function appears in both the maximum entropy objective of Equation 5.10 and the variational objective for degenerate libraries of Equation 5.14. In these cases the "search model" is a probability distribution representing a sequence library. The objective for degenerate libraries, Equation 5.14, is particularly interesting because it represents an entropy-regularized EDA, which is closely related to the so-called 'rank-μ' updates used in the EDA algorithm CMA-ES to prevent convergence to a local optima by increasing the entropy of the search model. Connections such as these may open up a range of possibilities for future exploration in sequence library design.

Chapter 6

Concluding remarks

In this dissertation, we have discussed a range of problems related to the application of machine learning to biological sequence design. Although we have explored a wide breadth of topics, ranging from a theoretical study of the forward problem in Chapter 4 to an experimentally-realizable solution to an inverse problem in Chapter 5, these explorations represent only a small slice of the rapidly growing field that is data-drive sequence engineering. The field will only continue to grow in the coming years as new experimental technologies are introduced that can probe the fitness of sequences with both increased throughput and precision. As this occurs, the field will have to wrestle with a number of technical and philosophical challenges, some of which were already apparent in this dissertation.

One important challenge for future data-driven sequence design pursuits is determining the appropriate role of top-down engineering approaches versus bottom-up approaches based on biological and physical knowledge. In the Introduction we discussed top-down and bottom-up approaches in the context of building forward models of fitness functions; however, this classification can also be applied to inverse problems. For instance, a black-box generative model from which one can sample nucleotide sequences that are expected to have high fitness is an example of a top-down approach, while a method that incorporates knowledge about, say, GC content into designing those nucleotide sequences is an example of a bottom-up approach. The tension between the bottom-up and top-down paradigms is apparent in some of the work in this dissertation. In particular, in Chapter 5 we used a bottom-up approach to develop a loss function for training forward models that resulted in more accurate models than a standard loss function (Figure 5.1b); however, the neural network forward model with an architecture chosen based on top-down machine learning principles outperformed simpler linear models with biological interpretation (Figure 5.1b). This example not only demonstrates the tension between top-down and bottom-up approaches, but also shows how the approaches can complement one another, since ultimately the most powerful model used a top-down architecture with a bottom-up loss function. Future modeling techniques for sequence design should strive to find a similar balance that allows for both powerful predictive (or generative) performance and meaningful biological interpretation.

It will additionally be important for the field to embrace the important role that theoret-

ical explorations can play in guiding future computational, and possibly even experimental, approaches to sequence design. In machine learning broadly, and in its specific applications to sequence design, our ability to build powerful models has far outpaced our theoretical understanding of those models. Machine learning techniques have produced some remarkable achievements in sequence design; for example, Biswas et al. [22] recently engineered novel variants of the Green Fluorescent Protein and TEM-1 β-lactamase protein using fitness measurements for fewer than 100 variants, and a number of groups have used models originally designed for natural language processing to predict structural contacts in proteins with remarkable accuracy [143, 139, 107]. For the most part, there is not an adequate explanation for these phenomena (although progress has recently been made in understanding the latter example [21]), which makes it difficult to build further techniques on top of this work. Although theoretical results in simplified systems may not be as powerful for achieving particular engineering goals as advanced machine learning methods, they are often able to make predictions in novel scenarios, and thus can be used to inform future techniques. For instance, it would be straightforward to use the data of Poelwijk et al. [136] to construct a top-down forward that predicts the fitness of variants of the mTagBFP2, which cannot be used to understand any other fitness functions. In contrast, in Chapter 4 we use this data in conjunction with a robust theory to predict (i) how higher-order epistasis arises in protein fitness functions in general and (ii) how many fitness measurements are required to learn protein fitness functions. This type of exploration can be used to inform both future modeling approaches, as well as experimental fitness probes, and thus demonstrates the potential power of complementing advanced black-box engineering approaches with foundational theoretical knowledge.

Biological sequence engineering in the near future will be driven primarily by experimental techniques, with computation playing a supporting role. Thus the most important challenge for the field of data-driven sequence engineering is to develop symbiotic relationships with experimental groups that foster improvements to both computational and experimental approaches. These relationships allow computational researchers to learn about and come to appreciate the complex biology that underlies experiments, and incorporate this knowledge into models. Additionally, collaboration allows experimentalists to understand what types of data are most useful to collect for modeling purposes, and cater their techniques to these needs. Such a collaboration resulted in the work presented in Chapter 5, where we worked closely with the experimental group of Professor David V. Schaffer to engineer AAV capsid sequences. Further, our work on understanding the sample complexity of fitness function regression in Chapter 4 was motivated by a desire to inform experimentalists on how to collect data such that it can be used to build a model of a fitness function. We hope the work presented in this dissertation will provide motivation for the both computational and experimental researchers to develop fruitful collaborations, which will be the key to achieving the ultimate goal of generalizable biological sequence engineering.

Bibliography

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015.

[2] Atish Agarwala and Daniel S. Fisher. "Adaptive walks on high-dimensional fitness landscapes and seascapes with distance-dependent statistics". *Theor. Popul. Biol.* 130 (2019), pp. 13–49.

[3] Amirali Aghazadeh, Adam Y. Lin, Mona A. Sheikh, Allen L. Chen, Lisa M. Atkins, Coreen L. Johnson, Joseph F. Petrosino, Rebekah A. Drezek, and Richard G. Baraniuk. "Universal microbial diagnostics using random DNA probes". *Science Advances* 2.9 (2016).

[4] Amirali Aghazadeh, Hunter Nisonoff, Orhan Ocal, Yijie Huang, O. Ozan Koyluoglu, Jennifer Listgarten, and Kannan Ramchandran. "Sparse Epistatic Regularization of Deep Neural Networks for Inferring Fitness Functions". *bioRxiv* (2020).

[5] Amirali Aghazadeh, Orhan Ocal, and Kannan Ramchandran. "CRISPRLand: Interpretable large-scale inference of DNA repair landscape based on a spectral approach". *Bioinformatics* 36.1 (2020), pp. i560–i568.

[6] Takuyo Aita, Yuuki Hayashi, Hitoshi Toyota, Yuzuru Husimi, Itaru Urabe, and Tetsuya Yomo. "Extracting characteristic properties of fitness landscape from in vitro molecular evolution: A case study on infectivity of fd phage to E.coli". *J. Theor. Biol.* 246.3 (2007), pp. 538–550.

[7] Youhei Akimoto, Yuichi Nagata, Isao Ono, and Shigenobu Kobayashi. "Theoretical foundation for CMA-ES from information geometry perspective". *Algorithmica* 64.4 (2012), pp. 698–716.

[8] Dave W. Anderson, Alesia N. McKeown, and Joseph W. Thornton. "Intermolecular epistasis shaped the function and evolution of an ancient transcription factor and its DNA binding sites". *eLife* 4 (2015).

[9] Christof Angermueller, David Dohan, David Belanger, Ramya Deshpande, Kevin Murphy, and Lucy Colwell. "Model-based reinforcement learning for biological sequence design". In: *Proceedings of the International Conference on Learning Representations*. 2020.

[10] Ivan Anishchenko, Sergey Ovchinnikov, Hetunandan Kamisetty, and David Baker. "Origins of coevolution between residues distant in protein 3D structures". *Proc. Natl. Acad. Sci. U.S.A.* 114.34 (2017), pp. 9122–9127.

[11] Ruben Armananzas, Yvan Saeys, Inaki Inza, Miguel Garcia-Torres, Concha Bielza, Yves van de Peer, and Pedro Larranaga. "Peakbin Selection in Mass Spectrometry Data Using a Consensus Approach with Estimation of Distribution Algorithms". *IEEE/ACM Trans. Comput. Biol. Bioinformatics* 8.3 (2011), pp. 760–774.

[12] Frances H. Arnold. "Design by Directed Evolution". *Acc. Chem. Res.* 31.3 (1998), pp. 125–131.

[13] Gurinder S. Atwal and Justin B. Kinney. "Learning Quantitative Sequence–Function Relationships from Massively Parallel Experiments". *J. Stat. Phys.* 162.5 (2016), pp. 1203–1243.

[14] Sivaraman Balakrishnan, Martin J. Wainwright, and Bin Yu. "Statistical guarantees for the EM algorithm: From population to sample-based analysis". *Ann. Statist.* 45.1 (2017), pp. 77–120.

[15] Aditya Ballal, Caroline Laurendon, Melissa Salmon, Maria Vardakou, Jitender Cheema, Marianne Defernez, Paul E O'Maille, and Alexandre V Morozov. "Sparse Epistatic Patterns in the Evolution of Terpene Synthases". *Mol. Biol. Evol.* 37.7 (2020), pp. 1907–1924.

[16] Claudia Bank, Sebastian Matuszewski, Ryan T. Hietpas, and Jeffrey D. Jensen. "On the (un)predictability of a large intragenic fitness landscape". *Proc. Natl. Acad. Sci. U.S.A.* 113.49 (2016), pp. 14085–14090.

[17] Melissa A. Bartel, John R. Weinstein, and David V. Schaffer. "Directed evolution of novel adeno-associated viruses for therapeutic gene delivery". *Gene Ther.* 19.6 (2012), pp. 694–700.

[18] Leonard E. Baum and Ted Petrie. "Statistical Inference for Probabilistic Functions of Finite State Markov Chains". *Ann. Math. Statist.* 37.6 (1966), pp. 1554–1563.

[19] Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. "Automatic Differentiation in Machine Learning: a Survey". *J. Mach. Learn. Res.* 18.153 (2018), pp. 1–43.

[20] Matthew J. Beal. "Variational Algorithms for Approximate Bayesian Inference". PhD thesis. 1998.

[21] Nicholas Bhattacharya, Neil Thomas, Roshan Rao, Justas Dauparas, Peter K. Koo, David Baker, Yun S. Song, and Sergey Ovchinnikov. "Single Layers of Attention Suffice to Predict Protein Contacts". *bioRxiv* (2020).

[22] Surojit Biswas, Grigory Khimulya, Ethan C. Alley, Kevin M. Esvelt, and George M. Church. "Low-N protein engineering with data-efficient deep learning". *Nat. Methods* 18.4 (2021), pp. 389–396.

[23] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. "Variational Inference: A Review for Statisticians". *J. Am. Stat. Assoc.* 112.518 (2017), pp. 859–877.

[24] Daniel N. Bolon and Stephen L. Mayo. "Enzyme-like proteins by computational design". *Proc. Natl. Acad. Sci. U.S.A* 98.25 (2001), pp. 14274–14279.

[25] Sylvie Boutin, Virginie Monteilhet, Philippe Veron, Christian Leborgne, Olivier Benveniste, Marie Françoise Montus, and Carole Masurier. "Prevalence of Serum IgG and Neutralizing Factors Against Adeno-Associated Virus (AAV) Types 1, 2, 5, 6, 8, and 9 in the Healthy Population: Implications for Gene Therapy Using AAV Vectors". *Hum. Gene Ther.* 21.6 (2010), pp. 704–712.

[26] David Brookes, Akosua Busia, Clara Fannjiang, Kevin Murphy, and Jennifer Listgarten. "A View of Estimation of Distribution Algorithms through the Lens of Expectation-Maximization". In: *GECCO*. 2020, pp. 189–190.

[27] David H. Brookes and Jennifer Listgarten. "Design by adaptive sampling". In: *Workshop at Conference on Neural Information Processing Systems*. 2018.

[28] David H. Brookes, Hahnbeom Park, and Jennifer Listgarten. "Conditioning by adaptive sampling for robust design". In: *Proceedings of the International Conference of Machine Learning*. 2019, pp. 773–782.

[29] Drew H. Bryant, Ali Bashir, Sam Sinai, Nina K. Jain, Pierce J. Ogden, Patrick F. Riley, George M. Church, Lucy J. Colwell, and Eric D. Kelsic. "Deep diversification of an AAV capsid protein by machine learning". *Nat. Biotechnol.* (2021), pp. 1–6.

[30] Jeffrey Buzas and Jeffrey Dinitz. "An Analysis of NK landscapes: Interaction structure, statistical properties, and expected number of local optima". *IEEE Trans. Evol. Comput.* 18.6 (2014), pp. 807–818.

[31] Leah C. Byrne, Timothy P. Day, Meike Visel, Jennifer A. Strazzeri, Cécile Fortuny, Deniz Dalkara, William H. Merigan, David V. Schaffer, and John G. Flannery. "In vivo–directed evolution of adeno-associated virus in the primate retina". *JCI Insight* 5.10 (2020).

[32] Paulo R. A. Campos, Christoph Adami, and Claus O. Wilke. "Optimal adaptive performance and delocalization in NK fitness landscapes". *Phys. A Stat. Mech. its Appl.* 304.3-4 (2002), pp. 495–506.

[33] Emmanuel J. Candes and Yaniv Plan. "A Probabilistic and RIPless Theory of Compressed Sensing". *IEEE Transactions on Information Theory* 57.11 (2011), pp. 7235–7254.

[34] Emmanuel J. Candes and Michael B. Wakin. "An Introduction To Compressive Sampling". *IEEE Signal Process. Mag.* 25.2 (2008), pp. 21–30.

[35] Kauê Cardoso. "Principal eigenvector of the signless Laplacian matrix". *Comput. Appl. Math.* 40.2 (2021), p. 50.

[36] Volkan Cevher. "Learning with Compressible Priors". In: *Advances in Neural Information Processing Systems*. Vol. 22. 2009, pp. 261–269.

[37] David Chandler. *Introduction to Modern Statistical Mechanics*. Oxford University Press, 1987.

[38] Keqin Chen and Frances H. Arnold. "Enzyme Engineering for Nonaqueous Solvents: Random Mutagenesis to Enhance Activity of Subtilisin E in Polar Organic Media". *Biotechnology (N Y)* 9.11 (1991), pp. 1073–1077.

[39] Albert Cohen, Wolfgang Dahmen, and Ronald DeVore. "Compressed sensing and best k-term approximation". *J. Am. Math. Soc.* 22 (2009), pp. 211–231.

[40] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. USA: Wiley-Interscience, 2006.

[41] Deniz Dalkara, Leah C. Byrne, Ryan R. Klimczak, Meike Visel, Lu Yin, William H. Merigan, John G. Flannery, and David V. Schaffer. "In Vivo–Directed Evolution of a New Adeno-Associated Virus for Therapeutic Outer Retinal Gene Delivery from the Vitreous". *Sci. Transl. Med.* 5.189 (2013), 189ra76–189ra76.

[42] Michael W. Davidson and Robert E. Campbell. "Engineered fluorescent proteins: innovations and applications". *Nat. Methods* 6.10 (2009), pp. 713–717.

[43] J. Arjan G.M. De Visser and Joachim Krug. "Empirical fitness landscapes and the predictability of evolution". *Nat. Rev. Genet.* 15.7 (2014), pp. 480–490.

[44] Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. "Maximum likelihood from incomplete data via the EM algorithm". *J. R. Stat. Soc* 39.1 (1977), pp. 1–38.

[45] Benjamin E. Deverman, Piers L. Pravdo, Sripriya Ravindra Simpson Bryan P .and Kumar, Ken Y. Chan, Abhik Banerjee, Wei-Li Wu, Bin Yang, Nina Huber, Sergiu P. Pasca, and Viviana Gradinaru. "Cre-dependent selection yields AAV variants for widespread gene transfer to the adult brain". *Nat. Biotechnol.* 34.2 (2016), pp. 204–209.

[46] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. "Density estimation using Real NVP". In: *Proceedings of the International Conference on Learning Representations*. 2017.

[47] D. Allan Drummond, Brent L. Iverson, George Georgiou, and Frances H. Arnold. "Why high-error-rate random mutagenesis libraries are enriched in functional and improved proteins". *J. Mol. Biol.* 350.4 (2005), pp. 806–816.

[48] Richard Durrett and Vlada Limic. "Rigorous Results for the NK Model". *Ann. Probab.* 31.4 (2003), pp. 1713–1753.

[49] Juan José Egozcue, Jan Graffelman, M Isabel Ortego, and Vera Pawlowsky-Glahn. "Some thoughts on counts in sequencing studies". *NAR Genomics and Bioinformatics* 2.4 (2020).

[50] Andrew D. Ellington and Jack W. Szostak. "In vitro selection of RNA molecules that bind specific ligands". *Nature* 346.6287 (1990), pp. 818–822.

