Toward Trustworthy Scientific Inquiry and Design with Machine Learning

Clara Wong-Fannjiang

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2023-191
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2023/EECS-2023-191.html

June 20, 2023

Toward Trustworthy Scientific Inquiry and Design with Machine Learning

By

Clara Wong-Fannjiang

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering – Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Michael I. Jordan, Co-chair
Professor Jennifer Listgarten, Co-chair
Professor David V. Schaffer

Summer 2023

Toward Trustworthy Scientific Inquiry and Design with Machine Learning

Abstract

Toward Trustworthy Scientific Inquiry and Design with Machine Learning

by

Clara Wong-Fannjiang

Doctor of Philosophy in Engineering – Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Michael I. Jordan, Co-chair

Professor Jennifer Listgarten, Co-chair

The last decade has witnessed rapid development and deployment of machine-learning systems across science. Such systems can supply predictions about scientific phenomena far more quickly and cheaply than gold-standard experiments, and are being used in efforts to both discover scientific knowledge and design new biomolecules. However, an important question remains unanswered: since machine-learning systems make errors, how can we use them in a trustworthy way for scientific discovery and design? This dissertation takes steps toward helping to ensure that the biomolecules we design and the scientific conclusions we draw using machine learning can be trusted.

We begin in the setting of machine learning-based design. The goal in this setting is to propose novel objects such as proteins, small molecules, or materials with desired properties, in a way that is guided by machine-learning models of such properties. Toward addressing model trustworthiness for design, we propose (i) a method for learning models that accounts for the distribution shifts inherent to design, and (ii) a method for constructing statistically valid confidence sets for the properties of objects designed using machine learning.

Finally, we examine the trustworthy use of machine learning for drawing scientific conclusions. In particular, we consider the increasingly relevant setting of treating predictions made by machine-learning systems as “data” in estimating quantities of scientific interest. We propose *prediction-powered inference*, a novel statistical framework for constructing valid confidence sets in this setting, which enables researchers to incorporate evidence from machine-learning systems into their scientific inquiry in a standardized and principled way.

In loving memory of Pei-rong Wang,
who hoped his children's children could chase dreams.

Test all things; hold fast what is good.
– 1 Thessalonians 5:21

Contents

List of Figures

List of Tables

Acknowledgments

When I first arrived at Berkeley, I was not entirely certain I was capable of finishing this degree. It is only by undeserved grace that I have, most clearly manifest in the extraordinary mentors, colleagues, and friends I was so fortunate to meet along the way, without whom many of the ideas I spent time on never would have come to fruition. I am struck by their brilliance, but more importantly, their kindness.

I must first thank my co-advisors, Michael I. Jordan and Jennifer Listgarten, for taking a chance on me. I had no formal exposure to statistics before coming to Berkeley, and my tastes and style of research were deeply shaped by Mike's teaching and mentorship. Mike's endless commitment to learning—not just producing research—continues to inspire me, as does his contagious optimism. I benefited greatly from Jenn's foresight into high-impact problems at the frontiers of biotechnology. Jenn guided me through the process of distilling practical challenges in scientific applications into clean technical problems—ideally, that appeal to both practitioners and methods researchers—an art I will keep trying to grow in for the rest of my career. The synergy of ideas that arose from working in both their groups enabled me to keep a foot in both methodological innovation and scientific applications. I am deeply grateful to both of them, not least of all for their confidence in me.

I am also grateful to David Schaffer and Yun S. Song for their contributions as members of my qualifying exam committee, and furthermore to David for also serving on my dissertation committee. The directions of my research during this last year of graduate school, and inevitably going forward, were shaped by their insights regarding the role of epistasis in protein evolution. I am also grateful to have been able to vet some harebrained methodological ideas by trying them out on high-throughput AAV packaging data that David and his lab generously provided. I would also like to thank Will Fithian, as I benefitted immensely from his wonderful theoretical statistics lectures, as well as Ben Recht and Jacob Steinhardt for their insightful lectures on optimization and robust statistics, respectively.

I consider myself incredibly lucky to have spent the summer of 2022 interning at Salesforce Research with Ali Madani and Nikhil Naik, on a formative collaboration with Polly Fordyce, James S. Fraser, Eric R. Greene, Craig Markin, Micah Olivas, and Margaux Pinney. Ali and Nikhil were invaluable mentors and helped me navigate the challenges of incorporating wet-lab experimental constraints into practical machine-learning solutions. I had the privilege of benefiting from the collective expertise of Polly, Jaime, Eric, Craig, Micah, and Margaux, whose backgrounds in biophysics, structural biology, enzymology, and assay development gave me invaluable insights into enzyme design. I am humbled to have been able to work closely with Eric and Micah, whose patience and thorough responses to my naive questions have been immensely enriching. Finally, I would also like to extend my appreciation to Jaime for his career advice and encouragement.

None of the ideas in this dissertation would exist without the extraordinary colleagues I was so lucky to work with. During my time at Berkeley I was fortunate to collaborate with Anastasios N. Angelopoulos, David H. Brookes, Stephen Bates, Akosua Busia, Polly Fordyce, James S. Fraser, Eric R. Greene, Chloe Hsu, Ghassen Jerfel, Michael I. Jordan, Jennifer

Listgarten, Ali Madani, Nikhil Naik, Hunter Nisonoff, Micah Olivas, Margaux Pinney, Serena L. Wang, and Tijana Zrnic. I have learned so much from each of you.

I am also grateful for the hours of brainstorming and freewheeling research conversations with Amirali Aghazadeh, Carlos Albors, Alan Aw, Frances Ding, Mariano Gabitto, Milind Jagota, Hanlun Jiang, Karl Krauth, Lydia T. Liu, Yian Ma, Sebastián Prillo, Nilesh Tripuraneni, Yixin Wang, Bear Xiong, and Bonnie Zhu. Thank you to the intrepid Mrunali Manjrekar and Renzo Soatto for taking a chance on some crazy ideas with me, and to Brian Hie, Kadina Johnston, Bruce J. Wittmann, Zachary Wu, and Kevin K. Yang for giving me the opportunity to work on a book chapter together.

I wouldn't have made it through the first two years without the technical and emotional companionship of Akosua Busia, Wenshuo Guo, Lydia Liu, Juanky Perdomo, Serena Wang, and Vickie Ye. Thank you also to Nilesh, who took it upon himself to be our preliminary exam guide, and to Kelvin Xu for the coffee breaks, spicy perspectives, and invaluable job-search advice. I owe each of you some random number of Acme baguettes.

I am eternally grateful to my husband, Ivan, for his support and endless sacrifice to make every step of this degree possible. There are no words to describe my debt to my parents, Jean and Albert, who put their lives on hold to support us as new parents in the middle of grad school and the pandemic. You made the way for me. Thank you.

Chapter 1

Introduction

Machine-learning systems are increasingly being used as inexpensive, high-throughput proxies for gold-standard scientific experiments. In the design of proteins, small molecules, and materials, for example, predictions made by machine-learning models are used to inform the design of objects with desired novel properties. In scientific inquiry, predictions are being used to draw scientific conclusions, in domains ranging from proteomics [127, 145] and genomics [77, 139] to the neurobiology of aging [142] and anthropogenic effects on food web complexity [132]. Though the predictive power of modern machine-learning systems gives us reason to be optimistic about these developments, an uncomfortable truth remains: predictions contain errors. Obtaining predictions from a machine-learning system is simply not the same as obtaining data from a gold-standard experiment. How, then, can one leverage such predictions to make trustworthy decisions—both in designing novel objects, and in drawing scientific conclusions? This dissertation takes a small step toward tackling this question.

The unifying theme and title of this work did not emerge until my last year in graduate school, as is perhaps true of many dissertations. Without a doubt I have found the most challenging—and rewarding—aspect of graduate school, beyond the technical training, to be the process of discovering and owning one's personal style in research. As I have always been intrigued by the origin stories of creative work, I wrote this introduction as an abridged version of how the three chapters of this dissertation—which are based on works co-authored with Anastasios N. Angelopoulos, Stephen Bates, Michael I. Jordan, Jennifer Listgarten, and Tijana Zrnic [92, 131, 144]—came to be. The ideas presented here are as much theirs as they are mine.

Chapters 1 & 2: Machine learning-based design

When I started graduate school in the fall of 2018, it was evident that the use of machine-learning systems in science was here to stay—not just for making predictions, but for drawing conclusions and make consequential decisions using those predictions. I had the vague thought that I wanted to study the unique problems that arise from this development, but

I needed a concrete instantiation in which to ground my thinking. A year of conversations with one of my co-advisors, Jennifer Listgarten, convinced me that machine learning-based protein design was a compelling setting. It didn't hurt that the same fall, Frances Arnold won the Nobel Prize in Chemistry for pioneering directed evolution, an approach that mimics natural selection in the laboratory in order to design enzymes and other proteins with enhanced functionality. Her group had also begun leveraging machine-learning methods to further facilitate the design of proteins with enhanced properties [41, 75, 85].

In parallel, the use of machine learning for design—the specification of novel objects with desired properties—was emerging in a number of other fields, including the design of small molecules [63, 68, 73] and materials [72]. The aspiration in these settings is often to design novel, desirable objects given a single batch of data. For example, given a fixed data set—say, of different enzyme sequences paired with experimental measurements of their catalytic activity on a certain reaction—can one propose novel enzyme sequences that are even more catalytically active, without collecting further data? One approach for attempting this is to fit a predictive model to the data, and then run some *design algorithm* that consults that model in order to propose objects believed to have the desired properties [63, 68, 72, 73, 85, 110, 122]. The first two chapters of this dissertation tackle questions that emerge with this approach. For concreteness, these are described next in the context of protein design, but the ideas apply broadly to machine learning-based design in other domains.

How should one learn a predictive model for design?

Since the goal of protein design is to find novel properties unobserved in known proteins, designed proteins must come from a different distribution than the training proteins—what we call the *design distribution*. Though distribution shifts arise in many machine learning settings, what is unique about the design setting is that the shift is purposely induced rather than passively observed; we revisit this idiosyncrasy in the next chapter. Due to this shift, the model's predictions must be trustworthy over regions of protein space "away from" the training proteins—in particular, regions characterized by the design distribution, as well as regions the design algorithm examines en route to deciding the design distribution. If one knew the design distribution in advance, then one could deploy strategies for fitting the model to be accurate over the designed rather than the training proteins [12, 27, 37–39]. The chicken-and-egg dilemma, however, is that one does not know the distribution of designed proteins until after the predictive model has been learned, as the latter dictates the former. Nevertheless, can one try to anticipate and account for plausibly relevant distribution shifts?

Many design algorithms move through protein space in an iterative fashion to search for promising proteins, where each move induces, either implicitly or explicitly, an intermediary distribution of proteins currently under consideration [90]. In the first chapter, we describe a strategy for relearning the predictive model in lockstep with such design algorithms—without access to new data—such that it is more accurate over each intermediary distribution [92]. We first formalize machine learning-based design as a game between the predictive model and the design algorithm, rather than as an optimization problem, which naturally gives rise to this strategy.

How can one quantify predictive uncertainty in design?

Although I believe there are significant, underexplored benefits to addressing distributional shift in machine learning-based protein design, our work [92] felt unsatisfying for a few reasons. One, which would mark a turning point in my research, was the realization that far more effort had been invested in developing machine-learning methods to try to design proteins more effectively, than in assessing how likely any such strategy was to be effective. Given how costly—in terms of time, money, resources, and personnel—the synthesis and experimental characterization of designed proteins can be, it seemed the latter warranted more attention. In particular, the pursuit of novelty requires any design algorithm to consider regions of sequence space away from the training data, but these regions are precisely where any learned model is least trustworthy. Given this dilemma, when one uses a machine-learning model to design novel proteins, how can one trust the model's predictions for the designed proteins?

Around this time in 2020, my co-advisor, Michael Jordan, directed his reading group to work through a short series of papers on *conformal prediction*, an approach for predictive uncertainty quantification under a frequentist lens [23, 143]. Conformal prediction methods produce confidence sets for test points that satisfy a frequentist notion of statistical validity called *coverage*: the sets are guaranteed to contain the true label of the test point with high probability. It was a fitting topic for the beginning of the pandemic lock-down, when uncertainty about the future abounded.

It was also serendipitously timely for me. Conversations with two colleagues, Anastasios Angelopoulos and Stephen Bates, led to an exploration of how to extend conformal prediction to the machine learning-based design setting, which resulted in the work in Chapter 2 [131]. Although conformal prediction has been extended to handle various distribution shifts [84, 91, 112, 119, 126], the technical challenge in our setting was that the test (*i.e.*, design) distribution is chosen based on the training data, such that the training and test data are dependent rather than simply drawn independently from different distributions. We formalized this shift as a generalization of the common covariate shift [12] and dubbed it *feedback covariate shift* (FCS), which turns out to describe a panoply of settings beyond machine learning-based design, including active learning and adaptive experimental design. Finally, we extended conformal prediction to provide coverage under FCS, taking heavy inspiration from the technical machinery developed by Tibshirani et al. [84] for handling covariate shift. The resulting method provides coverage for objects designed using learned models, for any model class and any design algorithm that one deems appropriate.

The jury is still out on whether Bayesian or frequentist notions of uncertainty better serve the purposes of machine learning-based design—or, more realistically, what distinguishes the settings in which one is more suitable the other. My own convictions change by the month.

Chapter 3: Trustworthy scientific inquiry using predictions as data

Our work in [131] led to conversations with Tijana Zrnic, in which we pivoted from thinking about how machine-learning predictions affect design to how such predictions affect scientific conclusions—in particular, the estimation of quantities of scientific interest. Personally, I was deeply inspired by the works of Bludau et al. [127] and Vaishnav et al. [139], in which predictive models were developed not just for the intermediary goal of improving various performance metrics, but to draw novel scientific conclusions using the resulting predictions. Bludau et al. [127] used Alphafold-predicted protein structures, in place of scarce gold-standard experimental structures, to quantify associations between structural features and regulatory biochemical modifications of proteins. Vaishnav et al. [139] used a transformer model trained to predict gene expression induced by regulatory DNA sequences to study how gene expression levels change under different types of evolution. These works inspired the following question: how can one estimate a quantity of scientific interest—say, the median gene expression level of promoter sequences naturally found in yeast—using predictions, rather than gold-standard data, in a way that is statistically valid?

Our work on this question led to *prediction-powered inference* [144], described in Chapter 3, a framework for performing valid statistical inference when one has access to just a small amount of gold-standard data, but a large amount of "imputed data" comprising predictions. A common use case we foresee is when a scientist wishes to leverage predictions from an already-trained model in the literature, such as AlphaFold [116], as additional evidence toward a scientific conclusion. The task the model was originally trained for need not exactly match the problem at hand; as long as its predictions are somewhat informative about the gold-standard quantity of interest, our framework can use them to effectively increase sample size for inference, while preserving validity. For example, large language models have been trained for next-token prediction on databases of naturally occurring protein sequences. A protein sequence's likelihood under such models has been shown to correlate with how well it performs its native function [117, 136, 137], even though these models were not explicitly trained to predict this quantity. Prediction-powered inference could enable one to use such predictions to study populations of protein variants in a statistically valid way.

Spiritually, prediction-powered inference enables scientists to more effectively build upon data collected by the research community, as distilled into trained models that yield informative predictions. As such, we hope the framework will help accelerate the process of trustworthy, machine learning-assisted scientific inquiry.

Chapter 2

Autofocused surrogates for design

The material in this chapter is based on work co-authored with Jennifer Listgarten [92].[1]

2.1 Surrogates for design

The design of objects with desired properties, such as novel proteins, molecules, or materials, has a rich history in bioengineering, chemistry, and materials science. In these domains, design has historically been performed through iterative, labor-intensive experimentation [65] (*e.g.*, measuring protein binding affinity) or compute-intensive physics simulations [35] (*e.g.*, computing low-energy structures for nanomaterials). Increasingly, however, attempts are being made to replace these costly and time-consuming steps with cheap and fast calls to a predictive model, trained on labeled data [72, 75, 85, 86, 110]. Herein, we refer to such a model as a *surrogate*, and assume that acquisition of training data for the surrogate is complete, as in [68, 72, 73, 75, 110].[2] The key issue addressed by our work is how best to train a surrogate for use in design, given fixed training data.

In contrast to the traditional use of predictive models, design is distinguished by the fact that it seeks solutions—and therefore, will query the surrogate—in regions of the design space that are not well-represented by the surrogate's training data. If this is not the case, the design problem is easy in that the solution is within the region of the training data. Furthermore, one does not know beforehand which parts of the design space a design procedure will navigate through. As such, a major challenge arises when an surrogate is

[1] We rectify two major terminology choices from the original work. First, the original use of "oracle" to refer to a predictive model has been replaced with "surrogate", to avoid misinterpretation of the former as the ground truth. Second, "model-based design" has been replaced with simply "design". "Model" in the original usage of "model-based design" referred to a density model from which designed objects are sampled, but it is perhaps (understandably) more likely to be misinterpreted as referring to the predictive model (*i.e.* the surrogate, formerly the oracle). We do not believe any clarity in exposition is lost from simplifying this phrase to "design".

[2] For many applications in protein, molecule, and material design, even if one performs iterative rounds of data acquisition, at some point, the acquisition phase concludes due to finite resources.

employed for design: its outputs, including its uncertainty estimates, become unreliable beyond the training data [57, 76]. Successful design using surrogates thus involves an inherent trade-off between the need to stay "near" the training data in order to trust the surrogate, and the need to depart from it in order to make improvements. While trust region approaches have been developed to help address this trade-off [76, 110], herein, we take a different approach and ask: *what is the most effective way to use a fixed, labeled dataset to train a surrogate for design*?

Contributions We develop a novel approach for surrogate-based design that specifies how to update the surrogate as the design space is explored—what we call *autofocusing* the surrogate. In particular, we (i) formalize surrogate-based design as a non-zero-sum game, (ii) derive a surrogate-updating strategy for seeking a Nash equilibrium, and (iii) demonstrate empirically that autofocusing holds promise for improving surrogate-based design.

2.2 Model-based optimization for design

Design problems can be cast as seeking points in the design space, $\mathbf{x} \in \mathcal{X}$, that with high probability satisfy desired conditions on a property random variable, $y \in \mathbb{R}$. For example, one might want to design a superconducting material by specifying its chemical composition, $\mathbf{x}$, such that the resulting material has critical temperature greater than some threshold, $y \geq y_\tau$, or has maximal critical temperature, $y = y_{\max}$. We specify the desired properties using a constraint set, S, such as $S = \{y \colon y \geq y_\tau\}$ for some y_τ. The design goal is then to solve $\arg\max_{\mathbf{x}} P(y \in S \mid \mathbf{x})$. This optimization problem over the inputs, $\mathbf{x}$, can be converted to one over *distributions* over the design space. Specifically, model-based optimization (MBO) seeks the parameters, θ, of a "search model", $p_\theta(\mathbf{x})$, that maximizes an objective that bounds the original objective [20, 90]:

$$\max_{\mathbf{x}} P(y \in S \mid \mathbf{x}) \geq \max_{\theta \in \Theta} \mathbb{E}_{p_\theta(\mathbf{x})}[P(y \in S \mid \mathbf{x})] = \max_{\theta \in \Theta} \mathbb{E}_{p_\theta(\mathbf{x})}\left[\int_S p(y \mid \mathbf{x}) dy\right]. \tag{2.1}$$

The original optimization problem over $\mathbf{x}$, and the MBO problem over θ, are equivalent when the search model has the capacity to place point masses on optima of the original objective. Reasons for using the MBO formulation include that it requires no gradients of $p(y \mid \mathbf{x})$, thereby allowing the use of arbitrary surrogates for design, including those that are not differentiable with respect to the design space and otherwise require specialized treatment. MBO also naturally allows one to obtain not just a single design candidate, but a diverse set of candidates, by sampling from the final search distribution (whose entropy can be adjusted by adding regularization to Eq. (2.1)). Finally, MBO introduces the language of probability into the optimization, thereby allowing coherent incorporation of probabilistic constraints such as implicit trust regions [76]. The search model can be any parameterized probability distribution that can be sampled from, and whose parameters can be estimated using weighted maximum likelihood estimation (MLE) or approximations thereof. Examples include mixtures of Gaussians, hidden Markov models, variational autoencoders [54], and Potts models [46]. Notably, the search model distribution can be over discrete or continuous random variables, or a combination thereof.

Herein, we focus on the use of MBO to solve design problems. In particular, we study *surrogate-based design*, by which we mean efforts to solve Eq. (2.1) by replacing costly and time-consuming queries of the ground truth[3], $p(y \mid \mathbf{x})$, with calls to a trained regression model (*i.e.*, surrogate), $p_\beta(y \mid \mathbf{x})$, with parameters, $\beta \in B$. Given access to a fixed dataset, $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, the surrogate is typically trained once using standard techniques and thereafter considered fixed [61, 68, 72, 73, 75, 76, 79, 110]. In what follows, we describe why such a strategy is sub-optimal and how to re-train the surrogate in order to better achieve design goals. First, however, we briefly review a common approach for performing MBO, as we will leverage such algorithms in our approach.

Solving model-based optimization problems

MBO problems are often tackled with an Estimation of Distribution Algorithm (EDA) [6, 13], a class of iterative optimization algorithms that can be seen as Monte Carlo expectation-maximization [90]; EDAs are also connected to the cross-entropy method [8, 10] and reward-weighted regression in reinforcement learning [32]. Given a surrogate, $p_\beta(y \mid \mathbf{x})$, and an initial search model, $p_{\theta^{(t=0)}}$, an EDA typically proceeds at iteration t with two core steps:

1. "E-step": Sample from the current search model, $\tilde{\mathbf{x}}_i \sim p_{\theta^{(t-1)}}(\mathbf{x})$ for all $i \in \{1, \ldots, m\}$. Compute a weight for each sample, $v_i := V(P_\beta(y \in S \mid \tilde{\mathbf{x}}_i))$, where $V(.)$ is a method-specific, monotonic transformation.

2. "M-step": Perform weighted MLE to yield an updated search model, $p_{\theta^{(t)}}(\mathbf{x})$, which tends to have more mass where $P_\beta(y \in S \mid \mathbf{x})$ is high. (Some EDAs can be seen as performing *maximum a posteriori* inference instead, which results in smoothed parameter updates [76].)

Upon convergence of the EDA, design candidates can be sampled from the final search model if it is not a point mass; one may also choose to use promising samples from earlier iterations. Notably, the surrogate, $p_\beta(y \mid \mathbf{x})$, remains fixed in the steps above. Next, we motivate a new formalism for surrogate-based design that yields a principled approach for updating the surrogate at each iteration.

2.3 Autofocused surrogates for design

The common approach of substituting the surrogate, $p_\beta(y \mid \mathbf{x})$, for the ground-truth, $p(y \mid \mathbf{x})$, does not address the fact that the surrogate is only likely to be reliable over the distribution

[3] We refer to the *ground truth* as the distribution of direct property measurements, which are inevitably stochastic due to sensor noise.

from which its training data were drawn [33, 40, 57]. To address this problem, we now reformulate the surrogate-based design problem as a non-zero-sum game, which suggests an algorithmic strategy for iteratively updating the surrogate within any MBO algorithm.

Surrogate-based design as a game

When the objective in Eq. (2.1) is replaced with a surrogate-based version,

$$\arg\max_{\theta \in \Theta} \mathbb{E}_{p_\theta(\mathbf{x})}[P_\beta(y \in S \mid \mathbf{x})], \tag{2.2}$$

the solution to the surrogate-based problem will, in general, be sub-optimal with respect to the original objective that uses the ground truth, $P(y \in S \mid \mathbf{x})$. This sub-optimality can be extreme due to pathological behavior of the surrogate when the search model, $p_\theta(\mathbf{x})$, strays too far from the training distribution during the optimization [76].

Since one cannot access the ground truth, we seek a practical alternative wherein we can leverage a surrogate, but also infer when the values of the ground-truth and surrogate-based objectives (in Eq. (2.1) and Eq. (2.2), respectively) are likely to be close. To do so, we introduce the notion of the *surrogate gap*, defined as $\mathbb{E}_{p_\theta(\mathbf{x})}[|P(y \in S \mid \mathbf{x}) - P_\beta(y \in S \mid \mathbf{x})|]$. When this quantity is small, then by Jensen's inequality the surrogate-based and ground-truth objectives are close. Consequently, our insight for improving surrogate-based design is to use the surrogate that minimizes the surrogate gap,

$$\arg\min_{\beta \in B} \textsc{OracleGap}(\theta, \beta) = \arg\min_{\beta \in B} \mathbb{E}_{p_\theta(\mathbf{x})}[|P(y \in S \mid \mathbf{x}) - P_\beta(y \in S \mid \mathbf{x})|]. \tag{2.3}$$

Together, Eq. (2.2) and Eq. (2.3) define the coupled objectives of two players, namely the search model (with parameters θ) and the surrogate (with parameters β), in a non-zero-sum game. To attain good objective values for both players, our goal will be to search for a Nash equilibrium—that is, a pair of values (θ^*, β^*) such that neither can improve its objective given the other. To do so, we develop an alternating ascent-descent algorithm, which alternates between (i) fixing the surrogate parameters and updating the search model parameters to increase the objective in Eq. (2.2) (the ascent step), and (ii) fixing the search model parameters and updating the surrogate parameters to decrease the objective in Eq. (2.3) (the descent step). In the next section, we describe this algorithm in more detail.

Practical interpretation of the surrogate-based design game. Interpreting the usefulness of this game formulation requires some subtlety. The claim is not that every Nash equilibrium yields a search model that provides a high value of the (unknowable) ground-truth objective in Eq. (2.1). However, for any pair of values, (θ, β), the value of the surrogate gap provides a certificate on the value of the ground-truth objective. In particular, if one has a surrogate and search model that yield a surrogate gap of ϵ, then by Jensen's inequality the ground-truth objective is within ϵ of the surrogate-based objective. Therefore, to the extent that we are able to minimize the surrogate gap, Eq. (2.3), we can trust the value of our surrogate-based objective, Eq. (2.2). Note that a small, or even zero surrogate gap only implies that the surrogate-based objective is trustworthy; successful design also entails achieving a *high* surrogate-based objective, the potential for which depends on an appropriate surrogate class and suitably informative training data (as it always does for surrogate-based design, regardless of whether our framework is used).

Although the surrogate gap as a certificate is useful conceptually for motivating our approach, at present it is not clear how to estimate it. In our experiments, we found that we could demonstrate the benefits of autofocusing without directly estimating the surrogate gap, relying solely on the principle of minimizing it. We also note that in practice, what matters is not whether we converge to a Nash equilibrium, just as what matters in empirical risk minimization is not whether one exactly recovers the global optimum, only a useful point. That is, if we can find parameters, (θ, β), that yield better designs than alternative methods, then we have developed a useful method.

An alternating ascent-descent algorithm for the surrogate-based design game

Our approach alternates between an ascent step that updates the search model, and a descent step that updates the surrogate. The ascent step is relatively straightforward as it leverages existing MBO algorithms. The descent step, however, requires some creativity. In particular, for the ascent step, we run a single iteration of an MBO algorithm as described in Section 2.2, to obtain a search model that increases the objective in Eq. (2.2). For the descent step, we aim to minimize the surrogate gap in Eq. (2.3) by making use of the following observation (proof in Section 2.8).

Proposition 1. *For any search model, $p_\theta(\mathbf{x})$, if the surrogate parameters, β, satisfy*

$$\mathbb{E}_{p_\theta(\mathbf{x})}[D_{KL}(p(y \mid \mathbf{x}) \mid\mid p_\beta(y \mid \mathbf{x}))] = \int_{\mathcal{X}} D_{KL}(p(y \mid \mathbf{x}) \mid\mid p_\beta(y \mid \mathbf{x}))\, p_\theta(\mathbf{x}) d\mathbf{x} \leq \epsilon, \tag{2.4}$$

where $D_{KL}(p \mid\mid q)$ is the Kullback-Leibler (KL) divergence between distributions p and q, then the following bound holds:

$$\mathbb{E}_{p_\theta(\mathbf{x})}[|P(y \in S \mid \mathbf{x}) - P_\beta(y \in S \mid \mathbf{x})|] \leq \sqrt{\frac{\epsilon}{2}}.$$

As a consequence of Proposition 1, given any search model, $p_\theta(\mathbf{x})$, a surrogate that minimizes the expected KL divergence in Eq. (2.4) also minimizes an upper bound on the surrogate gap. Our descent strategy is therefore to minimize this expected divergence. In particular, as shown in the Section 2.8, the resulting surrogate parameter update at iteration t can be written as $\beta^{(t)} = \arg\max_{\beta \in B} \mathbb{E}_{p_{\theta^{(t)}}(\mathbf{x})}\mathbb{E}_{p(y|\mathbf{x})}[\log p_\beta(y \mid \mathbf{x})]$, where we refer to the objective as the log-likelihood under the search model. Although we cannot generally access the ground truth, $p(y \mid \mathbf{x})$, we do have labeled training data, $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, whose labels come from the ground-truth distribution, $y_i \sim p(y \mid \mathbf{x} = \mathbf{x}_i)$. We therefore use importance sampling with the training distribution, $p_0(\mathbf{x})$, as the proposal distribution, to obtain a now-practical surrogate parameter update,

$$\beta^{(t)} = \arg\max_{\beta \in B} \frac{1}{n} \sum_{i=1}^{n} \frac{p_{\theta^{(t)}}(\mathbf{x}_i)}{p_0(\mathbf{x}_i)} \log p_\beta(y_i \mid \mathbf{x}_i). \tag{2.5}$$

The training points, $\mathbf{x}_i$, are used to estimate some model for $p_0(\mathbf{x})$, while $p_{\theta^{(t)}}(\mathbf{x})$ is given by the search model. We discuss the variance of the importance weights, $w_i := p_\theta(\mathbf{x}_i)/p_0(\mathbf{x}_i)$, shortly.

Together, the ascent and descent steps amount to appending a “Step 3” to each iteration of the generic two-step MBO algorithm outlined in Section 2.2, in which the surrogate is retrained on re-weighted training data according to Eq. (2.5). We call this strategy *autofocusing* the surrogate, as it retrains the surrogate in lockstep with the search model, to keep the surrogate likelihood maximized on the most promising regions of the design space. Pseudo-code for autofocusing can be found in the Algs. 1 and 2. As shown in the experiments, autofocusing tends to improve the outcomes of design procedures, and when it does not, no harm is incurred relative to the naive approach with a fixed surrogate. Before discussing such experiments, we first make some remarks.

Remarks on autofocusing

Controlling variance of the importance weights. It is well known that importance weights can have high, even infinite, variance [51], which may prevent the importance-sampled estimate of the log-likelihood from being useful for retraining the surrogate effectively. That is, solving Eq. (2.5) may not reliably yield surrogate parameter estimates that minimize the log-likelihood under the search model. To monitor the reliability of the importance-sampled estimate, one can compute and track an *effective sample size* of the re-weighted training data, $n_e := (\sum_{i=1}^n w_i)^2 / \sum_{i=1}^n w_i^2$, which reflects the variance of the importance weights [51]. If one has some sense of a suitable sample size for the application at hand (*e.g.*, based on the surrogate model capacity), then one could monitor n_e and choose not to retrain when it is too small. Another variance control strategy is to use a trust region to constrain the movement of the search model, such as in [76], which automatically controls the variance (see Proposition 2.8.1). Indeed, our experiments show how autofocusing works synergistically with a trust-region approach. Finally, two other common strategies are: (i) self-normalizing the weights, which provides a biased but consistent and lower-variance estimate [51], and (ii) flattening the weights [33] to w_i^α according to a hyperparameter, $\alpha \in [0, 1]$. The value of α interpolates between the original importance weights ($\alpha = 1$), which provide an unbiased but high-variance estimate, and all weights equal to one ($\alpha = 0$), which is equivalent to naively training the surrogate (*i.e.*, no autofocusing).