[51] Jesse Engel, Matthew Hoffman, and Adam Roberts. "Latent Constraints: Learning to Generate Conditionally from Unconditional Generative Models". In: *Proceedings of the International Conference on Learning Representations*. 2018.

[52] Daniel Esposito, Jochen Weile, Jay Shendure, Lea M Starita, Anthony T Papenfuss, Frederick P Roth, Douglas M Fowler, and Alan F Rubin. "MaveDB: an open-source platform to distribute and interpret data from multiplexed assays of variant effect". *Genome Biol.* 20.1 (2019), p. 223.

[53] Matteo Figliuzzi, Pierre Barrat-Charlaix, and Martin Weigt. "How Pairwise Coevolutionary Models Capture the Collective Residue Variability in Proteins?" *Mol. Biol. Evol* 35.4 (2018), pp. 1018–1027.

[54] US Food and Drug Administration. "FDA approves innovative gene therapy to treat pediatric patients with spinal muscular atrophy, a rare disease and leading genetic cause of infant mortality" (2019).

[55] US Food and Drug Administration. "FDA approves novel gene therapy to treat patients with a rare form of inherited vision loss" (2017).

[56] Douglas M. Fowler and Stanley Fields. *Deep mutational scanning: A new style of protein science*. 2014.

[57] Richard Fox. "Directed molecular evolution by machine learning and the influence of nonlinear interactions". *J. Theor. Biol.* 234.2 (2005), pp. 187–199.

[58] Marc Garcia-Borràs, Kendall N. Houk, and Gonzalo Jiménez-Osés. "Computational Design of Protein Function". In: *Computational Tools for Chemical Biology*. The Royal Society of Chemistry, 2018, pp. 87–107.

[59] Daniel G. Gibson, John I. Glass, Carole Lartigue, Vladimir N. Noskov, Ray-Yuan Chuang, Mikkel A. Algire, Gwynedd A. Benders, Michael G. Montague, Li Ma, Monzia M. Moodie, Chuck Merryman, Sanjay Vashee, Radha Krishnakumar, Nacyra Assad-Garcia, Cynthia Andrews-Pfannkoch, Evgeniya A. Denisova, Lei Young, Zhi-Qing Qi, Thomas H. Segall-Shapiro, Christopher H. Calvey, Prashanth P. Parmar, Clyde A. Hutchison, Hamilton O. Smith, and J. Craig Venter. "Creation of a Bacterial Cell Controlled by a Chemically Synthesized Genome". *Science* 329.5987 (2010), pp. 52–56.

[60] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. 1st. USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

[61] Rafael Gómez-Bombarelli, Jennifer N. Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D. Hirzel, Ryan P. Adams, and Alán Aspuru-Guzik. "Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules". *ACS Cent. Sci.* 4.2 (2018), pp. 268–276.

[62] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems*. 2014, pp. 2672–2680.

[63] Geoffrey R. Grimmett and David R. Stirzaker. *Probability and random processes*. Vol. 80. 391. Oxford university press, 2001.

[64] Anvita Gupta and James Zou. "Feedback GAN for DNA optimizes protein functions". *Nat. Mach. Intell.* 1.2 (2019), pp. 105–111.

[65] Richard Hammack, Wilfried Imrich, and Sandi Klavzar. *Handbook of Product Graphs, Second Edition*. 2nd. USA: CRC Press, Inc., 2011.

[66] Nikolaus Hansen. "The CMA Evolution Strategy: A Tutorial". *arXiv* 1412.6980 (2006).

[67] Nikolaus Hansen, Anne Auger, Olaf Mersmann, Tea Tusar, and Dimo Brockhoff. *COCO: A Platform for Comparing Continuous Optimizers in a Black-Box Setting*. Tech. rep. 2016.

[68] Nikolaus Hansen, Sibylle D. Müller, and Petros Koumoutsakos. "Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES)". *Evol. Comput.* 11.1 (2003), pp. 1–18.

[69] Nikolaus Hansen and Andreas Ostermeier. "Completely Derandomized Self-Adaptation in Evolution Strategies". *Evol. Comput.* 9.2 (2001), pp. 159–195.

[70] Trevor Hastie, Robert Tibshirani, and Martin Wainwright. *Statistical Learning with Sparsity: The Lasso and Generalizations*. Chapman & Hall/CRC, 2015.

[71] Wilfred K. Hastings. "Monte Carlo Sampling Methods Using Markov Chains and Their Applications". *Biometrika* 57.1 (1970), pp. 97–109.

[72] Yuuki Hayashi, Takuyo Aita, Hitoshi Toyota, Yuzuru Husimi, Itaru Urabe, and Tetsuya Yomo. "Experimental Rugged Fitness Landscape in Protein Sequence Space". *PLoS One* 1.1 (2006), e96.

[73] Robert B. Heckendorn and Darrell Whitley. "A Walsh Analysis of NK-Landscapes". In: *Proceedings of the Seventh International Conference on Genetic Algorithms*. Morgan Kaufmann, 1997, pp. 41–48.

[74] Trevor Hinkley, João Martins, Colombe Chappey, Mojgan Haddad, Eric Stawiski, Jeannette M Whitcomb, Christos J Petropoulos, and Sebastian Bonhoeffer. "A systems analysis of mutational effects in HIV-1 protease and reverse transcriptase". *Nat. Genet.* 43.5 (2011), pp. 487–489.

[75] Thomas Hofmann. "Unsupervised Learning by Probabilistic Latent Semantic Analysis". *Machine Learning* 42.1 (2001), pp. 177–196.

[76] Thomas A. Hopf, John B. Ingraham, Frank J. Poelwijk, Charlotta P I Schärfe, Michael Springer, Chris Sander, and Debora S Marks. "Mutation effects predicted from sequence co-variation". *Nat. Biotechnol.* 35.2 (2017), pp. 128–135.

[77] Chloe Hsu, Hunter Nisonoff, Clara Fannjiang, and Jennifer Listgarten. "Combining evolutionary and assay-labelled data for protein fitness prediction". *bioRxiv* (2021).

[78] Jiaqiao Hu and Ping Hu. "Annealing adaptive search, cross-entropy, and stochastic approximation in global optimization". *Nav. Res. Logist.* 58.5 (2011), pp. 457–477.

[79] Fumitaka Inoue and Nadav Ahituv. "Decoding enhancers using massively parallel reporter assays". *Genomics* 106.3 (2015). Recent advances in functional assays of transcriptional enhancers, pp. 159–164.

[80] Timothy M. Jacobs, Hayretin Yumerefendi, Brian Kuhlman, and Andrew Leaver-Fay. "SwiftLib: Rapid degenerate-codon-library optimization through dynamic programming". *Nucleic Acids Res.* 43.5 (2015), p. 34.

[81] Eric Jang, Shane Gu, and Ben Poole. "Categorical Reparameterization with Gumbel-Softmax". In: *Proceedings of the International Conference on Learning Representations*. 2017.

[82] Edwin T. Jaynes. "Information Theory and Statistical Mechanics". *Phys. Rev.* 106 (4 1957), pp. 620–630.

[83] José I. Jiménez, Ramon Xulvi-Brunet, Gregory W. Campbell, Rebecca Turk-MacLeod, and Irene A. Chen. "Comprehensive experimental fitness landscape and evolutionary network for small RNA". *Proc. Natl. Acad. Sci. U.S.A.* 110.37 (2013), pp. 14984–14989.

[84] D. Katz, J. Baptista, S. P. Azen, and M. C. Pike. "Obtaining Confidence Intervals for the Risk Ratio in Cohort Studies". *Biometrics* 34.3 (1978), pp. 469–474.

[85] Stuart Kauffman and Simon Levin. "Towards a general theory of adaptive walks on rugged landscapes". *J. Theor. Biol.* 128.1 (1987), pp. 11–45.

[86] Stuart A. Kauffman and Edward D. Weinberger. "The NK model of rugged fitness landscapes and its application to maturation of the immune response". *J. Theor. Biol.* 141.2 (1989), pp. 211–245.

[87] Ronald W. Keener. *Theoretical Statistics: Topics for a Core Course*. Springer Texts in Statistics. Springer New York, 2010.

[88] Andrea Kern, Kristen Schmidt, Christopher Leder, Oliver J. Müller, Christiane E. Wobus, K. Bettinger, Claus W. Von der Lieth, Jason A. King, and Jurgen A. Kleinschmidt. "Identification of a Heparin-Binding Motif on Adeno-Associated Virus Type 2 Capsids". *J. Virol.* 77.20 (2003), pp. 11072–11081.

[89] Stefan Kern, Sibylle D. Müller, Nikolaus Hansen, Dirk Büche, Jiri Ocenasek, and Petros Koumoutsakos. "Learning probability distributions in continuous evolutionary algorithms – a comparative review". *Natural Computing* 3.1 (2004), pp. 77–112.

[90] Nathan Killoran, Leo J. Lee, Andrew Delong, David Duvenaud, and Brendan J. Frey. "Generating and designing DNA with deep generative models". *arXiv* 1712.06148 (2017).

[91] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *Proceedings of the International Conference on Learning Representations*. 2015.

[92] Diederik P. Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: *Proceedings of the International Conference on Learning Representations*. 2014.

[93] John F. C. Kingman. "A Simple Model for the Balance between Selection and Mutation". *J. Appl. Probab.* 15.1 (1978), pp. 1–12.

[94] Justin B. Kinney and David M. McCandlish. "Massively Parallel Assays and Quantitative Sequence–Function Relationships". *Annu. Rev. Genomics Hum. Genet.* 20.1 (2019), pp. 99–127.

[95] Gert Kiss, Nihan Çelebi-Ölçüm, Rocco Moretti, David Baker, and K. N. Houk. "Computational Enzyme Design". *Angew. Chem. Int. Ed.* 52.22 (2013), pp. 5700–5725.

[96] Jack P.C. Kleijnen and Reuven Y. Rubinstein. *Optimization and sensitivity analysis of computer simulation models by the score function method*. Tech. rep. 3. 1996, pp. 413–427.

[97] Melissa A. Kotterman and David V. Schaffer. "Engineering adeno-associated viruses for clinical gene therapy". *Nat. Rev. Genet.* 15.7 (2014), pp. 445–451.

[98] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. "Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles". In: *Advances in Neural Information Processing Systems*. 2017, pp. 6402–6413.

[99] Pedro Larraanaga and Jose A. Lozano. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. USA: Kluwer Academic Publishers, 2001.

[100] Matthew R. Lee, Jerry Tsai, David Baker, and Peter A. Kollman. "Molecular dynamics in the endgame of protein structure prediction". *J. Mol. Biol.* 313.2 (2001), pp. 417–430.

[101] Michael Levitt and Shneior Lifson. "Refinement of protein conformations using a macromolecular energy minimization procedure". *J. Mol. Biol.* 46.2 (1969), pp. 269–279.

[102] Junwei Li, Maria T. Arévalo, and Mingtao Zeng. "Engineering influenza viral vectors". eng. *Bioengineered* 4.1 (2013), pp. 9–14.

[103] Kezhi Li and Shuang Cong. "State of the art and prospects of structured sensing matrices in compressed sensing". *Front. Comput. Sci.* 9.5 (2015), pp. 665–677.

[104] Johannes Linder, Nicholas Bogard, Alexander B. Rosenberg, and Georg Seelig. "A Generative Neural Network for Maximizing Fitness and Diversity of Synthetic DNA and Protein Sequences". *Cell Syst.* 11.1 (2020), 49–62.e16.

[105] Emilie Macé, Romain Caplette, Olivier Marre, Abhishek Sengupta, Antoine Chaffiol, Peggy Barbe, Mélissa Desrosiers, Ernst Bamberg, Jose-Alain Sahel, Serge Picaud, Jens Duebel, and Deniz Dalkara. "Targeting Channelrhodopsin-2 to ON-bipolar Cells With Vitreally Administered AAV Restores ON and OFF Visual Responses in Blind Mice". *Molecular Therapy* 23.1 (2015), pp. 7–16.

[106] David J. C. MacKay. *Information theory, inference, and learning algorithms*. Cambridge University Press, 2003.

[107] Ali Madani, Bryan McCann, Nikhil Naik, Nitish Shirish Keskar, Namrata Anand, Raphael R. Eguchi, Po-Ssu Huang, and Richard Socher. "ProGen: Language Modeling for Protein Generation". *bioRxiv* (2020).

[108] Stephan Mandt, James McInerney, Farhan Abrol, Rajesh Ranganath, and David Blei. "Variational Tempering". In: *Proceedings of the International Conference on Artificial Intelligence and Statistics*. 2016, pp. 704–712.

[109] Adam C. Mater, Mahakaran Sandhu, and Colin J. Jackson. "The NK Landscape as a Versatile Benchmark for Machine Learning Driven Protein Engineering". *bioRxiv* (2020).

[110] Sebastian Matuszewski, Marcel E. Hildebrandt, Ana Hermina Ghenu, Jeffrey D. Jensen, and Claudia Bank. "A statistical guide to the design of deep mutational scanning experiments". *Genetics* 204.1 (2016), pp. 77–87.

[111] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. "Human-level control through deep reinforcement learning". *Nature* 518.7540 (2015), pp. 529–533.

[112] Faruck Morcos, Andrea Pagnani, Bryan Lunt, Arianna Bertolino, Debora S. Marks, Chris Sander, Riccardo Zecchina, José N. Onuchic, Terence Hwa, and Martin Weigt. "Direct-coupling analysis of residue coevolution captures native contacts across many protein families". *Proc. Natl. Acad. Sci. U.S.A* 108.49 (2011), E1293–E1301.

[113] Heinz Mühlenbein and Gerhard Paaß. "From recombination of genes to the estimation of distributions I. Binary parameters". In: *Parallel Problem Solving from Nature — PPSN IV*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 178–187.

[114] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[115] Radford M. Neal and Geoffrey E. Hinton. "Learning in Graphical Models". In: ed. by Michael I. Jordan. Cambridge, MA, USA: MIT Press, 1999. Chap. A View of the EM Algorithm That Justifies Incremental, Sparse, and Other Variants, pp. 355–368.

[116] Johannes Neidhart, Ivan G. Szendro, and Joachim Krug. "Exact results for amplitude spectra of fitness landscapes". *J. Theor. Biol.* 332 (2013), pp. 218–227.

[117] Anh Nguyen, Jeff Clune, Yoshua Bengio, Alexey Dosovitskiy, and Jason Yosinski. "Plug & Play Generative Networks: Conditional Iterative Generation of Images in Latent Space". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017.

[118] Anh Nguyen, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, and Jeff Clune. "Synthesizing the preferred inputs for neurons in neural networks via deep generator networks". In: *Advances in Neural Information Processing Systems*. Vol. 29. Curran Associates, Inc., 2016.

[119] Anh Mai Nguyen, Jason Yosinski, and Jeff Clune. "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 427–436.

[120] Stefan Nowak and Joachim Krug. "Analysis of adaptive walks on NK fitness landscapes with different interaction schemes". *J. Stat. Mech. Theory Exp.* 2015.6 (2015), P06014.

[121] Jiri Ocenasek and Josef Schwarz. "Estimation of Distribution Algorithm for mixed continuous-discrete optimization problems". In: *Proceedings of the International Conference on Machine Learning*. 2012.

[122] Pierce J. Ogden, Eric D. Kelsic, Sam Sinai, and George M. Church. "Comprehensive AAV capsid fitness landscape reveals a viral gene and enables machine-guided design." *Science* 366.6469 (2019), pp. 1139–1143.

[123] Yann Ollivier, Ludovic Arnold, Anne Auger, and Nikolaus Hansen. "Information-geometric Optimization Algorithms: A Unifying Picture via Invariance Principles". *J. Mach. Learn. Res.* 18.1 (2017), pp. 564–628.

[124] H. Allen Orr. "The Population Genetics of Adaptation on Correlated Fitness Landscapes: The Block Model". *Evolution* 60.6 (2006), pp. 1113–1124.

[125] Jakub Otwinowski. "Biophysical inference of epistasis and the effects of mutations on protein stability and function". *Mol. Biol. Evol.* 35.10 (2018), pp. 2345–2354.

[126] Jakub Otwinowski, David M. McCandlish, and Joshua B. Plotkin. "Inferring the shape of global epistasis". *Proc. Natl. Acad. Sci. U.S.A.* 115.32 (2018), E7550–E7558.

[127] Jakub Otwinowski and Joshua B. Plotkin. "Inferring fitness landscapes by regression produces biased estimates of epistasis". *Proc. Natl. Acad. Sci. U.S.A.* 111.22 (2014).