Surrogate bias-variance trade-off. If the surrogate equals the ground truth over all parts of the design space encountered during the design procedure, then autofocusing should not improve upon using a fixed surrogate. In practice, however, this is unlikely to ever be the case—the surrogate is almost certain to be misspecified and ultimately mislead the design procedure with incorrect inductive bias. It is therefore interesting to consider what autofocusing does from the perspective of the bias-variance trade-off of the surrogate, with respect to the search model distribution. On the one hand, autofocusing retrains the surrogate using an unbiased estimate of the log-likelihood over the search model. On the other hand, as the search model moves further away from the training data, the effective sample size available to train the surrogate decreases; correspondingly, the variance of the surrogate increases. In other words, when we use a fixed surrogate (no autofocusing), we prioritize minimal variance at the expense of greater bias. With pure autofocusing, we prioritize reduction in bias at the expense of higher variance. Autofocusing with techniques to control the variance of the importance weights [33, 48] enables us to make a suitable trade-off between these two extremes.

Autofocusing corrects design-induced covariate shift. In adopting an importance-sampled estimate of the training objective, Eq. (2.5) is analogous to the classic covariate shift adaptation strategy known as importance-weighted empirical risk minimization [33, 48]. We can therefore interpret autofocusing as dynamically correcting for covariate shift induced by a design procedure, where, at each iteration, a new "test" distribution is given by the updated search model. Furthermore, we are in the fortunate position of knowing the exact parametric form of the test density at each iteration, which is simply that of the search model. This view highlights that the goal of autofocusing is not necessarily to increase exploration of the design space, but to provide a more useful surrogate wherever the search model does move (as dictated by the underlying method to which autofocusing is added).

2.4 Related work

Surrogate-based design in the fixed-data setting is gaining prominence in several application areas, including the design of proteins and nucleotide sequences [61, 76, 79, 95, 110], molecules [63, 68, 73], and materials [43, 72]. Within such work, the danger in extrapolating beyond the training distribution is not always acknowledged or addressed. In fact, proposed design procedures often are validated under the assumption that the surrogate is always correct [61, 63, 73, 79, 97]. Some exceptions include Conditioning by Adaptive Sampling (CbAS) [76], which employs a probabilistic trust-region approach using a model of the training distribution, and [110], which uses a hard distance-based threshold. Similar in spirit to [76], Linder et al. regularize the designed sequences based on their likelihood under a model of the training distribution [97]. In another approach, a variational autoencoder implicitly enforces a trust region by constraining design candidates to the probabilistic image of the decoder [68]. Finally, Kumar & Levine tackle design by learning the inverse of a ground-truth function, which they constrain to agree with a surrogate, so as to discourage too much departure from the training data [95]. None of these approaches update the surrogate. However, autofocusing is entirely complementary to and does not preclude the additional use of any of these approaches. For example, we demonstrate in our experiments that autofocusing improves the outcomes of CbAS, which implicitly inhibits the movement of the search model away from the training distribution.

Related to the design problem is that of active learning in order to optimize a function, using for example Bayesian optimization [47]. Such approaches are fundamentally distinct from our setting in that they dynamically acquire new labeled data, thereby more readily allowing for correction of surrogate modeling errors. In a similar spirit, evolutionary algorithms sometimes use a "surrogate" model of the function of interest (equivalent to our surrogate), to help guide the acquisition of new data [45]. In such settings, the surrogate may be updated using an *ad hoc* subset of the data [50] or perturbation of the surrogate parameters [36]. Similarly, a recent reinforcement-learning based approach to biological sequence design relies on new data to refine the surrogate when moving into a region of design space where the surrogate is unreliable [89].

Offline reinforcement learning (RL) [96] shares similar characteristics with our problem in that the goal is to find a policy that optimizes a reward function, given only a fixed dataset of trajectories sampled using another policy. In particular, offline model-based RL leverages a learned model of dynamics that may not be accurate everywhere. Methods in that setting have attempted to account for the shift away from the training distribution using uncertainty estimation and trust-region approaches [44, 67, 100]; importance sampling has also been used for off-policy evaluation [14, 59].

As noted in the previous section, autofocusing operates through iterative retraining of the surrogate in order to correct for covariate shift induced by the movement of the search model. It can therefore be connected to ideas from domain adaptation more broadly [40]. Finally, we note that mathematically, surrogate-based design is related to the decision-theoretic framework of performative prediction [99]. Perdomo et al. formalize the phenomenon in which using predictive models to perform actions induces distributional shift, then present theoretical analysis of repeated retraining with new data as a solution. Our problem has two major distinctions from this setting: first, the ultimate goal in design is to maximize an unknowable ground-truth objective, not to minimize risk of the surrogate. The latter is only relevant to the extent that it helps us achieve the former, and our work operationalizes that connection by formulating and minimizing the surrogate gap. Second, we are in a fixed-data setting. Our work demonstrates the utility of adaptive retraining even in the absence of new data.

2.5 Experiments

We now demonstrate empirically, across a variety of both experimental settings and MBO algorithms, how autofocusing can help us better achieve design goals. First we leverage an intuitive example to gain detailed insights into how autofocus behaves. We then conduct a

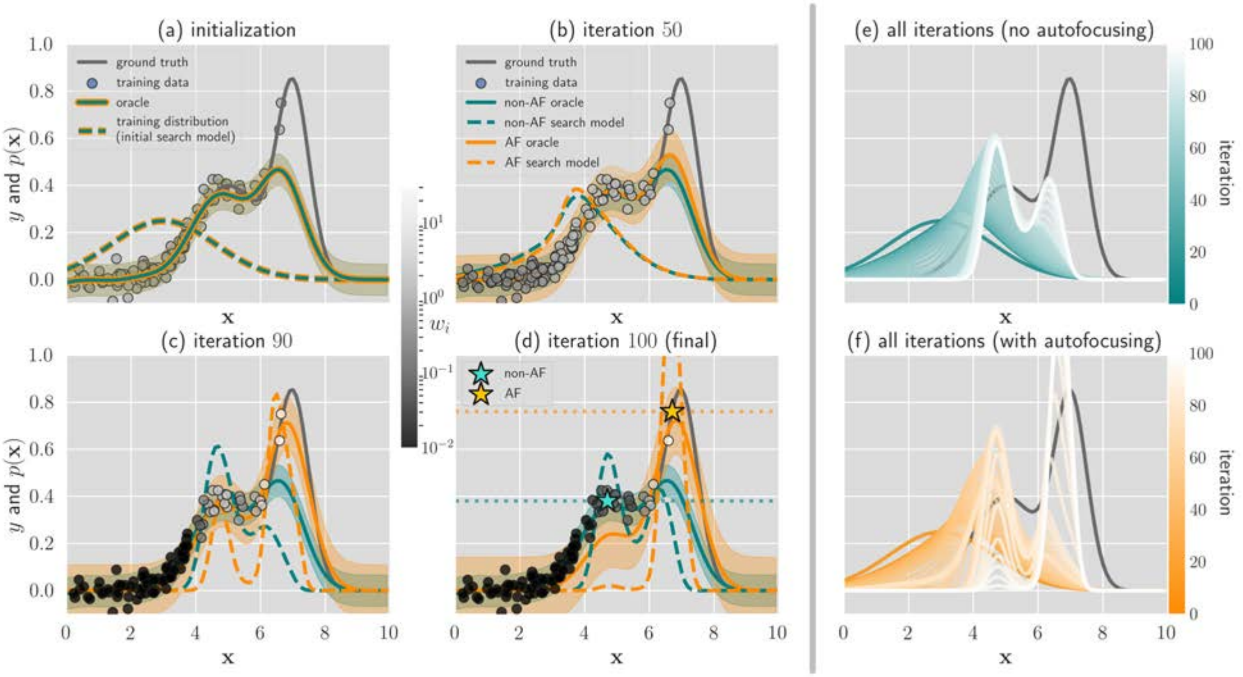


Figure 2.1: **Illustrative example of autofocusing for design.** Panels (a-d) show detailed snapshots of the MBO algorithm, CbAS [76], with and without autofocusing (AF) in each panel. The vertical axis represents both y values (for the surrogate and ground truth) and probability density values (of the training distribution, $p_0(\mathbf{x})$, and search distributions, $p_{\theta^{(t)}}(\mathbf{x})$). Shaded envelopes correspond to ±1 standard deviation of the oracles, $\sigma_{\beta^{(t)}}$, with the surrogate expectations, $\mu_{\beta^{(t)}}(\mathbf{x})$, shown as a solid line. Specifically, (a) at initialization, the surrogate and search model are the same for AF and non-AF. Intermediate and final iterations are shown in (b-d), where the non-AF and AF oracles and search models increasingly diverge. Grayscale of training points corresponds to their importance weights used for autofocusing. In (d), each star and dotted horizontal line indicate the ground-truth value corresponding to the point of maximum density of the final search model, indicative of its usefulness for design (higher is better). The values of $(\sigma_\epsilon, \sigma_0)$ used here correspond to the ones marked by an × in Figure 2.2, which summarizes results across a range of settings. Panels (e,f) show the search model over all iterations without and with autofocusing, respectively.

detailed study on a more realistic problem of designing superconducting materials. Code for our experiments is available at `https://github.com/clarafy/autofocused_oracles`.

An illustrative example

To investigate how autofocusing works in a setting that can be understood intuitively, we constructed a one-dimensional design problem where the goal was to maximize a multi-modal ground-truth function, $f(\mathbf{x}) : \mathbb{R} \rightarrow \mathbb{R}^+$, given fixed training data (Figure 2.1a). The training distribution from which training points were drawn, $p_0(\mathbf{x})$, was a Gaussian with variance, σ_0^2, centered at 3, a point where $f(\mathbf{x})$ is small relative to the global maximum at 7. This captures the common scenario where the surrogate training data do not extend out to global optima

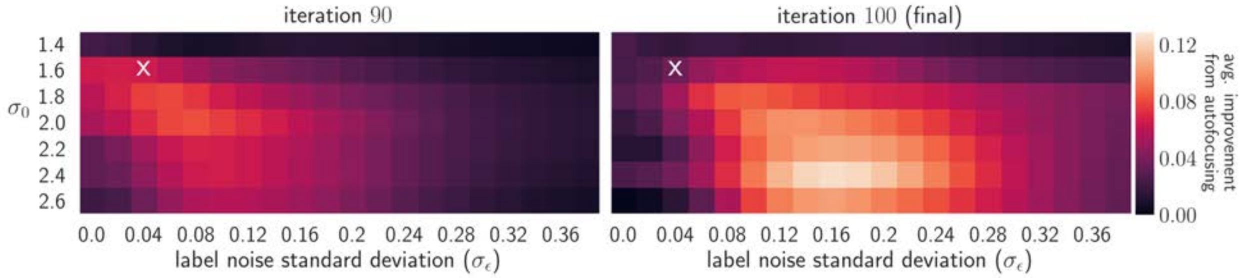


Figure 2.2: **Improvement from autofocusing for a range of settings of the illustrative example.** Each colored square shows the improvement (averaged over 50 trials) conferred by autofocusing (AF) for one setting, $(\sigma_\epsilon, \sigma_0)$, of, respectively, the standard deviations of the training distribution and the label noise. Improvement is quantified as the difference between the ground-truth objective in Eq. (2.1) achieved by the final search model with and without AF. A positive value means AF yielded higher ground-truth values (*i.e.*, performed better than without AF), while zero means it neither helped nor hurt. Similar plots to Figure 2.1 are shown for other settings in Figures 2.3, 2.4.

of the property of interest. As we increase the variance of the training distribution, σ_0^2, the training data become more and more likely to approach the global maximum of $f(\mathbf{x})$. The training labels are drawn from $p(y \mid \mathbf{x}) = \mathcal{N}(f(\mathbf{x}), \sigma_\epsilon^2)$, where σ_ϵ^2 is the variance of the label noise. For this example, we used CbAS [76], an MBO algorithm that employs a probabilistic trust region. We did not control the variance of the importance weights.

An MBO algorithm prescribes a sequence of search models as the optimization proceeds, where each successive search model is fit using weighted MLE to samples from its predecessor. However, in our one-dimensional example, one can instead use numerical quadrature to directly compute each successive search model [76]. Such an approach enables us to abstract out the particular parametric form of the search model, thereby more directly exposing the effects of autofocusing. In particular, we used numerical quadrature to compute the search model density at iteration t as $p^{(t)}(\mathbf{x}) \propto P_{\beta^{(t)}}(y \in S^{(t)} \mid \mathbf{x})p_0(\mathbf{x})$, where $S^{(t)}$ belongs to a sequence of relaxed constraint sets such that $S^{(t)} \supseteq S^{(t+1)} \supseteq S$ [76]. We computed this sequence of search models in two ways: (i) without autofocusing, that is, with a fixed surrogate trained once on equally weighted training data, and (ii) with autofocusing, that is, where the surrogate was retrained at each iteration. In both cases, the surrogate was of the form $p_\beta(y \mid \mathbf{x}) = \mathcal{N}(\mu_\beta(\mathbf{x}), \sigma_\beta^2)$, where $\mu_\beta(\mathbf{x})$ was fit by kernel ridge regression with a radial basis function kernel and σ_β^2 was set to the mean squared error between $\mu_\beta(\mathbf{x})$ and the labels (see Section 2.9 for more details). Since this was a maximization problem, the desired condition was set as $S = \{y : y \geq \max_{\mathbf{x}} \mu_\beta(\mathbf{x})\}$ (where $\mu_\beta(\mathbf{x}) = 0.68$ for the initial surrogate). We found that autofocusing more effectively shifts the search model toward the ground-truth global maximum as the iterations proceed (Figure 2.1b-f), thereby providing improved design candidates.

To understand the effect of autofocusing more systematically, we repeated the experiment just described across a range of settings of the variances of the training distribution, σ_0^2, and of the label noise, σ_ϵ^2 (Figure 2.2). Intuitively, both these variances control how informative the training data are about the ground-truth global maximum: as σ_0^2 increases, the training data are more likely to include points near the global maximum, and as σ_ϵ^2 decreases, the training labels are less noisy. Therefore, if the training data are either too uninformative (small σ_0^2 and/or large σ_ϵ^2) or too informative (large σ_0^2 and/or small σ_ϵ^2), then one would not expect autofocusing to substantially improve design. In intermediate regimes, autofocusing should be particularly useful. Such a phenomenon is seen in our experiments (Figure 2.2). Importantly, this kind of intermediate regime is one in which practitioners are likely to find themselves: the motivation for design is often sparked by the existence of a few examples with property values that are exceptional compared to most known examples, yet the design goal is to push the desired property to be more exceptional still. In contrast, if the true global optimum already resides in the training data, one cannot hope to design anything better anyway. However, even in regimes where autofocusing does not help, on average it does not hurt relative to a naive approach with a fixed surrogate (Figure 2.2 and Section 2.5).

Designing superconductors with maximal critical temperature

Designing superconducting materials with high critical temperatures is an active research problem that impacts engineering applications from magnetic resonance imaging systems to the Large Hadron Collider. To assess autofocusing in a more realistic scenario, we used a dataset comprising $21,263$ superconducting materials paired with their critical temperatures [69], the maximum temperature at which a material exhibits superconductivity. Each material is represented by a feature vector of length eighty-one, which contains real-valued properties of the material's constituent elements (*e.g.*, their atomic radius and valence). We outline our experiments here, with details deferred to the Section 2.9.

Unlike *in silico* validation of a predictive model, one cannot hold out data to validate a design algorithm because one will not have ground-truth labels for proposed design candidates. Thus, similarly to [76], we created a "ground-truth" model by training gradient-boosted regression trees [69, 74] on the whole dataset and treating the output as the ground-truth expectation, $\mathbb{E}[y \mid \mathbf{x}]$, which can be called at any time. Next, we generated training data to emulate the common scenario in which design practitioners have labeled data that are not dense near ground-truth global optima. In particular, we selected the $n = 17,015$ training points from the dataset whose ground-truth expectations were in the bottom 80^{th} percentile. We used MLE with these points to fit a full-rank multivariate normal, which served as the training distribution, $p_0(\mathbf{x})$, from which we drew n simulated training points, $\{\mathbf{x}_i\}_{i=1}^n$. For each $\mathbf{x}_i$ we drew one sample, $y_i \sim \mathcal{N}(\mathbb{E}[y \mid \mathbf{x}_i], 1)$, to obtain a noisy ground-truth label. Finally, for our surrogate, we used $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ to train an ensemble of three neural networks that output both $\mu_\beta(\mathbf{x})$ and $\sigma_\beta^2(\mathbf{x})$, to provide predictions of the form $p_\beta(y \mid \mathbf{x}) = \mathcal{N}(\mu_\beta(\mathbf{x}), \sigma_\beta^2(\mathbf{x}))$ [62].

We ran six different MBO algorithms, each with and without autofocusing, with the goal of designing materials with maximal critical temperatures. In all cases, we used a full-rank multivariate normal for the search model, and flattened the importance weights used for autofocusing to w_i^α [33] with $\alpha = 0.2$ to help control variance. The MBO algorithms were: (i)

Table 2.1: **Designing superconducting materials.** We ran six different MBO methods, each with and without autofocusing. For each method, we extracted those samples with surrogate expectations above the 80th percentile and computed their ground-truth expectations. We report the median and maximum of those ground-truth expectations (both in degrees K), their percent chance of improvement (PCI, in percent) over the maximum label in the training data, as well as the Spearman correlation (ρ) and root mean squared error (RMSE, in degrees K) between the surrogate and ground-truth expectations. Each reported score is averaged over 10 trials, where, in each trial, a different training set was sampled from the training distribution. "Mean Diff." is the average difference between the score when using autofocusing compared to not. Bold values with one star (*) and two stars (**), respectively, mean p-values < 0.05 and < 0.01 from a two-sided Wilcoxon signed-rank test on the 10 paired score differences between a method with autofocus and without ('Original'). For all scores but RMSE, a higher value means autofocusing yielded better results (as indicated by the arrow ↑); for RMSE, the opposite is true (as indicated by the arrow ↓).

	Median ↑	Max ↑	PCI ↑	ρ ↑	RMSE ↓	Median ↑	Max ↑	PCI ↑	ρ ↑	RMSE ↓
			CbAS					**DbAS**		
Original	51.5	103.8	0.11	0.05	17.2	57.0	98.4	0.11	0.01	29.6
Autofocused	76.4	119.8	3.78	0.56	12.9	78.9	111.6	4.4	0.01	24.5
Mean Diff.	**24.9****	**16.0****	**3.67****	**0.51****	**-4.4****	**21.9****	**13.2****	**4.2****	0.01	**-5.1***
			RWR					**FB**		
Original	43.4	102.0	0.05	0.92	7.4	49.2	100.6	0.14	0.09	17.5
Autofocused	71.4	114.0	1.60	0.65	12.7	64.2	111.6	0.86	0.49	11.1
Mean Diff.	**28.0****	**12.0****	**1.56****	**-0.27****	**5.4****	**15.0****	**11.0****	**0.73****	**0.40****	**-6.4****
			CEM-PI					**CMA-ES**		
Original	34.5	55.8	0.00	-0.16	148.3	42.1	69.4	0.00	0.27	27493.2
Autofocused	67.0	98.0	1.69	0.13	29.4	50.2	85.8	0.01	0.52	9499.8
Mean Diff.	**32.6****	**42.3***	**1.69***	0.29	**-118.9****	**8.1***	**16.3***	0.01	**0.25***	**-17993.5***

Conditioning by Adaptive Sampling (CbAS) [76]; (ii) Design by Adaptive Sampling (DbAS) [76]; (iii) reward-weighted regression (RWR) [32]; (iv) the "feedback" mechanism proposed in [79] (FB); (v) the cross-entropy method used to optimize probability of improvement (CEM-PI) [47, 76]; and (vi) Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [26]. These are briefly described in the Section 2.9.

To quantify the success of each algorithm, we did the following. At each iteration, t, we first computed the surrogate expectations, $\mathbb{E}_{\beta^{(t)}}[y \mid \mathbf{x}]$, for each of n samples drawn from the search model, $p_{\theta^{(t)}}(\mathbf{x})$. We then selected the iteration where the 80th percentile of these surrogate expectations was greatest. For that iteration, we computed various summary statistics on the *ground-truth* expectations of the best samples, as judged by the surrogate from that iteration (*i.e.*, samples with surrogate expectations greater than the 80th percentile; Table 2.1). See Algorithm 3 for pseudocode of this procedure. Our evaluation procedure emulates the typical setting in which a practitioner has limited experimental resources, and can only evaluate the ground truth for the most promising candidates [72, 75, 85, 110].

Across the majority of evaluation metrics, for all MBO methods, autofocusing a method provided a statistically significant improvement upon the original method. The percent chances of improvement (PCI, the percent chances that the best samples had greater ground-truth expectations than the maximum label in the training data), expose the challenging nature of the design problem. All methods with no autofocusing had a PCI less than 0.14%, which although small, still reflects a marked improvement over a naive baseline of simply drawing n new samples from the training distribution itself, which achieves $5.9 \times 10^{-3}\%$. Plots of design trajectories from these experiments, and results from experiments without variance control and with surrogate architectures of higher and lower capacities, can be found in Figures 2.6, 2.7, 2.8 and Tables 2.2, 2.3, 2.4.

2.6 Discussion

We have introduced a new formulation of surrogate-based design as a non-zero-sum game. From this formulation, we developed a new approach for design wherein the surrogate—the predictive model that replaces costly and time-consuming laboratory experiments—is iteratively retrained so as to “autofocus” it on the current region of design candidates under consideration. Our autofocusing approach can be applied to any design procedure that uses model-based optimization. We recommend using autofocusing with an MBO method that uses trust regions, such as CbAS [76], which automatically helps control the variance of the importance weights used for autofocusing. For autofocusing an MBO algorithm without a trust region, practical use of the surrogate gap certificate and/or effective sample size should be further investigated. Nevertheless, even without these, we have demonstrated empirically that autofocusing can provide benefits.

Autofocusing can be seen as dynamically correcting for covariate shift as the design procedure explores design space. It can also be understood as enabling a design procedure to navigate a trade-off between the bias and variance of the surrogate, with respect to the search model distribution. One extension of this idea is to also perform surrogate model selection at each iteration, such that the model capacity is tailored to the level of importance weight variance.

Further extensions to consider are alternate strategies for estimating the importance weights [48]. In particular, training discriminative classifiers to estimate these weights may be fruitful when using search models that are implicit generative models, or whose likelihood cannot otherwise be computed in closed form, such as variational autoencoders [48, 78]. We believe this may be a promising approach for applying autofocusing to biological sequence design and other discrete design problems, which often leverage such models. One can also imagine extensions of autofocusing to gradient-based design procedures [61]—for example, using techniques for test-time surrogate retraining, in order to evaluate the current point most accurately [104].

2.7 Pseudocode

Algorithm 1 gives pseudocode for autofocusing a broad class of model-based optimization (MBO) algorithms known as estimation of distribution algorithms (EDAs), which can be seen as performing Monte-Carlo expectation-maximization [90]. EDAs proceed at each iteration with a sampling-based "E-step" (Steps 2 and 3 in Algorithm 1) and a weighted maximum likelihood estimation (MLE) "M-step" (Step 4; see [90] for more details). Different EDAs are distinguished by method-specific monotonic transformations $V(\cdot)$, which determine the sample weights used in the M-step. In some EDAs, this transformation is not explicitly defined, but rather implicitly implemented by constructing and using a sequence of relaxed constraint sets, $S^{(t)}$, such that $S^{(t)} \supseteq S^{(t+1)} \supseteq S$ [8, 10, 76].

Algorithm 2 gives pseudocode for autofocusing a particular EDA, Conditioning by Adaptive Sampling (CbAS) [76], which uses such a sequence of relaxed constraint sets, as well as M-step weights that induce an implicit trust region for the search model update. For simplicity, the algorithm is instantiated with the design goal of maximizing the property of interest. It can easily be generalized to the goal of achieving a specific value for the property, or handling multiple properties (see Sections S2-3 of [76]).

Use of [.] in the pseudocode denotes an optional input argument with default values.

2.8 Proofs and derivations

Proof of Proposition 1. For any distribution $p_\theta(\mathbf{x})$, if

$$\mathbb{E}_{p_\theta(\mathbf{x})}\left[D_{\mathrm{KL}}(p(y \mid \mathbf{x}) \mid\mid p_\phi(y \mid \mathbf{x}))\right] \leq \epsilon,$$

then it holds that

$$\begin{aligned} \mathbb{E}_{p_\theta(\mathbf{x})}\left[|P(y \in S \mid \mathbf{x}) - P_\phi(y \in S \mid \mathbf{x})|^2\right] &\leq \mathbb{E}_{p_\theta(\mathbf{x})}\left[\delta(p(y \mid \mathbf{x}), p_\phi(y \mid \mathbf{x}))^2\right] \\ &\leq \frac{1}{2}\mathbb{E}_{p_\theta(\mathbf{x})}\left[D_{\mathrm{KL}}(p(y \mid \mathbf{x}) \mid\mid p_\phi(y \mid \mathbf{x}))\right] \\ &\leq \frac{\epsilon}{2}. \end{aligned}$$

where $\delta(p, q)$ is the total variation distance between probability distributions p and q, and the second inequality is due to Pinsker's inequality. Finally, applying Jensen's inequality yields

$$\mathbb{E}_{p_\theta(\mathbf{x})}\left[|P(y \in S \mid \mathbf{x}) - P_\phi(y \in S \mid \mathbf{x})|\right] \leq \sqrt{\frac{\epsilon}{2}}.$$

□

Algorithm 1 Autofocused model-based optimization algorithm

Input: Training data, $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$; surrogate model class, $p_\beta(y \mid \mathbf{x})$, with parameters, β, that can be estimated with MLE; search model class, $p_\theta(\mathbf{x})$, with parameters, θ, that can be estimated with weighted MLE or approximations thereof; desired constraint set, S (*e.g.*, $S = \{y \mid y \geq y_\tau\}$); maximum number of iterations, T; number of samples to generate, m; EDA-specific monotonic transformation, $V(\cdot)$ (see Section 2.9 for examples).

Initialization: Obtain $p_0(\mathbf{x})$ by fitting to $\{\mathbf{x}_i\}_{i=1}^n$ with the search model class. For the search model, set $p_{\theta^{(0)}}(\mathbf{x}) \leftarrow p_0(\mathbf{x})$. For the surrogate, $p_{\beta^{(0)}}(y \mid \mathbf{x})$, use MLE with equally weighted training data.

Output: Sequence of search models, $\{p_{\theta^{(t)}}(\mathbf{x})\}_{t=1}^T$, and sequence of samples, $\{(\tilde{\mathbf{x}}_i^{(t)}, \ldots, \tilde{\mathbf{x}}_m^{(t)})\}_{t=1}^T$, from all iterations. One may use these in a number of different ways. For example, one may sample design candidates from the final search model, $p_{\theta^{(T)}}(\mathbf{x})$, or use the most promising candidates among $\{(\tilde{\mathbf{x}}_i^{(t)}, \ldots, \tilde{\mathbf{x}}_m^{(t)})\}_{t=1}^T$, as judged by the appropriate surrogate (*i.e.*, corresponding to the iteration at which the candidate was generated).

```
for $i = 1, \ldots, T$ do
    Sample from the current search model, $\tilde{\mathbf{x}}_i^{(t)} \sim p_{\theta^{(t-1)}}(\mathbf{x}), \forall i \in \{1, \ldots, m\}$.
    $v_i \leftarrow V(P_{\beta^{(t-1)}}(y \in S \mid \tilde{\mathbf{x}}_i^{(t)})), \forall i \in \{1, \ldots, m\}$
    Fit the updated search model, $p_{\theta^{(t)}}(\mathbf{x})$, using weighted MLE with the samples, $\{\tilde{\mathbf{x}}_i^{(t)}\}_{i=1}^m$, and their corresponding EDA weights, $\{v_i\}_{i=1}^m$.
    Compute importance weights for the training data, $w_i \leftarrow p_{\theta^{(t)}}(\mathbf{x}_i)/p_{\theta^{(0)}}(\mathbf{x}_i), i = 1, \ldots, n$.
    Retrain the surrogate using the re-weighted training data,
        $\beta^{(t)} \leftarrow \arg\max_{\beta \in B} \frac{1}{n} \sum_{i=1}^{n} w_i \log p_\beta(y_i \mid \mathbf{x}_i).$
end for
```

Derivation of the descent step to minimize the surrogate gap

Here, we derive the descent step of the alternating ascent-descent algorithm described in Section 2.3. At iteration t, given the search model parameters, $\theta^{(t)}$, our goal is to update the surrogate parameters according to

$$\beta^{(t)} = \arg\min_{\beta \in B} \mathbb{E}_{p_{\theta^{(t)}}(\mathbf{x})}[D_{\mathrm{KL}}(p(y \mid \mathbf{x}) \mid\mid p_\beta(y \mid \mathbf{x}))].$$

Algorithm 2 Autofocused Conditioning by Adaptive Sampling (CbAS) [76]

```
Input: Training data, $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$; surrogate model class, $p_\beta(y \mid \mathbf{x})$ with parameters, $\beta$, that can be estimated with MLE; search model class, $p_\theta(\mathbf{x})$, with parameters, $\theta$, that can be estimated with weighted MLE or approximations thereof; maximum number of iterations, $T$; number of samples to generate, $m$; [percentile threshold, $Q = 90$].
Initialization: Obtain $p_0(\mathbf{x})$ by fitting to $\{\mathbf{x}_i\}_{i=1}^n$ with the search model class. For the search model, set $p_{\theta^{(0)}}(\mathbf{x}) \leftarrow p_0(\mathbf{x})$. For the surrogate, $p_{\beta^{(0)}}(y \mid \mathbf{x})$, use MLE with equally weighted training data. Set $\gamma_0 = -\infty$.
Output: Sequence of search models, $\{p_{\theta^{(t)}}(\mathbf{x})\}_{t=1}^T$, and sequence of samples, $\{(\tilde{\mathbf{x}}_i^{(t)}, \dots, \tilde{\mathbf{x}}_m^{(t)})\}_{t=1}^T$, from all iterations. One may use these in a number of different ways (see Algorithm 1).
for $i = 1, \dots, T$ do
    Sample from the current search model, $\tilde{\mathbf{x}}_i^{(t)} \sim p_{\theta^{(t-1)}}(\mathbf{x}), \forall i \in \{1, \dots, m\}$.
    $q_t \leftarrow Q$-th percentile of the surrogate expectations of the samples, $\{\mu_\beta(\tilde{\mathbf{x}}_i^{(t)})\}_{i=1}^m$
    $\gamma_t \leftarrow \max\{\gamma_{t-1}, q_t\}$
    $v_i \leftarrow (p_0(\tilde{\mathbf{x}}_i^{(t)})/p_{\theta^{(t-1)}}(\tilde{\mathbf{x}}_i^{(t)}))P_{\beta^{(t-1)}}(y \geq \gamma_t \mid \tilde{\mathbf{x}}_i^{(t)}), \forall i \in \{1, \dots, m\}$
    Fit the updated search model, $p_{\theta^{(t)}}(\mathbf{x})$, using weighted MLE with the samples, $\{\tilde{\mathbf{x}}_i^{(t)}\}_{i=1}^m$, and their corresponding EDA weights, $\{v_i\}_{i=1}^m$.
    Compute importance weights for the training data, $w_i \leftarrow p_{\theta^{(t)}}(\mathbf{x}_i)/p_{\theta^{(0)}}(\mathbf{x}_i), i = 1, \dots, n$.
    Retrain the surrogate using the re-weighted training data,
        $$\beta^{(t)} \leftarrow \arg\max_{\beta \in B} \frac{1}{n} \sum_{i=1}^n w_i \log p_\beta(y_i \mid \mathbf{x}_i).$$
end for
```

Note that

$$\begin{aligned}
\beta^{(t)} &= \arg\min_{\beta \in B} \mathbb{E}_{p_{\theta^{(t)}}(\mathbf{x})} \left[\int_{\mathbb{R}} p(y \mid \mathbf{x}) \log p(y \mid \mathbf{x}) dy - \int_{\mathbb{R}} p(y \mid \mathbf{x}) \log p_\beta(y \mid \mathbf{x}) dy \right] \\
&= \arg\max_{\beta \in B} \mathbb{E}_{p_{\theta^{(t)}}(\mathbf{x})} \left[\int_{\mathbb{R}} p(y \mid \mathbf{x}) \log p_\beta(y \mid \mathbf{x}) dy \right] \\
&= \arg\max_{\beta \in B} \mathbb{E}_{p_{\theta^{(t)}}(\mathbf{x})} \mathbb{E}_{p(y|\mathbf{x})} [\log p_\beta(y \mid \mathbf{x})].
\end{aligned}$$

We cannot query the ground truth, $p(y \mid \mathbf{x})$, but we do have labeled training data, $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where $\mathbf{x}_i \sim p_0(\mathbf{x})$ and $y_i \sim p(y \mid \mathbf{x} = \mathbf{x}_i)$ by definition. We therefore lever-

Algorithm 3 Procedure for evaluating MBO algorithms in superconductivity experiments. For each MBO algorithm in Tables 2.1, 2.2, 2.3, 2.4, the reported scores were the outputs of this procedure, averaged over 10 trials. Recall that $\mu_{\beta^{(t)}}(\mathbf{x}) \coloneqq \mathbb{E}_{\beta^{(t)}}[y \mid \mathbf{x}]$ denotes the expectation of the surrogate model at iteration t, while $\mathbb{E}[y \mid \mathbf{x}]$ denotes the ground-truth expectation.

```
Input: Sequence of samples, $\{(\tilde{\mathbf{x}}_i^{(t)}, \ldots, \tilde{\mathbf{x}}_m^{(t)})\}_{t=1}^T$, from each iteration of an MBO algorithm; their surrogate expectations, $\{(\mu_{\beta^{(t)}}(\tilde{\mathbf{x}}_i^{(t)}), \ldots, \mu_{\beta^{(t)}}(\tilde{\mathbf{x}}_m^{(t)}))\}_{t=1}^T$; percentile threshold, $Q = 80$.
Output: $\text{median}(\mu_{\text{GT,best}}), \max(\mu_{\text{GT,best}}), \rho$, PCI, RMSE
for $i = 1, \ldots, T$ do
    Compute $q_t \leftarrow Q$-th percentile of the surrogate expectations, $\{\mu_{\beta^{(t)}}(\tilde{\mathbf{x}}_i^{(t)})\}_{i=1}^m$.
end for
$t_{\text{best}} \leftarrow \arg\max_t q_t$ (pick the best iteration)
$\mathcal{I} \leftarrow \{i \in \{1, \ldots, m\} : \mu_{\beta^{(t_{\text{best}})}}(\tilde{\mathbf{x}}_i^{(t_{\text{best}})}) \geq q_{t_{\text{best}}}\}$ (pick best samples from best iteration)
$\mu_{\text{GT,best}} \leftarrow \{\mathbb{E}[y \mid \tilde{\mathbf{x}}_i^{t_{\text{best}}}] : i \in \mathcal{I}\}$
$\mu_{\text{GT}} \leftarrow (\mathbb{E}[y \mid \tilde{\mathbf{x}}_1^{t_{\text{best}}}], \ldots, \mathbb{E}[y \mid \tilde{\mathbf{x}}_m^{t_{\text{best}}}])$
$\mu_{\text{surrogate}} \leftarrow (\mu_{\beta^{(t_{\text{best}})}}(\tilde{\mathbf{x}}_1^{t_{\text{best}}}), \ldots, \mu_{\beta^{(t_{\text{best}})}}(\tilde{\mathbf{x}}_m^{t_{\text{best}}}))$
PCI $\leftarrow 100 \cdot \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \mathbf{1}[\mathbb{E}[y \mid \tilde{\mathbf{x}}_i^{t_{\text{best}}}] >$ maximum label in training data)]
$\rho \leftarrow \textsc{Spearman}(\mu_{\text{GT}}, \mu_{\text{surrogate}})$
RMSE $\leftarrow \text{RMSE}(\mu_{\text{GT}}, \mu_{\text{surrogate}})$
```

age importance sampling, using $p_0(\mathbf{x})$ as the proposal distribution, to obtain

$$\beta^{(t)} = \arg\max_{\beta \in B} \mathbb{E}_{p_0(\mathbf{x})} \mathbb{E}_{p(y|\mathbf{x})} \left[\frac{p_{\theta^{(t)}}(\mathbf{x})}{p_0(\mathbf{x})} \log p_\beta(y \mid \mathbf{x}) \right]. \tag{2.6}$$

Finally, we instantiate an importance sampling estimate of the objective in Eq. (2.6) with our labeled training data, to get a practical surrogate parameter update,

$$\beta^{(t)} = \arg\max_{\beta \in B} \frac{1}{n} \sum_{i=1}^{n} \frac{p_{\theta^{(t)}}(\mathbf{x}_i)}{p_0(\mathbf{x}_i)} \log p_\beta(y_i \mid \mathbf{x}_i). \tag{2.7}$$

This update is equivalent to fitting the surrogate parameters, $\beta^{(t)}$, by performing weighted MLE with the labeled training data, $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, and corresponding weights, $\{w_i\}_{i=1}^n$, where $w_i \coloneqq p_{\theta^{(t)}}(\mathbf{x}_i)/p_0(\mathbf{x}_i)$.