[128] Sergey Ovchinnikov, Hetunandan Kamisetty, and David Baker. "Robust and accurate prediction of residue–residue interactions across protein interfaces using evolutionary information". *eLife* 3 (2014), e02030.

[129] Andrew S. Parker, Karl E. Griswold, and Chris Bailey-Kellogg. "Optimization of combinatorial mutagenesis". *J. Comput. Biol.* 18.11 (2011), pp. 1743–1756.

[130] Alan S. Perelson and Catherine A. Macken. "Protein evolution on partially correlated landscapes". *Proc. Natl. Acad. Sci. U.S.A* 92.21 (1995), pp. 9657–9661.

[131] Jan Peters and Stefan Schaal. "Reinforcement Learning by Reward-weighted Regression for Operational Space Control". In: *Proceedings of the International Conference on Machine Learning*. 2007, pp. 745–750.

[132] Niles A. Pierce and Erik Winfree. "Protein Design is NP-hard". *Protein Engineering, Design and Selection* 15.10 (2002), pp. 779–782.

[133] K. L. Pietersz, R. M. Martier, M. S. Baatje, J. M. Liefhebber, C. C. Brouwers, S. M. Pouw, L. Fokkert, J. Lubelski, H. Petry, G. J.M. Martens, S. J. van Deventer, P. Konstantinova, and B. Blits. "Transduction patterns in the CNS following various routes of AAV-5-mediated gene delivery". *Gene Ther.* (2020).

[134] Jason N. Pitt and Adrian R. Ferré-D'Amare. "Rapid construction of empirical RNA fitness landscapes". *Science* 330.6002 (2010), pp. 376–379.

[135] Frank J. Poelwijk, Vinod Krishna, and Rama Ranganathan. "The Context-Dependence of Mutations: A Linkage of Formalisms". *PLoS Comput. Biol.* 12.6 (2016).

[136] Frank J. Poelwijk, Michael Socolich, and Rama Ranganathan. "Learning the pattern of epistasis linking genotype and phenotype in a protein". *Nat. Commun.* 10.1 (2019).

[137] Victoria O. Pokusaeva, Dinara R. Usmanova, Ekaterina V. Putintseva, Lorena Espinar, Karen S. Sarkisyan, Alexander S. Mishin, Natalya S. Bogatyreva, Dmitry N. Ivankov, Arseniy V. Akopyan, Sergey Ya. Avvakumov, Inna S. Povolotskaya, Guillaume J. Filion, Lucas B. Carey, and Fyodor A. Kondrashov. "An experimental assay of the interactions of amino acids from orthologous sequences shaping a complex fitness landscape". *PLOS Genetics* 15.4 (2019), pp. 1–30.

[138] Alexander Radovic, Mike Williams, David Rousseau, Michael Kagan, Daniele Bonacorsi, Alexander Himmel, Adam Aurisano, Kazuhiro Terao, and Taritree Wongjirad. "Machine learning at the energy and intensity frontiers of particle physics". *Nature* 560.7716 (2018), pp. 41–48.

[139] Roshan Rao, Nicholas Bhattacharya, Neil Thomas, Yan Duan, Peter Chen, John Canny, Pieter Abbeel, and Yun Song. "Evaluating Protein Transfer Learning with TAPE". In: *Advances in Neural Information Processing Systems*. Vol. 32. 2019, pp. 9689–9701.

[140] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. "Stochastic Backpropagation and Approximate Inference in Deep Generative Models". In: *Proceedings of the International Conference on Machine Learning*. 2014, pp. 1278–1286.

[141] Benjamin Ricaud, Pierre Borgnat, Nicolas Tremblay, Paulo Gonçalves, and Pierre Vandergheynst. "Fourier could be a data scientist: From graph Fourier transform to signal processing on graphs". *Comptes Rendus Phys.* 20.5 (2019), pp. 474–488.

[142] Adam J. Riesselman, John B. Ingraham, and Debora S. Marks. "Deep generative models of genetic variation capture the effects of mutations". *Nat. Methods* 15.10 (2018), pp. 816–822.

[143] Alexander Rives, Siddharth Goyal, Joshua Meier, Demi Guo, Myle Ott, C. Lawrence Zitnick, Jerry Ma, and Rob Fergus. "Biological Structure and Function Emerge from Scaling Unsupervised Learning to 250 Million Protein Sequences". *bioRxiv* (2019).

[144] Gabriel J. Rocklin, Tamuka M. Chidyausiku, Inna Goreshnik, Alex Ford, Scott Houliston, Alexander Lemak, Lauren Carter, Rashmi Ravichandran, Vikram K. Mulligan, Aaron Chevalier, Cheryl H. Arrowsmith, and David Baker. "Global analysis of protein folding using massively parallel design, synthesis, and testing". *Science* (2017).

[145] Nathan J. Rollins, Kelly P. Brock, Frank J. Poelwijk, Michael A. Stiffler, Nicholas P. Gauthier, Chris Sander, and Debora S. Marks. "Inferring protein 3D structure from deep mutation scans". *Nat. Genet.* 51.7 (2019), pp. 1170–1176.

[146] Philip A. Romero, Andreas Krause, and Frances H. Arnold. "Navigating the protein fitness landscape with Gaussian processes". *Proc. Natl. Acad. Sci. U.S.A* 110.3 (2013), E193–E201.

[147] Daniela Röthlisberger, Olga Khersonsky, Andrew M Wollacott, Lin Jiang, Jason DeChancie, Jamie Betker, Jasmine L Gallaher, Eric A Althoff, Alexandre Zanghellini, Orly Dym, Shira Albeck, Kendall N Houk, Dan S Tawfik, and David Baker. "Kemp elimination catalysts by computational enzyme design". *Nature* 453.7192 (2008), pp. 190–195.

[148] William Rowe, Mark Platt, David C Wedge, Philip J Day, Douglas B Kell, and Joshua Knowles. "Analysis of a complete DNA-protein affinity landscape". *J. R. Soc. Interface* 7.44 (2010), pp. 397–408.

[149] Alan F. Rubin, Hannah Gelman, Nathan Lucas, Sandra M. Bajjalieh, Anthony T. Papenfuss, Terence P. Speed, and Douglas M. Fowler. "A statistical framework for analyzing deep mutational scanning data". *Genome Biol.* 18.1 (2017).

[150] Reuven Rubinstein and William Davidson. "The Cross-Entropy Method for Combinatorial and Continuous Optimization". *Methodol. Comput. Appl. Probab.* 1 (1999), pp. 127–190.

[151] Reuven Y. Rubinstein. "Optimization of computer simulation models with rare events". *Eur. J. Oper. Res.* 99.1 (1997), pp. 89–112.

[152] Zachary R. Sailer and Michael J. Harms. "Detecting High-Order Epistasis in Nonlinear Genotype-Phenotype Maps". *Genetics* 205.3 (2017), pp. 1079–1088.

[153] Ruslan Salakhutdinov, Sam Roweis, and Zoubin Ghahramani. "Optimization with EM and Expectation-Conjugate-Gradient". In: *Proceedings of the International Conference on Machine Learning*. 2003, pp. 672–679.

[154] Karen S. Sarkisyan, Dmitry A. Bolotin, Margarita V. Meer, Dinara R. Usmanova, Alexander S. Mishin, George V. Sharonov, Dmitry N. Ivankov, Nina G. Bozhanova, Mikhail S. Baranov, Onuralp Soylemez, Natalya S. Bogatyreva, Peter K. Vlasov, Evgeny S. Egorov, Maria D. Logacheva, Alexey S. Kondrashov, Dmitry M. Chudakov, Ekaterina V. Putintseva, Ilgar Z. Mamedov, Dan S. Tawfik, Konstantin A. Lukyanov, and Fyodor A. Kondrashov. "Local fitness landscape of the green fluorescent protein". *Nature* 533 (2016), p. 397.

[155] Jörn M. Schmiedel and Ben Lehner. "Determining protein structures using deep mutagenesis". *Nat. Genet.* 51.7 (2019), pp. 1177–1186.

[156] Tatjana Schütze, Barbara Wilhelm, Nicole Greiner, Hannsjörg Braun, Franziska Peter, Mario Mörl, Volker A. Erdmann, Hans Lehrach, Zoltán Konthur, Marcus Menger, Peter F. Arndt, and Jörn Glökler. "Probing the SELEX process with next-generation sequencing". *PLoS One* 6.12 (2011).

[157] Andrew W. Senior, Richard Evans, John Jumper, James Kirkpatrick, Laurent Sifre, Tim Green, Chongli Qin, Augustin **v**Zídek, Alexander W. R. Nelson, Alex Bridgland, Hugo Penedones, Stig Petersen, Karen Simonyan, Steve Crossan, Pushmeet Kohli, David T. Jones, David Silver, Koray Kavukcuoglu, and Demis Hassabis. "Improved protein structure prediction using potentials from deep learning". *Nature* 577.7792 (2020), pp. 706–710.

[158] Siddhartha Shakya and John McCall. "Optimization by estimation of distribution with DEUM framework based on Markov random fields". *Int. J. Autom. Comput.* 4.3 (2007), pp. 262–272.

[159] Wen-Jun Shen, Hau-San Wong, Quan-Wu Xiao, Xin Guo, and Stephen Smale. "Introduction to the Peptide Binding Problem of Computational Immunology: New Results". *Found. Comput. Math.* 14.5 (2014), pp. 951–984.

[160] Yukio Shibata and Yosuke Kikuchi. "Graph Products Based on the Distance in Graphs". *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* E83A (2000).

[161] Vui A. Shim, Kay C. Tan, Jun Y. Chia, and Abdullah Al Mamun. "Multi-Objective Optimization with Estimation of Distribution Algorithm in a Noisy Environment". *Evol. Comput.* 21.1 (2013), pp. 149–177.

[162] Tyler C. Shimko, Polly M. Fordyce, and Yaron Orenstein. "DeCoDe: degenerate codon design for complete protein-coding DNA libraries". *Bioinformatics* 36.11 (2020), pp. 3357–3364.

[163] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps". In: *Workshop at International Conference on Learning Representations*. 2014.

[164] Sam Sinai, Richard Wang, Alexander Whatley, Stewart Slocum, Elina Locane, and Eric D. Kelsic. *AdaLead: A simple and robust adaptive greedy search algorithm for sequence design*. 2020.

[165] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. "Practical Bayesian Optimization of Machine Learning Algorithms". In: *Advances in Neural Information Processing Systems*. 2012, pp. 2951–2959.

[166] Peter F. Stadler. "Towards a theory of landscapes". In: *Complex Syst. Bin. Networks*. Vol. 461. 1995, pp. 77–163.

[167] Peter F. Stadler, Rudi Seitz, and Günter P. Wagner. "Population Dependent Fourier Decomposition of Fitness Landscapes over Recombination Spaces: Evolvability of Complex Characters". *Bull. Math. Biol.* 62.3 (2000), pp. 399–428.

[168] Robert Stadler Peter F.and Happel. "Random field models for fitness landscapes". *J. Math. Biol.* 38.5 (1999), pp. 435–478.

[169] Joe Staines and David Barber. "Optimization by Variational Bounding". In: *European Symposium on ANNs*. 2013.

[170] David Steele, Alexis Kertsburg, and Garrett A Soukup. "Engineered Catalytic RNA and DNA". *Am. J. Pharmacogenomics* 3.2 (2003), pp. 131–144.

[171] Gary D. Stormo. "Maximally Efficient Modeling of DNA Sequence Motifs at All Levels of Complexity". *Genetics* 187.4 (2011), pp. 1219–1224.

[172] Oksana M. Subach, Vladimir N. Malashkevich, Wendy D. Zencheck, Kateryna S. Morozova, Kiryl D. Piatkevich, Steven C. Almo, and Vladislav V. Verkhusha. "Structural Characterization of Acylimine-Containing Blue and Red Chromophores in mTagBFP and TagRFP Fluorescent Proteins". *Chem. Biol.* 17.4 (2010), pp. 333–341.

[173] Christopher M. Summa and Michael Levitt. "Near-native structure refinement using in vacuo energy minimization". *Proc. Natl. Acad. Sci. U.S.A* 104.9 (2007), pp. 3177–3182.

[174] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. "Intriguing properties of neural networks". In: *International Conference on Learning Representations*. 2014.

[175] Ammar Tareen, William Ireland, Anna Posfai, David McCandlish, and Justin Kinney. "MAVE-NN: Quantitative Modeling of Genotype-Phenotype Maps as Information Bottlenecks". *bioRxiv* (2020), p. 2020.07.14.201475.

[176] D. Gowanlock R. Tervo, Bum-Yeol Hwang, Sarada Viswanathan, Thomas Gaj, Maria Lavzin, Kimberly D. Ritola, Sarah Lindo, Susan Michael, Elena Kuleshova, David Ojala, Cheng-Chiu Huang, Charles R. Gerfen, Jackie Schiller, Joshua T. Dudman, Adam W. Hantman, Loren L. Looger, David V. Schaffer, and Alla Y. Karpova. "A Designer AAV Variant Permits Efficient Retrograde Access to Projection Neurons". *Neuron* 92.2 (2016), pp. 372–382.

[177] Nobuhiko Tokuriki, Colin J Jackson, Livnat Afriat-Jurnou, Kirsten T Wyganowski, Renmei Tang, and Dan S Tawfik. "Diminishing returns and tradeoffs constrain the laboratory optimization of an enzyme". *Nat. Commun.* 3.1 (2012), p. 1257.

[178] Craig Tuerk and Larry Gold. "Systematic evolution of ligands by exponential enrichment: RNA ligands to bacteriophage T4 DNA polymerase". *Science* 249.4968 (1990), pp. 505–510.

[179] Hanna Tuomisto. "A diversity of beta diversities: straightening up a concept gone awry. Part 1. Defining beta diversity as a function of alpha and gamma diversity". *Ecography* 33.1 (2010), pp. 2–22.

[180] Eeshit Dhaval Vaishnav, Carl G. de Boer, Moran Yassour, Jennifer Molinet, Lin Fan, Xian Adiconis, Dawn A. Thompson, Francisco A. Cubillos, Joshua Z. Levin, and Aviv Regev. "A comprehensive fitness landscape model reveals the evolutionary history and future evolvability of eukaryotic cis-regulatory DNA sequences". *bioRxiv* (2021).

[181] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. 2017, pp. 5998–6008.

[182] Deeptak Verma, Gevorg Grigoryan, and Chris Bailey-Kellogg. "Pareto optimization of combinatorial mutagenesis libraries". *IEEE/ACM Trans. Comput. Biol. Bioinforma.* 16.4 (2019), pp. 1143–1153.

[183] Edward D. Weinberger. "Fourier and Taylor series on fitness landscapes". *Biol. Cybern.* 65.5 (1991), pp. 321–330.

[184] Daniel M Weinreich, Yinghong Lan, C Scott Wylie, and Robert B Heckendorn. "Should evolutionary geneticists worry about higher-order epistasis?" *Curr. Opin. Genet. Dev.* 23.6 (2013), pp. 700–707.

[185] Daniel M. Weinreich, Yinghong Lan, Jacob Jaffe, and Robert B. Heckendorn. "The Influence of Higher-Order Epistasis on Biological Fitness Landscape Topography". *J. Stat. Phys.* 172.1 (2018), pp. 208–225.

[186] Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, Jan Peters, and Jürgen Schmidhuber. "Natural Evolution Strategies". *J. Mach. Learn. Res.* ().

[187] Daan Wierstra, Tom Schaul, Jan Peters, and Jürgen Schmidhuber. "Natural Evolution Strategies". In: *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*. 2008, pp. 3381–3387.

[188] Sewall Wright. "The roles of mutation, inbreeding, crossbreeding and selection in evolution". *Proc. 6th Int. Cong. Genet.* 1 (), pp. 356–366.

[189] Nicholas C. Wu, Lei Dai, C. Anders Olson, James O. Lloyd-Smith, and Ren Sun. "Adaptation in protein fitness landscapes is facilitated by indirect paths". *eLife* 5 (2016), e16965.

[190] C. Scott Wylie and Eugene I. Shakhnovich. "A biophysical protein folding model accounts for most mutational fitness effects in viruses". *Proc. Natl. Acad. Sci. U.S.A.* 108.24 (2011), pp. 9916–9921.

[191] Gloria Yang, Dave W. Anderson, Florian Baier, Elias Dohmen, Nansook Hong, Paul D. Carr, Shina Caroline Lynn Kamerlin, Colin J. Jackson, Erich Bornberg-Bauer, and Nobuhiko Tokuriki. "Higher-order epistasis shapes the fitness landscape of a xenobiotic-degrading enzyme". *Nat. Chem. Biol.* 15.11 (2019), pp. 1120–1128.

[192] Jianyi Yang and Yang Zhang. "I-TASSER server: new development for protein structure and function predictions". *Nucleic Acids Res.* 43.W1 (2015), W174–W181.

[193] Kevin K. Yang, Zachary Wu, and Frances H. Arnold. "Machine-learning-guided directed evolution for protein engineering". *Nat. Methods* 16.8 (2019), pp. 687–694.

[194] Kun Yao, John E. Herr, David W. Toth, Ryker Mckintyre, and John Parkhill. "The TensorMol-0.1 model chemistry: a neural network augmented with long-range physics". *Chem. Sci.* 9 (8 2018), pp. 2261–2269.

[195] Sun Yi, Daan Wierstra, Tom Schaul, and Jürgen Schmidhuber. "Stochastic Search Using the Natural Gradient". In: *Proceedings of the International Conference on Machine Learning*. 2009, pp. 1161–1168.

[196] Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. "Defending Against Neural Fake News". In: *Advances in Neural Information Processing Systems*. Vol. 32. 2019.

[197] Wei Zheng, Alan M Friedman, and Chris Bailey-Kellogg. "Algorithms for Joint Optimization of Stability and Diversity in Planning Combinatorial Libraries of Chimeric Proteins". *J. Comput. Biol.* 16.8 (2009).

[198] Mark Zlochin, Mauro Birattari, Nicolas Meuleau, and Marco Dorigo. "Model-based search for combinatorial optimization: A critical survey". *Ann. Oper. Res.* 131.1-4 (2004), pp. 373–395.

Appendix A

Proofs of sparsity calculations

In this appendix we provide proofs for the theoretical results presented in Chapter 4.

Graph theory preliminaries

Much of the following requires substantial graph-theoretic construction, so we first present the requisite notation and simple definitions. We will use the notation $V(G)$ and $E(G)$ to denote the vertex and edge sets of a graph G. The graph is then specified by $G = (V(G), E(G))$. The "degree" of a vertex is the number of other vertices that it is adjacent to. A k-regular graph is a graph in which every vertex has degree equal to k. The Graph Laplacian of a graph G with vertices $V(G) = \{g_i\}_{i=1}^{n}$ is given by $\mathbf{L}(G) \coloneqq \mathbf{D}(G) - \mathbf{A}(G)$ where $\mathbf{D}(G)$ is an $n \times n$ diagonal matrix whose i^{th} diagonal element is equal to the degree of vertex i and $\mathbf{A}(G)$ is the $n \times n$ adjacency matrix of G with elements given by

$$\mathbf{A}_{ij}(G) = \begin{cases} 1 & \text{if } g_i \text{ is adjacent to } g_j \text{ in } G, \\ 0 & \text{otherwise.} \end{cases}$$

Graph Laplacians and adjacency matrices are real, symmetric matrices and thus have orthonormal sets of eigenvectors. In the case of a k-regular graph, $\mathbf{L}(G) = k\mathbf{I} - \mathbf{A}(G)$, and thus the Laplacian and adjacency matrices share eigenvectors, and the eigenvalues of the Laplacian are given by $\lambda_j(\mathbf{L}) = k - \lambda_j(\mathbf{A})$ for $j = 1, ..., L$ where $\lambda_j(\mathbf{A})$ are the eigenvalues of the adjacency matrix.

We will make use of the Cartesian product of graphs, defined below:

Definition A.1 (Cartesian Product of Graphs)**.** The Cartesian product between two graphs $G = (V(G), E(G))$ and $H = (V(H), E(H))$ is defined as $G \square H = (V(G) \times V(H), E(G \square H))$, where two vertices (g, h) and (g', h') are adjacent in $G \square H$ if and only if either

1. $g = g'$ and h is adjacent to h' in H, or
2. $h = h'$ and g is adjacent to g' in G.

A direct consequence of Definition A.1 is that the adjacency matrix of the Cartesian product can be constructed from the adjacency matrices of its components as [160]:

$$\mathbf{A}(G\square H) = \mathbf{A}(G) \otimes \mathbf{I}_m + \mathbf{I}_n \otimes \mathbf{A}(H), \tag{A.1}$$

where $m = |V(H)|$ and $n = |V(G)|$ are the number of vertices in H and G, respectively.

We will additionally make use of the Lexicographic product of graphs [65].

Definition A.2 (Lexicographic Product of Graphs)**.** The Lexicographic product between two graphs $G = (V(G), E(G))$ and $H = (V(H), E(H))$ is defined as $G \circ H = (V(G) \times V(H), E(G \circ H))$, where two vertices (g, h) and (g', h') are adjacent in $G \circ H$ if and only if either

1. g is adjacent to g' in G, or
2. $g = g'$ and h is adjacent to h' in H.

The adjacency matrix of a Lexicographic product of graphs is given by [160]:

$$\mathbf{A}(G \circ H) = \mathbf{A}(G) \otimes \mathbf{J}_m + \mathbf{I}_n \otimes \mathbf{A}(H), \tag{A.2}$$

where $\mathbf{J}_m$ is the $m \times m$ matrix with every element equal to one.

The graphs described up until now have been 'simple' graphs, where each edge connects exactly two vertices and vertices are connected by at most one edge. We will also discuss 'hypergraphs', where 'edges' (dubbed hyperedges) are sets that may contain more than two vertices. Let $H = (V(H), E(H))$ be a hypergraph with n vertices, $V(H) = \{h_i\}_{i=1}^n$, and p hyperedges, $E(H) = \{e_j\}_{j=1}^p$. The *incidence* matrix of H, denoted $\mathbf{F}(H)$ is the $n \times p$ matrix with elements

$$\mathbf{F}_{ij}(H) = \begin{cases} 1 & \text{if } g_i \in e_i, \\ 0 & \text{otherwise.} \end{cases}$$

The degree of a vertex in a hypergraph is equal to the number of hyperedges that contain that vertex, and a k-regular hypergraph is one in which all vertices have degree equal to k. The *clique multigraph* corresponding to a hypergraph is the multigraph (another extension of simple graphs where two vertices can have multiple simple edges between them) with the same vertices as the hypergraph, and as many edges between two vertices as the number of times those vertices co-occur in a hyperedge of the hypergraph (i.e., if two vertices are both in two separate hyperedges of the hypergraph, then they will have two edges between them in the clique multigraph) [35]. The $(i, j)^{\text{th}}$ element adjacency matrix of a multigraph is equal to the number of edges that connect the i^{th} and j^{th} vertices.

Below are two Lemmas regarding the spectrum of Cartesian and lexicographic graph products that will be useful.

Lemma A.1. *Let G and H be regular graphs with n and m vertices, respectively. Let $\mathbf{A}(G) = \mathbf{P}\mathbf{\Lambda}_G\mathbf{P}^T$ and $\mathbf{A}(H) = \mathbf{Q}\mathbf{\Lambda}_H\mathbf{Q}^T$ be eigendecompositions of the adjacency matrices of G and H. Then the adjacency matrix of the Cartesian product $G\square H$ has the eigendecomposition given by:*

$$\mathbf{A}(G\square H) = \mathbf{R}\mathbf{\Lambda}_\square\mathbf{R}^T \tag{A.3}$$

where $\mathbf{R} = \mathbf{P} \otimes \mathbf{Q}$ and $\mathbf{\Lambda}_\square = \mathbf{\Lambda}_G \otimes \mathbf{I}_m + \mathbf{I}_n \otimes \mathbf{\Lambda}_H$.

Proof. We can use Equation A.1 to prove the proposed Lemma directly:

$$\begin{aligned}
\mathbf{R}\mathbf{\Lambda}_\square\mathbf{R}^T &= [\mathbf{P} \otimes \mathbf{Q}][\mathbf{\Lambda}_G \otimes \mathbf{I}_m + \mathbf{I}_n \otimes \mathbf{\Lambda}_H][\mathbf{P} \otimes \mathbf{Q}]^T \\
&= [\mathbf{P}\mathbf{\Lambda}_G \otimes \mathbf{Q} + \mathbf{P} \otimes \mathbf{Q}\mathbf{\Lambda}_H][\mathbf{P}^T \otimes \mathbf{Q}^T] \\
&= \mathbf{P}\mathbf{\Lambda}_G\mathbf{P}^T \otimes \mathbf{Q}\mathbf{Q}^T + \mathbf{P}\mathbf{P}^T \otimes \mathbf{Q}\mathbf{\Lambda}_H\mathbf{Q}^T \\
&= \mathbf{A}(G) \otimes \mathbf{I}_m + \mathbf{I}_n \otimes \mathbf{A}(H) \\
&= \mathbf{A}(G\square H)
\end{aligned}$$

where in the second and third lines we have used the property of Kronecker products that $(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = (\mathbf{AB} \otimes \mathbf{CD})$, in the second line we have also used the fact that the transpose is distributive over the Kronecker product, $(\mathbf{C} \otimes \mathbf{D})^T = \mathbf{C}^T \otimes \mathbf{D}^T$, and the final line is a result of Equation A.1. □

Lemma A.2. *Let G and H be regular graphs with n and m vertices, respectively. Let $\mathbf{A}(G) = \mathbf{P}\mathbf{\Lambda}_G\mathbf{P}^T$ and $\mathbf{A}(H) = \mathbf{Q}\mathbf{\Lambda}_H\mathbf{Q}^T$ be eigendecompositions of the adjacency matrices of G and H. Then the adjacency matrix of the lexicographic product $G \circ H$ has the eigendecomposition given by:*

$$\mathbf{A}(G \circ H) = \mathbf{R}\mathbf{\Lambda}_\circ\mathbf{R}^T \tag{A.4}$$

where $\mathbf{R} = \mathbf{P} \otimes \mathbf{Q}$ and $\mathbf{\Lambda}_\circ = \mathbf{\Lambda}_G \otimes \mathbf{B} + \mathbf{I}_n \otimes \mathbf{\Lambda}_H$ and $\mathbf{B} = m\mathbf{e}_1\mathbf{e}_1^T$ (i.e., $B_{ij} = m$ if $i = j = 1$ and zero otherwise).

Proof. The adjacency matrix of any k-regular graph with m vertices has two eigenvalues, k and 0 with multiplicities 1 and $m - 1$, respectively. The normalized eigenvector associated with the eigenvalue k is $\frac{1}{\sqrt{m}}\mathbf{1}_m$ and the normalized eigenvectors associated with the eigenvalue 0 are any set of length-m orthogonormal vectors that are orthogonal to $\mathbf{1}_m$ (i.e., vectors that sum to one). Since H is a regular graph, we then have that

$$(\mathbf{Q}^T\mathbf{J}_m)_{ij} = \begin{cases} \sqrt{m} & \text{if } i = 1 \\ 0 & \text{otherwise} \end{cases}$$

and

$$(\mathbf{Q}^T\mathbf{J}_m\mathbf{Q})_{ij} = \begin{cases} m & \text{if } i = j = 1 \\ 0 & \text{otherwise} \end{cases}$$

Therefore, $(\mathbf{Q}^T\mathbf{J}_m\mathbf{Q}) = \mathbf{B}$ and further $\mathbf{Q}\mathbf{B}\mathbf{Q}^T = \mathbf{J}_m$. Then we can use Equation (A.2) to show:

$$\begin{aligned}\mathbf{R}\mathbf{\Lambda}_\circ\mathbf{R}^T &= [\mathbf{P}\otimes\mathbf{Q}][\mathbf{\Lambda}_G\otimes\mathbf{B} + \mathbf{I}_n\otimes\mathbf{\Lambda}_H][\mathbf{P}\otimes\mathbf{Q}]^T \\ &= \mathbf{P}\mathbf{\Lambda}_G\mathbf{P}^T\otimes\mathbf{Q}\mathbf{B}\mathbf{Q}^T + \mathbf{P}\mathbf{P}^T\otimes\mathbf{Q}\mathbf{\Lambda}_H\mathbf{Q}^T \\ &= \mathbf{A}(G)\otimes\mathbf{J}_m + \mathbf{I}_n\otimes\mathbf{A}(H) \\ &= \mathbf{A}(G\circ H)\end{aligned}$$

□

Proofs of Propositions 4.1 and 4.2

Proof of Proposition 4.1. The Graph Laplacian of the complete graph $K(q)$ is given by $\mathbf{L}(K(q)) = q\mathbf{I}_q - \mathbf{J}_q$ (i.e., the $q\times q$ matrix with q on the diagonal and all other elements equal to -1). This matrix has two eigenvalues, 0 and q, with multiplicities 1 and $q-1$, respectively. The normalized eigenvector corresponding to the zero eigenvalue is $\frac{1}{\sqrt{q}}\mathbf{1}_q$. Since the graph Laplacian is symmetric, the eigenvectors are orthogonal and therefore the remaining eigenvectors are any set of $n-1$ orthogonal vectors that are orthogonal to $\mathbf{1}_q$ (i.e., vectors that sum to zero) and each other. In order to show that the columns of the Householder matrix given in Equation (4.7) are orthonormal eigenvectors of the complete graph, we will prove (i) that the first column of $\mathbf{P}_q$ is equal to $\frac{1}{\sqrt{q}}\mathbf{1}_q$ and (ii) that $\mathbf{P}_q$ is an orthogonal matrix:

(i) We can more explicitly write the $\mathbf{w} = [w_1, w_2, ..., w_q]$ vector as $\mathbf{w} = \mathbf{1}_q - \sqrt{q}\mathbf{e}_1 = [1-\sqrt{q}, 1, 1, ..., 1]^T$. Therefore, $\|\mathbf{w}\|_2^2 = (1-\sqrt{q})^2 + (q-1) = 2(q-\sqrt{q})$. Let α_i for $i = 1, 2, ..q$ be the elements of the first column of $\mathbf{P}_q$, respectively. Then,

$$\begin{aligned}\alpha_1 &= 1 - \frac{2w_1w_1}{\|w\|_2^2} \\ &= 1 - \frac{(1-\sqrt{q})^2}{q-\sqrt{q}} \\ &= 1 - \left(1 - \frac{1}{\sqrt{q}}\right) = \frac{1}{\sqrt{q}},\end{aligned}$$

and for $j = 2, 3, ..., q$, we have

$$\begin{aligned}\alpha_j &= \frac{2w_jw_1}{\|w\|_2^2} \\ &= \frac{1-\sqrt{q}}{q-\sqrt{q}} \\ &= \frac{1}{\sqrt{q}}.\end{aligned}$$

Therefore, all elements of the first columns of $\mathbf{P}_q$ are equal to $\frac{1}{\sqrt{q}}$.

(ii) The orthogonality of $\mathbf{P}_q$ follows directly from the definition:

$$\begin{aligned}(\mathbf{P}_q)^T\mathbf{P}_q &= \left(\mathbf{I}_q - \frac{2\mathbf{w}\mathbf{w}^T}{\|\mathbf{w}\|_2^2}\right)^T\left(\mathbf{I}_q - \frac{2\mathbf{w}\mathbf{w}^T}{\|\mathbf{w}\|_2^2}\right)\\ &= \mathbf{I}_q - \frac{4\mathbf{w}\mathbf{w}^T}{\|\mathbf{w}\|_2^2} + \frac{4\mathbf{w}\mathbf{w}^T\mathbf{w}\mathbf{w}^T}{\|\mathbf{w}\|_2^4}\\ &= \mathbf{I}_q.\end{aligned}$$

Thus, $\mathbf{P}_q$ is an orthogonal matrix whose first column is $\frac{1}{\sqrt{q}}\mathbf{1}_q$, and further, the columns are $\mathbf{P}_q$ are an orthonormal set of eigenvectors of $K(q)$. □

To prove Proposition 4.2, we need a couple of preliminary results. First note that the Hamming graph $H(L,q)$ is the L-fold Cartesian product of the complete graph $K(q)$. We have the following result regarding the eigenvectors and eigenvalues of the adjacency matrices of Cartesian products of regular graphs.