Variance of importance weights

The estimate of the log-likelihood used to retrain the surrogate, Eq. (2.7), is unbiased, but may have high variance due to the variance of the importance weights.

CbAS naturally controls the importance weight variance. Design procedures that leverage a trust region can naturally bound the variance of the importance weights. For instance, CbAS [76], developed in the context of a surrogate with fixed parameters, β, proposes estimating the training distribution conditioned on S as the search model:

$$p_\theta(\mathbf{x}) = p_0(\mathbf{x} \mid S) = P_\beta(S \mid \mathbf{x}) p_0(\mathbf{x}) / P_0(S), \tag{2.8}$$

where $P_0(S) = \int P_\beta(S \mid \mathbf{x}) p_0(\mathbf{x}) dx$. This prescribed search model yields the following importance weight variance.

Proposition 2.8.1. *For $p_\theta(\mathbf{x}) = p_0(\mathbf{x} \mid S)$, it holds that*

$$\mathrm{Var}_{p_0(\mathbf{x})}\left(\frac{p_\theta(\mathbf{x})}{p_0(\mathbf{x})}\right) = \frac{1}{P_0(S)} - 1.$$

That is, so long as S has non-neglible mass under data drawn from the training distribution, $p_0(\mathbf{x})$, we have reasonable control on the variance of the importance weights. Of course, if $P_0(S)$ is too small, this bound is not useful, but to have any hope of successful data-driven design it is only reasonable to expect this quantity to be non-negligible. This is similar to the experimental requirement, in directed evolution for protein design, that the initial data exhibit some "minimal functionality" with regards to the property of interest [86].

Proof. The variance of the importance weights can be written as

$$\mathrm{Var}_{p_0(\mathbf{x})}\left(\frac{p_0(\mathbf{x} \mid S)}{p_0(\mathbf{x})}\right) = d_2(p_0(\mathbf{x} \mid S) || p_0(\mathbf{x})) - 1,$$

where $d_2(p_0(\mathbf{x} \mid S) || p_0(\mathbf{x})) = \mathbb{E}_{p_0(\mathbf{x})}[(p_0(\mathbf{x} \mid S)/p_0(\mathbf{x}))^2]$ is the exponentiated Rényi-2 divergence. Then we have

$$\mathrm{Var}_{p_0(\mathbf{x})}\left(\frac{p_\theta(\mathbf{x})}{p_0(\mathbf{x})}\right) = d_2(p_0(\mathbf{x} \mid S) || p_0(\mathbf{x})) - 1 = \frac{1}{p_0(S)} - 1,$$

where the second equality is due to the property in Example 1 of [55]. □

This variance yields the following expression for the population version of the effective sample size:

$$n_e^* := \frac{n \mathbb{E}_{p_0(x)}\left[p_\theta(\mathbf{x})/p_0(\mathbf{x})\right]^2}{\mathbb{E}_{p_0(x)}\left[(p_\theta(\mathbf{x})/p_0(\mathbf{x}))^2\right]} = \frac{n}{\mathbb{E}_{p_0(x)}\left[(p_\theta(\mathbf{x})/p_0(\mathbf{x}))^2\right]} = nP_0(S).$$

Furthermore, CbAS proposes an iterative procedure to estimate $p_\theta(\mathbf{x})$. At iteration t, the algorithm seeks a variational approximation to $p^{(t)}(\mathbf{x}) \propto P_\beta(S^{(t)} \mid \mathbf{x}) p_0(\mathbf{x})$, where $S^{(t)} \supseteq S$. Since $P_0(S^{(t)} \mid \mathbf{x}) \geq P_0(S \mid \mathbf{x})$, the expressions above for the importance weight variance and effective sample size for the final search model prescribed by CbAS translate into upper and lower bounds, respectively, on the importance weight variance and effective sample size for the distributions $p^{(t)}(\mathbf{x})$ prescribed at each iteration.

2.9 Experimental details

Illustrative example

Ground truth and surrogate. For the ground-truth function $f : \mathbb{R} \to \mathbb{R}^+$, we used the sum of the densities of two Gaussian distributions, $\mathcal{N}_1(5, 1)$ and $\mathcal{N}_2(7, 0.25)$. For the expectation of the surrogate model, $\mu_\beta(\mathbf{x}) := \mathbb{E}_\beta[y \mid \mathbf{x}]$, we used kernel ridge regression with a radial basis function kernel as implemented in `scikit-learn`, with the default values for all hyperparameters. The variance of the surrogate model, $\sigma^2_\beta := \mathrm{Var}_\beta[y \mid \mathbf{x}]$, was set to the mean squared error between $\mu_\beta(\mathbf{x})$ and the training data labels, as estimated with 4-fold importance-weighted cross-validation when autofocusing [33].

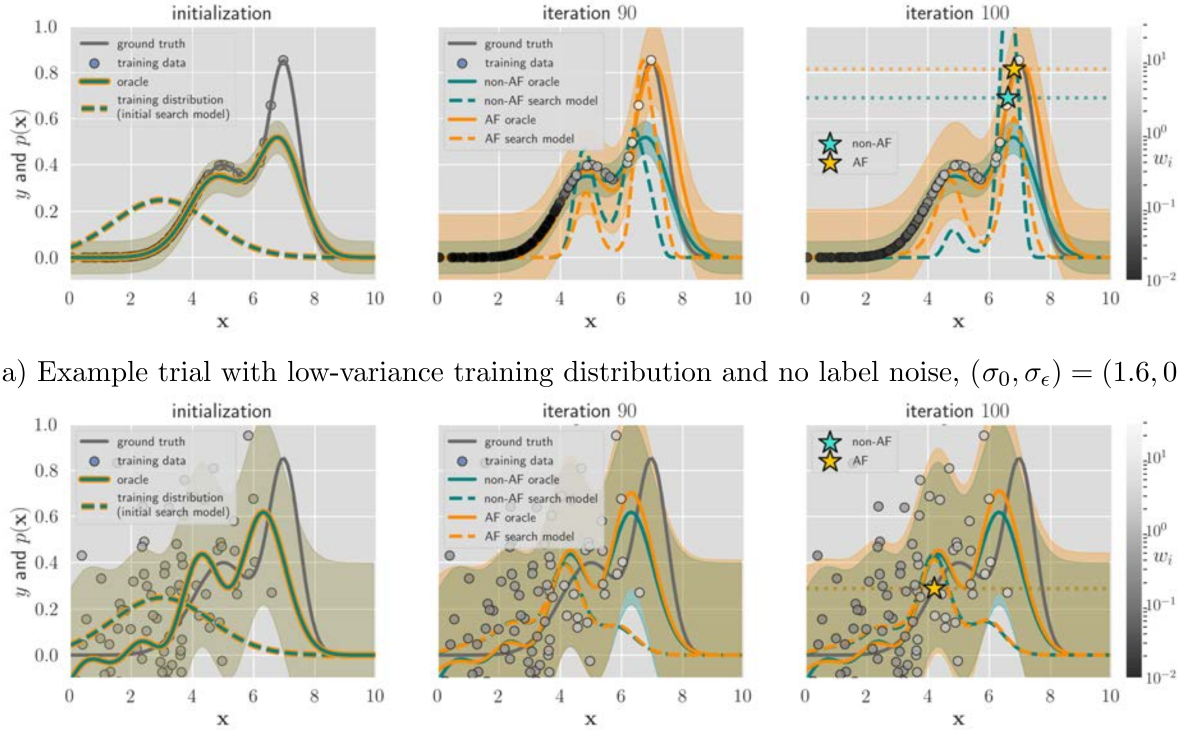
MBO algorithm. We used CbAS as follows. At iteration $t = 1, \ldots, 100$, similar to [76], we used the relaxed constraint set $S^{(t)} = \{y : y \geq \gamma_t\}$ where γ_t was the t^{th} percentile of the surrogate expectation, $\mu_\beta(\mathbf{x})$, when evaluated over $\mathbf{x} \in [0, 10]$. At the final iteration, $t = 100$, the constraint set is equivalent to the design goal of maximizing the surrogate expectation, $S^{(100)} = S = \{y : y \geq \max_{\mathbf{x}} \mu_\beta(\mathbf{x})\}$, which is the surrogate-based proxy to maximizing the ground-truth function, $f(\mathbf{x})$. At each iteration, we used numerical quadrature (`scipy.integrate.quad`) to compute the search model,

$$p^{(t)}(\mathbf{x}) = \frac{P_{\beta^{(t)}}(y \in S^{(t)} \mid \mathbf{x})\, p_0(\mathbf{x})}{\int_{\mathcal{X}} P_{\beta^{(t)}}(y \in S^{(t)} \mid \mathbf{x})\, p_0(\mathbf{x})}.$$

Numerical integration enabled us to use CbAS without a parametric search model, which otherwise would have been used to find a variational approximation to this distribution [76]. We also used numerical integration to compute the value of the design objective (Eq. (2.1)) achieved by the final search model, both with and without autofocusing.

Additional plots and discussion

For all tested settings of the variance of the training distribution, σ^2_0, and the variance of the label noise, σ^2_ϵ, autofocusing yielded positive improvement to the design objective (Eq. (2.1)) on average over 50 trials (Figure 2.2). For a more comprehensive understanding of the effects of autofocusing, here we pinpoint specific trials where autofocusing decreased the objective, compared to a naive approach with a fixed surrogate. Such trials were rare, and occurred in regimes where one would not reasonably expect autofocusing to provide a benefit. In particular, as discussed in Section 2.5, such regimes include when σ^2_0 is too small, such that training data are unlikely to be close to the global maximum, and when σ^2_0 is too large, such that the training data already include points around the global maximum and a fixed surrogate should be suitable for successful design. Similarly, when the label noise variance, σ^2_ϵ, is too large, the training data are no longer informative and no procedure should hope to perform well systematically. We now walk through each of these regimes.



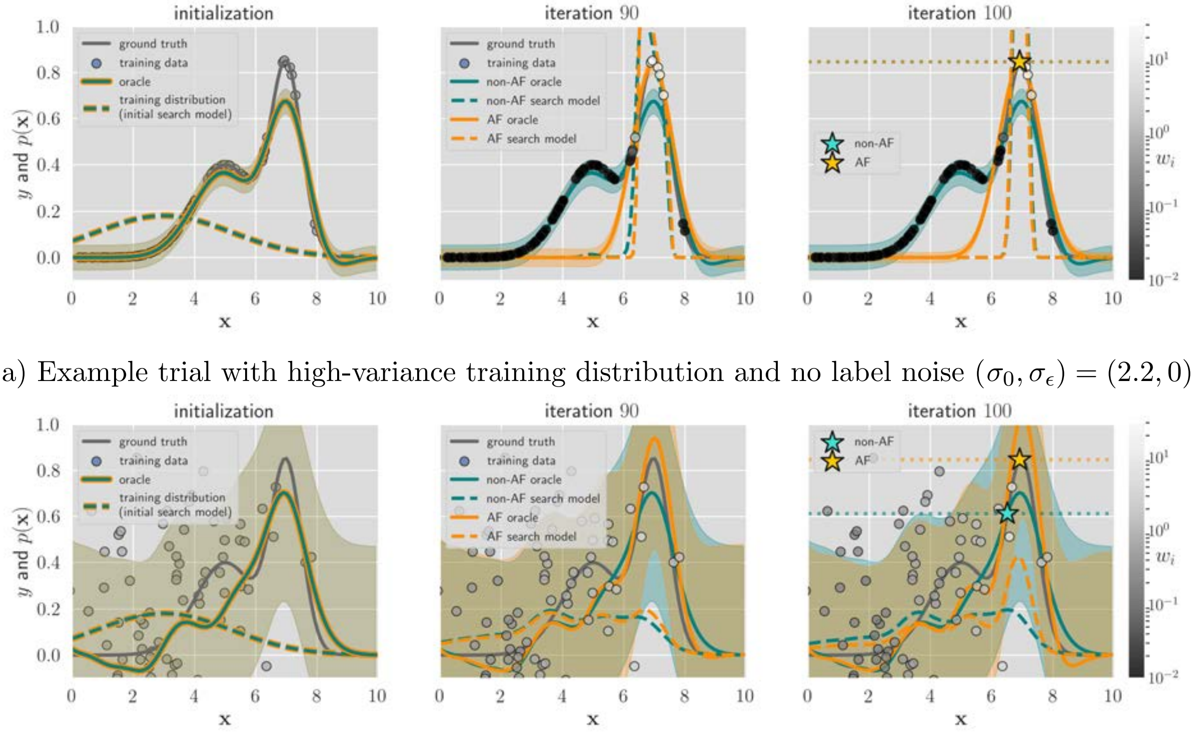
(a) Example trial with low-variance training distribution and no label noise, $(\sigma_0, \sigma_\epsilon) = (1.6, 0)$.

(b) Example trial with low-variance training distribution and high label noise, $(\sigma_0, \sigma_\epsilon) = (1.6, 0.38)$.

Figure 2.3: **Effect of autofocusing (AF) compared to without (non-AF) for low-variance training distributions and varying amounts of label noise.** Each row shows snapshots of CbAS in a different experimental regime, from initialization (leftmost panel), to an intermediate iteration (middle panel), to the final iteration (rightmost panel). As in Figure 2.1, the vertical axis represents both y values (for the surrogate and ground truth) and probability density values (of the training distribution, $p_0(\mathbf{x})$, and search distributions, $p_{\theta^{(t)}}(\mathbf{x})$). Shaded envelopes correspond to ±1 standard deviation of the oracles, $\sigma_{\beta^{(t)}}$, with the surrogate expectations, $\mu_{\beta^{(t)}}(\mathbf{x})$, shown as a solid line. Greyscale of training points corresponds to their importance weights used in autofocusing. In the rightmost panels, for easy visualization of the final search models achieved with and without AF, the stars and dotted horizontal lines indicate the ground-truth values corresponding to the points of maximum density.

When σ_0^2 was small and there was no label noise, we observed a few trials where the final search model placed less mass under the global maximum with autofocusing than without. This effect was due to increased standard deviation of the autofocused surrogate, induced by high variance of the importance weights (Figure 2.3a). When σ_0^2 was small and σ_ϵ^2 was extremely large, a few trials yielded lower final objectives with autofocusing by insignificant margins; in such cases, the label noise was overwhelming enough that the search model did not move much anyway, either with or without autofocusing (Figure 2.3b). Similarly, when σ_0^2 was large and there was no label noise, a few trials yielded lower final objectives with autofocusing than without, by insignificant margins (Figure 2.4a).

Interestingly, when the variances of both the training distribution and label noise were



(a) Example trial with high-variance training distribution and no label noise $(\sigma_0, \sigma_\epsilon) = (2.2, 0)$.

(b) Example trial with high-variance training distribution and high label noise $(\sigma_0, \sigma_\epsilon) = (2.2, 0.38)$.

Figure 2.4: **Effect of autofocusing (AF) compared to without (non-AF) for high-variance training distributions and varying amounts of label noise.** Continuation of Figure 2.3.

high, autofocusing yielded positive improvement for all trials. In this regime, by encouraging the surrogate to fit most accurately to the points with the highest labels, autofocusing resulted in search models with greater mass under the global maximum than the fixed-surrogate approach, which was more influenced by the extreme label noise (Figure 2.4b).

As discussed in Section 2.5, in practice it is often the case that 1) practitioners can collect reasonably informative training data for the application of interest, such that some exceptional examples are measured (corresponding to sufficiently large σ_0^2), and 2) there is always label noise, due to measurement error (corresponding to non-zero σ_ϵ^2). Thus, we expect many design applications in practice to fall in the intermediate regime where autofocusing tends to yield positive improvements over a fixed-surrogate approach (Figure 2.2, Table 2.1).

Designing superconductors with maximal critical temperature

Pre-processing. Each of the 21, 263 materials in the superconductivity data from [69] is represented by a vector of eighty-one real-valued features. We zero-centered and normalized

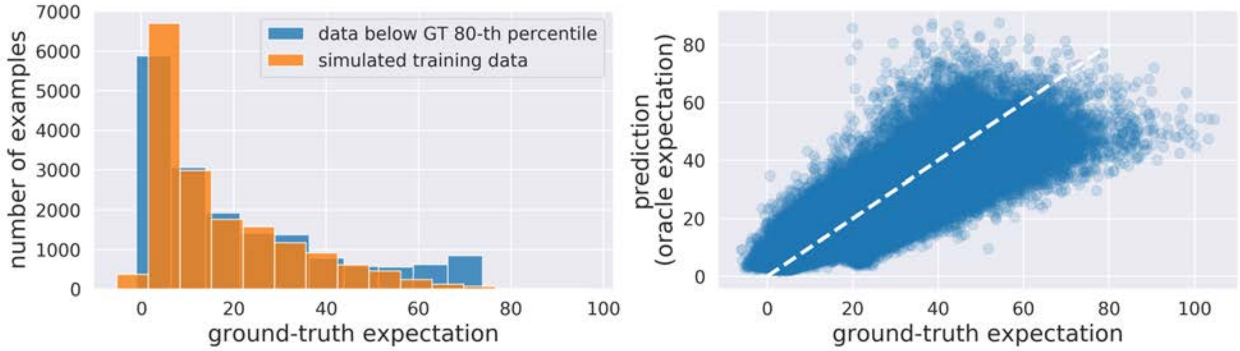


Figure 2.5: **Training distribution and initial surrogate for designing superconductors.** Simulated training data were generated from a training distribution, $p_0(\mathbf{x})$, which was a multivariate Gaussian fit to data points with ground-truth expectations below the 80th percentile. The left panel shows histograms of the ground-truth expectations of these original data points, and the ground-truth expectations of simulated training data. The right panel illustrates the error of an initial surrogate used in the experiments, by plotting the ground-truth and predicted labels of $10,000$ test points drawn from the training distribution. The RMSE here was 7.31.

each feature to have unit variance.

Ground-truth model. To construct the model of the ground-truth expectation, $\mathbb{E}[y \mid \mathbf{x}]$, we fit gradient-boosted regression trees using `xgboost` and the same hyperparameters reported in [69], which selected them using grid search. The one exception was that we used 200 trees instead of 750 trees, which yielded a hold-out root mean squared error (RMSE) of 9.51 compared to the hold-out RMSE of 9.5 reported in [69]. To remove collinear features noted in [69], we also performed feature selection by thresholding `xgboost`'s in-built feature weights, which quantifies how many times a feature is used to split the data across all trees. We kept the sixty most important features according to this score, which decreased the hold-out RMSE from 9.51 when using all the features to 9.45. The resulting input space for design was then $\mathcal{X} = \mathbb{R}^{60}$.

Training distribution. To construct the training distribution, we selected the $17,015$ points from the dataset whose ground-truth expectations were below the 80th percentile (equivalent to 73.8 degrees Kelvin, compared to the maximum of 138.3 degrees Kelvin in the full dataset). We used MLE with these points to fit a full-rank multivariate normal, which served as the training distribution, $p_0(\mathbf{x})$, from which we drew $n = 17,015$ simulated training points, $\{\mathbf{x}_i\}_{i=1}^n$, for each trial. For each $\mathbf{x}_i$ we drew one sample, $y_i \sim \mathcal{N}(\mathbb{E}[y \mid \mathbf{x}_i], 1)$, to obtain a noisy ground-truth label. This training distribution produced simulated training points with a distribution of ground-truth expectations, $\mathbb{E}[y \mid \mathbf{x}]$, reasonably comparable to that of the points from the original dataset (Figure 2.5, left panel).

Surrogate. For the surrogate, we trained an ensemble of three neural networks to maximize log-likelihood according to the method described in [62] (without adversarial examples). Each model in the ensemble had the architecture `Input(60)` → `Dense(100)` → `Dense(100)` → `Dense(100)` → `Dense(100)` → `Dense(10)` → `Dense(2)`, with `elu` nonlinearities everywhere except for linear output units. Out of the range of hidden layer numbers and sizes we tested, this architecture minimized RMSE on held-out data. Each model was trained using Adam [56] with a learning rate of 5×10^{-4} for a maximum of 2000 epochs, with a batch size of 64 and early stopping based on the log-likelihood of a validation set. Across the 10 trials, the initial oracles had hold-out RMSEs between 6.95 and 7.40 degrees Kelvin (Figure 2.5, right panel).

Autofocusing. During autofocusing, each model in the surrogate ensemble was retrained with the re-weighted training data, using the same optimization hyperparameters as the initial surrogate, except early stopping was based on the re-weighted log-likelihood of the validation set. For the results in Table 2.1, to help control the variance of the importance weights, we flattened the importance weights to w_i^α where $\alpha = 0.2$ [33] and also self-normalized them [51]. We found that autofocusing yielded similarly widespread benefits for a wide range of values of α, including $\alpha = 1$, which corresponds to a "pure" autofocusing strategy without variance control (Table 2.2).

MBO algorithms. Here, we provide a brief description of the different MBO algorithms used in the superconductivity experiments (Tables 2.1, 2.2, 2.3, 2.4, Figures 2.6, 2.7, 2.8). Wherever applicable, we anchor these descriptions in the notation and procedure of Algorithm 1.

- *Design by Adaptive Sampling (DbAS)* [76]. A basic EDA that anneals a sequence of relaxed constraint sets, $S^{(t)}$, such $S^{(t)} \supseteq S^{(t+1)} \supseteq S$, to iteratively solve Eq. (2.2). At iteration t, DbAS uses
$$V(\tilde{\mathbf{x}}_i^{(t)}) = P_{\beta^{(t-1)}}(y \in S^{(t)} \mid \tilde{\mathbf{x}}_i^{(t)}).$$

- *Conditioning by Adaptive Sampling (CbAS)* [76]. Seeks to estimate the training distribution conditioned on the desired constraint set S (Eq. (2.8)). Similar mechanistically to DbAS, as it involves constructing a sequence of relaxed constraint sets, but also incorporates an implicit trust region based on the training distribution. At iteration t, CbAS uses $V(\tilde{\mathbf{x}}_i^{(t)}) = (p_0(\tilde{\mathbf{x}}_i^{(t)})/p_{\theta^{(t-1)}}(\tilde{\mathbf{x}}_i^{(t)}))P_{\beta^{(t-1)}}(y \in S^{(t)} \mid \tilde{\mathbf{x}}_i^{(t)})$. See Algorithm 2; non-autofocused CbAS excludes Steps 7 and 8.

- *Reward-Weighted Regression (RWR)* [32]. An EDA used in the reinforcement learning community. At iteration t, RWR uses
$$V(\tilde{\mathbf{x}}_i^{(t)}) = v'_i / \sum_{j=1}^{m} v'_j,$$

where $v_i' = \exp(\gamma \mathbb{E}_{\beta^{(t-1)}}[y \mid \tilde{\mathbf{x}}_i^{(t)}])$ and $\gamma > 0$ is a hyperparameter.

- *"Feedback" Mechanism (FB)* [79]. A heuristic version of CbAS, which maintains samples from previous iterations to prevent the search model from changing too rapidly. At Step 4 in Algorithm 1, FB uses samples from the current iteration with surrogate expectations that surpass some percentile threshold, along with a subset of promising samples from previous iterations.

- *Cross-Entropy Method with Probability of Improvement (CEM-PI)* [76]. A baseline EDA that uses the cross-entropy method [8, 10] to maximize the probability of improvement, an acquisition function commonly used in Bayesian optimization [47]. (At iteration t, CEM-PI uses $V(\tilde{\mathbf{x}}_i^{(t)}) = \mathbf{1}[P_{\beta^{(t)}}(y \geq y_{\max} \mid \tilde{\mathbf{x}}_i^{(t)}) \geq \gamma_t]$, where $y_{\max}$ is the maximum label observed in the training data, and, following the cross-entropy method, γ_t is some percentile of the probabilities of improvement according to the surrogate, $\{P_{\beta^{(t)}}(y \geq y_{\max} \mid \tilde{\mathbf{x}}_i^{(t)})\}_{i=1}^m$.)

- *Covariance Matrix Adaptation Evolution Strategy (CMA-ES)* [26]. A state-of-the-art EDA developed for the special case of multivariate Gaussian search models. We used it to maximize the probability of improvement according to the surrogate,

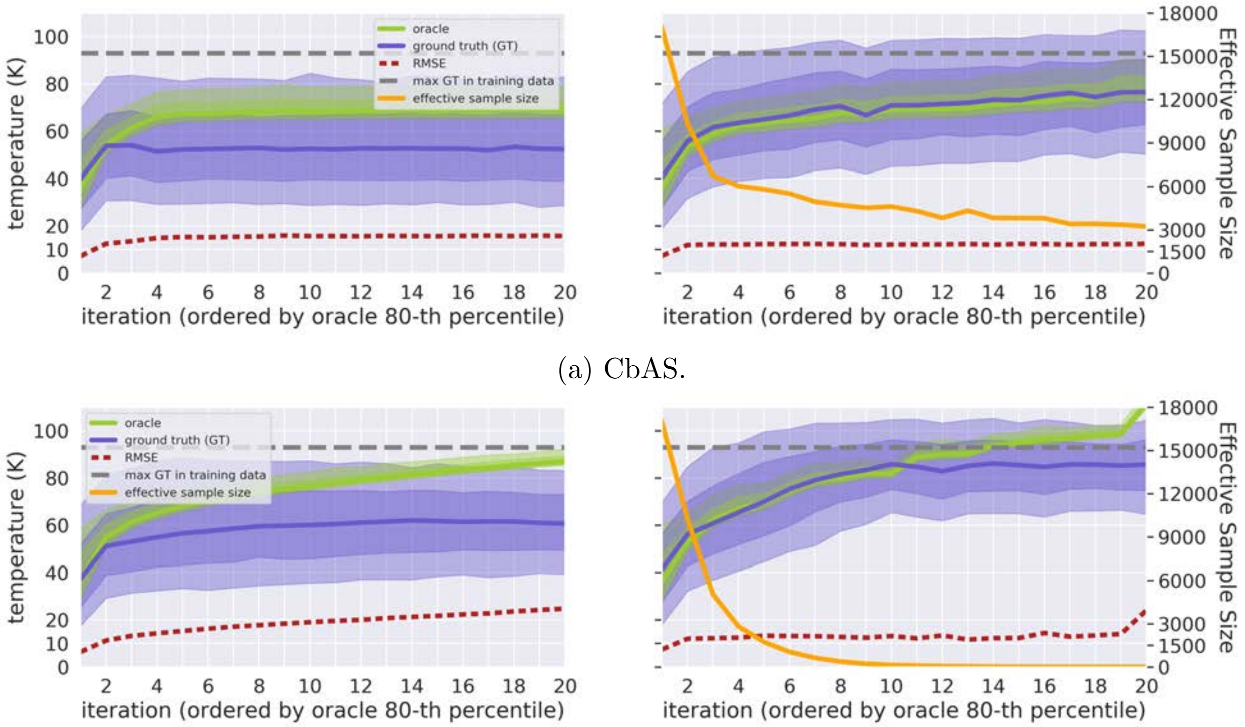
$$P_{\beta^{(t)}}(y \geq y_{\max} \mid \tilde{\mathbf{x}}_i^{(t)}).$$

CbAS, DbAS, FB, and CEM-PI all have hyperparameters corresponding to a percentile threshold (for CbAS and DbAS, this is used to construct the relaxed constraint sets). We set this hyperparameter to 90 for all these methods. For RWR, we set $\gamma = 0.01$, and for CMA-ES, we set the step size hyperparameter to $\sigma = 0.01$.

Additional experiments

Importance weight variance control. To see how much importance weight variance affects autofocusing, we conducted the same experiments as Table 2.1, except without flattening the weights to reduce variance (Table 2.2). For CbAS, DbAS, RWR, FB, and CEM-PI, autofocusing without variance control yielded statistically significant improvements to the majority of scores, though with somewhat lesser effect sizes than in Table 2.1 when the weights were flattened with $\alpha = 0.2$. For CMA-ES, the only significant improvement autofocusing rendered was to the Spearman correlation between the surrogate and the ground-truth expectations. Note that CMA-ES is a state-of-the-art method for optimizing a given objective with a multivariate Gaussian search model [26], which likely led to liberal movement of the search model away from the training distribution and therefore high importance weight variance.

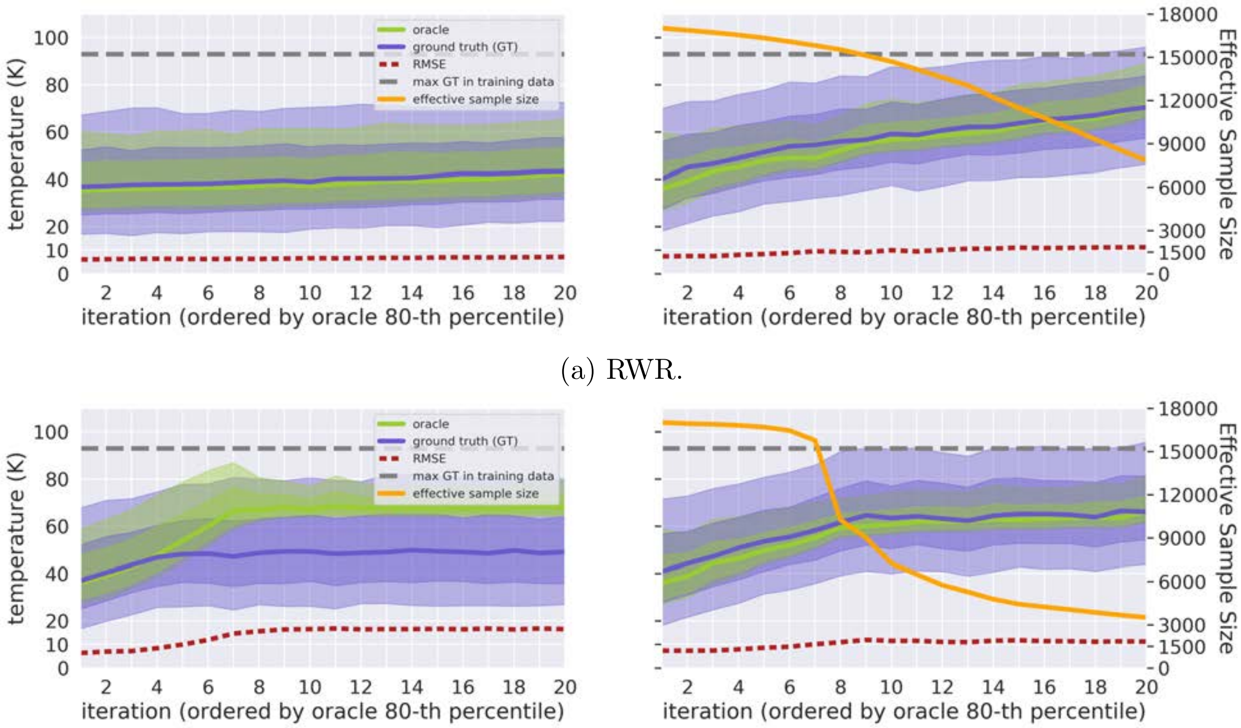
Surrogate capacity. To see how different surrogate capacities affect the improvements gained from autofocusing, we ran the same experiments as Table 2.1 with two different surrogate architectures. One architecture had higher capacity than the original surrogate (hidden



(a) CbAS.

(b) DbAS.

Figure 2.6: **Trajectories of DbAS and CbAS without (left) and with (right) autofocusing for designing superconducting materials.** Trajectories are shown for one example of the trials used to compute Table 2.1. At each iteration, we extract the samples with surrogate expectations greater than the 80^{th} percentile. For these samples, solid lines give the median surrogate (green) and ground-truth (indigo) expectations. The shaded regions capture 70 and 95 percent of these quantities. The RMSE at each iteration is between the surrogate and ground-truth expectations of all samples. The horizontal axis is sorted by increasing 80^{th} percentile of surrogate expectations (*i.e.*, the samples plotted at iteration 1 are from the iteration whose 80^{th} percentile of surrogate expectations was lowest). This ordering exposes the trend of whether the surrogate expectations of samples were correlated to their ground-truth expectations. Four more algorithms are shown in Figures 2.7,2.8.



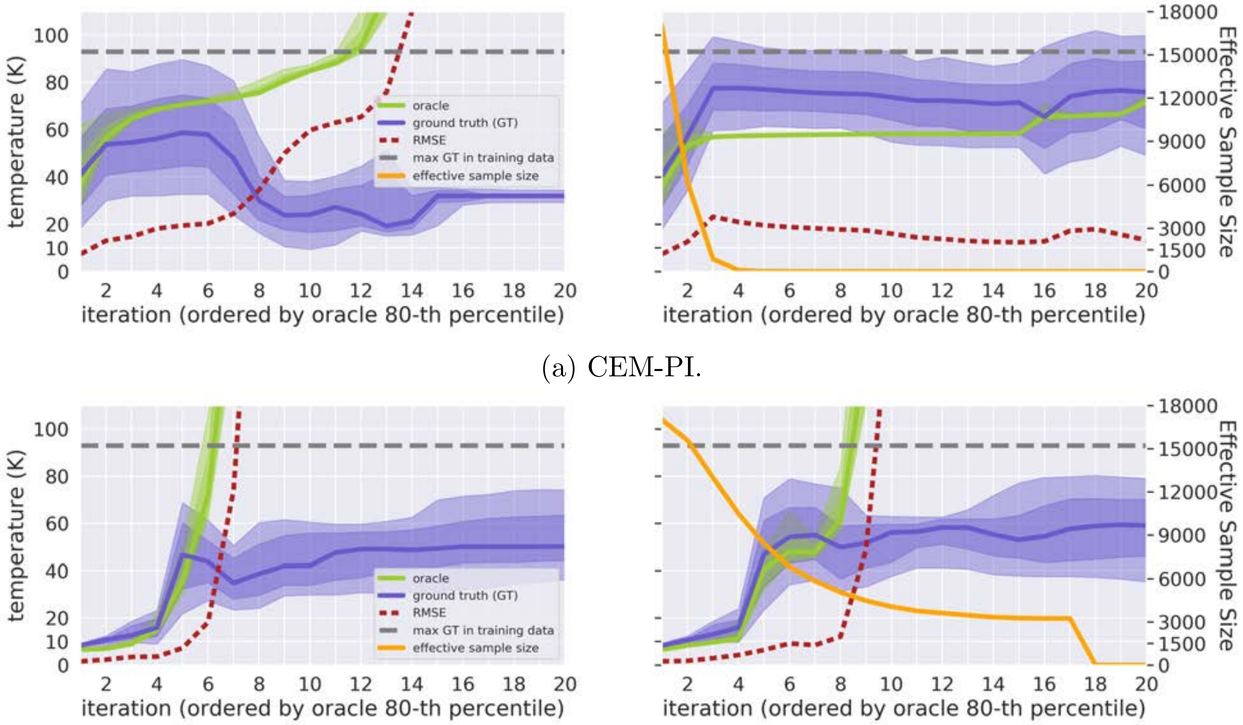
(a) RWR.

(b) FB.

Figure 2.7: **Trajectories of RWR and FB without (left) and with (right) autofocusing for designing superconducting materials.** Continuation of Figure 2.6.

Table 2.2: **Designing superconducting materials without importance weight variance control.** Same experiments and caption as Table 2.1, except with $\alpha = 1$ (no flattening of the importance weights to control variance).

	Median ↑	Max ↑	PCI ↑	ρ ↑	RMSE ↓	Median ↑	Max ↑	PCI ↑	ρ ↑	RMSE ↓
			CbAS					**DbAS**		
Original	51.5	103.8	0.11	0.05	17.2	57.0	98.4	0.11	0.01	29.6
Autofocused	73.2	116.0	2.29	0.56	12.8	69.4	109.9	0.68	0.01	27.4
Mean Diff.	**21.8****	**12.2****	**2.18****	**0.51****	**-4.4****	**12.4****	**11.5****	**0.58****	0.01	-2.2
			RWR					**FB**		
Original	43.4	102.0	0.05	0.92	7.4	9.2	100.6	0.14	40.09	17.5
Autofocused	68.5	113.4	1.34	0.63	14.2	63.4	110.8	0.63	0.49	11.2
Mean Diff.	**25.1****	**11.5****	**1.30****	**-0.29****	**6.8****	**14.2****	**10.2***	**0.50****	**0.40****	**-6.3****
			CEM-PI					**CMA-ES**		
Original	34.5	55.8	0.00	-0.16	148.3	42.1	69.4	0.00	0.27	27493.2
Autofocused	59.5	89.1 0.39		0.02	46.9	45.1	70.7	0.00	0.50	27944.9
Mean Diff.	**25.0***	**33.3***	**0.39***	0.18	**-101.4****	3.1	1.2	0	**0.22***	451.7



(a) CEM-PI.

(b) CMA-ES.

Figure 2.8: **Trajectories of CEM-PI and CMA-ES without (left) and with (right) autofocusing for designing superconducting materials.** Continuation of Figure 2.6.

layer sizes of `(200, 200, 100, 100, 10)` compared to `(100, 100, 100, 100, 10)`; Table 2.3), and one one had lower capacity (hidden layer sizes of `(100, 100, 10)`; Table 2.4).

Table 2.3: **Designing superconducting materials with higher-capacity model.** Same experiments and caption as Table 1, except using a surrogate architecture with hidden layers 200 → 200 → 100 → 100 → 10.

	Median ↑	Max ↑	PCI ↑	ρ ↑	RMSE ↓	Median ↑	Max ↑	PCI ↑	ρ ↑	RMSE ↓
			CbAS					**DbAS**		
Original	48.3	100.8	0.05	0.03	19.6	55.3	98.6	0.025	-0.02	32.1
Autofocused	79.0	119.4	4.35	0.55	13.5	81.6	113.3	5.33	0.01	27.0
Mean Diff.	**30.7****	**18.6****	**4.30****	**0.52****	**-6.1****	**26.4****	**14.8****	**5.30****	0.03	-5.1
			RWR					**FB**		
Original	36.5	81.3	0.00	-0.24	55.5	47.8	101.5	0.09	0.06	18.3
Autofocused	73.4	114.8	2.05	0.72	12.7	63.5	113.1	0.58	0.58	10.7
Mean Diff.	**36.9****	**33.4****	**2.05****	**0.97****	**-42.8****	**15.7****	**11.7****	**0.49****	**0.51****	**-7.5****
			CEM-PI					**CMA-ES**		
Original	48.2	58.3	0.00	0.09	271.4	39.0	63.1	0.00	0.26	6774.6
Autofocused	64.5	84.1	0.48	-0.14	61.07	53.1	79.0	0.01	0.48	10183.7
Mean Diff.	16.3	**25.9***	0.48	-0.22	**-210.3**	**14.1***	**15.9***	0.01	0.23	3409.1

Table 2.4: **Designing superconducting materials with lower-capacity model.** Same experiments and caption as Table 1, except using a surrogate architecture with hidden layers 100 → 100 → 10.

	Median ↑	Max ↑	PCI ↑	ρ ↑	RMSE ↓	Median ↑	Max ↑	PCI ↑	ρ ↑	RMSE ↓
Original	0.06	46.8	98.5	-0.03	23.8	0.02	56.3	97.7	0.00	37.0
Autofocused	1.4	67.0	114.3	0.52	13.0	1.3	72.5	108.4	0.04	27.6
Mean Diff.	**1.3****	**20.2****	**15.8****	**0.55****	**-10.9****	**1.3****	**16.2****	**10.7****	0.03	**-9.4****
			RWR					**FB**		
Original	0.00	30.9	76.8	-0.33	83.5	0.04	47.2	100.4	0.02	19.9
Autofocused	0.68	66.0	112.6	0.57	18.3	0.43	58.2	111.4	0.50	12.3
Mean Diff.	**0.68****	**35.1****	**35.8****	**0.90****	**-65.2****	**0.40****	**11.0****	**11.0****	**0.48****	**-7.6****
			CEM-PI					**CMA-ES**		
Original	0.00	36.3	46.2	-0.01	382.4	0.00	36.9	62.3	0.10	9587.1
Autofocused	0.04	53.9	71.3	-0.04	210.8	0.00	43.9	80.0	0.29	40858.6
Mean Diff.	0.04	17.6	**25.1***	-0.03	-171.6	0	**7.0***	**17.7***	**0.19****	**31271.5****

Chapter 3

Conformal prediction under feedback covariate shift for biomolecular design

The material in this chapter is based on work co-authored with Stephen Bates, Anastasios N. Angelopoulos, Jennifer Listgarten, and Michael I. Jordan [131].

3.1 Uncertainty quantification under feedback loops

Consider a protein engineer who is interested in designing a protein with high *fitness*—some real-valued measure of its desirability, such as fluorescence or therapeutic efficacy. The engineer has a data set of various protein sequences, denoted X_i, labeled with experimental measurements of their fitnesses, denoted Y_i, for $i = 1, \ldots, n$. The *design problem* is to propose a novel sequence, X_{test}, that has higher fitness, Y_{test}, than any of these. To this end, the engineer trains a regression model on the data set, then identifies a novel sequence that the model predicts to be more fit than the training sequences. Can she trust the model's prediction for the designed sequence?

This is an important question to answer, not just for the protein design problem just described, but for any deployment of machine learning where the test data depends on the training data. More broadly, settings ranging from Bayesian optimization to active learning to strategic classification involve *feedback loops* in which the learned model and data influence each other in turn. As feedback loops violate the standard assumptions of machine learning algorithms, we must be able to diagnose when a model's predictions can and cannot be trusted in their presence.

In this work, we address the problem of uncertainty quantification when the training and test data exhibit a type of dependence that we call *feedback covariate shift* (FCS). A joint distribution of training and test data falls under FCS if it satisfies two conditions. First, the test input, X_{test}, is selected based on independently and identically distributed (i.i.d.) training data, $(X_1, Y_1), \ldots, (X_n, Y_n)$. That is, the distribution of X_{test} is a function of the training data. Second, $P_{Y|X}$, the ground-truth distribution of the label, Y, given any input,

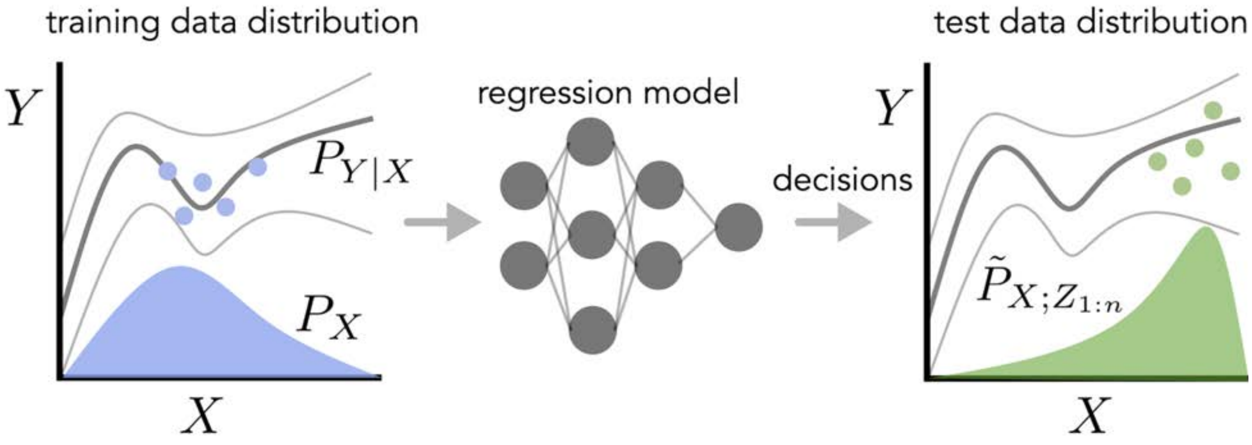


Figure 3.1: **Feedback covariate shift.** In the left graph, the blue distribution represents the training input distribution, P_X. The dark gray line sandwiched by lighter gray lines represents the mean ± the standard deviation of $P_{Y|X}$, the conditional distribution of the label given the input, which does not change between the training and test data distributions (left and right graphs, respectively). The blue dots represent training data, $Z_{1:n} = \{Z_1, \ldots, Z_n\}$, where $Z_i = (X_i, Y_i)$, which is used to fit a regression model (middle). Algorithms that use that trained model to make decisions, such as in design problems, active learning, and Bayesian optimization give rise to a new test-time input distribution, $P_{X;Z_{1:n}}$ (right graph, green distribution). The green dots represent test data.

X, does not change between the training and test data distributions. For example, returning to the example of protein design, the training data is used to select the designed protein, X_{test}; the distribution of X_{test} is determined by some optimization algorithm that calls the regression model in order to design the protein. However, since the fitness of any given sequence is some property dictated by nature, $P_{Y|X}$ stays fixed. Representative examples of FCS include:

- **Algorithms that use predictive models to explicitly choose the test distribution**, including the design of proteins, small molecules, and materials with favorable properties, active learning, adaptive experimental design, Bayesian optimization, and machine learning-guided scientific discovery.

- **Algorithms that use predictive models to perform actions that change a system's state**, such as autonomous driving algorithms that use computer vision systems.

We anchor our discussion and experiments by focusing on protein design problems. However, the methods and insights developed herein are applicable to a variety of FCS problems.

Quantifying uncertainty with valid confidence sets

Given a regression model of interest, μ, we quantify its uncertainty on an input with a *confidence set*. A confidence set is a function, $C : \mathcal{X} \to 2^{\mathbb{R}}$, that maps a point from some input space, $\mathcal{X}$, to a set of real values that the model considers to be plausible labels.[1] Informally, we will examine the model's error on the training data in order to quantify its uncertainty about the label, Y_{test}, of an input, X_{test}. Formally, using the notation $Z_i = (X_i, Y_i), i = 1, \ldots, n$ and $Z_{\text{test}} = (X_{\text{test}}, Y_{\text{test}})$, our goal is to construct confidence sets that have the frequentist statistical property known as *coverage*.

Definition 1. *Consider data points from some joint distribution, $(Z_1, \ldots, Z_n, Z_{test}) \sim \mathcal{P}$. Given a miscoverage level, $\alpha \in (0, 1)$, a confidence set, $C : \mathcal{X} \to 2^{\mathbb{R}}$, which may depend on $Z_1, \ldots, Z_n$, provides coverage under $\mathcal{P}$ if*

$$\mathbb{P}\left(Y_{test} \in C(X_{test})\right) \geq 1 - \alpha, \tag{3.1}$$

where the probability is over all $n + 1$ data points, $(Z_1, \ldots, Z_n, Z_{test}) \sim \mathcal{P}$.

There are three important aspects of this definition. First, coverage is with respect to a particular joint distribution of the training and test data, $\mathcal{P}$, as the probability statement in Eq. (3.1) is over random draws of all $n + 1$ data points. That is, if one draws $(Z_1, \ldots, Z_n, Z_{\text{test}}) \sim \mathcal{P}$ and constructs the confidence set for X_{test} based on a regression model fit to $(Z_1, \ldots, Z_n)$, then the confidence set contains the true test label, Y_{test}, a fraction of $1 - \alpha$ of the time. In this work, $\mathcal{P}$ can be any distribution captured by FCS, as we describe later in more detail.

Second, note that Eq. (3.1) is a finite-sample statement: it holds for any number of training data points, n. Finally, coverage is a marginal probability statement, which averages over all the randomness in the training and test data; it is not a statement about conditional probabilities, such as $\mathbb{P}(Y_{\text{test}} \in C(X_{\text{test}}) \mid X_{\text{test}})$. We will call a family of confidence sets, C_α, indexed by the miscoverage level, $\alpha \in (0, 1)$, *valid* if they provide coverage for all $\alpha \in (0, 1)$.

When the training and test data are exchangeable (*e.g.*, independently and identically distributed), *conformal prediction* is an approach for constructing valid confidence sets for any regression model [11, 23, 71]. Though recent work has extended the methodology to certain forms of distribution shift [84, 91, 112, 118, 119], to our knowledge no existing approach can produce valid confidence sets when *the test data depends on the training data.* Here, we generalize conformal prediction to the FCS setting, enabling uncertainty quantification under this prevalent type of dependence between training and test data.

Our contributions

First, we formalize the concept of feedback covariate shift, which describes a type of distribution shift that emerges under feedback loops between learned models and the data they

[1] We will use the term *confidence set* to refer to both this function and the output of this function for a particular input; the distinction will be clear from the context.

operate on. Second, we introduce a generalization of conformal prediction that produces valid confidence sets under feedback covariate shift for any regression model. We also introduce randomized versions of these confidence sets that achieve a stronger property called *exact coverage*. Finally, we demonstrate the use of our method to quantify uncertainty for the predicted fitness of designed proteins, using several real data sets.

We recommend using our method for design algorithm selection, as it enables practitioners to identify settings of algorithm hyperparameters that achieve acceptable trade-offs between high predictions and low predictive uncertainty.

Prior work

Our study investigates uncertainty quantification in a setting that brings together the well-studied concept of covariate shift [12, 22, 33, 40] with feedback between learned models and data distributions, a widespread phenomenon in real-world deployments of machine learning [58, 99]. Indeed, beyond the design problem, feedback covariate shift is one way of describing and generalizing the dependence between data at successive iterations of active learning, adaptive experimental design, and Bayesian optimization.

Our work builds upon *conformal prediction*, a framework for constructing confidence sets that satisfy the finite-sample coverage property in Eq. (3.1) for arbitrary model classes [9, 23, 144]. Though originally based on the premise of exchangeable (*e.g.*, independently and identically distributed) training and test data, the framework has since been generalized to handle various forms of distribution shift, including covariate shift [84, 118], label shift [119], arbitrary distribution shifts in an online setting [112], and test distributions that are nearby the training distribution [91]. Conformal approaches have also been used to detect distribution shift [93, 105, 108, 134, 135, 138, 146].

We call particular attention to the work of Tibshirani et al. [84] on conformal prediction in the context of covariate shift, whose technical machinery we adapt to generalize conformal prediction to feedback covariate shift. In covariate shift, the training and test input distributions differ, but, critically, the training and test data are still independent; we henceforth refer to this setting as *standard* covariate shift. The chief innovation of our work is to formalize and address a ubiquitous type of dependence between training and test data that is absent from standard covariate shift, and, to the best of our knowledge, absent from any other form of distribution shift to which conformal approaches have been generalized.

For the design problem, in which a regression model is used to propose new inputs—such as a protein with desired properties—it is important to consider the predictive uncertainty of the designed inputs, so that we do not enter "pathological" regions of the input space where the model's predictions are desirable but untrustworthy [76, 92]. Gaussian process regression (GPR) models are popular tools for addressing this issue, and algorithms that leverage their posterior predictive variance [16, 47] have been used to design enzymes with enhanced thermostability and catalytic activity [52, 113]. Despite these successes, it is not clear how to obtain practically meaningful theoretical guarantees for the posterior predictive variance, and consequently to understand in what sense we can trust it. Similarly, ensembling strategies such as [62], which are increasingly being used to quantify uncertainty for deep neural networks [76, 87, 92, 98], as well as uncertainty estimates that are explicitly learned by deep models [121] do not come with formal guarantees. A major advantage of conformal prediction is that it can be applied to any modelling strategy, and can be used to calibrate any existing uncertainty quantification approach, including those aforementioned.

3.2 Conformal prediction under feedback covariate shift

Feedback covariate shift

We begin by formalizing feedback covariate shift (FCS), which describes a setting in which the test data depends on the training data, but the relationship between inputs and labels remains fixed.

We first set up our notation. Recall that we let $Z_i = (X_i, Y_i)$, $i = 1, \ldots, n$, denote n independently and identically distributed (i.i.d.) training data points comprising inputs, $X_i \in \mathcal{X}$, and labels, $Y_i \in \mathbb{R}$. Similarly, let $Z_{\text{test}} = (X_{\text{test}}, Y_{\text{test}})$ denote the test data point. We use $Z_{1:n} = \{Z_1, \ldots, Z_n\}$ to denote the multiset of the training data, in which values are unordered but multiple instances of the same value appear according to their multiplicity. We also use the shorthand $Z_{-i} = Z_{1:n} \setminus \{Z_i\}$, which is a multiset of $n-1$ values that we refer to as the *i-th leave-one-out training data set*.

FCS describes a class of joint distributions over $(Z_1, \ldots, Z_n, Z_{\text{test}})$ that have the dependency structure described informally in the Introduction. Formally, we say that training and test data exhibit FCS when they can be generated according to the following three steps.

1. The training data, $(Z_1, \ldots, Z_n)$, are drawn i.i.d. from some distribution:

$$\begin{aligned} X_i &\overset{\text{i.i.d}}{\sim} P_X, \\ Y_i &\sim P_{Y \mid X_i},\ i = 1, \ldots, n. \end{aligned}$$

2. The realized training data induces a new input distribution over $\mathcal{X}$, denoted $\tilde{P}_{X;Z_{1:n}}$ to emphasize its dependence on the training data, $Z_{1:n}$.

3. The test input is drawn from this new input distribution, and its label is drawn from the unchanged conditional distribution:

$$\begin{aligned} X_{\text{test}} &\sim \tilde{P}_{X;Z_{1:n}} \\ Y_{\text{test}} &\sim P_{Y \mid X_{\text{test}}}. \end{aligned}$$

The key object in this formulation is the test input distribution, $\tilde{P}_{X;Z_{1:n}}$. Prior to collecting the training data, $Z_{1:n}$, the specific test input distribution is not yet known. The observed training data induces a distribution of test inputs, $\tilde{P}_{X;Z_{1:n}}$, that the model encounters at test time (for example, through any of the mechanisms summarized in the Introduction).

This is an expressive framework: the object $\tilde{P}_{X;Z_{1:n}}$ can be an arbitrarily complicated mapping from a data set of size n to an input distribution, so long as it is invariant to the order of the data points. There are no other constraints on this mapping; it need not exhibit any notion of smoothness, for example. In particular, FCS encapsulates any design algorithm that makes use of a regression model fit to the training data, $Z_{1:n}$, in order to propose designed inputs.

Conformal prediction for exchangeable data

To explain how to construct valid confidence sets under FCS, we first walk through the intuition behind conformal prediction in the setting of exchangeable data, then present the adaptation to accommodate FCS.

Score function. First, we introduce the notion of a *score function*, $S : (\mathcal{X} \times \mathbb{R}) \times (\mathcal{X} \times \mathbb{R})^m \to \mathbb{R}$, which is an engineering choice that quantifies how well a given data point "conforms" to a multiset of m data points, in the sense of evaluating whether the data point comes from the same conditional distribution, $P_{Y|X}$, as the data points in the multiset.[2] A representative example is the residual score function, $S((X, Y), D) = |Y - \mu_D(X)|$, where D is a multiset of m data points and μ_D is a regression model trained on D. A large residual signifies a data point that the model cannot easily predict, which suggests it does not obey the input-label relationship present in the training data.

More generally, we can choose the score to be any notion of uncertainty of a trained model on the point (X, Y), heuristic or otherwise, such as the posterior predictive variance of a Gaussian process regression model [52, 113], the variance of the predictions from an ensemble of neural networks [62, 87, 98], uncertainty estimates learned by deep models [88], or even the outputs of other calibration procedures [70]. Regardless of the choice of the score function, conformal prediction produces valid confidence sets; however, the particular choice of score function will determine the size, and therefore, informativeness, of the resulting sets. Roughly speaking, a score function that better reflects the likelihood of observing the given point, (X, Y), under the true conditional distribution that governs D, $P_{Y|X}$, results in smaller valid confidence sets.

Imitating exchangeable scores. At a high level, conformal prediction is based on the observation that when the training and test data are exchangeable, their scores are also exchangeable. More concretely, assume we use the residual score function, $S((X, Y), D) = |Y - \mu_D(X)|$, for some regression model class. Now imagine that we know the label, Y_{test}, for

[2] Since the second argument is a multiset of data points, the score function must be invariant to the order of these data points. For example, when using the residual as the score, the regression model must be trained in a way that is agnostic to the order of the data points.

the test input, X_{test}. For each of the $n+1$ training and test data points, $(Z_1, \ldots, Z_n, Z_{\text{test}})$, we can compute the score using a regression model trained on the remaining n data points; the resulting $n+1$ scores are exchangeable.

In reality, of course, we do not know the true label of the test input. However, this key property—that the scores of exchangeable data yield exchangeable scores—enables us to construct valid confidence sets by including all "candidate" values of the test label, $y \in \mathbb{R}$, that yield scores for the $n+1$ data points (the training data points along with the candidate test data point, (X_{test}, y)) that appear to be exchangeable. For a given candidate label, the conformal approach assesses whether or not this is true by comparing the score of the candidate test data point to an appropriately chosen quantile of the training data scores.

Conformal prediction under FCS

When the training and test data are under FCS, their scores are no longer exchangeable, since the training and test inputs are neither independent nor from the same distribution. Our solution to this problem will be to weight each training and test data point to take into account these two factors. Thereafter, we can proceed with the conformal approach of including all candidate labels such that the (weighted) candidate test data point is sufficiently similar to the (weighted) training data points. Toward this end, we introduce two quantities: (1) a likelihood ratio function, which will be used to define the weights, and (2) the quantile of a distribution, which will be used to assess whether a candidate test data point conforms to the training data.

The likelihood ratio function for an input, X, which depends on a multiset of data points, D, is given by

$$v(X; D) = \frac{\tilde{p}_{X;D}(X)}{p_X(X)}, \tag{3.2}$$

where $\tilde{p}_{X;D}$ and p_X denote the densities of the test and training input distributions, respectively, and the test input distribution is the particular one indexed by the data set, D.

This quantity is the ratio of the likelihoods under these two distributions, and as such, is reminiscent of weights used to modify various statistical procedures to accommodate standard covariate shift [22, 33, 84]. What distinguishes its use here is that our particular likelihood ratio is indexed by a multiset and depends on which data point is being evaluated as well as the candidate label, as will become clear shortly.

Consider a discrete distribution with probability masses $p_1, \ldots, p_n$ located at support points $s_1, \ldots, s_n$, respectively, where $s_i \in \mathbb{R}$ and $p_i \geq 0, \sum_i p_i = 1$. We define the *$\beta$-quantile* of this distribution as

$$\textsc{Quantile}_\beta\left(\sum_{i=1}^{n} p_i\, \delta_{s_i}\right) = \inf\left\{s : \sum_{i: s_i \leq s} p_i \geq \beta\right\},$$

where δ_{s_i} is a unit point mass at s_i.

We now define the confidence set. For any score function, S, any miscoverage level, $\alpha \in (0,1)$, and any test input, $X_{\text{test}} \in \mathcal{X}$, define the *full conformal* confidence set as

$$C_\alpha(X_{\text{test}}) = \left\{ y \in \mathbb{R} : S_{n+1}(X_{\text{test}}, y) \leq \textsc{Quantile}_{1-\alpha}\left(\sum_{i=1}^{n+1} w_i^y(X_{\text{test}})\, \delta_{S_i(X_{\text{test}}, y)} \right) \right\}, \tag{3.3}$$

where

$$\begin{aligned} S_i(X_{\text{test}}, y) &= S(Z_i, Z_{-i} \cup \{(X_{\text{test}}, y)\}),\ i = 1, \dots, n, \\ S_{n+1}(X_{\text{test}}, y) &= S((X_{\text{test}}, y), Z_{1:n}), \end{aligned}$$

which are the scores for each of the training and candidate test data points, when compared to the remaining n data points, and the weights for these scores are given by

$$\begin{aligned} w_i^y(X_{\text{test}}) &\propto v(X_i; Z_{-i} \cup \{(X_{\text{test}}, y)\}),\ i = 1, \dots, n, \\ w_{n+1}^y(X_{\text{test}}) &\propto v(X_{\text{test}}; Z_{1:n}), \end{aligned} \tag{3.4}$$

which are normalized such that $\sum_{i=1}^{n+1} w_i^y(X_{\text{test}}) = 1$.