Lemma A.3. *Let G and H be regular graphs and let $\mathbf{P}$ and $\mathbf{Q}$ be matrices whose columns are eigenvectors of the Graph Laplacians of G and H, respectively. Then the columns of $\mathbf{P}\otimes\mathbf{Q}$ are eigenvectors of the Graph Laplacian of the Cartesian product $G\square H$.*

Proof. For a regular graph, the degree matrix is a constant multiplied by the identity matrix. Thus, the Graph Laplacian and adjacency matrices differ only by a constant added to the diagonal (and a constant multiplication of -1), and these two matrices therefore have the same eigenvectors. The result then follows from Lemma A.1. □

Proof of Proposition 4.2. The Hamming graph $H(L,q)$ is defined as the L-fold Cartesian product of the complete graph $K(q)$ [166]:

$$H(L,q) = \square_{i=1}^{L} K(q). \tag{A.5}$$

Thus, by Lemma A.3, the eigenvectors of the Graph Laplacian of $H(L,q)$ are the L-fold Kronecker product of the eigenvectors of the Graph Laplacian of $K(q)$, as given in Equation (4.8). □

Proofs of Theorems 4.1 and 4.2

In order prove Theorems 4.1 and 4.2, we will first provide an alternative definition of the GNK model in terms of hypergraphs. To start, we'll now assign an index to every sequence in the space of sequences, so $\mathcal{S}^{(L,q)} = \{\mathbf{s}_i\}_{i=1}^{q^L}$. As in the main text, $\mathbf{s}_i^{[j]}$ refers to the subsequence of $\mathbf{s}_i$ corresponding to the indices in the neighborhood $V^{[j]}$.

Each neighborhood in the GNK model induces a hypergraph over sequence space, where the vertices represent all sequences in $\mathcal{S}^{(L,q)}$ and edges contain sequences that share subsequences corresponding to the indices in the neighborhood. We formally define this hypergraph below.

Definition A.3 (GNK hypergraph)**.** Let $G(V) = (\mathcal{S}^{(L,q)}, E(V))$ be a 'GNK hypergraph' corresponding to a neighborhood V for a GNK model defined for sequences of length L and alphabet size q. The edge set, $E(V)$, corresponds to every possible subsequence of length $|V|$, and two sequences co-occur in an edge if and only if they share the subsequence corresponding to the positions in V. Additionally, let $\mathbf{F}(V) \coloneqq \mathbf{F}(G(V))$ be the incidence matrix of $G(V)$, $C(V)$ be the clique multigraph of $G(V)$ and $\mathbf{A}(V) \coloneqq \mathbf{A}(C(V))$ be the adjacency matrix of $C(V)$. Finally, when it is appropriate to consider the indexed neighborhoods $V^{[j]}$, then we will use this indexing for all of the GNK hypergraph quantities. Specifically, define $G^{[j]} \coloneqq G(V^{[j]})$, $\mathbf{F}^{[j]} \coloneqq \mathbf{F}(V^{[j]})$, $C^{[j]} \coloneqq C(V^{[j]})$, and $\mathbf{A}^{[j]} \coloneqq \mathbf{A}(V^{[j]})$.

The following Lemma gives an immediate useful result of this definition.

Lemma A.4. *Every GNK hypergraph, $G(V)$, is a 1-regular hypergraph.*

Proof. Every sequence (i.e., vertex of $G(V)$) contains exactly one subsequence corresponding to the position indices in V. Therefore, each vertex is contained in exactly one edge of $G(V)$. □

We will use the GNK hypergraphs to provide an alternative definition of the GNK model, which is shown in the following result. Note that this is equivalent to the matrix definition of the GNK model of Buzas and Dinitz [30].

Lemma A.5. *Define the matrix $\mathbf{F}$ as the column-wise concatenation of the incidence matrices $\mathbf{F}^{[j]}$ for $j = 1, 2, ..., L$:*

$$\mathbf{F} \coloneqq \left[\mathbf{F}^{[1]} \,\middle|\, \mathbf{F}^{[2]} \,\middle|\, \dots \,\middle|\, \mathbf{F}^{[L]} \right] \tag{A.6}$$

Additionally, let $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ be a length $\sum_{j=1}^{L} q^{K_j}$ normally distributed random vector. Then, $\mathbf{f} = \mathbf{F}\mathbf{w}$ contains all fitness evaluations of a fitness function f that is distributed according to GNK$(L, q, \mathcal{V})$ (i.e., $\mathbf{f} = (f(\mathbf{s}))_{\mathbf{s} \in \mathcal{S}^{(L,q)}}$ and $f \sim$ GNK$(L, q, \mathcal{V})$).

Proof. In order to prove this, we need to show (i) that the above formulation results in L unit normally distributed subsequence fitness values being assigned to each sequence, where each subsequence corresponds to the position indices in a neighborhood $V^{[j]}$ (ii) that sequences share subsequence fitness values when they share the corresponding subsequence, and (iii) that the L subsequence fitness values are summed to produce the total fitness value assigned to each sequence.

A direct result of Definition A is that $F^{[j]}$ has elements given by:

$$\mathbf{F}^{[j]}_{ik} = \begin{cases} 1 & \text{if } \mathbf{s}_i^{[j]} = \tilde{\mathbf{s}}_k \\ 0 & \text{otherwise,} \end{cases} \tag{A.7}$$

where we define $\tilde{\mathbf{s}}_k$ as the k^{th} possible subsequence of length K_j (i.e., the k^{th} element in $\mathcal{S}^{(K_j,q)}$). Since each hyperedge in $G^{[j]}$ represents a subsequence of length K_j, each hyperedge

contains vertices that represent sequences that share subsequence fitness values in the GNK model. Therefore, letting $\mathbf{w}^{[j]} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ be a length q^{K_j} normally distributed random vector representing the subsequence fitness values randomly assigned to each subsequence, then

$$\mathbf{f}^{[j]} = \mathbf{F}^{[j]}\mathbf{w}^{[j]},$$

where $\mathbf{f}^{[j]} = [f_j(\mathbf{s}_1), f_j(\mathbf{s}_2), ..., f_j(\mathbf{s}_{q^L})]^T$ is the vector of subsequence fitness values corresponding to neighborhood j that are assigned to each sequence in $\mathcal{S}^{(L,q)}$. Since $G^{[j]}$ is a 1-regular hypergraph (Lemma A.4), each row of $\mathbf{F}^{[j]}$ contains exactly one nonzero element and therefore the subsequence fitness values of each sequence are distributed as $\mathcal{N}(0, 1)$, as in the original definition of the GNK model given in the main text. Additionally, the structure of the incidence matrix shown in Equation (A.7) ensures that two sequences that share a subsequence corresponding to the position indices in $V^{[j]}$ also share a subsequence fitness value in $\mathbf{f}^{[j]}$, as required by the GNK model.

Now, let $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ be the random vector that is the concatenation of the $\mathbf{w}^{[j]}$ random vectors containing subsequence fitness values. Then we have:

$$\begin{aligned}\mathbf{f} &= \mathbf{F}\mathbf{w} \\ &= \sum_{j=1}^{L} \mathbf{F}^{[j]}\mathbf{w}^{[j]} \\ &= \sum_{j=1}^{L} \mathbf{f}^{[j]}.\end{aligned}$$

Therefore, the elements of $\mathbf{f}$ are simply the sums of the L subsequence fitness values corresponding to each sequence, which is the final step in definition of the GNK model given in the main text. □

The following conclusions regarding the statistics of the Fourier coefficients in the GNK model are immediate from this definition of the model.

Lemma A.6. *The Fourier coefficients, $\boldsymbol{\beta}$, of fitness functions distributed according to GNK$(L, q, \mathcal{V})$ are normally distributed with $\langle\boldsymbol{\beta}\rangle = 0$ and covariance $\langle\boldsymbol{\beta}\boldsymbol{\beta}^T\rangle = \left(\boldsymbol{\Phi}^{(L,q)}\right)^T \mathbf{F}\mathbf{F}^T\boldsymbol{\Phi}^{(L,q)}$, where $\boldsymbol{\Phi}^{(L,q)}$ is the Fourier basis defined in Equation* (4.8) *and $\mathbf{F}$ is the column-wise concatenation of the incidence matrices of GNK hypergraphs defined in Equation* (A.6).

Proof. For ease of notation, let $\boldsymbol{\Phi} \leftarrow \boldsymbol{\Phi}^{(L,q)}$. The proposed Lemma follows immediately from recognizing that $\mathbf{f} = \mathbf{F}\mathbf{w} = \boldsymbol{\Phi}\boldsymbol{\beta}$, and therefore $\boldsymbol{\beta} = \boldsymbol{\Phi}^T\mathbf{F}\mathbf{w}$. The Fourier coefficients are thus a linear transformation of a normally distributed random vector, $\mathbf{w}$, and are therefore normally distributed with mean $\langle\boldsymbol{\beta}\rangle = \boldsymbol{\Phi}^T\mathbf{F}\langle\mathbf{w}\rangle = \mathbf{0}$ and covariance matrix $\langle\boldsymbol{\beta}\boldsymbol{\beta}^T\rangle = \boldsymbol{\Phi}^T\mathbf{F}\langle\mathbf{w}\mathbf{w}^T\rangle\mathbf{F}^T\boldsymbol{\Phi} = \boldsymbol{\Phi}^T\mathbf{F}\mathbf{F}^T\boldsymbol{\Phi}$. □

Lemma A.6 provides a straightforward path towards proving Theorem 4.1. We now need to show that (i) $\mathbf{\Phi}$ diagonalizes $\mathbf{F}\mathbf{F}^T$ (i.e., $\mathbf{\Phi}$ is a basis of eigenvectors for $\mathbf{F}\mathbf{F}^T$) and (ii) that the eigenvalues of $\mathbf{F}\mathbf{F}^T$ are given by Equation (4.12). The problem can be further simplified by first noting the following simple result, which follows straightforwardly from the multiplication of block matrices.

Lemma A.7. $\mathbf{F}\mathbf{F}^T = \sum_{j=1}^{L} \mathbf{F}^{[j]}(\mathbf{F}^{[j]})^T$.

This result tells us that if possible, it is sufficient to prove that $\mathbf{\Phi}$ diagonalizes each $\mathbf{F}^{[j]}(\mathbf{F}^{[j]})^T$ in order to prove that $\mathbf{\Phi}$ diagonalizes $\mathbf{F}\mathbf{F}^T$. Then the eigenvalues of $\mathbf{F}\mathbf{F}^T$ will simply be given by the sum of the eigenvalues of the $\mathbf{F}^{[j]}(\mathbf{F}^{[j]})^T$. We are further assisted by the following result regarding the outer product of incidence matrices of regular hypergraphs, due to [35].

Lemma A.8. *Let C be the clique multigraph of a k-regular hypergraph H with incidence matrix $\mathbf{F}(H)$. Then $\mathbf{F}(H)\mathbf{F}(H)^T = \mathbf{A}(C) + k\mathbf{I}$, where $\mathbf{A}(C)$ is the adjacency matrix of C.*

Proof. Each element of $\mathbf{F}(H)\mathbf{F}(H)^T$ is the inner product of two rows in $\mathbf{F}(H)$. Since each element in row i of $\mathbf{F}(H)$ indicates whether vertex i is in a particular edge, the inner product of row i and row j ($i \neq j$) counts the number of edges that contain both vertex i and vertex j. Of course, this is also the number of edges connecting vertex i and vertex j in the clique multigraph, and thus the off-diagonal elements of $\mathbf{F}(H)\mathbf{F}(H)^T$ are equal to the elements of $\mathbf{A}(C)$. The diagonal elements of $\mathbf{F}(H)\mathbf{F}(H)^T$ are equal to the total number of edges containing vertex i, which is L for every vertex. □

Lemma A.8 tells us if we can determine the spectrum of the adjacency matrices $\mathbf{A}^{[j]}$ of the clique multigraphs $C^{[j]}$, then it is straightforward to calculate the spectrum of $\mathbf{F}^{[j]}(\mathbf{F}^{[j]})^T$. In order to begin to calculate the spectrum of $\mathbf{A}^{[j]}$ we recognize the following simple fact regarding these clique multigraphs (remember that $G^{[j]}$ is a 1-regular hypergraph).

Lemma A.9. *The clique multigraph of a 1-regular hypergraph is a simple graph.*

Proof. Each vertex in a 1-regular hypergraph is in exactly one hyperedge, and thus the clique multigraph has at most one edge between any two vertices. □

We have therefore reduced the problem to determining the spectrum of the simple graphs $C^{[j]}$. $C^{[j]}$ contains edges between any two sequences that share a subsequence corresponding to the indices in the j^{th} neighborhood $V^{[j]}$. In order to calculate the spectrum of $C^{[j]}$, we will first show how these clique multigraphs can be constructed recursively. In the next few Lemmas, we will provide results for graphs associated with a generic neighborhood V, and then return to considering the indexed neighborhoods $V^{[j]}$ when necessary.

Lemma A.10. *Let $V \subseteq \{1, 2, ..., L\}$ be a GNK neighborhood. Additionally, let $O(q)$ be the empty graph of size q (i.e., the graph containing q vertices and no edges) and define the graphs $B_l(V)$, via the recursion relation:*

$$B_{l+1}(V) = \begin{cases} B_l(V) \square O(q) & \text{if } i+1 \in V \\ B_l(V) \circ K(q) & \text{otherwise} \end{cases} \tag{A.8}$$

for $i = 1, 2, ..., L-1$, where

$$B_1(V) = \begin{cases} O(q) & \text{if } 1 \in V \\ K(q) & \text{otherwise} \end{cases} \tag{A.9}$$

Then the vertices of $B_l(V)$ represent all sequences in $\mathcal{S}^{(l,q)}$ and two sequences $\mathbf{s}_i = [s_{i,1}, s_{i,2}, ..., s_{i,l}] \in \mathcal{S}^{(l,q)}$ and $\mathbf{s}_j = [s_{j,1}, s_{j,2}, ..., s_{j,l}] \in \mathcal{S}^{(l,q)}$ are adjacent in $B_l(V)$ if and only if $s_{i,k} = s_{j,k}$ for every $k \in V_{(l)}$ where we define $V_{(l)} \coloneqq \{m \in V : m \leq l\}$ to be the l smallest elements of V.

Proof. First, both the lexicographic and Cartesian products result in graphs whose vertex sets are the (set) Cartesian product of the vertex sets of the multiples. Since the vertex sets of both $O(q)$ and $K(q)$ represent elements of the alphabet of size q, an l-fold graph product of these graphs will result in each vertex representing a sequence of length l.

We will prove the adjacency property of these product graphs with induction. For ease of notation, we drop the dependence of $B_l(V)$ on V and let $B_l \leftarrow B_l(V)$. Assume that two sequences $\mathbf{s}_i = [s_{i,1}, s_{i,2}, ..., s_{i,l}] \in \mathcal{S}^{(l,q)}$ and $\mathbf{s}_j = [s_{j,1}, s_{j,2}, ..., s_{j,l}] \in \mathcal{S}^{(l,q)}$ are adjacent in B_l if and only if the adjacency condition, $s_{i,k} = s_{j,k}$ for every $k \in V_{(l)}$, is satisfied. We will show that these adjacency conditions remain true for $l+1$. There are two cases to consider: (i) $l+1 \in V$ and (ii) $l+1 \notin V$.

(i) $(l+1 \in V)$. Let $\mathbf{s}_i = [\tilde{\mathbf{s}}_i | s_{i,l+1}] \in \mathcal{S}^{(l+1,q)}$ and $\mathbf{s}_j = [\tilde{\mathbf{s}}_j | s_{j,l+1}] \in \mathcal{S}^{(l+1,q)}$ be sequences of length $l+1$, where $\tilde{\mathbf{s}}_i$ contains the first l elements of $\mathbf{s}_i$. Since in this case $l+1 \in V_{(l+1)}$, we must prove that $\mathbf{s}_i$ and $\mathbf{s}_j$ are adjacent in B_{l+1} if and only if the adjacency condition is satisfied for $\tilde{\mathbf{s}}_i$ and $\tilde{\mathbf{s}}_j$ in B_l (which is true by inductive assumption) and $s_{i,l+1} = s_{j,l+1}$. By Equation A.8, $B_{l+1} = B_l \square O(q)$ in this case. Note that the vertices in B_l represent the $\tilde{\mathbf{s}}_i$ sequences of length l and the vertices in $O(q)$ represent the new elements of the sequence, $s_{i,l+1}$. According to the definition of the graph Cartesian product (Definition A.1), $\mathbf{s}_i$ and $\mathbf{s}_j$ are adjacent in B_{l+1} if and only if $\tilde{\mathbf{s}}_i$ and $\tilde{\mathbf{s}}_j$ are adjacent in B_l and $s_{i,l+1} = s_{j,l+1}$. Thus, in this case, the adjacency condition remains true for $l+1$ under the inductive assumption.