In words, the confidence set in Eq. (3.3) includes all real values, $y \in \mathbb{R}$, such that the "candidate" test data point, (X_{test}, y), has a score that is sufficiently similar to the scores of the training data. Specifically, the score of the candidate test data point needs to be smaller than the $(1-\alpha)$-quantile of the weighted scores of all $n+1$ data points (the n training data points as well as the candidate test data point), where the i-th data point is weighted by $w_i^y(X_{\text{test}})$.

Our main result is that this confidence set provides coverage under FCS (see Section 3.5 for the proof).

Theorem 1. *Suppose data are generated under feedback covariate shift and assume $\tilde{P}_{X;D}$ is absolutely continuous with respect to P_X for all possible values of D. Then, for any miscoverage level, $\alpha \in (0,1)$, the full conformal confidence set, C_α, in Eq.* (3.3) *satisfies the coverage property in Eq.* (3.1)*, namely, $\mathbb{P}(Y_{test} \in C_\alpha(X_{test})) \geq 1 - \alpha$.*

Since we can supply any domain-specific notion of uncertainty as the score function, this result implies we can interpret the condition in Eq. (3.3) as a calibration of the provided score function that guarantees coverage. That is, our conformal approach can complement any existing uncertainty quantification method by endowing it with coverage under FCS.

We note that although Theorem 1 provides a lower bound on the probability $\mathbb{P}(Y_{\text{test}} \in C_\alpha(X_{\text{test}}))$, one cannot establish a corresponding upper bound without further assumptions on the training and test input distributions. However, by introducing randomization to the β-quantile, we can construct a randomized version of the confidence set, $C_\alpha^{\text{rand}}(X_{\text{test}})$, that is not conservative and satisfies $\mathbb{P}(Y_{\text{test}} \in C_\alpha^{\text{rand}}(X_{\text{test}})) = 1 - \alpha$, a property called *exact coverage*. See Theorem 3 for details.

Estimating confidence sets in practice. In practice, one cannot check all possible candidate labels, $y \in \mathbb{R}$, to construct a confidence set. Instead, as done in previous work on conformal prediction, we estimate $C_\alpha(X_{\text{test}})$ by defining a finite grid of candidate labels, $\mathcal{Y} \subset \mathbb{R}$, and checking the condition in Eq. (3.3) for all $y \in \mathcal{Y}$. Algorithm 4 outlines a generic recipe for computing $C_\alpha(X_{\text{test}})$ for a given test input; see Section 3.2 for important special cases in which $C_\alpha(X_{\text{test}})$ can be computed more efficiently.

Algorithm 4 Pseudocode for approximately computing $C_\alpha(X_{\text{test}})$

Input: Training data, $(Z_1, \dots, Z_n)$, where $Z_i = (X_i, Y_i)$; test input, X_{test}; finite grid of candidate labels, $\mathcal{Y} \subset \mathbb{R}$; likelihood ratio function subroutine, $v(\cdot\,;\cdot)$; and score function subroutine $S(\cdot,\cdot)$.

Output: Confidence set, $C_\alpha(X_{\text{test}}) \subset \mathcal{Y}$.

```
$C_\alpha(X_{\text{test}}) \leftarrow \emptyset$
Compute $v(X_{\text{test}}; Z_{1:n})$
for $y \in \mathcal{Y}$ do
    for $i = 1, \dots, n$ do
        Compute $S_i(X_{\text{test}}, y)$ and $v(X_i; Z_{-i} \cup \{(X_{\text{test}}, y)\})$
    end for
    Compute $S_{n+1}(X_{\text{test}}, y)$
    for $i = 1, \dots, n+1$ do
        Normalize $w_i^y(X_{\text{test}})$ according to Eq. (3.4)
    end for
    $q_y \leftarrow \text{QUANTILE}_{1-\alpha}\left(\sum_{i=1}^{n+1} w_i^y(X_{\text{test}})\, \delta_{S_i(X_{\text{test}},y)}\right)$
    if $S_{n+1}(X_{\text{test}}, y) \le q_y$ then
        $C_\alpha(X_{\text{test}}) \leftarrow C_\alpha(X_{\text{test}}) \cup \{y\}$
    end if
end for
```

Relationship with exchangeable and standard covariate shift settings. The weights assigned to each score, $w_i^y(X_{\text{test}})$ in Eq. (3.4), are the distinguishing factor between the confidence sets constructed by conformal approaches for the exchangeable, standard covariate shift, and FCS settings. For exchangeable training and test data, these weights are simply $1/(n+1)$. To accommodate standard covariate shift, where the training and test data are independent, these weights are also normalized likelihood ratios—but, importantly, the test input distribution in the numerator is fixed, rather than data-dependent as in the FCS setting [84]. That is, the weights are defined using one fixed likelihood ratio function, $v(\cdot) = \tilde{p}_X(\cdot)/p_X(\cdot)$, where $\tilde{p}_X$ is the density of the single test input distribution under consideration.

In contrast, under FCS, observe that the likelihood ratio that is evaluated in Eq. (3.4), $v(\cdot; D)$ from Eq. (3.2), is different for each of the $n+1$ training and candidate test data points and for each candidate label, $y \in \mathbb{R}$. To weight the i-th training score, we evaluate the likelihood ratio of X_i where the test input distribution is the one induced by $Z_{-i} \cup \{(X_{\text{test}}, y)\}$,

$$v(X_i; Z_{-i} \cup \{(X_{\text{test}}, y)\}) = \frac{\tilde{p}_{X;Z_{-i}\cup\{(X_{\text{test}},y)\}}(X_i)}{p_X(X_i)}.$$

That is, the weights under FCS take into account not just a single test input distribution, but every test input distribution that can be induced when we treat a leave-one-out training data set combined with a candidate test data point, $Z_{-i} \cup \{(X_{\text{test}}, y)\}$, as the training data.

To further appreciate the relationship between the standard and feedback covariate shift settings, consider the weights used in the standard covariate shift approach if we treat $P_{X;Z_{1:n}}$ as the test input distribution. The extent to which $P_{X;Z_{1:n}}$ differs from $P_{X;Z_{-i}\cup\{(X_{\text{test}},y)\}}$, for any $i = 1, \dots, n$ and $y \in \mathbb{R}$, determines the extent to which the weights used under standard covariate shift deviate from those used under FCS. In other words, since $Z_{1:n}$ and $Z_{-i} \cup \{(X_{\text{test}}, y\}$ differ in exactly one data point, the similarity between the standard covariate shift and FCS weights depends on the "smoothness" of the mapping from D to $\tilde{P}_{X;D}$. For example, the more algorithmically stable the learning algorithm through which $\tilde{P}_{X;D}$ depends on D is, the more similar these weights will be.

Input distributions are often known in the design problem. The design problem is a unique setting in which we have control over the data-dependent test input distribution, $P_{X;D}$, since we choose the procedure used to design an input. In the simplest case, some design procedures sample from a distribution whose form is explicitly chosen, such as an energy-based model whose energy function is proportional to the predictions from a trained regression model [110], or a model whose parameters are set by solving an optimization problem (*e.g.*, the training of a generative model) [73, 76, 80, 92, 101, 107, 114, 120, 124]. In either setting, we know the exact form of the test input distribution, which also absolves the need for density estimation.

In other cases, the design procedure involves iteratively applying a gradient to, or otherwise locally modifying, an initial input in order to produce a designed input [61, 68, 97, 103, 109, 111]. Due to randomness in either the initial input or the local modification rule, such procedures implicitly result in some distribution of test inputs. Though we do not have access to its explicit form, knowledge of the design procedure can enable us to estimate it much more readily than in a naive density estimation setting. For example, we can simulate the design procedure as many times as is needed to sufficiently estimate the resulting density, whereas in density estimation in general, we cannot control how many test inputs we can access.

The training input distribution, P_X, is also often known explicitly. In protein design problems, for example, training sequences are often generated by introducing random substitutions to a single wild type sequence [76, 110, 111], by recombining segments of several "parent" sequences [31, 52, 75, 113], or by independently sampling the amino acid at each position from a known distribution [124, 140]. Conveniently, we can then compute the weights in Eq. (3.4) exactly without introducing approximation error due to density ratio estimation.

Finally, we note that, by construction, the design problem tends to result in test input distributions that place considerable probability mass on regions where the training input distribution does not. The further the test distribution is from the training distribution in this regard, the larger the resulting weights on candidate test points, and the larger the confidence set in Eq. (3.3) will tend to be. This phenomenon agrees with our intuition about epistemic uncertainty: we should have more uncertainty—that is, larger confidence sets—in regions of input space where there is less training data.

Efficient computation of confidence sets under feedback covariate shift

Using Algorithm 4 to construct the full conformal confidence set, $C_\alpha(X_{\text{test}})$, requires computing the scores and weights, $S_i(X_{\text{test}}, y)$ and $w_i^y(X_{\text{test}})$, for all $i = 1, \ldots, n+1$ and all candidate labels, $y \in \mathcal{Y}$. When the dependence of $\tilde{P}_{X;D}$ on D arises from a model trained on D, then naively, we must train $(n+1) \times |\mathcal{Y}|$ models in order to compute these quantities. We now describe two important, practical cases in which this computational burden can be reduced to fitting $n+1$ models, removing the dependence on the number of candidate labels. In such cases, we can post-process the outputs of these $n+1$ models to calculate all $(n+1) \times |\mathcal{Y}|$ required scores and weights (see Alg. 6 for pseudocode); we refer to this as computing the confidence set efficiently.

In the following two examples and in our experiments, we use the residual score function, $S((X, Y), D) = |Y - \mu_D(X)|$, where μ_D is a regression model trained on the multiset D. To understand at a high level when efficient computation is possible, first let μ_{-i}^y denote the regression model trained on $Z_{-i}^y = Z_{-i} \cup \{(X_{\text{test}}, y)\}$, the i-th leave-one-out training data set combined with a candidate test data point. The scores and weights can be computed efficiently when $\mu_{-i}^y(X_i)$ is a computationally simple function of the candidate label, y, for all i—for example, a linear function of y. We discuss two such cases in detail.

Ridge regression. Suppose we fit a ridge regression model, with ridge regularization hyperparameter γ, to the training data. Then, we draw the test input vector from a distribution which places more mass on regions of $\mathcal{X}$ where the model predicts more desirable values, such as high fitness in protein design problems. Recent studies have employed this relatively simple approach to successfully design novel enzymes with enhanced catalytic efficiencies and thermostabilities [30, 31, 110].

In the ridge regression setting, the quantity $\mu_{-i}^y(X_i)$ can be written in closed form as

$$\begin{aligned}\mu_{-i}^y(X_i) &= \left[\left(\mathbf{X}_{-i}^T\mathbf{X}_{-i} + \gamma I\right)^{-1}\mathbf{X}_{-i}^T Y_{-i}^y\right]^T X_i \\ &= \left(\sum_{j=1}^{n-1} Y_{-i;j}\mathbf{A}_{-i;j}\right)^T X_i + (\mathbf{A}_{-i;n}^T X_i)y,\end{aligned} \tag{3.5}$$

where the rows of the matrix $\mathbf{X}_{-i} \in \mathbb{R}^{n \times p}$ are the input vectors in Z^y_{-i}, $Y^y_{-i} = (Y_{-i}, y) \in \mathbb{R}^n$ contains the labels in Z^y_{-i}, the matrix $\mathbf{A}_{-i} \in \mathbb{R}^{n \times p}$ is defined as $\mathbf{A}_{-i} = \left(\mathbf{X}^T_{-i}\mathbf{X}_{-i} + \gamma I\right)^{-1} \mathbf{X}^T_{-i}$, $\mathbf{A}_{-i;j}$ denotes the j-th column of A_{-i}, and $Y_{-i;j}$ denotes the j-th element of Y_{-i}.

Note that the expression in Eq. (3.5) is a linear function of the candidate label, y. Consequently, as formalized by Alg. 6, we first compute and store the slopes and intercepts of these linear functions for all i, which can be calculated as byproducts of fitting $n+1$ ridge regression models. Using these parameters, we can then compute $\mu^y_{-i}(X_i)$ for all candidate labels, $y \in \mathcal{Y}$, by simply evaluating a linear function of y instead of retraining a regression model on Z^y_{-i}. Altogether, beyond fitting $n + 1$ ridge regression models, Alg. 6 requires $O(n \cdot p \cdot |\mathcal{Y}|)$ additional floating point operations to compute the scores and weights for all the candidate labels, the bulk of which can be implemented as one outer product between an n-vector and a $|\mathcal{Y}|$-vector, and one Kronecker product between an $(n \times p)$-matrix and a $|\mathcal{Y}|$-vector.

Gaussian process regression. Similarly, suppose we fit a Gaussian process regression model to the training data. We then select a test input vector according to a likelihood that is a function of the mean and variance of the model's prediction; such functions are referred to as *acquisition functions* in the Bayesian optimization literature.

For a linear kernel, the expression for the mean prediction, $\mu^y_{-i}(X_i)$, is the same as for ridge regression (Eq. (3.5)). For arbitrary kernels, the expression can be generalized and remains a linear function of y (see Section 3.7 for details). We can therefore mimic the computations described for the ridge regression case to compute the scores and weights efficiently.

Data splitting

For settings with abundant training data, or model classes that do not afford efficient computations of the scores and weights, one can turn to *data splitting* to construct valid confidence sets. To do so, we first randomly partition the labeled data into disjoint training and calibration sets. Next, we use the training data to fit a regression model, which induces a test input distribution. If we condition on the training data, thereby treating the regression model as fixed, we have a setting in which (1) the calibration and test data are drawn from different input distributions, but (2) are independent (even though the test and training data are not). Thus, data splitting returns us to the setting of standard covariate shift, under which we can use the data splitting approach in [84] to construct valid *split conformal* confidence intervals (see Section 3.6).

We also introduce randomized data splitting approaches that give exact coverage; see Section 3.6 for details.

3.3 Simulated protein design experiments

To demonstrate practical applications of our work, we turn to examples of uncertainty quantification for designed proteins. Given a fitness function[3] of interest, such as fluorescence, a typical goal of protein design is to seek a protein with high fitness—in particular, higher than we have observed in known proteins. Historically, this has been accomplished in the wet lab through several iterations of expensive, time-consuming experiments. Recently, efforts have been made to augment such approaches with machine learning-based strategies; see reviews by Yang et al. [86], Sinai & Kelsic [102], and Wu et al. [123] and references therein. For example, one might train a regression model on protein sequences with experimentally measured fitnesses, then use an optimization algorithm or fit a generative model that leverages that regression model to propose promising new proteins [30, 52, 75, 76, 85, 89, 97, 110, 113, 122, 124]. Special attention has been given to the *single-shot* case in which we are given just a single batch of training data, due to its obvious practical convenience.

The use of regression models for design involves balancing (1) the desire to explore regions of input space far from the training inputs, in order to find new desirable inputs, with (2) the need to stay close enough to the training inputs that we can trust the regression model. As such, estimating predictive uncertainty in this setting is important. Furthermore, the training and designed data are described by feedback covariate shift: since the fitness is some quantity dictated by nature, the conditional distribution of fitness given any sequence stays fixed, but the distribution of designed sequences is chosen based on a trained regression model.[4]

Our experimental protocol is as follows. Given training data consisting of protein sequences labeled with experimental measurements of their fitnesses, we fit a regression model, then sample test sequences (representing designed proteins) according to design algorithms used in recent work [110, 124] (Fig. 3.2). We then construct confidence sets with guaranteed coverage for the designed proteins, and examine various characteristics of those sets to evaluate the utility of our approach. In particular, we show how our method can be used to select design algorithm hyperparameters that achieve acceptable trade-offs between high predicted fitness and low predictive uncertainty for the designed proteins. Code reproducing these experiments is available at `https://github.com/clarafy/conformal-for-design`.

Design experiments using combinatorially complete fluorescence data sets

The challenge when evaluating *in silico* design methods is that in general, we do not have labels for the designed sequences. One workaround, which we take here, is to make use of

[3] We use the term *fitness function* to refer to a particular property that can be exhibited by proteins, while the *fitness* of a protein refers to the extent to which it exhibits that property.

[4] In this section, we will use "test" and "designed" interchangeably when describing data. We will also sometimes say "sequence" instead of "input," but this does not imply any constraints on how the protein is represented or featurized.

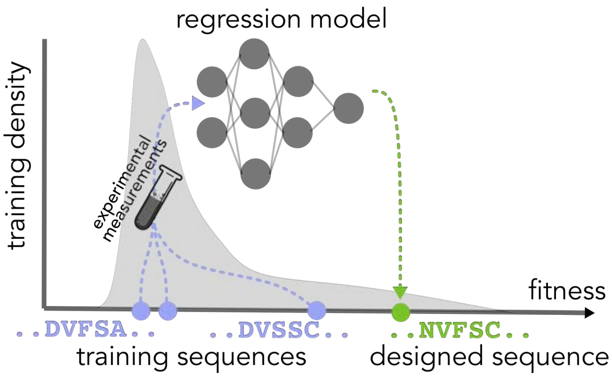


Figure 3.2: **Single-shot protein design.** The gray distribution represents the distribution of fitnesses under the training sequence distribution. The blue circles represent the fitnesses of three training sequences, and the goal is to propose a sequence with even higher fitness. To that end, we fit a regression model to the training sequences labeled with experimental measurements of their fitnesses, then deploy some design procedure that uses that trained model to propose a new sequence believed to have a higher fitness (green circle).

combinatorially complete protein data sets [83, 85, 122, 128], in which a small number of fixed positions are selected from some wild type sequence, and all possible variants of the wild type that vary in those selected positions are measured experimentally. Such data sets enable us to simulate protein design problems where we always have labels for the designed sequences. In particular, we can use a small subset of the data for training, then deploy a design procedure that proposes novel proteins (restricted to being variants of the wild type at the selected positions), for which we have labels.

We used data of this kind from Poelwijk et al. [83], which focused on two “parent” fluorescent proteins that differ at exactly thirteen positions in their sequences, and are identical at every other position. All $2^{13} = 8,192$ sequences that have the amino acid of either parent at those thirteen sites (and whose remaining positions are identical to the parents) were experimentally labeled with a measurement of brightness at both a “red” wavelength and a “blue” wavelength, resulting in combinatorially complete data sets for two different fitness functions. In particular, for both wavelengths, the label for each sequence was an enrichment score based on the ratio of its counts before and after brightness-based selection through fluorescence-activated cell sorting. The enrichment scores were then normalized so that the same score reflects comparable brightness for both wavelengths.

Finally, each time we sampled from this data set to acquire training or designed data, as described below, we added simulated measurement noise to each label by sampling from a noise distribution estimated from the combinatorially complete data set (see 3.8 for details). This step simulates the fact that sampling and measuring the same sequence multiple times results in different measurements.

Protocol for design experiments

Our training data sets consisted of n data points, $Z_{1:n}$, sampled uniformly at random from the combinatorially complete data set. We used $n \in \{96, 192, 384\}$ as is typical of realistic scenarios [52, 75, 85, 110, 122]. We represented each sequence as a feature vector containing all first- and second-order interaction terms between the thirteen variable sites, and fit a ridge regression model, $\mu_{Z_{1:n}}(x)$, to the training data, where the regularization strength was set to 10 for $n = 96$ and 1 otherwise. Linear models of interaction terms between sequence positions have been observed to be both theoretically justified and empirically useful as models of protein fitness functions [83, 128, 133] and thus may be particularly useful for protein design, particularly with small amounts of training data.

Sampling designed sequences. Following ideas in [110, 124], we designed a protein by sampling from a sequence distribution whose log-likelihood is proportional to the prediction of the regression model:

$$\tilde{p}_{X;Z_{1:n}}(X_{\text{test}}) \propto \exp(\lambda \cdot \mu_{Z_{1:n}}(X_{\text{test}})), \tag{3.6}$$

where $\lambda > 0$, the *inverse temperature*, is a hyperparameter. Larger values of λ result in distributions of designed sequences that are more likely to have high predicted fitnesses according to the model, but are also, for this same reason, more likely to be in regions of sequence space that are further from the training data and over which the model is more uncertain. Analogous hyperparameters have been used in recent protein design work to control this trade-off between exploration and exploitation [101, 110, 124, 147]. We took $\lambda \in \{0, 2, 4, 6\}$ to investigate how the behavior of our confidence sets varies along this trade-off.

Constructing confidence sets for designed sequences. For each setting of n and λ, we generated n training data points and one designed data point as just described $T = 2000$ times. For each of these T trials, we used Alg. 6 to construct the full conformal confidence set, $C_\alpha(X_{\text{test}})$, using a grid of real values between 0 and 2.2 spaced $\Delta = 0.02$ apart as the set of candidate labels, $\mathcal{Y}$. This range contained the ranges of fitnesses in both the blue and red combinatorially complete data sets, $[0.091, 1.608]$ and $[0.025, 1.692]$, respectively.[5]

We used $\alpha = 0.1$ as a representative miscoverage value, corresponding to coverage of $1 - \alpha = 0.9$. We then computed the *empirical coverage* achieved by the confidence sets, defined as the fraction of the T trials where the true fitness of the designed protein was

[5] In general, a reasonable approach for constructing a finite grid of candidate labels, $\mathcal{Y}$, is to span an interval beyond which one knows label values are impossible in practice, based on prior knowledge about the measurement technology. The presence or absence of any such value in a confidence set would not be informative to a practitioner. The size of the grid spacing, Δ, determines the resolution at which we evaluate coverage; that is, in terms of coverage, including a candidate label is equivalent to including the Δ-width interval centered at that label value. Generally, one should therefore set Δ as small as possible, subject to one's computational budget.

within half a grid spacing from some value in the confidence set, namely, $\min\{|Y_{\text{test}} - y| : y \in C_\alpha(X_{\text{test}})\} \leq \Delta/2$. Based on Theorem 1, assuming $\mathcal{Y}$ is both a large and fine enough grid to encompass all possible fitness values, the expected empirical coverage is lower bounded by $1 - \alpha = 0.9$. However, there is no corresponding upper bound, so it will be of interest to examine any excess in the empirical coverage, which corresponds to the confidence sets being conservative (larger than necessary). Ideally, the empirical coverage is exactly 0.9, in which case the sizes of the confidence sets reflect the minimal predictive uncertainty we can have about the designed proteins while achieving coverage.

In our experiments, the computed confidence sets tended to comprise grid-adjacent candidate labels, suggestive of confidence intervals. As such, we hereafter refer to the *width* of *confidence intervals*, defined as the grid spacing size times the number of values in the confidence set, $\Delta \cdot |C_\alpha(X_{\text{test}})|$.

Results

Here we discuss results for the blue fluorescence data set. Analogous results for the red fluorescence data set are presented in Section 3.8.

Effect of inverse temperature. First we examined the effect of the inverse temperature, λ, on the fitnesses of designed proteins (Fig. 3.3a). Note that $\lambda = 0$ corresponds to a uniform distribution over all sequences in the combinatorially complete data set (i.e., the training distribution), which mostly yields label values less than 0.5. For $\lambda \geq 4$, we observe a considerable mass of designed proteins attaining fitnesses around 1.5, so these values of λ represent settings where the designed proteins are more likely to be fitter than the training proteins. This observation is consistent with the use of this and other analogous hyperparameters to tune the outcomes of design algorithms [101, 110, 124, 147], and is meant to provide an intuitive interpretation of the hyperparameter to readers unfamiliar with its use in design problems.

Empirical coverage and confidence interval widths. Despite the lack of a theoretical upper bound, the empirical coverage does not tend to exceed the theoretical lower bound of $1 - \alpha = 0.9$ by much (Fig. 3.3b), reaching at most 0.924 for $n = 96, \lambda = 6$. Loosely speaking, this observation suggests that the confidence intervals are nearly as small, and therefore as informative, as they can be while achieving coverage.

As for the widths of the confidence intervals, we observe that for any value of λ, the intervals tend to be smaller for larger amounts of training data (Fig. 3.3c). Also, for any value of n, the intervals tend to get larger as λ increases. The first phenomenon agrees with the intuition that training a model on more data should generally reduce predictive uncertainty. The second phenomenon arises because greater values of λ lead to designed sequences with higher predicted fitnesses, which the model is more uncertain about. Indeed, for $\lambda = 4, n = 96$ and $\lambda = 6, n \in \{96, 192\}$, many confidence intervals contain the entire range of fitnesses in the combinatorially complete data set. In these regimes, the regression model cannot glean enough information from the training data to have much certainty about the designed protein.

Comparison to standard covariate shift Deploying full conformal prediction as prescribed for standard covariate shift (SCS) [84], a heuristic with no formal guarantees in this setting, often results in more conservative confidence sets than those produced by our method (Fig. 3.3). To understand when the outputs of these two methods will differ more or less, we can compare the forms of the weights that both methods introduce on the training and candidate test data points when considering a candidate label.

First, recall that for both feedback covariate shift (FCS) and SCS, the weight assigned to the i-th training score is a normalized ratio of the likelihood of X_i under a test input distribution and the training input distribution, p_X, namely:

$$\begin{aligned} v(X_i; Z_{-i} \cup \{(X_{\text{test}}, y)\}) &= \tilde{p}_{X; Z_{-i} \cup \{(X_{\text{test}}, y)\}}(X_i)/p_X(X_i), \\ v(X_i) &= \tilde{p}_{X; Z_{1:n}}(X_i)/p_X(X_i), \end{aligned}$$

for FCS and SCS, respectively. For FCS, the test input distribution, $\tilde{p}_{X; Z_{-i} \cup \{(X_{\text{test}}, y)\}}$, is induced by a regression model trained on $Z_{-i} \cup \{(X_{\text{test}}, y)\}$, and therefore depends on the candidate label, y, and also differs for each of the n training inputs. Consequently, for FCS the weight on the i-th training score depends on the candidate label under consideration, y. In contrast, for SCS the test input distribution, $p_{X; Z_{1:n}}$, is simply the one induced by the training data, $Z_{1:n}$, and is therefore fixed for all training scores and all candidate labels.

Note, however, that the SCS and FCS weights depend on data sets, $Z_{1:n}$ and $Z_{-i} \cup \{(X_{\text{test}}, y)\}$, respectively, that differ only in a single data point: the former contains Z_i, while the latter contains (X_{test}, y). Therefore, the difference between the weights—and the resulting confidence sets—is determined by how sensitive the mapping from data set to test input distribution, $D \to \tilde{p}_{X;D}$ (given by Eq. (3.6) in this setting), is to changes of a single data point in D. Roughly speaking, the less sensitive this mapping, the more similar the FCS and SCS confidence sets will be. For example, using more training data (*e.g.*, $n = 384$ compared to $n = 96$ for a fixed λ) or a lower inverse temperature (*e.g.*, $\lambda = 2$ compared to $\lambda = 6$ for a fixed n) results in more similar SCS and FCS confidence sets (Figs. 3.3d, 3.7, 3.10). Similarly, using regression models with fewer features or stronger regularization also results in more similar confidence sets (Figs. 3.8, 3.9, 3.11).

One can therefore think of and use SCS confidence sets as a computationally cheaper approximation to FCS confidence sets, where the approximation is better for mappings $D \to \tilde{p}_{X;D}$ that are less sensitive to changes in D. Conversely, the extent to which SCS confidence sets are similar to FCS confidence sets will generally reflect this sensitivity. In our protein design experiments, SCS confidence sets tend to be more conservative than their FCS counterparts, where the extent of overcoverage generally increases with less training data, higher inverse temperature (Figs. 3.3d, 3.7), more complex features, and weaker regularization (Figs. 3.8, 3.9).

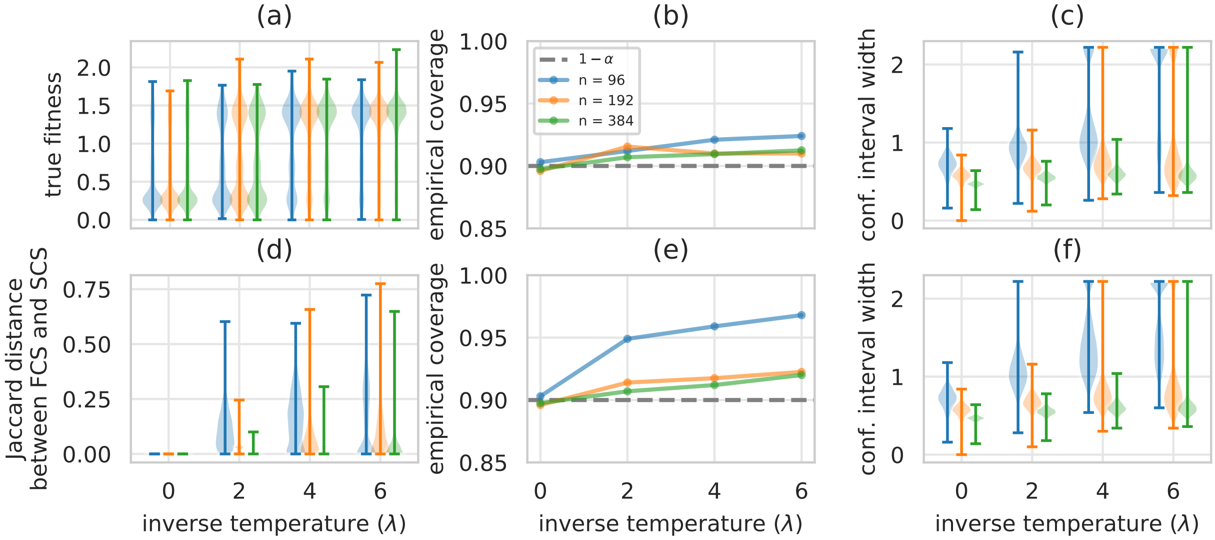


Figure 3.3: **Quantifying uncertainty for designed proteins (blue fluorescence).** (a) Distributions of labels of designed proteins, for different values of the inverse temperature, λ, and different amounts of training data, n. Labels surpass the fitness range observed in the combinatorially complete data set, $[0.091, 1.608]$, due to additional simulated measurement noise. (b) Empirical coverage, compared to the theoretical lower bound of $1 - \alpha = 0.9$ (dashed gray line), and (c) distributions of confidence interval widths achieved by full conformal prediction for feedback covariate shift (our method) over $T = 2000$ trials. In (a), and (c), the whiskers signify the minimum and maximum observed values. (d) Distributions of Jaccard distances between the confidence intervals produced by full conformal prediction for feedback covariate shift and standard covariate shift [84]. (e, f) Same as (b, c) but using full conformal prediction for standard covariate shift.

Using uncertainty quantification to set design procedure hyperparameters. As the inverse temperature, λ, in Eq. (3.6) varies, there is a trade-off between the mean predicted fitness and predictive certainty for designed proteins: both mean predicted fitness and mean confidence interval width grow as λ increases (Fig. 3.4a). To demonstrate how our method might be used to inform the design procedure itself, one can visualize this trade-off (Fig. 3.4) and use it to decide on a setting of λ that achieves both a mean predicted fitness and degree of certainty that one finds acceptable, given, for example, some resource budget for evaluating designed proteins in the wet lab. For data sets of different fitness functions, which may be better or worse approximated by our chosen regression model class and may have different amounts of measurement noise, this trade-off—and therefore, the appropriate setting of λ—will be different (Fig. 3.4).

For example, protein design experiments on the red fluorescence data set result in a less favorable trade-off between mean predicted fitness and predictive certainty than the blue fluorescence data set: the same amount of increase in mean predicted fitness corresponds to a greater increase in mean interval width for red compared to blue fluorescence (Fig. 3.4a). We

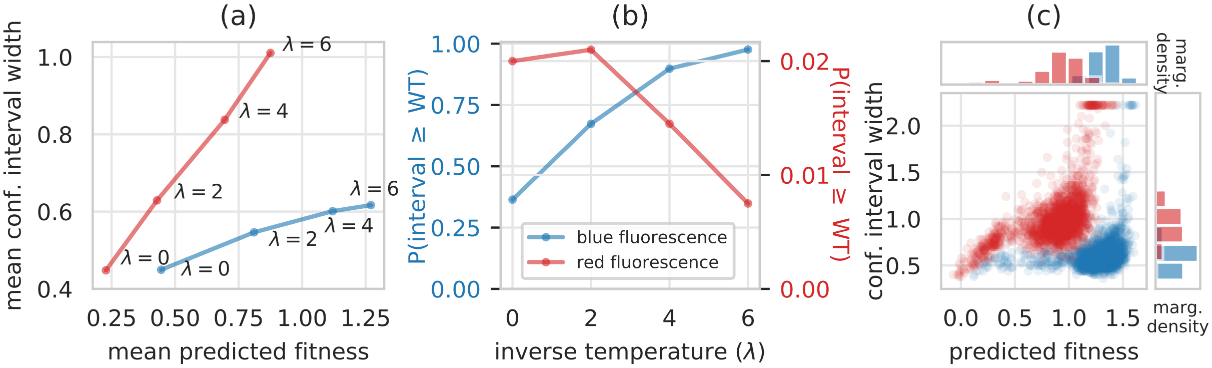


Figure 3.4: **Comparison of trade-off between predicted fitness and predictive certainty for red and blue fluorescence.** (a) Trade-off between mean confidence interval width and mean predicted fitness for different values of the inverse temperature, λ, and $n = 384$ training data points. (b) Empirical probability that the smallest fitness value in the confidence intervals of designed proteins exceeds the true fitness of one of the wild-type parent sequences, mKate2. (c) For $n = 384$ and $\lambda = 6$, the distributions of both confidence interval width and predicted fitnesses of designed proteins.

might therefore choose a smaller value of λ when designing proteins for the former compared to the latter. Indeed, predictive uncertainty grows so quickly for red fluorescence that, for $\lambda > 2$, the empirical probability that the smallest value in the confidence interval is greater than the true fitness of a wild type sequence decreases rather than increases (Fig. 3.4b) which suggests we may not want to set $\lambda > 2$. In contrast, if we had looked at the mean predicted fitness alone without assessing the uncertainty of those predictions, it grows monotonically with λ (Fig. 3.4a), which would not suggest any harm from setting λ to a higher value.

In contrast, for blue fluorescence, although the mean interval width also grows with λ, it does so at a much slower rate than for red fluorescence (Fig. 3.4a); correspondingly, the empirical frequency at which the confidence interval surpasses the fitness of the wild type also grows monotonically (Fig. 3.4b).

We can observe these differences in the trade-off between blue and red fluorescence even for a fixed value of λ. For example, for $n = 384, \lambda = 6$ (Fig. 3.4c), observe that proteins designed for blue fluorescence (blue circles) mostly lie in a flat horizontal band. That is, those with higher predicted fitnesses do not have much wider intervals than those with lower predicted fitnesses, except for a few proteins with the highest predicted fitnesses. In contrast, for red fluorescence, designed proteins with higher predicted fitnesses also tend to have wider confidence intervals.

Design experiments using adeno-associated virus (AAV) capsid packaging data

In contrast with Section 3.3, which represented a protein design problem with limited amounts of labeled data (at most a few hundred sequences), here we focus on a setting in which there is abundant labeled data. We can therefore employ data splitting as described in Section 3.2 to construct confidence sets, as an alternative to computing full conformal confidence sets (Eq. (3.3)) as done in Section 3.3. Specifically, we construct a randomized version of the split conformal confidence set (Section 3.6), which achieves exact coverage.

This subsection, together with the previous subsection, demonstrate that in both regimes—limited and abundant labeled data—our proposed methods provide confidence sets that give coverage, are not overly conservative, and can be used to visualize the trade-off between predicted fitness and predictive uncertainty inherent to a design algorithm.

Protein design problem: AAV capsid proteins with improved packaging ability

Adeno-associated viruses (AAVs) are a class of viruses whose capsid, the protein shell that encapsulates the viral genome, are a promising delivery vehicle for gene therapy. As such, the proteins that constitute the capsid have been modified to enhance various fitness functions, such as the ability to enter specific cell types and evade the immune system [28, 49, 64]. Such efforts usually start by sampling millions of proteins from some sequence distribution, then performing an experiment that selects out the fittest sequences. Sequence distributions commonly used today have relatively high entropy, and the resulting sequence diversity can lead to successful outcomes for a myriad of downstream selection experiments [111, 124]. However, most of these sequences fail to assemble into a capsid that packages the genetic payload [53, 64, 82]—a function called *packaging*, which is the minimum requirement of a gene therapy delivery mechanism, and therefore a prerequisite to any other desiderata.

If sequence distributions could be developed with higher packaging rate, without compromising sequence diversity, then the success rate of downstream selection experiments should improve. To this end, Zhu & Brookes et al. [124] use neural networks trained on sequence-packaging data to specify the parameters of sequence distributions that simultaneously have high entropy and yield sequences with high predicted packaging ability. The sequences in this data varied at seven promising contiguous positions identified in previous work [49], and elsewhere matched a wild type. To accommodate commonly used DNA synthesis protocols, the authors parameterized their sequence distributions as independent categorical distributions over the four nucleotides at each of twenty-one contiguous sites, corresponding to codons at each of the seven sites of interest.

Protocol for design experiments

We followed the methodology of Zhu & Brookes et al. [124] to find sequence distributions with high mean predicted fitness—in particular, higher than that of the commonly used “NNK” sequence distribution [49]. Specifically, we used simulated data based on their high-throughput data, which sampled millions of sequences from the NNK distribution and labeled each with an enrichment score quantifying its packaging fitness, based on its count before and after a packaging-based selection experiment. We introduced additional simulated measurement noise to these labels, where the parameters of the noise distribution were also estimated from the pre- and post-selection counts, resulting in labels ranging from -7.53 to 8.80 for $8,552,729$ sequences (see Section 3.8 for details).

We then randomly selected and held out one million of these data points, for calibration and test purposes described shortly, then trained a neural network on the remaining data to predict fitness from sequence. Finally, following [124], we approximately solved an optimization problem that leveraged this regression model in order to specify the parameters of sequence distributions with high mean predicted fitness. Specifically, let $\{p_\phi : \phi \in \Phi\}$ denote the class of sequence distributions parameterized as independent categorical distributions over the four nucleotides at each of twenty-one contiguous sequence positions. We set the parameters of the designed sequence distribution by using stochastic gradient descent to approximately solve the following problem:

$$\phi_\lambda = \arg\min_{\phi \in \Phi} D_{\mathrm{KL}}(p^\star_\lambda || p_\phi) \tag{3.7}$$

where $p^\star_\lambda(X) \propto \exp(\lambda \cdot \mu(X))$, μ is the neural network fit to the training data, and $\lambda \geq 0$ is an inverse temperature hyperparameter. After solving for ϕ_λ for a range of inverse temperature values, $\lambda \in \{1, 2, 3, 4, 5, 6, 7\}$, we sampled designed sequences from p_{ϕ_λ} as described below, then used a randomized data splitting approach to construct confidence sets that achieve exact coverage.

Sampling designed sequences. Unlike in Section 3.3, here we did not have a label for every sequence in the input space—that is, all sequences that vary at the seven positions of interest, and that elsewhere match a wild type. As an alternative, we used rejection sampling to sample from p_{ϕ_λ}. Specifically, recall that we held out a million of the labeled sequences. The input space was sampled uniformly and densely enough by the high-throughput data set that we treated $990,000$ of these held-out labeled sequences as samples from a proposal distribution (that is, the NNK distribution) and were able to perform rejection sampling to sample designed sequences from p_{ϕ_λ} for which we have labels.

Constructing confidence sets for designed sequences. Note that rejection sampling results in some random number, at most $990,000$, of designed sequences; in practice, this number ranged from single digits to several thousand for $\lambda = 7$ to $\lambda = 1$, respectively. To account for this variability, for each value of the inverse temperature, we performed $T = 500$ trials of the following steps. We randomly split the one million held-out labeled sequences into $990,000$ proposal distribution sequences and $10,000$ sequences to be used as calibration data. We used the former to sample some number of designed sequences, then used the latter to construct randomized staircase confidence sets (Alg. 5) for each of the designed sequences. The results we report next concern properties of these sets averaged over all $T = 500$ trials.

Results

Effect of inverse temperature. The inverse temperature hyperparameter, λ, in Eq. (3.7) plays a similar role as in Section 3.3: larger values result in designed sequences with higher mean true fitness (Fig. 3.5a). Note that the mean true fitness for all considered values of the inverse temperature is higher than that of the training distribution (the dashed black line, Fig. 3.5a).

Empirical coverage and confidence set sizes. For all considered values of the inverse temperature, the empirical coverage of the confidence sets is very close to the expected value of $1 - \alpha = 0.9$ (Fig. 3.5b, top). Note that some designed sequences, which the neural network is particularly uncertain about, are given a confidence set with infinite size (Fig. 3.5b, bottom). The fraction of sets with infinite size, as well as the mean size of non-infinite sets, both increase with the inverse temperature (Fig. 3.5b, bottom), which is consistent with our intuition that the neural network should be less confident about predictions that are much higher than fitnesses seen in the training data.

Using uncertainty quantification to set design procedure hyperparameters. As in Section 3.3, the confidence sets we construct expose a trade-off between predicted fitness and predictive uncertainty as we vary the inverse temperature. Generally, the higher the mean predicted fitness of the sequence distributions, the greater the mean confidence set size as well (Fig. 3.5c).[6] One can inspect this trade-off to decide on an acceptable setting of the inverse temperature. For example, observe that the mean set size does not grow appreciably between $\lambda = 1$ and $\lambda = 4$, even though the mean predicted fitness monotonically increases (Fig. 3.5b, bottom, 3.5c); similarly, the fraction of sets with infinite size also remains near zero for these values of λ (Fig. 3.5b, bottom). However, both of these quantities start to increase for $\lambda \geq 5$. By $\lambda = 7$, for instance, more than 17% of designed sequences are given a confidence set with infinite size, suggesting that p_{ϕ_7} has shifted too far from the training distribution for the neural network to be reasonably certain about its predictions. Therefore, one might conclude that using $\lambda \in \{4, 5\}$ achieves an acceptable balance of designed sequences with higher predicted fitness than the training sequences and low enough predictive uncertainty.

[6] The exception is the sequence distribution corresponding to $\lambda = 2$, which has a higher mean predicted fitness but on average smaller sets than $\lambda = 1$. One likely explanation is that experimental measurement noise is particularly high for very low fitnesses, making low-fitness sequences inherently difficult to predict.

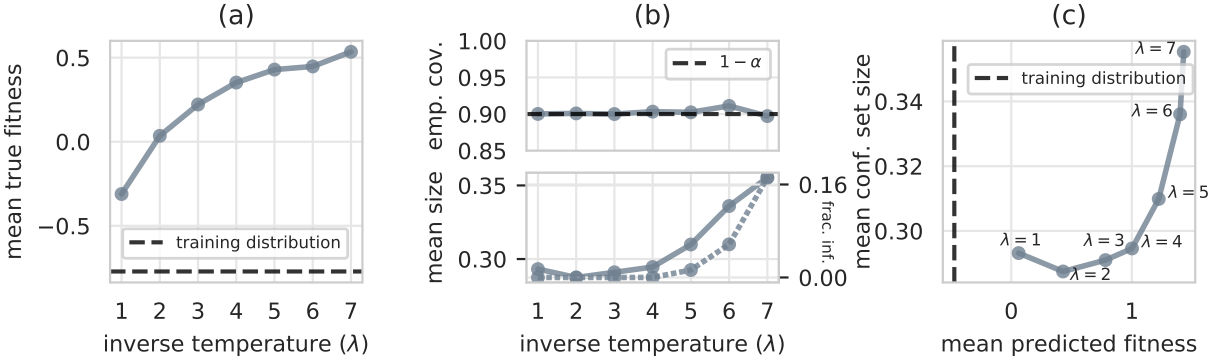


Figure 3.5: **Quantifying uncertainty for designed adeno-associated virus (AAV) capsid proteins.** (a) Mean true fitness of designed sequences resulting from different values of the inverse temperature, $\lambda \in \{1, 2, \ldots, 7\}$. The dashed black line is the mean true fitness of sequences drawn from the NNK sequence distribution (i.e., the training distribution). (b) Top: empirical coverage of randomized staircase confidence sets (Section 3.6) constructed for designed sequences. The dashed black line is the expected empirical coverage of $1 - \alpha = 0.9$. Bottom: fraction of confidence sets with infinite size (dashed gray line) and mean size of non-infinite confidence sets (solid gray line). The set size is reported as a fraction of the range of fitnesses in all the labeled data, $[-7.53, 8.80]$. (c) Trade-off between mean predicted fitness and mean confidence set size for $\lambda \in \{1, 2, \ldots, 7\}$. The dashed black line is the mean predicted fitness for sequences from the training distribution.

3.4 Discussion

The predictions made by machine learning models are increasingly being used to make consequential decisions, which in turn influence the data that the models encounter. Our work presents a methodology that allows practitioners to trust the predictions of learned models in such settings. In particular, our protein design examples demonstrate how our approach can be used to navigate the trade-off between desirable predictions and predictive certainty inherent to design problems.

Looking beyond the design problem, the formalism of feedback covariate shift (FCS) introduced here captures a range of problem settings pertinent to modern-day deployments of machine learning. In particular, FCS often occurs at each iteration of a feedback loop—for example, at each iteration of active learning, adaptive experimental design, and Bayesian optimization methods. Applications and extensions of our approach to such settings are exciting directions for future investigation.

3.5 Proofs

Proof of Theorem 1

Data from feedback covariate shift (FCS) are a special case of what we call *pseudo-exchangeable*[7] random variables.

Definition 2. *Random variables $V_1, \ldots, V_{n+1}$ are pseudo-exchangeable with factor functions $g_1, \ldots, g_{n+1}$ and core function h if the density, f, of their joint distribution can be factorized as*

$$f(v_1, \ldots, v_{n+1}) = \prod_{i=1}^{n+1} g_i(v_i;\, v_{-i}) \cdot h(v_1, \ldots, v_{n+1}),$$

where $v_{-i} = v_{1:(n+1)} \setminus v_i$,[8] *each $g_i(\cdot;\, v_{-i})$ is a function that depends on the multiset v_{-i} (that is, on the values in v_{-i} but not on their ordering), and h is a function that does not depend on the ordering of its $n+1$ inputs.*

The following lemma characterizes the distribution of the scores of pseudo-exchangeable random variables, which allows for a pseudo-exchangeable generalization of conformal prediction in Theorem 2. We then show that data generated under FCS are pseudo-exchangeable, and a straightforward application of Theorem 2 yields Theorem 1 as a corollary. Our technical development here builds upon the work of Tibshirani et al. [84], who generalized conformal prediction to handle "weighted exchangeable" random variables, including data under standard covariate shift.

The key insight is that if we condition on the values, but not the ordering, of the scores, we can exactly describe their distribution. The following proposition is a generalization of arguments found in the proof of Lemma 3 in [84]; the subsequent result in Lemma 1 is a generalization of that lemma.

Proposition 2. *Let $Z_1, \ldots, Z_{n+1}$ be pseudo-exchangeable random variables with a joint density function, f, that can be written with factor functions $g_1, \ldots, g_{n+1}$ and core function h. Let S be any score function and denote $S_i = S(Z_i, Z_{-i})$ where $Z_{-i} = Z_{1:(n+1)} \setminus \{Z_i\}$ for $i = 1, \ldots, n+1$. Define*

$$w_i(z_1, \ldots, z_{n+1}) \equiv \frac{\sum_{\sigma : \sigma(n+1) = i} \prod_{j=1}^{n+1} g_j\big(z_{\sigma(j)};\, z_{-\sigma(j)}\big)}{\sum_{\sigma} \prod_{j=1}^{n+1} g_j\big(z_{\sigma(j)};\, z_{-\sigma(j)}\big)}, \quad i = 1, \ldots, n+1, \tag{3.8}$$

[7] The name *pseudo-exchangeable* hearkens to the similarity of the factorized form to the pseudo-likelihood approximation of a joint density. Note, however, that each factor, $g_i(v_i;\, v_{-i})$, can only depend on the values and not the ordering of the other variables, $v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_n$, whereas each factor in the pseudo-likelihood approximation also depends on the identities (i.e., the ordering) of the other variables.

[8] With some abuse of notation, we denote $z_{-i} = z_{1:(n+1)} \setminus z_i$ whenever possible, as done here, but use $z_{-i} = z_{1:n} \setminus z_i$ whenever we need to append a candidate test point, as done in the main text and in Theorem 2 below. In either case, we will clarify.

where the summations are taken over permutations, σ, of the integers $1, \dots, n+1$. For values $z = (z_1, \dots, z_{n+1})$, let $s_i = S(z_i, z_{-i})$ and let E_z be the event that $\{Z_1, \dots, Z_{n+1}\} = \{z_1, \dots, z_{n+1}\}$ (that is, the multiset of values taken on by $Z_1, \dots, Z_{n+1}$ equals the multiset of the values in z). Then

$$S_{n+1} \mid E_z \sim \sum_{i=1}^{n+1} w_i(z_1, \dots, z_{n+1})\, \delta_{s_i}.$$

Proof. For simplicity, we treat the case where $S_1, \dots, S_{n+1}$ are distinct almost surely; the result also holds in the general case, but the notation that accommodates duplicate values is cumbersome. For $i = 1, \dots, n+1$,

$$\begin{aligned}
\mathbb{P}(S_{n+1} = s_i \mid E_z) &= \mathbb{P}(Z_{n+1} = z_i \mid E_z) \\
&= \frac{\sum_{\sigma:\sigma(n+1)=i} f\big(z_{\sigma(1)}, \dots, z_{\sigma(n+1)}\big)}{\sum_{\sigma} f\big(z_{\sigma(1)}, \dots, z_{\sigma(n+1)}\big)} \\
&= \frac{\sum_{\sigma:\sigma(n+1)=i} \prod_{j=1}^{n+1} g_j\big(z_{\sigma(j)}; z_{-\sigma(j)}\big) \cdot h\big(z_{\sigma(1)}, \dots, z_{\sigma(n+1)}\big)}{\sum_{\sigma} \prod_{j=1}^{n+1} g_j\big(z_{\sigma(j)}; z_{-\sigma(j)}\big) \cdot h\big(z_{\sigma(1)}, \dots, z_{\sigma(n+1)}\big)} \\
&= \frac{\sum_{\sigma:\sigma(n+1)=i} \prod_{j=1}^{n+1} g_j\big(z_{\sigma(j)}; z_{-\sigma(j)}\big) \cdot h\big(z_1, \dots, z_{n+1}\big)}{\sum_{\sigma} \prod_{j=1}^{n+1} g_j\big(z_{\sigma(j)}; z_{-\sigma(j)}\big) \cdot h\big(z_1, \dots, z_{n+1}\big)} \\
&= \frac{\sum_{\sigma:\sigma(n+1)=i} \prod_{j=1}^{n+1} g_j\big(z_{\sigma(j)}; z_{-\sigma(j)}\big)}{\sum_{\sigma} \prod_{j=1}^{n+1} g_j\big(z_{\sigma(j)}; z_{-\sigma(j)}\big)} \\
&= w_i(z_1, \dots, z_{n+1}).
\end{aligned}$$

□

Lemma 1. *Let $Z_1, \dots, Z_{n+1}$ be pseudo-exchangeable random variables with a joint density function, f, that can be written with factor functions $g_1, \dots, g_{n+1}$ and core function h. Let S be any score function and denote $S_i = S(Z_i, Z_{-i})$ where $Z_{-i} = Z_{1:(n+1)} \setminus \{Z_i\}$ for $i = 1, \dots, n+1$. For any $\beta \in (0,1)$,*

$$\mathbb{P}\left\{ S_{n+1} \leq \textsc{Quantile}_{\beta}\left(\sum_{i=1}^{n+1} w_i(Z_1, \dots, Z_{n+1})\, \delta_{S_i} \right) \right\} \geq \beta,$$

where $w_i(z_1, \dots, z_{n+1})$ is defined in Eq. (3.8).

Proof. Assume for simplicity of notation that $S_1, \dots, S_{n+1}$ are distinct almost surely (but the result holds generally). For data point values $z = (z_1, \dots, z_{n+1})$, let $s_i = S(z_i, z_{-i})$ and let E_z be the event that $\{Z_1, \dots, Z_{n+1}\} = \{z_1, \dots, z_{n+1}\}$. By Proposition 2,

$$S_{n+1} \mid E_z \sim \sum_{i=1}^{n+1} w_i(z_1, \dots, z_{n+1})\, \delta_{s_i},$$

and consequently

$$\mathbb{P}\left(S_{n+1} \leq \textsc{Quantile}_\beta\left(\sum_{i=1}^{n+1} w_i(z_1, \ldots, z_{n+1})\, \delta_{s_i}\right) \middle| E_z\right) \geq \beta,$$

by definition of the β-quantile; equivalently, since we condition on E_z,

$$\mathbb{P}\left(S_{n+1} \leq \textsc{Quantile}_\beta\left(\sum_{i=1}^{n+1} w_i(Z_1, \ldots, Z_{n+1})\, \delta_{S_i}\right) \middle| E_z\right) \geq \beta.$$

Since this inequality holds for all events E_z, where z is a vector of $n+1$ data point values, smoothing gives

$$\mathbb{P}\left(S_{n+1} \leq \textsc{Quantile}_\beta\left(\sum_{i=1}^{n+1} w_i(Z_1, \ldots, Z_{n+1})\, \delta_{S_i}\right)\right) \geq \beta.$$

□

Lemma 1 yields the following theorem, which enables a generalization of conformal prediction to pseudo-exchangeable random variables.

Theorem 2. *Suppose $Z_1, \ldots, Z_{n+1}$ where $Z_i = (X_i, Y_i) \in \mathcal{X} \times \mathbb{R}$ are pseudo-exchangeable random variables with factor functions $g_1, \ldots, g_{n+1}$. For any score function, S, and any miscoverage level, $\alpha \in (0, 1)$, define for any point $x \in \mathcal{X}$:*

$$C_\alpha(x) = \left\{y \in \mathbb{R} : S_{n+1}(x, y) \leq \textsc{Quantile}_{1-\alpha}\left(\sum_{i=1}^{n+1} w_i(Z_1, \ldots, Z_n, (x, y))\, \delta_{S_i(x,y)}\right)\right\} \tag{3.9}$$

where $S_i(x, y) = S(Z_i, Z_{-i} \cup \{(x, y)\})$ and $Z_{-i} = Z_{1:n} \setminus Z_i$ for $i = 1, \ldots, n$, $S_{n+1}(x, y) = S((x, y), Z_{1:n})$, and the weight functions w_i are as defined in Eq. (3.8)*. Then C_α satisfies*

$$\mathbb{P}\left(Y_{n+1} \in C_\alpha(X_{n+1})\right) \geq 1 - \alpha,$$

where the probability is over all $n+1$ data points, $Z_1, \ldots, Z_{n+1}$.

Proof. By construction, we have

$$\begin{aligned} &Y_{n+1} \in C_\alpha(X_{n+1}) \iff \\ &\qquad S_{n+1}(X_{n+1}, Y_{n+1}) \leq \textsc{Quantile}_{1-\alpha}\left(\sum_{i=1}^{n+1} w_i(Z_1, \ldots, Z_{n+1})\, \delta_{S_i(X_{n+1}, Y_{n+1})}\right). \end{aligned}$$

Applying Lemma 1 gives the result. □

Finally, Theorem 1 follows as a corollary of Theorem 2. Denoting $Z_{n+1} = Z_{\text{test}}$ and $Z_{-i} = Z_{1:(n+1)} \setminus \{Z_i\}$, observe that data, $(Z_1, \ldots, Z_{n+1})$, under FCS are pseudo-exchangeable with the core function

$$h(z_1, \ldots, z_{n+1}) = \prod_{i=1}^{n+1} p_X(x_i)\, p_{Y|X}(y_i \mid x_i),$$

and factor functions $g_i(z_i;\, z_{-i}) = 1$ for $i = 1, \ldots, n$ and

$$g_{n+1}(z_{n+1};\, z_{1:n}) = \frac{\tilde{p}_{X;z_{1:n}}(x_{n+1})\, p_{Y|X}(y_{n+1} \mid x_{n+1})}{p_X(x_{n+1})\, p_{Y|X}(y_{n+1} \mid x_{n+1})} = \frac{\tilde{p}_{X;z_{1:n}}(x_{n+1})}{p_X(x_{n+1})} = v(x_{n+1}; z_{1:n})$$

where $v(\cdot;\cdot)$ is the likelihood ratio function defined in Eq. (3.2). The weights, $w_i(z_1, \ldots, z_{n+1})$, in Eq. (3.8) then simplify as

$$\begin{aligned}
w_i(z_1, \ldots, z_{n+1}) &= \frac{\sum_{\sigma:\sigma(n+1)=i} \prod_{j=1}^{n+1} g_j(z_{\sigma(j)}; z_{-\sigma(j)})}{\sum_{\sigma} \prod_{j=1}^{n+1} g_j(z_{\sigma(j)}; z_{-\sigma(j)})} \\
&= \frac{\sum_{\sigma:\sigma(n+1)=i} \prod_{j=1}^{n+1} g_j(z_{\sigma(j)}; z_{-\sigma(j)})}{\sum_{k=1}^{n+1} \sum_{\sigma:\sigma(n+1)=k} \prod_{j=1}^{n+1} g_j(z_{\sigma(j)}; z_{-\sigma(j)})} \\
&= \frac{\sum_{\sigma:\sigma(n+1)=i} g_{n+1}(z_{\sigma(n+1)}; z_{-\sigma(n+1)})}{\sum_{k=1}^{n+1} \sum_{\sigma:\sigma(n+1)=k} g_{n+1}(z_{\sigma(n+1)}; z_{-\sigma(n+1)})} \\
&= \frac{\sum_{\sigma:\sigma(n+1)=i} g_{n+1}(z_i; z_{-i})}{\sum_{k=1}^{n+1} \sum_{\sigma:\sigma(n+1)=k} g_{n+1}(z_k; z_{-k})} \\
&= \frac{n! \cdot g_{n+1}(z_i; z_{-i})}{\sum_{k=1}^{n+1} n! \cdot g_{n+1}(z_k; z_{-k})} \\
&= \frac{v(x_i; z_{-i})}{\sum_{k=1}^{n+1} v(x_k; z_{-k})}.
\end{aligned}$$

These quantities are exactly the weight functions, w_i^y, defined in Eq. (3.4) and used in the full conformal confidence set in Eq. (3.3): $w_i^y(X_{\text{test}}) = w_i(Z_1, \ldots, Z_n, (X_{\text{test}}, y))$ for $i = 1, \ldots, n+1$. That is, Eq. (3.3) gives the confidence set defined in Eq. (3.9) for data under FCS. Applying Theorem 2 then yields Theorem 1.

A randomized confidence set achieves exact coverage

Here, we introduce the *randomized β-quantile* and a corresponding randomized confidence set that achieves exact coverage. To lighten notation, for a discrete distribution with probability masses $w = (w_1, \ldots, w_{n+1})$ on points $s = (s_1, \ldots, s_{n+1})$, where $s_i \in \mathbb{R}$ and $w_i \geq 0, \sum_{i=1}^{n+1} w_i = 1$, we will write $\textsc{Quantile}_\beta(s, w) = \textsc{Quantile}_\beta(\sum_{i=1}^{n} w_i \delta_{s_i})$. Observe that $\textsc{Quantile}_\beta(s,w)$ is always one of the support points, s_i. Now define the *β-quantile lower bound*:

$$\textsc{QuantileLB}_\beta\,(s,w) = \inf\left\{ s : \sum_{i:s_i \le s} w_i < \beta, \sum_{i:s_i\le s} w_i + \sum_{j:s_j = \textsc{Quantile}_\beta(s,w)} w_j \ge \beta \right\},$$

which is either a support point strictly less than the β-quantile, or negative infinity. Finally, letting $\mathrm{QF}_\beta(s,w)$ and $\mathrm{LF}_\beta(s,w)$ denote the CDF of the discrete distribution at $\textsc{Quantile}_\beta(s,w)$ and $\textsc{QuantileLB}_\beta(s,w)$), respectively, the randomized β-quantile is a random variable that takes on the value of either the β-quantile or the β-quantile lower bound:

$$\textsc{RandomizedQuantile}_\beta(s,w) = \begin{cases} \textsc{QuantileLB}_\beta(s,w) & \text{w. p. } \frac{\mathrm{QF}_\beta(s,w)-\beta}{\mathrm{QF}_\beta(s,w)-\mathrm{LF}_\beta(s,w)}, \\ \textsc{Quantile}_\beta(s,w) & \text{w. p. } 1-\frac{\mathrm{QF}_\beta(s,w)-\beta}{\mathrm{QF}_\beta(s,w)-\mathrm{LF}_\beta(s,w)}. \end{cases} \tag{3.10}$$

We use this quantity to define the *randomized full conformal* confidence set, which, for any miscoverage level, $\alpha \in (0,1)$, and $x \in \mathcal{X}$ is the following random variable:

$$\begin{aligned} C_\alpha^{\mathrm{rand}}(x) = \{y \in \mathbb{R} : &S((x,y), Z_{1:n}) \qquad (3.11)\\ &\le \textsc{RandomizedQuantile}_{1-\alpha}(s(Z_1,\dots,Z_n,(x,y)), w(Z_1,\dots,Z_n,(x,y))\} \end{aligned}$$

where $s(Z_1,\dots,Z_n,(x,y)) = (S_1,\dots,S_n,S((x,y),Z_{1:n})$ and $S_i = S(Z_i, Z_{-i}\cup\{(x,y)\})$ for $i=1,\dots,n$, and $w(Z_1,\dots,Z_n,(x,y)) = (w_1^y(x),\dots,w_{n+1}^y(x))$ where $w_i^y(x)$ is defined in Eq. (3.4). Note that for each candidate label, $y \in \mathbb{R}$, an independent randomized β-quantile is instantiated; some values will use the β-quantile as the threshold on the score, while the others will use the β-quantile lower bound. Randomizing the confidence set in this way yields the following result.

Theorem 3. *Suppose data, $Z_1,\dots,Z_n,Z_{test}$, are generated under feedback covariate shift and assume $\tilde{P}_{X;D}$ is absolutely continuous with respect to P_X for all possible values of D. Then, for any miscoverage level, $\alpha \in (0,1)$, the randomized full confidence set, C_α^{rand}, in Eq.* (3.11) *satisfies the exact coverage property:*

$$\mathbb{P}(Y_{test} \in C_\alpha^{rand}(X_{test})) = 1-\alpha, \tag{3.12}$$

where the probability is over $Z_1,\dots,Z_n,Z_{test}$ and the randomness in C_α^{rand}.