(ii) $(l+1 \notin V)$. In this case, $l+1 \notin V_{(l+1)}$ and therefore we need to prove that $\mathbf{s}_i$ and $\mathbf{s}_j$ are adjacent in B_{l+1} if and only if $\tilde{\mathbf{s}}_i$ and $\tilde{\mathbf{s}}_j$ are adjacent in B_l or $\tilde{\mathbf{s}}_i = \tilde{\mathbf{s}}_j$. In this case, $B_{l+1} = B_l \circ K(q)$. Due to the definition of the lexicographic product (Definition A.2), $\mathbf{s}_i$ and $\mathbf{s}_j$ are adjacent in B_{l+1} if and only if (1) $\tilde{\mathbf{s}}_i$ and $\tilde{\mathbf{s}}_j$ are adjacent in B_l or (2)

$\tilde{\mathbf{s}}_i = \tilde{\mathbf{s}}_j$ and $s_{i,l+1}$ is adjacent to $s_{j,l+1}$ in $K(q)$. Since all vertices in $K(q)$ are adjacent to one another, condition (2) simply results in $\mathbf{s}_i$ and $\mathbf{s}_j$ being adjacent in B_{l+1} if $\tilde{\mathbf{s}}_i = \tilde{\mathbf{s}}_j$. Thus, in this case, the required adjacency condition remains true for $l+1$ under the inductive assumption.

The base case of this induction is $l = 1$. If $1 \in V$, then $V_{(1)} = \{1\}$. Since $s_{i,1} \neq s_{j,1}$ for all $\mathbf{s}_i, \mathbf{s}_j \in \mathcal{S}^{(1,q)}$ with $i \neq j$ (i.e., since each length-one sequence represents an element of the alphabet, none of these sequences are equal to one another), the graph B_1 should contain no edges; this is indeed the case because $B_1 = O(q)$ in the case $1 \in V^{[j]}$ due to Equation (A.9). Similarly, if $i \notin V^{[j]}$, then $V^{[j]}_{(1)} = \emptyset$ and all vertices of B_1 should be adjacent to one another; this is indeed the case because $B_1 = K(q)$ when $1 \notin V^{[j]}$ due to Equation (A.9). □

The simple corollary of Lemma A.10 is that the clique graphs $C(V)$ are the final results of the recursion in Equation (A.8).

Lemma A.11. *Let $C(V)$ be the clique multigraph of a GNK hypergraph $G(V)$ and $B_L(V)$ be the graph defined by Equations* (A.8) *and* (A.9)*. Then $C(V) = B_L(V)$.*

Proof. The vertex sets of both $C(V)$ and $B_L(V)$ are given by the space of sequences $\mathcal{S}^{(L,q)}$. By definition, two sequences co-occur in an edge of the hypergraph $G(V)$ if and only if they share the subsequence corresponding to the indices in V. Therefore two sequences are adjacent in the clique multigraph $C(V)$ if and only if they share the subsequence corresponding to the indices in V. Additionally, by Lemma A.9, $C(V)$ is a simple graph. By Lemma A.10, $B_L(V)$ is a simple graph in which two sequences are adjacent if and only if they share the subsequence corresponding to the indices in V. Thus, the vertex and edge sets of $C(V)$ and $B_L(V)$ are equivalent, and the graphs are equivalent. □

This recursive definition of $C(V)$ will allow us to calculate the spectrum of the adjacency matrix $\mathbf{A}(V)$ using the spectral properties of graph products presented in Lemmas A.1 and A.2. In particular, we have the following result regarding the eigenvectors of $\mathbf{A}(V)$.

Lemma A.12. *The columns of the Fourier basis $\mathbf{\Phi}^{(L,q)}$ are a complete set of orthonormal eigenvectors of the adjacency matrix $\mathbf{A}(V) \coloneqq \mathbf{A}(C(V))$ of the clique multigraph $C(V)$.*

Proof. Recall from Equation (4.8) in the main text that $\mathbf{\Phi}^{(L,q)} = \bigotimes_{i=1}^{L} \mathbf{P}_q$, where $\mathbf{P}_q$ is a complete set of orthonormal eigenvectors of the complete graph $K(q)$. Additionally, recognize that the adjacency matrix of the empty graph, $\mathbf{A}(O(q))$ has every element equal to zero and therefore any nonzero vector is an eigenvector of $\mathbf{A}(O(q))$; for our purposes we will use $\mathbf{P}_q$ as the eigenvectors of $\mathbf{A}(O(q))$. Let the columns of $\mathbf{\Theta}_l$ be orthonormal eigenvectors of the graph $B_l(V)$, which is defined in Lemma A.10. Then we have

$$\mathbf{\Theta}_1 = \begin{cases} \mathbf{P}_q & \text{if } 1 \in V \\ \mathbf{P}_q & \text{otherwise} \end{cases}$$

where the first and second lines on the RHS are due to $\mathbf{P}_q$ being a set of orthonormal eigenvectors of $O(q)$ and $K(q)$, respectively. We additionally have the recursive relation:

$$\mathbf{\Theta}_{l+1} = \begin{cases} \mathbf{\Theta}_l \otimes \mathbf{P}_q & \text{if } i+1 \in V^{[j]} \\ \mathbf{\Theta}_l \otimes \mathbf{P}_q & \text{otherwise} \end{cases}$$

for $l = 1, 2, ...L-1$, where the first line on the RHS is due to Lemma A.1 and the second is due to Lemma A.2. Therefore, $\mathbf{\Theta}_L = \bigotimes_{i=1}^{L} \mathbf{P}_q = \mathbf{\Phi}^{(L,q)}$. The result follows from recognizing that $C(V) = B_L(V)$ by Lemma A.11, and therefore $\mathbf{\Theta}_L = \mathbf{\Phi}^{(L,q)}$ are a set of orthonormal eigenvectors of $\mathbf{A}(V)$. □

We could similarly use Lemmas A.1 and A.2 to calculate the eigenvalues of $\mathbf{A}(V)$; however, this would not allow us to connect the eigenvalues to epistatic interactions, as is required to prove Theorem 4.1. We will instead proceed by showing in Lemma A.13 that the columns of suitably defined matrix are eigenvectors of the adjacency matrix $\mathbf{A}(V)$ with eigenvalues equal to a summand of Equation (4.12) up to additive constant. Then, in Lemma A.14 we will show that this matrix is indeed equal to the columns of the Fourier basis corresponding to the epistatic interaction U.

Lemma A.13. *Let $U \subseteq 1, 2, ..., L$ be a set of position indices representing an epistatic interaction and $V \subseteq \{1, 2, ..., L\}$ be a GNK neighborhood. Define the matrix $\mathbf{Z}_l(U)$ with the recursion relation:*

$$\mathbf{Z}_{l+1}(U) = \begin{cases} \mathbf{Z}_l(U) \otimes \tilde{\mathbf{P}}_q & \textit{if} \quad l+1 \in U \\ \mathbf{Z}_l(U) \otimes \mathbf{1}_q & \textit{otherwise} \end{cases} \tag{A.10}$$

for $l = 1, 2, ..., L-1$, where

$$\mathbf{Z}_1(U) = \begin{cases} \tilde{\mathbf{P}}_q & \textit{if} \quad 1 \in U \\ \mathbf{1}_q & \textit{otherwise} \end{cases} \tag{A.11}$$

Then the columns of $\mathbf{Z}_L(U)$ are eigenvectors of the adjacency matrix $\mathbf{A}(V)$, all associated with the eigenvalue given by

$$\mu(U, V) = q^{L-|V|} I(U \subseteq V) - 1. \tag{A.12}$$

Proof. We will prove this by induction. For ease of notation, we will drop the dependence of the $Z_l(U)$ matrices on U, and let $\mathbf{Z}_l \leftarrow \mathbf{Z}_l(U)$. Define $\mathbf{A}_l \coloneqq \mathbf{A}(B_l(V))$ as the adjacency of the graph $B_l(V)$, which is the graph defined by Equations (A.8) and (A.9). Additionally let $V_{(l)} \coloneqq \{m \in V : m \leq l\}$ and $U_{(l)} \coloneqq \{m \in U : m \leq l\}$ be the l smallest elements of V and U, respectively. Our inductive assumption will be that

$$\mathbf{A}_l \mathbf{Z}_l = \mu_l \mathbf{Z}_l$$

where we define $\mu_l \coloneqq q^{l-|V_{(l)}|} I(U_{(l)} \subseteq V_{(l)}) - 1$. In other words, we will assume that the columns of $\mathbf{Z}_l$ are eigenvectors of $\mathbf{A}_l$ associated with the eigenvalue μ_l, and then will show that $\mathbf{A}_{l+1}\mathbf{Z}_{l+1} = \mu_{l+1}\mathbf{Z}_{l+1}$. There are four cases to consider: (i) $l+1 \in U$ and $l+1 \in V$, (ii) $l+1 \in U$ and $l+1 \notin V$, (iii) $l+1 \notin U$ and $l+1 \in V$, and (iv) $l+1 \notin U$ and $l+1 \notin V$

(i) ($l+1 \in U$ and $l+1 \in V$). In this case, $V_{(l+1)}$ and $U_{(l+1)}$ add the element $l+1$ to $V_{(l)}$ and $U_{(l)}$, respectively. Therefore, if $U_{(l)} \subseteq V_{(l)}$, it will be true that $U_{(l+1)} \subseteq V_{(l+1)}$, and if $U_{(l)} \not\subseteq V_{(l)}$, then $U_{(l+1)} \not\subseteq V_{(l+1)}$. Additionally, in this case, $|V_{(l+1)}| = |V_{(l)}| + 1$, so we have

$$\begin{aligned}\mu_{l+1} &= q^{l+1-|V_{(l+1)}|} I(U_{(l+1)} \subseteq V_{(l+1)}) - 1 \\ &= q^{l-|V_{(l)}|} I(U_{(l)} \subseteq V_{(l)}) - 1 \\ &= \mu_l\end{aligned}$$

Therefore, we must show that μ_l is the eigenvalue of $\mathbf{A}_{l+1}$ associated with the columns of $\mathbf{Z}_{l+1}$. In this case, $\mathbf{Z}_{l+1} = \mathbf{Z}_l \otimes \tilde{\mathbf{P}}_q$. Also in this case, $B_{l+1} = B_l \square O(q)$ (by Equation A.8), so by Equation (A.1), $\mathbf{A}_{l+1} = \mathbf{A}_l \otimes \mathbf{I}_q$. Then we have

$$\begin{aligned}\mathbf{A}_{l+1}\mathbf{Z}_{l+1} &= (\mathbf{A}_l \otimes \mathbf{I}_q)(\mathbf{Z}_l \otimes \tilde{\mathbf{P}}_q) \\ &= \mathbf{A}_l\mathbf{Z}_l \otimes \tilde{\mathbf{P}}_q \\ &= \mu_l \mathbf{Z}_l \otimes \tilde{\mathbf{P}}_q \\ &= \mu_l \mathbf{Z}_{l+1},\end{aligned}$$

where the third line results from the inductive assumption. Thus, $\mu_l = \mu_{l+1}$ is the eigenvalue of $\mathbf{A}_{l+1}$ associated with the columns of $\mathbf{Z}_{l+1}$.

(ii) ($l+1 \in U$ and $l+1 \notin V$). In this case, $U_{(l+1)} \not\subseteq V_{(l+1)}$ because the element $l+1$ is in $U_{(l+1)}$ but not $V_{(l+1)}$. Therefore, in this case $\mu_{l+1} = -1$, and we must prove the -1 is the eigenvalue of $\mathbf{A}_{l+1}$ associated with the columns of $\mathbf{Z}_{l+1}$. In this case, $\mathbf{Z}_{l+1} = \mathbf{Z}_l \otimes \tilde{\mathbf{P}}_q$. Also, in this case, $B_{l+1} = B_l \circ K(q)$, so by Equation (A.2),

$$\begin{aligned}\mathbf{A}_{l+1} &= \mathbf{A}_l \otimes \mathbf{J}_q + \mathbf{I} \otimes \mathbf{A}(K(q)) \\ &= \mathbf{A}_l \otimes \mathbf{J}_q + \mathbf{I} \otimes (\mathbf{J}_q - \mathbf{I}_q).\end{aligned}$$

Then we have,

$$\begin{aligned}\mathbf{A}_{l+1}\mathbf{Z}_{l+1} &= (\mathbf{A}_l \otimes \mathbf{J}_q + \mathbf{I} \otimes (\mathbf{J}_q - \mathbf{I}_q))(\mathbf{Z}_l \otimes \tilde{\mathbf{P}}_q) \\ &= \mathbf{A}_l\mathbf{Z}_l \otimes \mathbf{J}_q\tilde{\mathbf{P}}_q + \mathbf{Z}_l \otimes \mathbf{J}_q\tilde{\mathbf{P}}_q - \mathbf{Z}_l \otimes \tilde{\mathbf{P}}_q \\ &= \mathbf{A}_l\mathbf{Z}_l \otimes \mathbf{0}_q + \mathbf{Z}_l \otimes \mathbf{0}_q - \mathbf{Z}_l \otimes \tilde{\mathbf{P}}_q \\ &= -\mathbf{Z}_l \otimes \tilde{\mathbf{P}}_q \\ &= -\mathbf{Z}_{l+1}\end{aligned}$$

where the third line results from recognizing that each column of $\tilde{\mathbf{P}}_q$ sums to zero, so $\mathbf{J}_q\tilde{\mathbf{P}}_q = \mathbf{0}_q$, where $\mathbf{0}_q$ is the $q \times q$ matrix of all zeros. Thus, $\mu_{l+1} = -1$ is the eigenvalue of $\mathbf{A}_{l+1}$ associated with the columns of $\mathbf{Z}_{l+1}$.

(iii) ($l+1 \notin U$ and $l+1 \in V$). In this case, the element $l+1$ is in $V_{(l+1)}$ but not $U_{(l+1)}$. Therefore, if $U_{(l)} \subseteq V_{(l)}$, it will be true that $U_{(l+1)} \subseteq V_{(l+1)}$, and if $U_{(l)} \not\subseteq V_{(l)}$, then $U_{(l+1)} \not\subseteq V_{(l+1)}$. Additionally, in this case, $|V_{(l+1)}| = |V_{(l)}| + 1$, so, as in case (i), we have $\mu_{l+1} = \mu_l$, and we must prove the μ_l is the eigenvalue of $\mathbf{A}_{l+1}$ associated with the columns of $\mathbf{Z}_{l+1}$. In this case, $\mathbf{Z}_{l+1} = \mathbf{Z}_l \otimes \mathbf{1}_q$ and $\mathbf{A}_{l+1} = \mathbf{A}_l \otimes \mathbf{I}_q$ (as in case (i)). Then,

$$\begin{aligned}\mathbf{A}_{l+1}\mathbf{Z}_{l+1} &= (\mathbf{A}_l \otimes \mathbf{I}_q)(\mathbf{Z}_l \otimes \mathbf{1}_q) \\ &= \mathbf{A}_l\mathbf{Z}_l \otimes \mathbf{1}_q \\ &= \mu_l\mathbf{Z}_l \otimes \mathbf{1}_q \\ &= \mu_l\mathbf{Z}_{l+1},\end{aligned}$$

where the third line results from the inductive assumption. Thus, $\mu_l = \mu_{l+1}$ is the eigenvalue of $\mathbf{A}_{l+1}$ associated with the columns of $\mathbf{Z}_{l+1}$.

(iv) ($l+1 \notin U$ and $l+1 \notin V$). In this case, $V_{(l+1)} = V_{(l)}$ and $U_{(l+1)} = U_{(l)}$, so $I(U_{(l+1)} \subset V_{(l+1)}) = I(U_{(l)} \subseteq V_{(l)})$ and $|V_{(l+1)}| = |V_{(l)}|$. Therefore,

$$\begin{aligned}\mu_{l+1} &= q^{l+1-|V_{(l+1)}|}I(U_{(l+1)} \subseteq V_{(l+1)}) - 1 \\ &= q^{l+1-|V_{(l)}|}I(U_{(l)} \subseteq V_{(l)}) - 1 \\ &= q\left(q^{l-|V_{(l)}|}I(U_{(l)} \subseteq V_{(l)})\right) - 1 \\ &= q\left(q^{l-|V_{(l)}|}I(U_{(l)} \subseteq V_{(l)}) - 1\right) + q - 1 \\ &= q\mu_l + q - 1.\end{aligned}$$

Thus, we must prove that $q\mu_l + q - 1$ is the eigenvalue of $\mathbf{A}_{l+1}$ associated with the columns of $\mathbf{Z}_{l+1}$. In this case, $\mathbf{Z}_{l+1} = \mathbf{Z}_l \otimes \mathbf{1}_q$ (as in case (iii)) and $\mathbf{A}_{l+1} = \mathbf{A}_l \otimes \mathbf{J}_q + \mathbf{I} \otimes (\mathbf{J}_q - \mathbf{I}_q)$ (as in case (ii)). Then, we have

$$\begin{aligned}\mathbf{A}_{l+1}\mathbf{Z}_{l+1} &= \left(\mathbf{A}_l \otimes \mathbf{J}_q + \mathbf{I} \otimes (\mathbf{J}_q - \mathbf{I}_q)\right)(\mathbf{Z}_l \otimes \mathbf{1}_q) \\ &= \mathbf{A}_l\mathbf{Z}_l \otimes \mathbf{J}_q\mathbf{1}_q + \mathbf{Z}_l \otimes \mathbf{J}_q\mathbf{1}_q - \mathbf{Z}_l \otimes \mathbf{1}_q \\ &= \mu_l\mathbf{Z}_l \otimes q\mathbf{1}_q + \mathbf{Z}_l \otimes q\mathbf{1}_q - \mathbf{Z}_l \otimes \mathbf{1}_q \\ &= (q\mu_l + q - 1)\,\mathbf{Z}_l \otimes \mathbf{1}_q \\ &= (q\mu_l + q - 1)\,\mathbf{Z}_{l+1},\end{aligned}$$

where the third line results from the inductive assumption and recognizing that $\mathbf{J}_q\mathbf{1}_q = q\mathbf{1}_q$. Thus, $\mu_{l+1} = (q\mu_l + q - 1)$ is the eigenvalue of $\mathbf{A}_{l+1}$ associated with the columns of $\mathbf{Z}_{l+1}$.

We additionally have four analogous base cases for the induction: (i) $1 \in U$ and $1 \in V$, (ii) $1 \in U$ and $1 \notin V$, (iii) $1 \notin U$ and $1 \in V$, and (iv) $1 \notin U$ and $1 \notin V$:

(i) ($1 \in U$ and $1 \in V$). In this case, $U_{(1)} = V_{(1)} = \{1\}$, so $I(U_{(1)} \subseteq V_{(1)}) = 1$, $|V_{(1)}| = 1$, and therefore $\mu_1 = 0$. Additionally, $\mathbf{A}_1 = \mathbf{A}(O(q)) = \mathbf{0}_q$, so $\mathbf{A}_1\mathbf{Z}_1 = \mu_1\mathbf{Z}_1 = \mathbf{0}_q$.

(ii) ($1 \in U$ and $1 \notin V$). In this case, $V_{(1)} = \emptyset$, so $U_{(1)} \not\subseteq V_{(1)}$, $|V| = 0$, and $\mu_1 = -1$. Additionally, $\mathbf{Z}_1 = \tilde{\mathbf{P}}_q$ and $\mathbf{A}_1 = \mathbf{A}(K(q)) = \mathbf{J}_q - \mathbf{I}_q$, so we have $\mathbf{A}_1\mathbf{Z}_1 = \mathbf{J}_q\tilde{\mathbf{P}}_q - \tilde{\mathbf{P}}_q = -\tilde{\mathbf{P}}_q = -\mathbf{Z}_1$.

(iii) ($1 \notin U$ and $1 \in V$). In this case $U = \emptyset$ and $V = \{1\}$, so $U_{(1)} \subseteq V_{(1)}$, $|V| = 1$, and $\mu_1 = 0$. Since $\mathbf{A}_1 = \mathbf{A}(O(q)) = \mathbf{0}_q$, then $\mathbf{A}_1\mathbf{Z}_1 = \mu_1\mathbf{Z}_1 = \mathbf{0}$.

(iv) ($1 \notin U$ and $1 \notin V$). In this case, $U_{(1)} = V_{(1)} = \emptyset$, so $U_{(1)} \subseteq V_{(1)}$, $|V_{(1)}| = 0$ and thus $\mu_1 = q - 1$. Additionally, $\mathbf{Z}_1 = \mathbf{1}_q$ and $\mathbf{A}_1 = \mathbf{A}(K(q)) = \mathbf{J}_q - \mathbf{I}_q$, so we have

$$\begin{aligned}\mathbf{A}_1\mathbf{Z}_1 &= \mathbf{J}_q\mathbf{1}_q - \mathbf{1}_q \\ &= (q-1)\mathbf{1}_q \\ &= (q-1)\mathbf{Z}_1.\end{aligned}$$

Thus, μ_1 is the eigenvalue of $\mathbf{A}_1$ associated with the columns $\mathbf{Z}_1$ in each of these base cases.

By this induction, we have proved that μ_L is the eigenvalue of $\mathbf{A}_L$ associated with the columns of $\mathbf{Z}_L$. It is clear to see that $\mu_L = \mu(U, V)$. The result then follows from recognizing that, due to Lemma A.11, $B_L(V) = C(V)$ and therefore $\mathbf{A}_L = \mathbf{A}(V)$. Thus, from the induction, the columns of $\mathbf{Z}_L(U)$ are eigenvectors $A(V)$ associated with the eigenvalue $\mu_L = \mu(U, V)$. □

Lemma A.14. *Define $\mathbf{\Phi}_U^{(L,q)}$ as the matrix of $(q-1)^{|U|}$ columns of the Fourier basis $\mathbf{\Phi}^{(L,q)}$ corresponding to the epistatic interaction $U \subseteq \{1, 2, ..., L\}$:*

$$\mathbf{\Phi}_U^{(L,q)} \coloneqq \begin{bmatrix} — & \boldsymbol{\phi}_U^{(L)}(\mathbf{s}_1)^T & — \\ — & \boldsymbol{\phi}_U^{(L)}(\mathbf{s}_2)^T & — \\ & \vdots & \\ — & \boldsymbol{\phi}_U^{(L)}(\mathbf{s}_{q^L})^T & — \end{bmatrix} \tag{A.13}$$

where $\boldsymbol{\phi}_U^{(L)}(\mathbf{s}_i) \coloneqq \frac{1}{\sqrt{q^L}} \bigotimes_{j \in U} \mathbf{p}_q(s_{i,j})$ is the encoding of sequence $\mathbf{s}_i$ in terms of the epistatic interaction U in the Fourier basis[1]*. Then, $\frac{1}{\sqrt{q^L}}\mathbf{Z}_L(U) = \mathbf{\Phi}_U^{(L,q)}$, where $\mathbf{Z}_L(U)$ is defined by Equations* (A.10) *and* (A.11)*.*

Proof. Let $\hat{\boldsymbol{\phi}}_U^{(L)}(\mathbf{s}_i) \coloneqq \sqrt{q^L}\boldsymbol{\phi}_U^{(L)}(\mathbf{s}_i)$ for $i = 1, 2, ...q^L$ be the unnormalized rows of $\mathbf{\Phi}_U^{(L,q)}$. These can be defined recursively. In particular, we have

$$\hat{\boldsymbol{\phi}}_U^{(l+1)}(\mathbf{s}_i) = \begin{cases} \hat{\boldsymbol{\phi}}_U^{(l)}(\tilde{\mathbf{s}}_i) \otimes \mathbf{p}_q(s_{i,l+1}) & \text{if } l+1 \in U \\ \hat{\boldsymbol{\phi}}_U^{(l)}(\tilde{\mathbf{s}}_i) & \text{otherwise} \end{cases} \tag{A.14}$$

[1] $\boldsymbol{\phi}_U^{(L)}(\mathbf{s}_i)$ is equal to the $\boldsymbol{\phi}_U(\mathbf{s}_i)$ defined in the main text, but we have added the (L) index.

for $l = 1, 2, ..., L$, where $\tilde{\mathbf{s}}_i$ are the first l positions of $\mathbf{s}_i$, and

$$\hat{\boldsymbol{\phi}}_U^{(1)}(s_{i,1}) = \begin{cases} \mathbf{p}_q(s_{i,1}) & \text{if } 1 \in U \\ 1 & \text{otherwise} \end{cases} \tag{A.15}$$

For a given $\tilde{\mathbf{s}} \in \mathcal{S}^{(l,q)}$, there are q sequences $\mathbf{s} \in \mathcal{S}^{(l+1,q)}$ whose first l positions are $\tilde{\mathbf{s}}$. Further, each of these sequences has a unique element in the final position. Thus, in order to construct the sub-matrix of $\boldsymbol{\Phi}_U^{(l+1,q)}$ corresponding to a $\tilde{\mathbf{s}}$,

$$\begin{bmatrix} — \hat{\boldsymbol{\phi}}_U^{(l+1)}([\tilde{\mathbf{s}}, 1])^T — \\ — \hat{\boldsymbol{\phi}}_U^{(l+1)}([\tilde{\mathbf{s}}, 2])^T — \\ \vdots \\ — \hat{\boldsymbol{\phi}}_U^{(l+1)}([\tilde{\mathbf{s}}, q])^T — \end{bmatrix} = \begin{cases} \hat{\boldsymbol{\phi}}_U^{(l)}(\tilde{\mathbf{s}}) \otimes \tilde{\mathbf{P}}_q & \text{if } l+1 \in U \\ \hat{\boldsymbol{\phi}}_U^{(l)}(\tilde{\mathbf{s}}) \otimes \mathbf{1}_q & \text{otherwise.} \end{cases} \tag{A.16}$$

Now let, $\hat{\boldsymbol{\Phi}}_U^{(l,q)} \coloneqq \sqrt{q^l}\boldsymbol{\Phi}_U^{(l,q)}$. Applying Equation (A.16) to each row in $\hat{\boldsymbol{\Phi}}_U^{(l,q)}$ results in:

$$\hat{\boldsymbol{\Phi}}_U^{(l+1,q)} = \begin{cases} \hat{\boldsymbol{\Phi}}_U^{(l,q)} \otimes \tilde{\mathbf{P}}_q & \text{if } l+1 \in U \\ \hat{\boldsymbol{\Phi}}_U^{(l,q)} \otimes \mathbf{1}_q & \text{otherwise.} \end{cases} \tag{A.17}$$

which is equivalent to the recursion in Equation (A.10) that defines $\mathbf{Z}_l(U)$. Additionally, repeated application of Equation (A.15) to each element in the alphabet results in the equivalent base case to Equation (A.11). Carrying out the recursion of Equation (A.16) to $l = L$ then gives $\mathbf{Z}_L(U) = \hat{\boldsymbol{\Phi}}_U^{(L,q)} = \sqrt{q^L}\boldsymbol{\Phi}_U^{(L,q)}$. □

We are finally prepared to prove Theorem 4.1.

Proof of Theorem 4.1. In order to prove this theorem, we need to show (i) that the Fourier coefficients are normally distributed with zero mean and diagonal covariance and (ii) that the variance of the coefficients corresponding to a particular epistatic interaction are given by Equation (4.12).

First, Lemma A.6 proves that the Fourier coefficients are normally distributed with zero mean. Next, Lemma A.12 proves that the Fourier basis $\boldsymbol{\Phi}^{(L,q)}$ diagonalizes the adjacency matrix $\mathbf{A}^{[j]}$ of the clique multigraph of the GNK hypergraph $G^{[j]}$. Recalling that $G^{[j]}$ is a 1-regular hypergraph, then by Lemma A.8, $\mathbf{F}^{[j]}(\mathbf{F}^{[j]})^T = \mathbf{A}^{[j]} + \mathbf{I}$. Therefore, the Fourier basis diagonalizes $\mathbf{F}^{[j]}(\mathbf{F}^{[j]})^T$ for all $j = 1, 2, ..., L$, and thus also diagonalizes $\mathbf{F}\mathbf{F}^T$ due to Lemma A.7. Then, the covariance matrix of the Fourier coefficients, which is shown in Lemma A.6 to be $\langle \boldsymbol{\beta}\boldsymbol{\beta}^T \rangle = \left(\boldsymbol{\Phi}^{(L,q)}\right)^T \mathbf{F}\mathbf{F}^T \boldsymbol{\Phi}^{(L,q)}$, is diagonal. The eigenvalues of $\mathbf{F}\mathbf{F}^T$ are then equal to the variances of the Fourier coefficients.

Lemma A.13 shows that the columns of the matrix $\mathbf{Z}_L(U)$ defined by Equations (A.10) and (A.11) are eigenvectors of $\mathbf{A}^{[j]}$. Further, Lemma A.14 shows that this matrix is equal to the columns of the Fourier basis corresponding to the epistatic interaction U, $\boldsymbol{\Phi}_U^{(L,q)}$. Thus,

Lemma A.13 shows that the eigenvalue of $\mathbf{A}^{[j]}$ associated with the columns $\mathbf{\Phi}_U^{(L,q)}$ is given by:

$$\mu(U, V^{[j]}) = q^{L-K_j} I(U \subseteq V^{[j]}) - 1.$$

By Lemma A.8, the eigenvalues of $\mathbf{F}^{[j]}(\mathbf{F}^{[j]})^T$ are simply one plus those calculated with (A.12). Since the Fourier basis diagonalizes all $\mathbf{F}^{[j]}(\mathbf{F}^{[j]})^T$, the eigenvalues of $\mathbf{F}\mathbf{F}^T$ are simply the sum of those of the $\mathbf{F}^{[j]}(\mathbf{F}^{[j]})^T$. The eigenvalues of $\mathbf{F}\mathbf{F}^T$ associated with the eigenvectors given by the columns of $\mathbf{\Phi}_U^{(L,q)}$ are the variances of the Fourier coefficients corresponding to the epistatic interaction U. All of this together, we have:

$$\begin{aligned}
\langle \boldsymbol{\beta}_U \boldsymbol{\beta}_U^T \rangle &= \left(\mathbf{\Phi}_U^{(L,q)}\right)^T \mathbf{F}\mathbf{F}^T \mathbf{\Phi}_U^{(L,q)} \\
&= \sum_{j=1}^{L} \left(\mathbf{\Phi}_U^{(L,q)}\right)^T \mathbf{F}^{[j]}(\mathbf{F}^{[j]})^T \mathbf{\Phi}_U^{(L,q)} \\
&= \sum_{j=1}^{L} (\mu(U, V^{[j]}) + 1)\mathbf{I} \\
&= \sum_{j=1}^{L} \left(q^{L-K_j} I(U \subseteq V^{[j]})\right) \mathbf{I},
\end{aligned}$$

which is the desired result for the variances of the Fourier coefficients. □

Proof of Theorem 4.2. . Theorem 4.1 shows that all the Fourier coefficients associated with an epistatic interaction U are deterministically zero if $U \not\subseteq V^{[j]}$ for $j = 1, 2, ..., L$, which can be alternatively stated as $U \notin \mathscr{P}(V^{[j]})$ for $j = 1, 2, ..., L$. The epistatic interactions with non-zero Fourier coefficients are the $U \in \mathcal{U} \coloneqq \mathscr{P}(\{1, ..., L\})$ for which there exists a $j \in 1, 2, ..., L$ such that $U \in \mathscr{P}(V^{[j]})$. Since $\mathscr{P}(V^{[j]}) \subseteq \mathcal{U}$, we have

$$\{U \in \mathcal{U} : U \in \mathscr{P}(V^{[j]})\} = \mathscr{P}(V^{[j]})$$

and further,

$$\begin{aligned}
\{U \in \mathcal{U} : \exists j \in \{1, 2, ..., L\} \text{ s.t. } U \in \mathscr{P}(V^{[j]})\} &= \bigcup_{j=1}^{L} \{U \in \mathcal{U} : U \in \mathscr{P}(V^{[j]})\} \\
&= \bigcup_{j=1}^{L} \mathscr{P}(V^{[j]}).
\end{aligned}$$