Proof. Denote $Z_{n+1} = Z_{\mathrm{test}}$ and $Z = (Z_1,\dots,Z_{n+1})$. For a vector of $n+1$ data point values, $z = (z_1,\dots,z_{n+1})$, use the following shorthand:

$$\begin{aligned} Q_\beta(z) &= \textsc{Quantile}_\beta(s(z),w(z)),\\ L_\beta(z) &= \textsc{QuantileLB}_\beta(s(z),w(z)),\\ R_\beta(z) &= \textsc{RandomizedQuantile}_\beta(s(z),w(z)),\\ \mathrm{QF}_\beta(z) &= \mathrm{QF}_\beta(s(z),w(z)),\\ \mathrm{LF}_\beta(z) &= \mathrm{LF}_\beta(s(z),w(z)). \end{aligned}$$

As in the proof of Lemma 1, consider the event, E_z, that $\{Z_1, \dots, Z_{n+1}\} = \{z_1, \dots, z_{n+1}\}$. Assuming for simplicity that the scores are distinct almost surely, by Proposition 2

$$S(Z_{n+1}, Z_{1:n}) \mid E_z \sim \sum_{i=1}^{n+1} w_i(z_1, \dots, z_{n+1})\, \delta_{S(z_i, z_{-i})},$$

and consequently

$$\begin{aligned}
&\mathbb{P}(S(Z_{n+1}, Z_{1:n}) \leq \mathrm{R}_{1-\alpha}(z) \mid E_z) \\
&= \mathbb{P}(S(Z_{n+1}, Z_{1:n}) \leq \mathrm{R}_{1-\alpha}(z) \mid E_z, \mathrm{R}_{1-\alpha}(z) = \mathrm{Q}_{1-\alpha}(z)) \cdot \mathbb{P}(\mathrm{R}_{1-\alpha}(z) = \mathrm{Q}_{1-\alpha}(z) \mid E_z) + \\
&\qquad \mathbb{P}(S(Z_{n+1}, Z_{1:n}) \leq \mathrm{R}_{1-\alpha}(z) \mid E_z, \mathrm{R}_{1-\alpha}(z) = \mathrm{L}_{1-\alpha}(z)) \cdot \mathbb{P}(\mathrm{R}_{1-\alpha}(z) = \mathrm{L}_{1-\alpha}(z) \mid E_z) \\
&= \mathbb{P}(S(Z_{n+1}, Z_{1:n}) \leq \mathrm{Q}_{1-\alpha}(z) \mid E_z) \cdot \left(1 - \frac{\mathrm{QF}_{1-\alpha}(z) - (1-\alpha)}{\mathrm{QF}_{1-\alpha}(z) - \mathrm{LF}_{1-\alpha}(z)}\right) + \\
&\qquad \mathbb{P}(S(Z_{n+1}, Z_{1:n}) \leq \mathrm{L}_{1-\alpha}(z) \mid E_z) \cdot \frac{\mathrm{QF}_{1-\alpha}(z) - (1-\alpha)}{\mathrm{QF}_{1-\alpha}(z) - \mathrm{LF}_{1-\alpha}(z)} \\
&= \mathrm{QF}_{1-\alpha}(z) \cdot \left(1 - \frac{\mathrm{QF}_{1-\alpha}(z) - (1-\alpha)}{\mathrm{QF}_{1-\alpha}(z) - \mathrm{LF}_{1-\alpha}(z)}\right) + \mathrm{LF}_{1-\alpha}(z) \cdot \frac{\mathrm{QF}_{1-\alpha}(z) - (1-\alpha)}{\mathrm{QF}_{1-\alpha}(z) - \mathrm{LF}_{1-\alpha}(z)} \\
&= -\left(\mathrm{QF}_{1-\alpha}(z) - \mathrm{LF}_{1-\alpha}(z)\right) \cdot \frac{\mathrm{QF}_{1-\alpha}(z) - (1-\alpha)}{\mathrm{QF}_{1-\alpha}(z) - \mathrm{LF}_{1-\alpha}(z)} + \mathrm{QF}_{1-\alpha}(z) \\
&= -\mathrm{QF}_{1-\alpha}(z) + (1-\alpha) + \mathrm{QF}_{1-\alpha}(z) \\
&= 1 - \alpha.
\end{aligned}$$

Since we condition on E_z, we equivalently have

$$\mathbb{P}(S(Z_{n+1}, Z_{1:n}) \leq R_{1-\alpha}(Z) \mid E_z) = 1 - \alpha,$$

and since this equality holds for all events E_z, where z is a vector of $n+1$ data point values, taking an expectation over E_z yields

$$\mathbb{P}(S(Z_{n+1}, Z_{1:n}) \leq R_{1-\alpha}(Z)) = 1 - \alpha.$$

Finally, since

$$Y_{n+1} \in C_\alpha^{\mathrm{rand}}(X_{n+1}) \iff S(Z_{n+1}, Z_{1:n}) \leq R_{1-\alpha}(Z),$$

the result follows. □

Note that standard covariate shift is subsumed by feedback covariate shift, so Theorem 3 can be used to construct a randomized confidence set with exact coverage under standard covariate shift as well.

3.6 Data splitting

In general, computing the full conformal confidence set, $C_\alpha(x)$, using Alg. 4 requires fitting $(n+1) \times |\mathcal{Y}|$ regression models. A much more computationally attractive alternative is called a *data splitting* or *split conformal* approach [17, 71], in which we (i) randomly partition the labeled data into disjoint training and *calibration* data sets, (ii) fit a regression model to the training data, and (iii) use the scores that it provides for the calibration data (but not the training data) to construct confidence sets for test data points. Though this approach only requires fitting a single model, the trade-off is that it does not use the labeled data as efficiently: only some fraction of our labeled data can be used to train the regression model. This limitation may be inconsequential for settings with abundant data, but can be a nonstarter when labeled data is limited, such as in many protein design problems.

Here, we show how data splitting simplifies feedback covariate shift (FCS) to standard covariate shift. We then use the data splitting method from Tibshirani et al. [84] to produce confidence sets with coverage; the subsequent subsection shows how to introduce randomization to achieve exact coverage.

To begin, we recall the standard covariate shift model [12, 22]. The training data, $Z_1, \ldots, Z_n$ where $Z_i = (X_i, Y_i)$, are i.i.d. from some distribution: $X_i \sim P_X, Y_i \sim P_{Y|X_i}$ for $i = 1, \ldots, n$. A test data point, $Z_{\text{test}} = (X_{\text{test}}, Y_{\text{test}})$, is drawn from a different input distribution but the same conditional distribution, $X_{\text{test}} \sim \tilde{P}_X, Y_{\text{test}} \sim P_{Y|X_{\text{test}}}$, independently from the training data. In contrast to FCS, here the test input cannot be chosen in a way that depends on the training data.

Returning to FCS, suppose we randomly partition all our labeled data into disjoint training and calibration data sets. Let μ denote the regression model fit to the training data; we henceforth consider μ as fixed and make no further use of the training data. As such, without loss of generality we will use $Z_1, \ldots, Z_m$ to refer to the calibration data. Now suppose the test input distribution is induced by the trained regression model, μ; we write this as $\tilde{P}_{X;\mu}$. Observe that, conditioned on the training data, we now have a setting where the calibration and test data are drawn from different input distributions but the same conditional distribution, $P_{Y|X}$, and are independent of each other. That is, data splitting returns us to standard covariate shift.

To construct valid confidence sets under standard covariate shift, define the following likelihood ratio function:

$$v(x) = \frac{\tilde{p}_{X;\mu}(x)}{p_X(x)}, \tag{3.13}$$

where p_X and $\tilde{p}_{X;\mu}$ refer to the densities of the training and test input distributions, respectively. We restrict our attention to score functions of the following form [143]:

$$S(x, y) = \frac{|y - \mu(x)|}{u(x)}. \tag{3.14}$$

where u is any heuristic, nonnegative notion of uncertainty; one can also set $u(x) = 1$ to recover the residual score function. Note that, since we condition on the training data and treat the regression model as fixed, the score of a point, (x, y), is no longer also a function of other data points. Finally, for any miscoverage level, $\alpha \in (0, 1)$, and any $x \in \mathcal{X}$, define the *split conformal* confidence set as

$$\begin{aligned} C_\alpha^{\text{split}}(x) &= \mu(x) \pm q \cdot u(x), \\ q &= \textsc{Quantile}_{1-\alpha}\left(\sum_{i=1}^{m} w_i(x)\, \delta_{S_i} + w_{n+1}(x)\, \delta_\infty \right), \end{aligned} \tag{3.15}$$

where $S_i = S(X_i, Y_i)$ for $i = 1, \ldots, m$ and

$$\begin{aligned} w_i(x) &= \frac{v(X_i)}{\sum_{j=1}^{m} v(X_j) + v(x)}, \quad i = 1, \ldots, m, \\ w_{m+1}(x) &= \frac{v(x)}{\sum_{j=1}^{m} v(X_j) + v(x)}. \end{aligned} \tag{3.16}$$

For data under standard covariate shift, the split conformal confidence set achieves coverage, as first shown in [84].

Theorem 4 (Corollary 1 in [84]). *Suppose calibration and test data, $Z_1, \ldots, Z_m, Z_{test}$, are under standard covariate shift, and assume $\tilde{P}_{X;\mu}$ is absolutely continuous with respect to P_X. For score functions of the form in Eq.* (3.14)*, and any miscoverage level, $\alpha \in (0, 1)$, the split conformal confidence set, $C_\alpha^{split}(x)$, in Eq.* (3.15) *satisfies the coverage property in Eq.* (3.1)*.*

To achieve exact coverage, we can introduce randomization, as we discuss next.

Data splitting with randomization achieves exact coverage

Here, we stay in the setting and notation of the previous subsection and demonstrate how randomizing the β-quantile enables a data splitting approach to achieve exact coverage. For any score function of the form in Eq. (3.14), any miscoverage level, $\alpha \in (0, 1)$, the *randomized split conformal* confidence set is the following random variable for $x \in \mathcal{X}$:

$$\begin{aligned} C_\alpha^{\text{rand,split}}(x) = \{y \in \mathbb{R} : S(x, y) \\ \leq \textsc{RandomizedQuantile}_{1-\alpha}\left((S_1, \ldots, S_m, S(x, y)), (w_1(x), \ldots, w_{m+1}(x))\right)\}, \end{aligned} \tag{3.17}$$

where the randomized β-quantile, $\textsc{RandomizedQuantile}_\beta$ is defined in Eq. (3.10), $S_i = S(X_i, Y_i)$ for $i = 1, \ldots, m$, and $w_i(\cdot)$ for $i = 1, \ldots, m+1$ is defined in Eq. (3.16). Observe that for each candidate label, $y \in \mathbb{R}$, an independent randomized β-quantile is drawn, such that the scores of some values are compared to the β-quantile while the others are compared to the β-quantile lower bound. The exact coverage property of this confidence set is a consequence of Theorem 3.

Corollary 1. *Suppose calibration and test data, $Z_1, \dots, Z_m, Z_{test}$, are under standard covariate shift, and assume $\tilde{P}_{X;\mu}$ is absolutely continuous with respect to P_X. For score functions of the form in Eq.* (3.14)*, and any miscoverage level, $\alpha \in (0, 1)$, the randomized split conformal confidence set, $C_\alpha^{rand,split}(x)$, in Eq.* (3.17) *satisfies the exact coverage property in Eq.* (3.12)*.*

Proof. Since standard covariate shift is a special case of FCS, the calibration and test data can be described by FCS where $\tilde{P}_{X;D} = \tilde{P}_{X;\mu}$ for any multiset D. The randomized split conformal confidence set, $C_\alpha^{\text{rand,split}}$, is simply the randomized full conformal confidence set, C_α^{rand}, defined in Eq. (3.11), instantiated with the scores $S((x, y), Z_{1:m}) = S(x, y)$ and $S(Z_i, Z_{-i} \cup \{(x, y)\}) = S(Z_i)$ for $i = 1, \dots, m$, and weights resulting from $\tilde{P}_{X;D} = \tilde{P}_{X;\mu}$ for all D. The result then follows from Theorem 3. □

While we only need to fit a single regression model to compute the scores for data splitting, naively it might seem that in practice, we need to approximate $C_\alpha^{\text{rand,split}}(x)$ by introducing a discrete grid of candidate labels, $\mathcal{Y} \subset \mathbb{R}$, and computing a randomized β-quantile for $|\mathcal{Y}|$ different discrete distributions. Fortunately, we can construct an alternative confidence set that also achieves exact coverage, the *randomized staircase* confidence set, $C_\alpha^{\text{staircase}}$, which only requires sorting m scores and an additional $O(m)$ floating point operations to compute (see Alg. 5).

At a high level, its construction is based on the observation that for any $x \in \mathcal{X}$ and $y \in \mathbb{R}$, the quantity $\mathbb{P}(y \in C_\alpha^{\text{rand,split}}(x))$, where the probability is over the randomness in $C_\alpha^{\text{rand,split}}(x)$, is a piecewise constant function of y. Instead of testing each value of $y \in \mathbb{R}$, we can then construct this piecewise constant function, and randomly include entire intervals of y values that have the same value of $\mathbb{P}(y \in C_\alpha^{\text{rand,split}}(x))$.

Fig. 3.6 illustrates this observation, which we now explain. First, the discrete distribution in Eq. (3.17) has probability masses $w_1(x), \dots, w_{m+1}(x)$ at the points $S_1, \dots, S_m, S(x, y)$, respectively. Given the values of the m calibration data points and the test input, x, all of these quantities are fixed—except for the score of the candidate test data point, $S(x, y)$. That is, the only quantity that depends on the value of y is $S(x, y)$, which is the location of the probability mass $w_{m+1}(x)$; the remaining m support points and their corresponding probability masses do not not change with y.

Now consider the calibration scores, $S_1, \dots, S_m$, sorted in ascending order. Observe that for any pair of successive sorted scores, $S_{(i)}$ and $S_{(i+1)}$, the entire interval of y values such that $S(x, y) \in (S_{(i)}, S_{(i+1)})$ belongs to one of three categories: $S(x, y) \leq \beta$-quantile lower bound (of the discrete distribution with probability masses $w_1, \dots, w_{m+1}$ at support points $S_1, \dots, S_m, S(x, y)$), $S(x, y) = \beta$-quantile, or $S(x, y) > \beta$-quantile. An interval of y values that belongs to the first category is deterministically included in $C_\alpha^{\text{rand,split}}(x)$, regardless of the randomness in the randomized β-quantile (color-coded green in Fig. 3.6), while an interval that belongs to the last category is deterministically excluded (color-coded purple in Fig. 3.6). The only y values whose inclusion is not deterministic are those in the second category (color-coded teal and blue), which are randomly included with the probability, given in Eq. (3.10), that the randomized β-quantile equals the β-quantile. Consequently, we can

Algorithm 5 Randomized staircase confidence set

Input: Miscoverage level, $\alpha \in (0,1)$; calibration data, $Z_1, \dots, Z_m$, where $Z_i = (X_i, Y_i)$; test input, X_{test}; subroutine for likelihood ratio function, $v(\cdot)$, defined in Eq. (3.13); subroutine for uncertainty heuristic, $u(\cdot)$; subroutine for regression model prediction, $\mu(\cdot)$.

Output: Randomized staircase confidence set, $C = C_\alpha^{\text{staircase}}(X_{\text{test}})$.

```
for $i = 1, \dots, m$ do                                   ▷ Compute calibration scores
    $S_i \leftarrow |Y_i - \mu(X_i)|/u(X_i)$
    $v_i \leftarrow v(X_i)$
end for
$v_{m+1} \leftarrow v(X_{\text{test}})$
for $i = 1, \dots, m+1$ do                                 ▷ Compute calibration and test weights
    $w_i \leftarrow v_i / \sum_{j=1}^{m+1} v_j$
end for
$C \leftarrow \emptyset$
LowerBoundIsSet ← False
$S_{(0)} = 0, w_0 = 0$                                     ▷ Dummy values so for-loop will include $[0, S_{(1)}]$
for $i = 0, \dots, m-1$ do
    if $\sum_{j: S_j \le S_{(i)}} w_j + w_{m+1} < 1 - \alpha$ then      ▷ $S(x,y) \le \beta$-quantile lower bound, so include deterministically
        $C = C \cup \big[\mu(X_{\text{test}}) + S_{(i)} \cdot u(X_{\text{test}}), \mu(X_{\text{test}}) + S_{(i+1)} \cdot u(X_{\text{test}})\big] \cup \big[\mu(X_{\text{test}}) - S_{(i+1)} \cdot u(X_{\text{test}}), \mu(X_{\text{test}}) - S_{(i)} \cdot u(X_{\text{test}})\big]$
    else if $\sum_{j: S_j \le S_{(i)}} w_j + w_{m+1} \ge 1 - \alpha$ and $\sum_{j: S_j \le S_{(i)}} w_j < 1 - \alpha$ then      ▷ $S(x,y) = \beta$-quantile, so randomize inclusion
        if LowerBoundIsSet = False then
            LowerBoundIsSet ← True                         ▷ Set $\beta$-quantile lower bound
            $LF = \sum_{j: S_j \le S_{(i)}} w_j$
        end if
        $F \leftarrow \frac{\sum_{j: S_j \le S_{(i)}} w_j + w_{m+1} - (1-\alpha)}{\sum_{j: S_j \le S_{(i)}} w_j + w_{m+1} - LF}$
        $b \sim \text{Bernoulli}(1 - F)$
        if $b$ then
            $C = C \cup \big[\mu(X_{\text{test}}) + S_{(i)} \cdot u(X_{\text{test}}), \mu(X_{\text{test}}) + S_{(i+1)} \cdot u(X_{\text{test}})\big] \cup \big[\mu(X_{\text{test}}) - S_{(i+1)} \cdot u(X_{\text{test}}), \mu(X_{\text{test}}) - S_{(i)} \cdot u(X_{\text{test}})\big]$
        end if
    end if
end for
if $\sum_{i=1}^{m} w_i < 1 - \alpha$ then      ▷ For $S(x,y) > S_{(m)}$, either $S(x,y) = \beta$-quantile or $S(x,y) > \beta$-quantile
    if LowerBoundIsSet = False then
        $LF = \sum_{i=1}^{m} w_i$
    end if
    $F \leftarrow \frac{1-(1-\alpha)}{1-LF}$
    $b \sim \text{Bernoulli}(1 - F)$
    if $b$ then
        $C = C \cup \big[\mu(X_{\text{test}}) + S_{(m)} \cdot u(X_{\text{test}}), \infty\big] \cup \big[-\infty, \mu(X_{\text{test}}) - S_{(m)} \cdot u(X_{\text{test}})\big]$
    end if
end if
```

identify the intervals of y values belonging to each of these categories, and for those in the second category, compute the probability that the randomized β-quantile is instantiated as the β-quantile, which is $\mathbb{P}(y \in C_\alpha^{\text{rand,split}}(x))$.

This probability turns out to be a piecewise constant function of y. Note that it is computed from two quantities: the c.d.f. at the β-quantile and the c.d.f at the β-quantile lower bound (see Eq. (3.10)). As depicted in Fig. 3.6 (third panel from top), for any two successive sorted calibration scores, $S_{(i)}$ and $S_{(i+1)}$, both of these quantities are constant over $S(x, y) \in (S_{(i)}, S_{(i+1)})$. That is, both the c.d.f. at the β-quantile and the c.d.f. at β-quantile lower bound are piecewise constant functions of y, which only change values at the calibration scores, $S_1, \ldots, S_m$ (and can take on different values exactly at the calibration scores). Consequently, the probability $\mathbb{P}(y \in C_\alpha^{\text{rand,split}}(x))$ is also a piecewise constant function of y, which only changes values at the calibration scores. It attains its highest value at $\mu(x)$ and decreases as y moves further away from it, resembling a staircase, as depicted in Fig. 3.6 (fourth panel from the top).

Therefore, instead of computing a randomized β-quantile for all $y \in \mathbb{R}$, we can simply compute the value of this probability on the $m + 1$ intervals between neighboring sorted calibration scores: $[0, S_{(1)}), (S_{(1)}, S_{(2)}), \ldots, (S_{(m-1)}, S_{(m)}), (S_{(m)}, \infty]$, as well as its value exactly at the m calibration scores. These probabilities may equal 1 or 0, which correspond to the two cases earlier described wherein y is deterministically included or excluded, respectively. If the probability is not 1 or 0, then we can randomly include the entire set of values of y such that $S(x, y)$ falls in the interval. Due to the form of the score in Eq. (3.14), this set comprises two equal-length intervals on both sides of $\mu(x)$: $(\mu(x) - S_{(i+1)}, \mu(x) - S_{(i)}) \cup (\mu(x) + S_{(i+1)}, \mu(x) + S_{(i)})$.

Finally, if we assume that scores are distinct almost surely, then our treatment of values of y such that $S(x, y) = S_i$ for $i = 1, \ldots, m$, does not affect the exact coverage property. For simplicity, Alg. 5 therefore includes or excludes closed intervals that contain these y values as endpoints, rather than treating them separately.

More general score functions. In the reasoning above, we use the assumption that the score function takes the form in Eq. (3.14) only at the end of the argument, to infer the form of the sets of y values. We can relax this assumption as follows. For any continuous score function, consider the preimage of the intervals $[0, S_{(1)}), (S_{(1)}, S_{(2)}), \ldots, (S_{(m-1)}, S_{(m)}), (S_{(m)}, \infty]$ under the function $S(x, \cdot)$ (a function of the second argument with x held fixed), rather than the intervals given explicitly in Lines 14, 23, and 34 of Alg. 5. This algorithm then gives exact coverage for any continuous score function, although it will only be computationally feasible when these preimages can be computed efficiently.

3.7 Efficient algorithms for full conformal prediction

Ridge regression

When the likelihood of the test input is a function of the prediction from a ridge regression model, it is possible to compute the scores and weights for the full conformal confidence set by fitting $n+1$ models and $O(n \cdot p \cdot |\mathcal{Y}|)$ additional floating point operations, instead of naively fitting $(n+1) \times \mathcal{Y}$ models, as demonstrated in Alg. 6.

For the fluorescent protein design experiments, the TestInputLikelihood subroutine in Alg. 6 computed the likelihood in Eq. (3.6), that is,

$$\begin{aligned} \textsc{TestInputLikelihood}(a_i + b_i y) &\leftarrow \frac{\exp(\lambda \cdot (a_i + b_i y))}{\cdot \sum_{x \in \mathcal{X}} \exp(\lambda \cdot (C_i + y\mathbf{A}_{-i,n})^T x)}, \\ \textsc{TestInputLikelihood}(a_{n+1}) &\leftarrow \frac{\exp(\lambda \cdot a_{n+1})}{\cdot \sum_{x \in \mathcal{X}} \exp(\lambda \cdot \beta^T x)}, \end{aligned} \tag{3.18}$$

where the input space $\mathcal{X}$ was the combinatorially complete set of 8,192 sequences. The TrainInputLikelihood subroutine returned the likelihood under the training input distribution, which is simply equal to to 1/8192, since training sequences were sampled uniformly from the combinatorially complete data set. See `https://github.com/clarafy/conformal-for-design` for an implementation.

Computing the test input likelihoods was dominated by the $(n+1) \times |\mathcal{Y}|$ normalizing constants, which can be computed efficiently using a single tensor product between an $(n+1) \times p \times |\mathcal{Y}|$ tensor containing the model parameters, $C_i + y\mathbf{A}_{-i,n}$ and β, and an $|\mathcal{X}| \times p$ data matrix containing all inputs in $\mathcal{X}$. For domains, $\mathcal{X}$, that are too large for the normalizing constants to be computed exactly, one can turn to tractable Monte Carlo approximations.

Gaussian process regression

Here we describe how the scores and weights for the confidence set in Eq. (3.3) can be computed efficiently, when the likelihood of the test input distribution is a function of the predictive mean and variance of a Gaussian process regression model.

For an arbitrary kernel and two data matrices, $\mathbf{V} \in \mathbb{R}^{n_1 \times p}$ and $\mathbf{V}' \in \mathbb{R}^{n_2 \times p}$, let $K(\mathbf{V}, \mathbf{V}')$ denote the $n_1 \times n_2$ matrix where the (i,j)-th entry is the covariance between the i-th row of $\mathbf{V}$ and j-th row of $\mathbf{V}'$. The mean prediction for X_i of a Gaussian process regression model fit to the i-th augmented LOO data set, $\mu^y_{-i}(X_i)$, is then given by

$$\mu^y_{-i}(X_i) = K(X_i, \mathbf{X}_{-i})[K(\mathbf{X}_{-i}, \mathbf{X}_{-i}) + \sigma^2 I]^{-1} Y^y_{-i},$$

and the model's predictive variance at X_i is

$$K(X_i, X_i) - K(X_i, \mathbf{X}_{-i})[K(\mathbf{X}_{-i}, \mathbf{X}_{-i}) + \sigma^2 I]^{-1} K(\mathbf{X}_{-i}, X_i),$$

Algorithm 6 Efficient computation of scores and weights for ridge regression-based feedback covariate shift

Input: training data, $Z_1, \ldots, Z_n$, where $Z_i = (X_i, Y_i)$; test input, X_{n+1}; grid of candidate labels, $\mathcal{Y} \subset \mathbb{R}$; subroutine for test input likelihood, TestInputLikelihood(·), that takes an input's predicted fitness and outputs its likelihood under the test input distribution; subroutine for training input likelihood, TrainInputLikelihood(·).

Output: scores $S_i(X_{n+1}, y)$ and likelihood ratios $v(X_i, Z^y_{-i})$ for $i = 1, \ldots, n+1$, $y \in \mathcal{Y}$.

```
for $i = 1, \ldots, n$ do
    $C_i \leftarrow \sum_{j=1}^{n-1} Y_{-i;j} \mathbf{A}_{-i;j}$
    $a_i \leftarrow C_i^T X_i$
    $b_i \leftarrow \mathbf{A}^T_{-i;n} X_i$
end for
$\beta \leftarrow (\mathbf{X}^T \mathbf{X} + \gamma I)^{-1} \mathbf{X}^T Y$
$a_{n+1} \leftarrow \beta^T X_{n+1}$
for $i = 1, \ldots, n$ do
    for $y \in \mathcal{Y}$ do
        $S_i(X_{n+1}, y) \leftarrow |Y_i - (a_i + b_i y)|$    ▷ Can vectorize via outer product between $(b_1, \ldots, b_n)$ and vector of all $y \in \mathcal{Y}$.
        $v(X_i; Z_{-i,y}) \leftarrow$ TestInputLikelihood$(a_i + b_i y)$/TrainInputLikelihood$(X_i)$    ▷ Can vectorize (see commentary on Eq. (3.18)).
    end for
end for
$S_{n+1}(X_{n+1}, y) \leftarrow |y - a_{n+1}|$
$v(X_{n+1}; Z_{1:n}) \leftarrow$ TestInputLikelihood$(a_{n+1})$/TrainInputLikelihood$(X_{n+1})$
```

where the rows of the matrix $\mathbf{X}_{-i} \in \mathbb{R}^{n \times p}$ are the inputs in Z^y_{-i}, $Y^y_{-i} = (Y_{-i}, y) \in \mathbb{R}^n$ is the vector of labels in Z^y_{-i}, and σ^2 is the (unknown) variance of the label noise, whose value is set as a hyperparameter. Note that the mean prediction is a linear function of the candidate value, y, which is of the same form as the ridge regression prediction in Eq. (3.5); furthermore, the predictive variance is constant over y. Therefore, we can mimic Alg. 6 to efficiently compute scores and weights by training just $n+1$ rather than $(n+1) \times |\mathcal{Y}|$ models.

3.8 Experimental details and additional results

Designing fluorescent proteins

Features Each sequence was first represented as a length-thirteen vector of signed bits (-1 or 1), each denoting which of the two wild-type parents the amino acid at a site matches. The features for the sequence consisted of these thirteen signed bits, called the first-order terms in the main text, as well as all $\binom{13}{2}$ products between pairs of these thirteen bits, called the second-order interaction terms.

Additional simulated measurement noise. Each time the i-th sequence in the combinatorially complete data set was sampled, for either training or designed data, we introduced additional simulated measurement noise using the following procedure. Poelwijk et al. [83] found that the Walsh-Hadamard transform of the brightness fitness landscape included up to seventh-order statistically significant terms. Accordingly, we fit a linear model of up to seventh-order terms for each of the combinatorially complete data sets, then estimated the standard deviation of the i-th sequence's measurement noise, σ_i, as the residual between its label and this model's prediction. Each time the i-th sequence was sampled, for either training or designed data, we also sampled zero-mean Gaussian noise with standard deviation σ_i and added it to the i-th sequence's label. This was done to simulate the fact that multiple measurements of the same sequence will yield different labels, due to measurement noise.

Designing AAV capsids

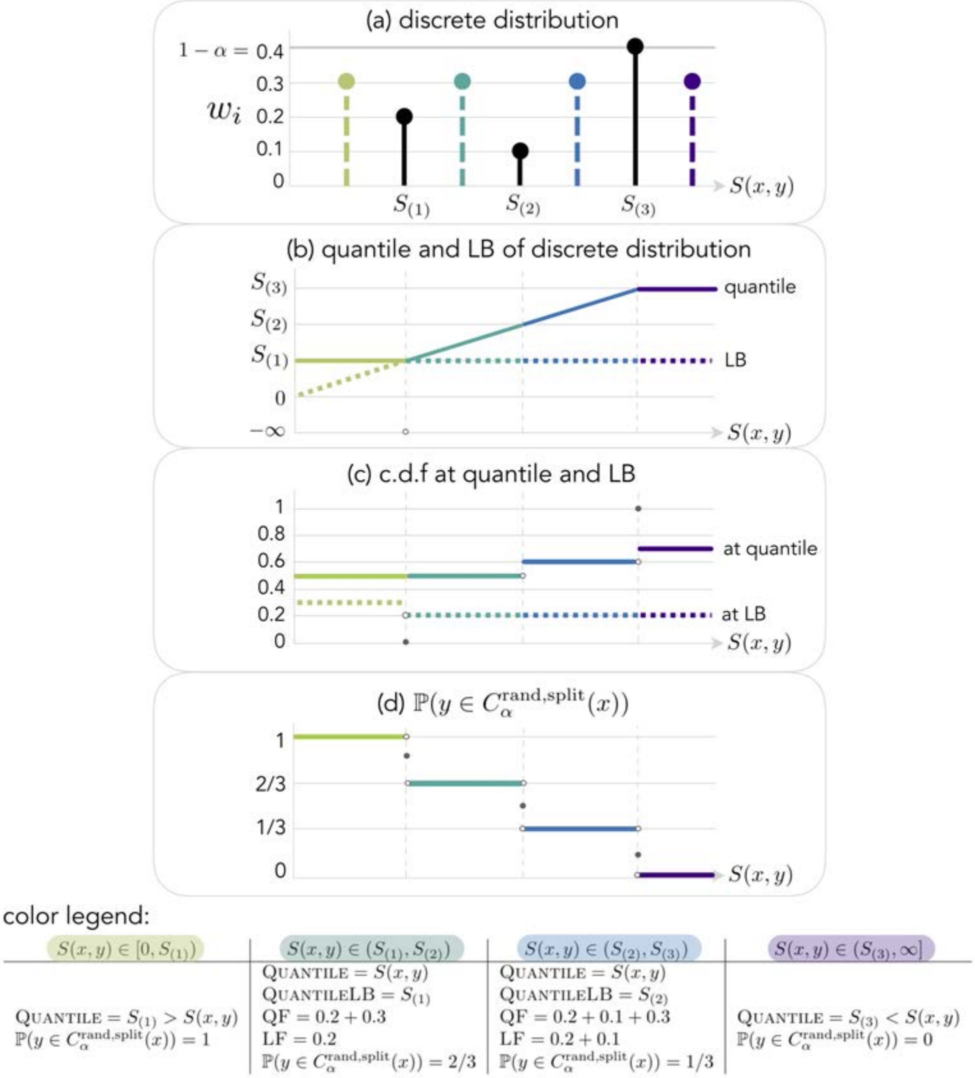
NNK sequence distribution. The NNK sequence distribution is parameterized by independent categorical distributions over the four nucleotides, where the probabilities of the nucleotides are intended to result in a high diversity of amino acids while avoiding stop codons. Specifically, for three contiguous nucleotides corresponding to a codon, the first two nucleotides are sampled uniformly at random from $\{\mathtt{A}, \mathtt{C}, \mathtt{T}, \mathtt{G}\}$, while the last nucleotide is sampled uniformly at random from only $\{\mathtt{T}, \mathtt{G}\}$.

Additional simulated measurement noise. Following Zhu & Brookes et al. [124], the fitness assigned to the i-th sequence was an enrichment score based on its counts before and after a selection experiment, $n_{i,\text{pre}}$ and $n_{i,\text{post}}$, respectively. The variance of this enrichment score for the i-th sequence was estimated as

$$\sigma_i^2 = \frac{1}{n_{i,\text{post}}}\left(1 - \frac{n_{i,\text{post}}}{N_\text{post}}\right) + \frac{1}{n_{i,\text{pre}}}\left(1 - \frac{n_{i,\text{pre}}}{N_\text{pre}}\right)$$

where N_pre and N_post denote the total counts of all the sequences before and after the selection experiment, respectively. Using this estimate, we introduced additional simulated measurement noise to the label of the i-th sequence by adding zero-mean Gaussian noise with a variance of $0.1 \cdot \sigma_i^2$.

Neural network details. As in [124], the neural network took one-hot-encoded sequences as inputs and had an architecture consisting of two fully connected hidden layers, with 100 units each and `tanh` activation functions. It was fit to the $7,552,729$ training data points with the built-in implementation of the Adam algorithm in Tensorflow, using the default hyperparameters and a batch size of 64 for 10 epochs, where each training data point was weighted according to its estimated variance as in [124].