There are $(q-1)^{|U|}$ Fourier coefficients associated with each $U \in \mathcal{U}$, so letting $\mathcal{T}(\mathcal{V}) \coloneqq \bigcup_{j=1}^{L} \mathscr{P}(V^{[j]})$, we have

$$\text{supp}(\boldsymbol{\beta}) \geq \sum_{U \in \mathcal{T}(\mathcal{V})} (q-1)^{|U|}$$

where the bound results from recognizing that the RHS sums overs all Fourier coefficients that are *deterministically* zero, but the coefficients with nonzero variances may still equal zero. However, recognizing that each $\beta \in \boldsymbol{\beta}$ with nonzero variance is a normal random variable that can equal zero with zero probability, we have

$$\text{supp}(\boldsymbol{\beta}) = \sum_{U \in \mathcal{T}(\mathcal{V})} (q-1)^{|U|}$$

almost surely. □

The sparsity of GNK fitness functions with standard neighborhood schemes

Proof of Proposition 4.3. Let $\mathcal{W}_r^{[j]} \coloneqq \{W \in \mathscr{P}(V^{[j]}) : |W| = r\}$ be the number of elements of the powerset of neighborhood j with cardinality r. Additionally define

$$\begin{aligned} n(r) &\coloneqq \left| \bigcup_{j=1}^{L} \mathcal{W}_r^{[j]} \right| \\ &= \#\{W \in \mathcal{T}(\mathcal{V}) : |W| = r\} \end{aligned}$$

as the number of elements in the union of powersets with cardinality r. For any set of neighborhoods, we have $n(0) = 1$ and $n(1) = L$. Additionally, for any $\mathcal{V}_K$ with uniform neighborhood size K, $n(r) = 0$ for $r > K$. Then, for $r = 2, 3..., K$, we have

$$\begin{aligned} n(r) &= \left| \bigcup_{j=1}^{L} \mathcal{W}_r^{[j]} \right| \\ &\leq \sum_{j=1}^{L} |\mathcal{W}_r^{[j]}| \\ &= \sum_{j=1}^{L} \binom{K}{r} \\ &= L\binom{K}{r} \end{aligned}$$

where the second line results from the union bound and the third from recognizing that there are $\binom{K}{r}$ sets of cardinality r in the powerset of a set with K elements. Using this within

Theorem 4.2 we then have

$$\begin{aligned} S(f) &= \sum_{U \in \mathcal{T}(\mathcal{V}_K)} (q-1)^{|U|} \\ &= \sum_{r=0}^{L} n(r)(q-1)^r \\ &\leq 1 + L(q-1) + L \sum_{r=0}^{K} \binom{K}{r} (q-1)^r. \end{aligned}$$

□

Now we formally present results for the sparsity of GNK models with neighborhoods constructed using the standard schemes.

Proof of Proposition 4.4. There are $\frac{L}{K}$ blocks. The blocks are fully connected, so all $\sum_{r=0}^{K} \binom{K}{r}(q-1)^r = q^K$ Fourier coefficients corresponding to intra-block epistatic interactions are nonzero. The only epistatic interaction shared by the blocks is the zeroth order interaction, so each block contributes $(q^K - 1)$ unique nonzero Fourier coefficients, and the total number of nonzero Fourier coefficients is given by Equation (4.15), where the final addition of one is due to the shared zeroth order interaction. □

Proof of Proposition 4.5. Define $\mathcal{W}_r^{[j]} \coloneqq \{W \in \mathscr{P}(V^{[j]}) : |W| = r\}$ and

$$n_l(r) \coloneqq \left| \bigcup_{j=1}^{l} \mathcal{W}_r^{[j]} \right|$$

for $l = 1, 2, ..., L$. For $l \leq L - K + 1$, and $r = 1, 2, ..., K$, we have

$$n_l(r) = l\binom{K}{r} - (l-1)\binom{K-1}{r}. \tag{A.18}$$

This can be shown by induction. In particular, assume Equation (A.18) is correct for $l <$

$L-K+1$ and then we find:

$$\begin{aligned}
n_{l+1}(r) &= \left|\bigcup_{j=1}^{l+1} \mathcal{W}_r^{[j]}\right| \\
&= \left|\left(\bigcup_{j=1}^{l} \mathcal{W}_r^{[j]}\right) \cup \mathcal{W}_r^{[l+1]}\right| \\
&= \left|\bigcup_{j=1}^{l} \mathcal{W}_r^{[j]}\right| + \left|\mathcal{W}_r^{[l+1]}\right| - \left|\left(\bigcup_{j=1}^{l} \mathcal{W}_r^{[j]}\right) \cap \mathcal{W}_r^{[l+1]}\right| \\
&= n_l(r) + \binom{K}{r} - \binom{K-1}{r} \\
&= (l+1)\binom{K}{r} - l\binom{K-1}{r}.
\end{aligned}$$

where the fourth line results from recognizing that $V^{[l+1]}$ when $l+1 \leq L-K+1$ contains exactly one position that is not in $\bigcup_{j=1}^{l} V^{[j]}$; there are then $\binom{K-1}{r-1}$ sets in $\mathcal{W}_r^{[l+1]}$ that contain this element and are thus unique to $\mathcal{W}_r^{[l+1]}$, which leads to

$$\begin{aligned}
\left|\left(\bigcup_{j=1}^{l} \mathcal{W}_r^{[j]}\right) \cap \mathcal{W}_r^{[l+1]}\right| &= \binom{K}{r} - \binom{K-1}{r-1} \\
&= \binom{K-1}{r}.
\end{aligned}$$

It is clear that $n_1(r) = \binom{K}{r}$, and thus Equation (A.18) is proved by induction for $l \leq L-K+1$.

Equation (A.18) accounts for redundancies in $\mathcal{W}_r^{[j]}$ that result from overlapping positions in the neighborhoods, without considering periodicity. There are additional redundancies that occur when $l > L-K+1$ due to the periodicity of the neighborhoods. In particular, for $l = L-K+2, ..., L$, due to periodicity $V^{[l]}$ contains $(l+k) \bmod L-1$ additional positions that are already in $\bigcup_{j=1}^{l} V^{[j]}$ (outside of those that are already $\bigcup_{j=1}^{l} V^{[j]}$ due to non-periodic overlap). Therefore $\mathcal{W}_r^{[l]}$ contains $\binom{(l+k) \bmod L-1}{r}$ additional sets that are already in $\bigcup_{j=1}^{l-1} \mathcal{W}_r^{[j]}$ due to periodicity. Then we have, for $l = L-K+2, ..., L$:

$$n_l(r) = l\binom{K}{r} - (l-1)\binom{K-1}{r} - \binom{(l+k) \bmod L-1}{r}.$$

At $l = L$, we then have

$$\begin{aligned} n_L(r) &= L\binom{K}{r} - (L-1)\binom{K-1}{r} - \binom{(L+k) \bmod L - 1}{r} \\ &= L\binom{K}{r} - (L-1)\binom{K-1}{r} - \binom{K-1}{r} \\ &= L\left(\binom{K}{r} - \binom{K-1}{r}\right) \\ &= L\binom{K-1}{r-1}. \end{aligned}$$

The result follows from recognizing that $n_L(0) = 1$, and therefore

$$\begin{aligned} S(f) &= \sum_{U \in \mathcal{T}(\mathcal{V}_{AN})} (q-1)^{|U|} \\ &= \sum_{r=0}^{L} n_L(r)(q-1)^r \\ &= 1 + +L\sum_{r=1}^{K} \binom{K-1}{r-1}(q-1)^r. \end{aligned}$$

□

Additionally, we are able to calculate the expected sparsity of GNK fitness functions with Random Neighborhoods, which is shown in the following result. The proof of this follows the analogous calculations of Nowak and Krug [120], and we correct a mistake in their calculations.

Proof of Proposition 4.6. Consider a set $W \subseteq \{1, 2, ..., L\}$ of cardinality r. Define $\alpha(r)$ as the the probability that W is a subset of the random neighborhood $V^{[j]}$ given that $j \in W$, which is given by

$$\begin{aligned} \alpha(r) &\coloneqq \Pr(W \subseteq V^{[j]} | j \in W) \\ &= \frac{\binom{L-r}{K-r}}{\binom{L-1}{K-1}} \\ &= \frac{(L-r)!}{(K-r)!}\frac{(K-1)!}{(L-1)!} \end{aligned}$$

where $\binom{L-1}{K-1}$ is the total number of ways to construct $V^{[j]}$ and $\binom{L-r}{K-r}$ is the number of ways to construct $V^{[j]}$ such that every element of W is in $V^{[j]}$. The probability that W is a subset

of the random neighborhood $V^{[j]}$ given that $j \notin W$ is similarly given by:

$$\begin{aligned}\Pr(W \subseteq V^{[j]} | j \notin W) &= \frac{\binom{L-r-1}{K-r-1}}{\binom{L-1}{K-1}} \\ &= \frac{(L-r-1)!}{(K-r-1)!}\frac{(K-1)!}{(L-1)!} \\ &= \alpha(r)\frac{K-r}{L-r}\end{aligned}$$

There are r neighborhoods $V^{[j]}$ for which $j \in W$, and $L-r$ neighborhoods for which $j \notin W$. Define $p(r)$ as the probability that W is a subset of at least one, which is then:

$$\begin{aligned}p(r) &\coloneqq \Pr(\exists j : W \subseteq V^{[j]}) \\ &= 1 - \Pr(\nexists j : W \subseteq V^{[j]}) \\ &= 1 - \Pr(\nexists j \in W : W \subseteq V^{[j]})\Pr(\nexists j \notin W : W \subseteq V^{[j]}) \\ &= 1 - (1-\alpha(r))^r \left(1 - \alpha(r)\frac{K-r}{L-r}\right)^{L-r}\end{aligned}$$

There are $\binom{L}{r}$ sets of cardinality r, and in expectation $\binom{L}{r}p(r)$ will be subsets of at least one neighborhood, and will therefore represent epistatic interactions or order r corresponding to $(q-1)^r$ nonzero Fourier coefficients. Equation (4.17) follows from summing over all possible cardinalities r. □

Appendix B

Further details of maximum entropy library design techniques

In this appendix, we provide derivations of some of the quantities calculated in Chapter 5.

Maximum entropy degenerate library gradients

Here we derive the gradients used to optimize the library parameters in Section 5.5, the components of which are shown in Equation 5.18. First, we recognize that the gradient of the entropy is given by

$$\begin{aligned}
\nabla_{\boldsymbol{\phi}} H[q_{\boldsymbol{\phi}}] &= -\nabla_{\boldsymbol{\phi}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{x})}[\log q_{\boldsymbol{\phi}}(\mathbf{x})] \\
&= -\sum_{\mathbf{x}\in\mathcal{X}} \nabla_{\boldsymbol{\phi}} q_{\boldsymbol{\phi}}(\mathbf{x}) \log q_{\boldsymbol{\phi}}(\mathbf{x}) \\
&= -\sum_{\mathbf{x}\in\mathcal{X}} (\log q_{\boldsymbol{\phi}}(\mathbf{x}) \nabla_{\boldsymbol{\phi}} q_{\boldsymbol{\phi}}(\mathbf{x}) + q_{\boldsymbol{\phi}}(\mathbf{x}) \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\mathbf{x})) \\
&= -\sum_{\mathbf{x}\in\mathcal{X}} (\log q_{\boldsymbol{\phi}}(\mathbf{x}) q_{\boldsymbol{\phi}}(\mathbf{x}) \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\mathbf{x}) + q_{\boldsymbol{\phi}}(\mathbf{x}) \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\mathbf{x})) \\
&= -\sum_{\mathbf{x}\in\mathcal{X}} q_{\boldsymbol{\phi}}(\mathbf{x})(1 + \log q_{\boldsymbol{\phi}}(\mathbf{x})) \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\mathbf{x}) \\
&= -\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{x})}[(1 + \log q_{\boldsymbol{\phi}}(\mathbf{x})) \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\mathbf{x})].
\end{aligned}$$

where in the third line we have used the equality $\nabla_{\boldsymbol{\phi}} q_{\boldsymbol{\phi}}(\mathbf{x}) = q_{\boldsymbol{\phi}}(\mathbf{x}) \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\mathbf{x})$.

We then have

$$\begin{aligned}\nabla_{\boldsymbol{\phi}} F(\boldsymbol{\phi}) &= \nabla_{\boldsymbol{\phi}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{x})}[f(\mathbf{x})] + \lambda \nabla_{\boldsymbol{\phi}} H[q_{\boldsymbol{\phi}}] \\ &= \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{x})}[f(\mathbf{x}) \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\mathbf{x})] - \lambda \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{x})}[(1 + \log q_{\boldsymbol{\phi}}(\mathbf{x})) \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\mathbf{x})] \\ &= \lambda \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{x})}[(f(\mathbf{x}) - \lambda(1 + \log q_{\boldsymbol{\phi}}(\mathbf{x}))) \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\mathbf{x})] \\ &= \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{x})}[w(\mathbf{x}) \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\mathbf{x})], \end{aligned} \tag{B.1}$$

where $w(\mathbf{x}) \coloneqq f(\mathbf{x}) - \lambda(1 + \log q_{\boldsymbol{\phi}}(\mathbf{x}))$. The individual components of $\nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\mathbf{x})$ are given by:

$$\begin{aligned}\frac{\partial}{\partial \phi_{jk}} \log q_{\boldsymbol{\phi}}(\mathbf{x}) &= \frac{\partial}{\partial \phi_{jk}} \log q_{\boldsymbol{\phi}_j}(x^j) \\ &= \frac{\partial}{\partial \phi_{jk}} \log \frac{e^{\phi_{j,x^j}}}{\sum_{l=1}^{K} e^{\phi_{jl}}} \\ &= \frac{\partial}{\partial \phi_{jk}} \phi_{j,x^j} - \frac{\partial}{\partial \phi_{jk}} \log \sum_{l=1}^{K} e^{\phi_{jl}} \\ &= \delta_k(x^j) - \frac{1}{\sum_{l=1}^{K} e^{\phi_{jl}}} \frac{\partial}{\partial \phi_{jk}} \sum_{l=1}^{K} e^{\phi_{jl}} \\ &= \delta_k(x^j) - \frac{e^{\phi_{jk}}}{\sum_{l=1}^{K} e^{\phi_{jl}}} e^{\phi_{jl}} \\ &= \delta_k(x^j) - q_{\boldsymbol{\phi}_j}(k). \end{aligned} \tag{B.2}$$

Using Equation (B.2) within Equation (B.1) gives the result of Equation (5.18) in the main text.

Expected Pairwise Distance

Here we derive the Expected Pairwise Distance between pairs of sequences sampled from a library design. In particular, consider a degenerate library design $q_{\boldsymbol{\phi}}(\mathbf{x})$ for sequences of length L and alphabet size K. The Hamming distance between two sequences, $\mathbf{x}_1$ and $\mathbf{x}_2$, is $d(\mathbf{x}_1, \mathbf{x}_2) = L - \sum_{i=1}^{L} \delta(x_1^j, x_2^j)$, where $\delta(x_1^j, x_2^j)$ is equal to one if $x_j^1 = x_2^j$, and zero otherwise.

The expected distance between two sequences sampled from $q_{\boldsymbol{\phi}}(\mathbf{x})$ is then:

$$\begin{aligned}
EPD(\boldsymbol{\phi}) &= \mathbb{E}_{\mathbf{x}_1 \sim q_{\boldsymbol{\phi}}(\mathbf{x}), x_2 \sim q_{\boldsymbol{\phi}}(\mathbf{x})}[d(\mathbf{x}_1, \mathbf{x}_2)] \\
&= L - \mathbb{E}_{\mathbf{x}_1 \sim q_{\boldsymbol{\phi}}(\mathbf{x}), x_2 \sim q_{\boldsymbol{\phi}}(\mathbf{x})} \left[\sum_{i=1}^{L} \delta(x_1^i, x_2^i) \right] \\
&= L - \sum_{i=1}^{L} \mathbb{E}_{x_1^i \sim q_{\boldsymbol{\phi}_i}(x^i), x_2^i \sim q_{\boldsymbol{\phi}_i}(x^i)} \left[\delta(x_1^i, x_2^i) \right] \\
&= L - \sum_{i=1}^{L} \sum_{j=1}^{K} \sum_{l=1}^{K} q_{\boldsymbol{\phi}_i}(j) q_{\boldsymbol{\phi}_i}(l) \delta(j, l) \\
&= L - \sum_{i=1}^{L} \sum_{j=1}^{K} q_{\boldsymbol{\phi}_i}(j)^2,
\end{aligned}$$

which is the desired result that is used in Equation (5.21).