$S(x,y) \in [0, S_{(1)})$	$S(x,y) \in (S_{(1)}, S_{(2)})$	$S(x,y) \in (S_{(2)}, S_{(3)})$	$S(x,y) \in (S_{(3)}, \infty]$
$\textsc{Quantile} = S_{(1)} > S(x,y)$ $\mathbb{P}(y \in C_\alpha^{\text{rand,split}}(x)) = 1$	$\textsc{Quantile} = S(x,y)$ $\textsc{QuantileLB} = S_{(1)}$ $\text{QF} = 0.2 + 0.3$ $\text{LF} = 0.2$ $\mathbb{P}(y \in C_\alpha^{\text{rand,split}}(x)) = 2/3$	$\textsc{Quantile} = S(x,y)$ $\textsc{QuantileLB} = S_{(2)}$ $\text{QF} = 0.2 + 0.1 + 0.3$ $\text{LF} = 0.2 + 0.1$ $\mathbb{P}(y \in C_\alpha^{\text{rand,split}}(x)) = 1/3$	$\textsc{Quantile} = S_{(3)} < S(x,y)$ $\mathbb{P}(y \in C_\alpha^{\text{rand,split}}(x)) = 0$

Figure 3.6: **Probability $\mathbb{P}(y \in C_\alpha^{\textbf{rand,split}}(x))$ is a piecewise constant function of y.** (a) Given the values of the calibration data and test input, the scores $S_1, \ldots, S_m$ and corresponding probability masses $w_1, \ldots, w_m$ (black stems), as well as the probability mass for the test input, $w_{m+1} = 0.3$, are fixed. The only quantity that depends on y is $S(x,y)$. Four example values are shown as dashed green, teal, blue, and purple stems, representing values in $[0, S_{(1)}), (S_{(1)}, S_{(2)}), (S_{(2)}, S_{(3)})$, and $(S_{(3)}, \infty]$, respectively (see color legend). Note that in this example, $1-\alpha = 0.4$. (b) The 0.4-quantile and 0.4-quantile lower bound of the discrete distribution in the top panel as a function of $S(x,y)$, where the colors correspond to values of $S(x,y)$ in the intervals just listed. Note the discontinuity in the 0.4-quantile lower bound at $S(x,y) = S_{(1)}$. (c) The c.d.f. of the discrete distribution at the 0.4-quantile and 0.4-quantile lower bound. Note the discontinuities when $S(x,y)$ equals a calibration score. (d) The probability $\mathbb{P}(y \in C_\alpha^{\text{rand,split}}(x))$, which equals 1 or 0 if $S(x,y) =$ 0.4-quantile lower bound or $S(x,y) >$ 0.4-quantile, respectively, and otherwise equals the probability in Eq. (3.10) that the randomized 0.4-quantile equals the 0.4-quantile: $1 - \frac{\text{QF}-0.4}{\text{QF}-\text{LF}}$, where QF and LF denote the c.d.f. at the 0.4-quantile and 0.4-quantile lower bound, respectively. Color legend: calculations of the plotted quantities (calculations for $S(x,y) = S_{(i)}$ omitted).

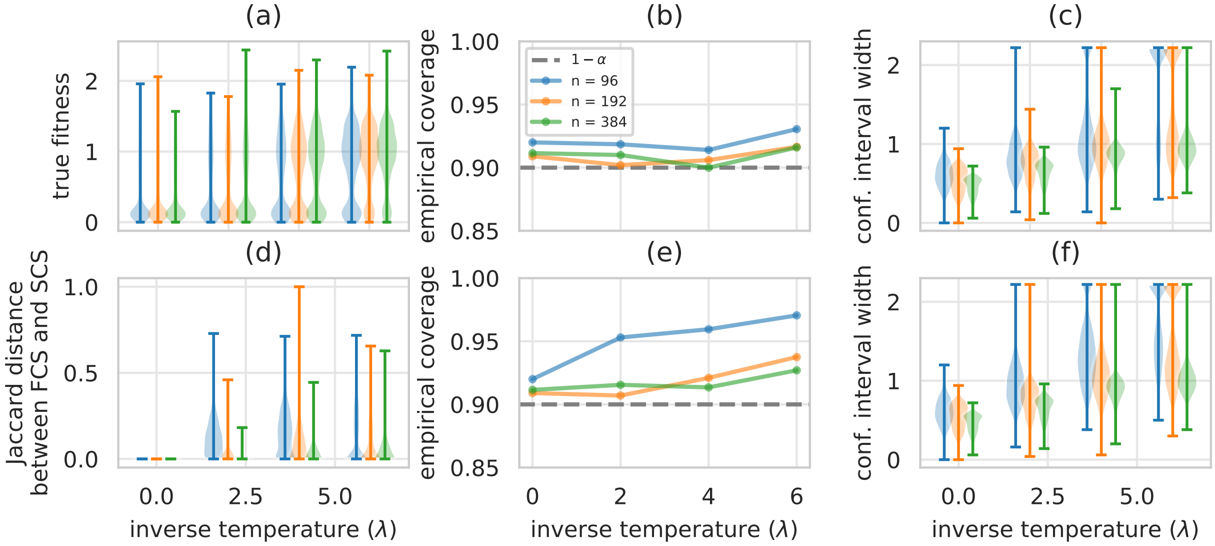


Figure 3.7: **Quantifying uncertainty for designed proteins (red fluorescence).** (a) Distributions of labels of designed proteins, for different values of the inverse temperature, λ, and different amounts of training data, n. Labels surpass the fitness range observed in the combinatorially complete data set, $[0.025, 1.692]$, due to additional simulated measurement noise. (b) Empirical coverage, compared to the theoretical lower bound of $1 - \alpha = 0.9$ (dashed gray line), and (c) distributions of confidence interval widths achieved by full conformal prediction for feedback covariate shift (our method) over $T = 2000$ trials. (d) Distributions of Jaccard distances between the confidence intervals produced by full conformal prediction for feedback covariate shift and standard covariate shift [84]. (e, f) Same as (b, c) but using full conformal prediction for standard covariate shift. In (a), (c), (d), and (f), the whiskers signify the minimum and maximum observed values.

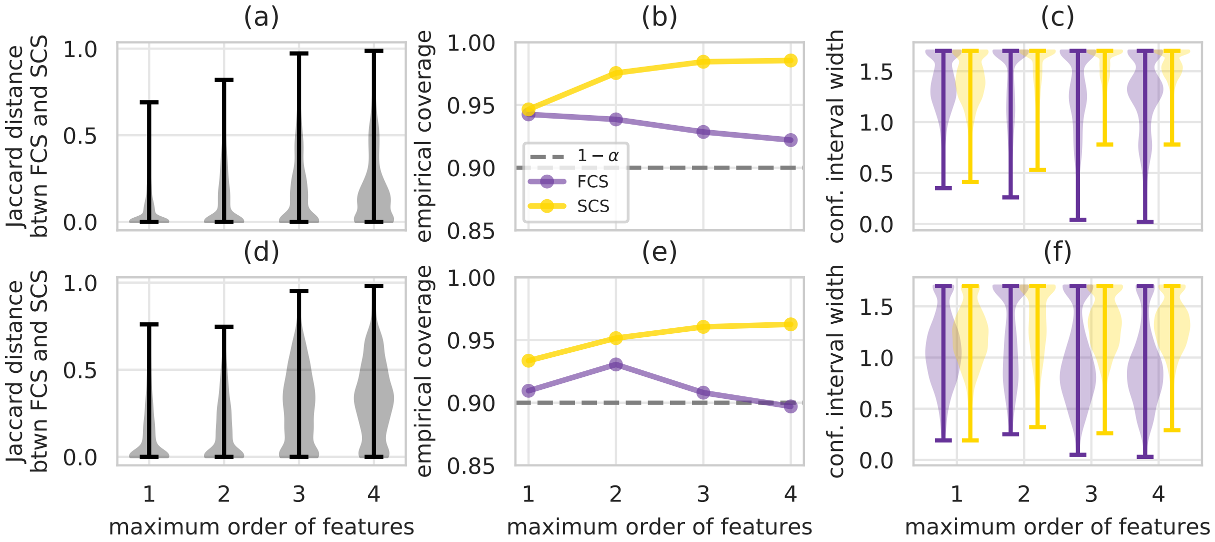


Figure 3.8: **Quantifying uncertainty for designed proteins, for $n = 48$ training data points, $\lambda = 6$, and ridge regression models with features of varying complexity.** In particular, the features consist of all interaction terms up to order d between the thirteen sequence sites, where the maximum order, d, is the x-axis of the following subplots. (a) Distributions of Jaccard distances between the confidence intervals produced by conformal prediction for feedback covariate shift (FCS, our method) and standard covariate shift (SCS) [84] for the blue fluorescence data set over $T = 2000$ trials. (b) Empirical coverage, compared to the theoretical lower bound of $1 - \alpha = 0.9$ (dashed gray line), achieved by conformal prediction for FCS and SCS over those trials. (c) Distributions of confidence interval widths using conformal prediction for FCS and SCS. (d-f) Same as (a-c) but for the red fluorescence data set. In (a), (c), (d), and (f), whiskers signify the minimum and maximum observed values.

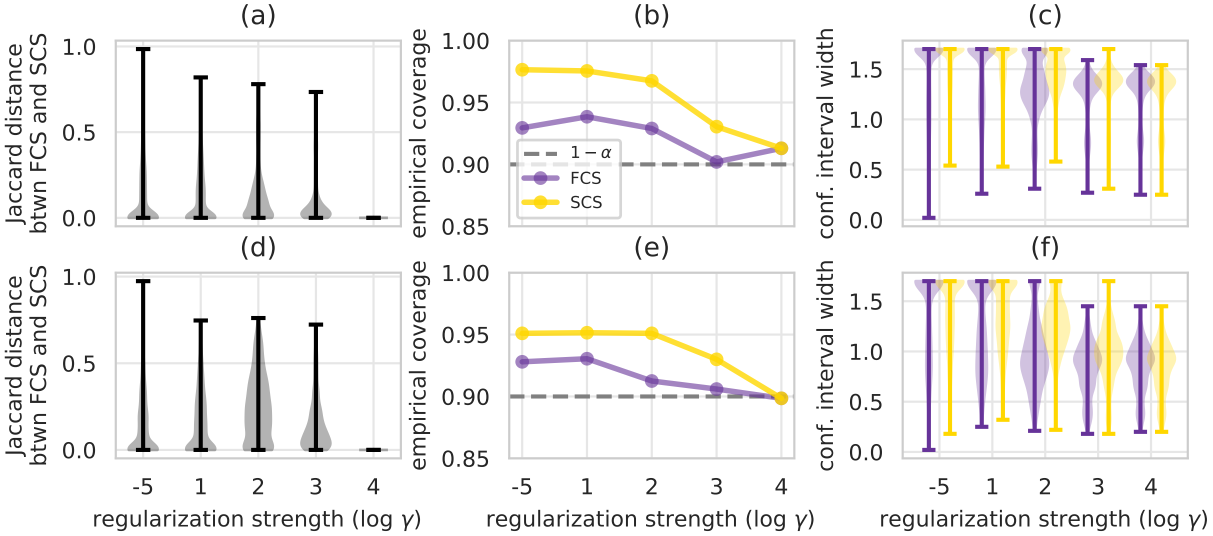


Figure 3.9: **Quantifying uncertainty for designed proteins, for $n = 48$ training data points, $\lambda = 6$, and varying ridge regularization strength, γ.** (a) Distributions of Jaccard distances between the confidence intervals produced by conformal prediction for feedback covariate shift (FCS, our method) and standard covariate shift (SCS) [84] for the blue fluorescence data set over $T = 2000$ trials. (b) Empirical coverage, compared to the theoretical lower bound of $1 - \alpha = 0.9$ (dashed gray line), achieved by conformal prediction for FCS and SCS over those trials. (c) Distributions of confidence interval widths using conformal prediction for FCS and SCS. (d-f) Same as (a-c) but for the red fluorescence data set. In (a), (c), (d), and (f), whiskers signify the minimum and maximum observed values.

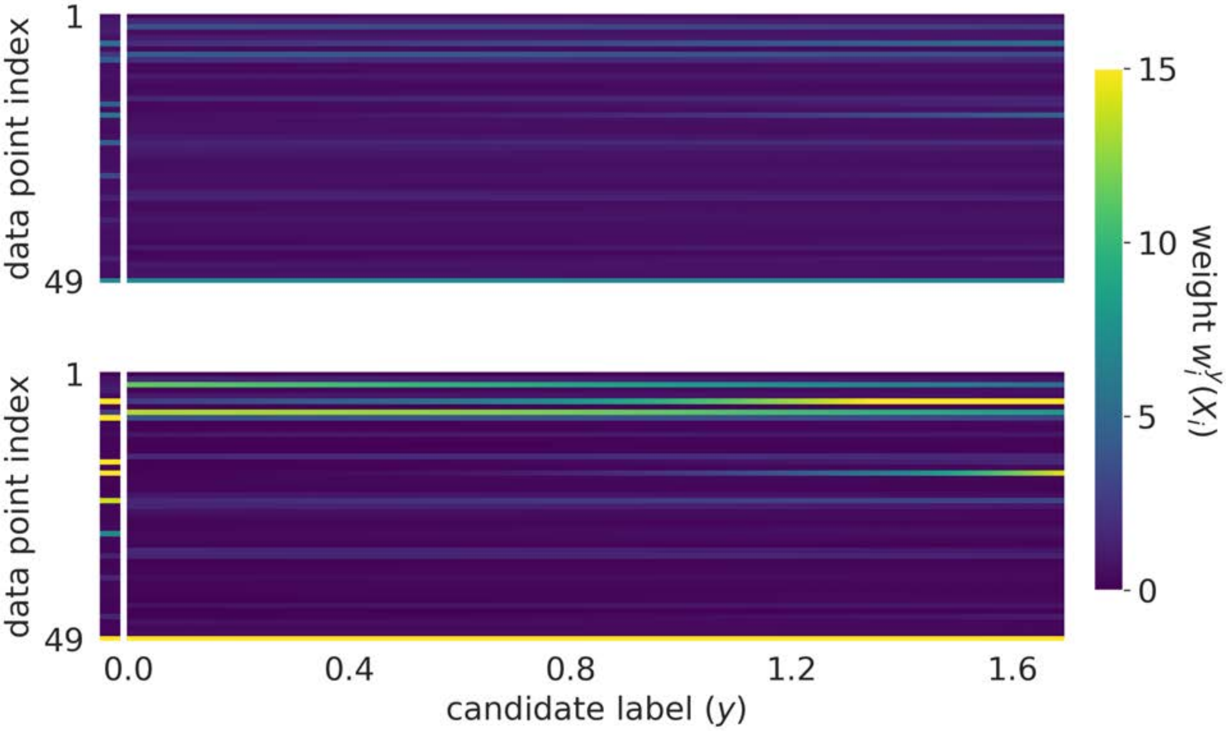


Figure 3.10: **Comparison of weights constructed by conformal prediction for feedback covariate shift (FSC, our method) and standard covariate shift (SCS) [84] for one example blue fluorescence training data set and resulting designed sequence, for $n = 48$ and two different settings of the inverse temperature, λ.** Top: For $\lambda = 2$, vector of the $n+1$ weights prescribed under SCS for the n training data points (data point indices 1 through 48) and the candidate test data points (data point index 49), alongside $(n+1) \times |\mathcal{Y}|$ matrix of the weights prescribed under FCS for those same $n+1$ training and candidate test data points. The weight for each of these data points depends on the candidate label, y (x-axis of heatmap), through a linear relationship with y (see Section 3.2). Bottom: same as top but for $\lambda = 6$.

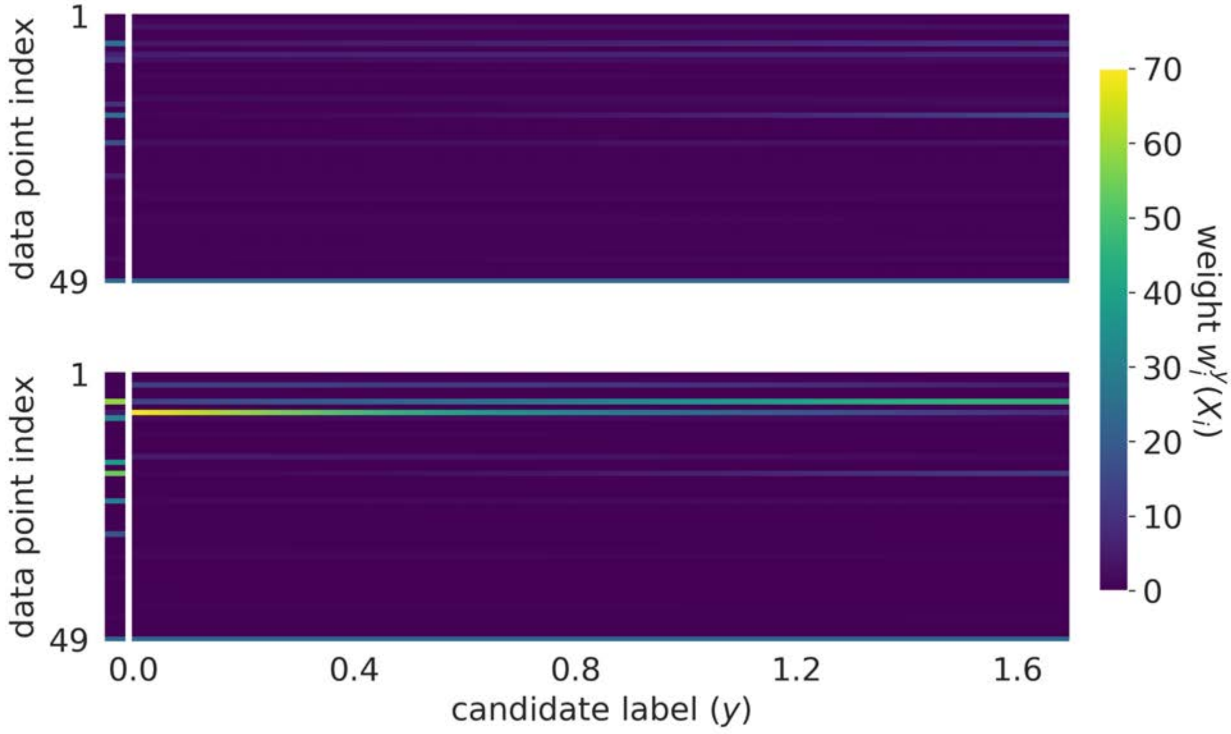


Figure 3.11: **Comparison of weights constructed by conformal prediction for feedback covariate shift (FSC, our method) and standard covariate shift (SCS) [84] for one example blue fluorescence training data set and resulting designed sequence, for $n = 48$ and two different ridge regularization strengths, γ.** Top: For $\gamma = 100$, vector of the $n + 1$ weights prescribed under SCS for the n training data points (data point indices 1 through 48) and the candidate test data points (data point index 49), alongside $(n + 1) \times |\mathcal{Y}|$ matrix of the weights prescribed under FCS for those same $n + 1$ training and candidate test data points. The weight for each of these data points depends on the candidate label, y (x-axis of heatmap), through a linear relationship with y (see Section 3.2). Bottom: same as top but for $\gamma = 10$.

Chapter 4

Prediction-powered inference

The material in this chapter is based on work co-authored with Anastasios N. Angelopoulos, Stephen Bates, Michael I. Jordan, and Tijana Zrnic [144]

4.1 Predictions as data for scientific inquiry

Imagine a scientist has a machine-learning system that can supply accurate predictions about a phenomenon far more cheaply than any gold-standard experimental technique. The scientist may wish to use these predictions as evidence in drawing scientific conclusions. For example, accurate predictions of three-dimensional structures have been made for a vast catalog of known protein sequences [116] and are now being used in proteomics studies [127, 145]. Such machine-learning systems are becoming increasingly common in modern scientific inquiry. However, predictions are not perfect, which may lead to incorrect conclusions. How can modern science leverage machine-learning predictions in a statistically principled way?

One way to use predictions, which we call the *imputation approach*, is to proceed as if they are gold-standard measurements. Although this lets the scientist draw conclusions cheaply and quickly due to the high-throughput nature of the machine-learning system, the conclusions may be invalid because the predictions may have biases.

An alternative approach is to ignore the machine-learning predictions altogether and only use the available gold-standard measurements, which are typically far less abundant. We call this the *classical approach*. The resulting discoveries will be statistically valid, but the smaller amount of data will limit the scope of possible discoveries.

We present *prediction-powered inference*, a framework that achieves the best of both worlds: extracting information from the predictions of a high-throughput machine-learning system, while guaranteeing statistical validity of the resulting conclusions. Prediction-powered inference provides a protocol for combining predictions, which are abundant but not always trustworthy, with gold-standard data, which is trusted but scarce, to compute confidence intervals and p-values. The resulting confidence intervals and p-values are statistically valid, as in the classical approach, but also leverage the information contained in the predictions, as in the imputation approach, to make the confidence intervals smaller and the p-values more powerful.

Prediction-powered inference can be used with any machine-learning system; as such, it absolves the need for case-by-case analyses dependent on the machine-learning algorithm on hand. The proposed protocol thereby enables researchers to report machine learning-based scientific conclusions in a fully standardized way.

General principle of prediction-powered inference

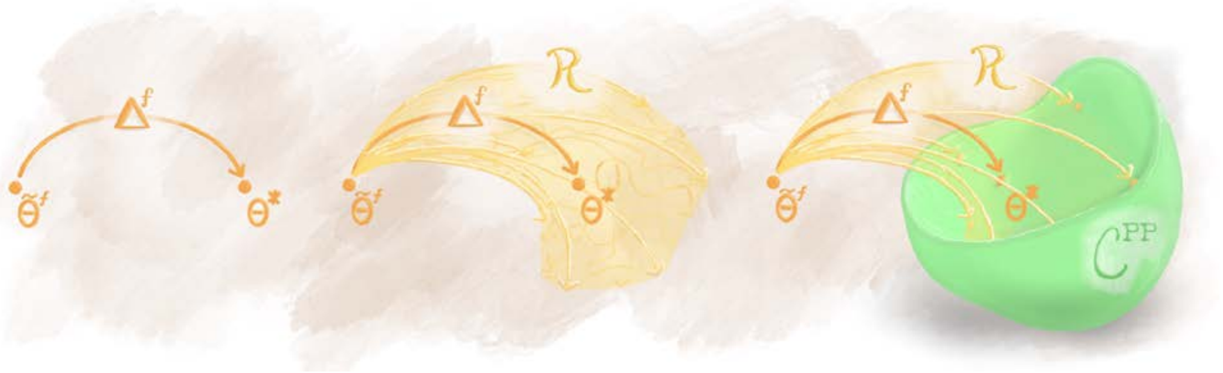
The goal in prediction-powered inference is to estimate a quantity, $\theta^\star$, such as the mean or median value of the label over a population of interest. Toward this end, we have access to a small *gold-standard* data set of features paired with labels, $(X, Y) = \big((X_1, Y_1), \dots, (X_n, Y_n)\big)$, as well as the features of a large *unlabeled* data set, $(\widetilde{X}, \widetilde{Y}) = \big((\widetilde{X}_1, \widetilde{Y}_1), \dots, (\widetilde{X}_N, \widetilde{Y}_N)\big)$, where we do not observe the labels, $\widetilde{Y}_1, \dots, \widetilde{Y}_N$. We focus on the practically relevant setting where $N \gg n$, and assume that (X, Y) and $(\widetilde{X}, \widetilde{Y})$ are independently and identically distributed from a common distribution, $\mathbb{P}$.

Next, we have a prediction rule, $f : \mathcal{X} \to \mathcal{Y}$, that is independent of the observed data. For example, it may be a machine-learning model trained on data independent from both the labeled and the unlabeled data. We let $f_i = f(X_i)$ and $\tilde{f}_i = f(\widetilde{X}_i)$ denote the predictions for the labeled and unlabeled data, respectively, and let $f = (f_1, \dots, f_n)$ and $\tilde{f} = (\tilde{f}_1, \dots, \tilde{f}_N)$, with slight abuse of notation.

Prediction-powered inference builds confidence intervals that are guaranteed to contain $\theta^\star$ with high probability. Imagine we have an estimator $\hat{\theta}$ of $\theta^\star$. One naive way to estimate $\theta^\star$, the imputation approach, is to treat the predictions as gold-standard labels and compute $\tilde{\theta}^f = \hat{\theta}(\widetilde{X}, \tilde{f})$. However, $\tilde{\theta}^f$ will generally be biased due to prediction error. Instead, our key idea is to use the gold-standard data set to quantify how the prediction errors affect the imputation estimate, and then construct a confidence set for $\theta^\star$ by adjusting for this effect.

More systematically, the first step and our key conceptual innovation is to introduce an estimand-specific notion of prediction error called the *rectifier*, denoted $\boldsymbol{\Delta}^f$. The rectifier captures how errors in the predictions lead to bias in $\tilde{\theta}^f$. The appropriate rectifier depends on the estimand of interest, $\theta^\star$, and we derive it for a broad class of estimands in Section 4.2. Next, we use the gold-standard data to construct a confidence set for the rectifier, $\mathcal{R}$. Finally, we form a confidence set for $\theta^\star$ by taking $\tilde{\theta}^f$ and "rectifying" it with each possible value in the set $\mathcal{R}$. The collection of these rectified values is the prediction-powered confidence set, $\mathcal{C}^{\mathrm{PP}}$, which is guaranteed to contain $\theta^\star$ with high probability.

Prediction-Powered Inference



1. Define rectifier

Define the *rectifier*, $\mathbf{\Delta}^f$, a notion of prediction error.

2. Estimate rectifier

With labeled data, build $\mathcal{R}$, a confidence set for the rectifier.

3. Prediction-powered confidence set

Construct confidence set $\mathcal{C}^{\text{PP}}$ by rectifying $\tilde{\theta}^f$ with each value in $\mathcal{R}$.

Prediction-powered inference yields efficient and provably valid confidence intervals and p-values for a broad class of estimands, enabling researchers to reliably incorporate machine learning into their analyses. We provide practical algorithms for constructing prediction-powered confidence intervals for means, quantiles, modes, regression coefficients, and other potential estimands of scientific interest. For conciseness, our technical statements and algorithms will focus on constructing confidence intervals; however, note that through the duality between confidence intervals and hypothesis tests, these results directly imply valid prediction-powered p-values and hypothesis tests as well.

Warm-up: Mean estimation

Before presenting our theoretical results, we use the example of mean estimation to build intuition. The goal is to give a valid confidence interval for the mean value of the label, $\theta^\star = \mathbb{E}[Y_1]$. A classical estimate of $\theta^\star$ is the average of the labels in the labeled data set,

$$\hat{\theta}^{\text{class}} = \frac{1}{n}\sum_{i=1}^{n} Y_i.$$

We now construct a prediction-powered estimate, $\hat{\theta}^{\mathrm{PP}}$, and show how it leads to tighter confidence intervals than $\hat{\theta}^{\mathrm{class}}$ if the prediction rule is accurate. Consider

$$\hat{\theta}^{\mathrm{PP}} = \underbrace{\frac{1}{N}\sum_{i=1}^{N} \tilde{f}_i}_{\tilde{\theta}^f} - \underbrace{\frac{1}{n}\sum_{i=1}^{n}(f_i - Y_i)}_{\hat{\boldsymbol{\Delta}}^f}.$$

The key idea is that if the predictions are accurate, then $\hat{\boldsymbol{\Delta}}^f \approx 0$ and $\hat{\theta}^{\mathrm{PP}} \approx \frac{1}{N}\sum_{i=1}^{N} \widetilde{Y}_i$, which has a much lower variance than $\hat{\theta}^{\mathrm{class}}$ since $N \gg n$.

More precisely, observe that $\hat{\theta}^{\mathrm{PP}}$ is an unbiased estimator for $\theta^\star$, and that it is the sum of two independent terms. Thus, we can invoke the central limit theorem to construct 95% confidence intervals for $\theta^\star$ as

$$\underbrace{\hat{\theta}^{\mathrm{class}} \pm 1.96\sqrt{\frac{\hat{\sigma}_Y^2}{n}}}_{\text{classical interval}} \qquad \text{or} \qquad \underbrace{\hat{\theta}^{\mathrm{PP}} \pm 1.96\sqrt{\frac{\hat{\sigma}_{f-Y}^2}{n} + \frac{\hat{\sigma}_{\tilde{f}}^2}{N}}}_{\text{prediction-powered interval}},$$

where $\hat{\sigma}_Y^2$, $\hat{\sigma}_{f-Y}^2$, and $\hat{\sigma}_{\tilde{f}}^2$ are the estimated variances of Y_i, $f_i - Y_i$, and $\tilde{f}_i$, respectively. Because $N \gg n$, the width of the prediction-powered interval is primarily determined by the term $\hat{\sigma}_{f-Y}^2$; furthermore, when the predictions are accurate, we have $\hat{\sigma}_{f-Y}^2 \ll \hat{\sigma}_Y^2$, meaning the prediction-powered interval will be smaller than the classical interval. Although this particular estimator for the mean can be found in many forms in the literature—see Section 4.1—we leverage this variance reduction idea to make prediction-powered intervals smaller than their classical counterparts for a broad range of estimands beyond the mean.

Related work

Our work generalizes tools from the model-assisted survey sampling literature [4], which improve inference from survey data using auxiliary information. In particular, the mean estimator in Section 4.1 is also known as the *difference estimator*, a variation of generalized regression estimators [1]. The use of predictions as auxiliary data is not at all new [15], and much work has gone into constructing asymptotically valid confidence intervals when the predictive model is fitted on the same data used for inference [60].

Our work is also related to the statistical literature on missing data and multiple imputation [81]. In particular, regression with missing data has been studied by Robins et al. [5], Robins and Rotnitzky [7], Chen and Breslow [19], and Yu and Nan [29]. Our setting is also related to measurement error [25], particularly as studied by Chen et al. [21], who focus on the estimation of parameters defined as solutions to estimating equations, as we do herein.

Recently, a body of work on estimation using both labeled and unlabeled data has been developed [3, 18, 34, 125], focusing on efficiency in semiparametric or high-dimensional regimes. In particular, Chakrabortty and Cai [66] study efficient estimation of linear regression parameters, Chakrabortty et al. [129] focus on efficient quantile estimation, Zhang and

Bradic [141] investigate mean estimation in a high-dimensional setting, and Hou et al. [115] explore an imputation approach for improving generalized linear models. Our work instead focuses on the setting where we have access to an accurate predictive model fit to separate data. This allows us to address a broader range of estimands, such as minimizers of any convex objective, and provide finite-sample guarantees without assumptions about the machine-learning model.

More distantly, our setting involving labeled and unlabeled data also appears in the semi-supervised learning literature [24, 42], which aims to improve prediction accuracy with unlabeled data. We also highlight the literature on surrogates in causal inference [94]. Thematically, our work is most similar to Wang et al. [106], who also propose a method to correct machine-learning predictions for inference. However, our approach provides provably valid confidence intervals with minimal assumptions about the data distribution, while Wang et al. rely on parametric assumptions about true relationship between features and labels.

4.2 Main theory: estimands that minimize convex objectives

Our main contribution is a technique for inference on any estimand that can be expressed as a solution to a convex optimization problems. In additions to means, this includes a wide range of quantities such as medians, quantiles, linear and logistic regression coefficients, and more. Formally, we consider estimands of the form

$$\theta^\star = \underset{\theta \in \mathbb{R}^p}{\arg\min} \; \mathbb{E}\left[\ell_\theta(X_1, Y_1)\right],$$

for an objective function $\ell_\theta : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ that is convex in $\theta \in \mathbb{R}^p$, for some $p \in \mathbb{N}$. Throughout, we assume the existence of the minimizer, $\theta^\star$, and in cases where it is not unique, our method returns a confidence set that contains all minimizers. Under mild conditions, convexity ensures that $\theta^\star$ can also be be expressed as the value solving

$$\mathbb{E}\left[g_{\theta^\star}(X_1, Y_1)\right] = 0, \tag{4.1}$$

where $g_\theta : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^p$ is a subgradient of ℓ_θ with respect to θ. We refer to estimation problems where $\theta^\star$ satisfies Eq. (4.1) as nondegenerate convex estimation problems, which describes many problems in practice. For example, if the objective is differentiable for all $\theta \in \mathbb{R}^p$, then the problem is immediately nondegenerate. Furthermore, if the data distribution does not have point masses and, for every θ, $L_\theta(x, y)$ is nondifferentiable only for a measure-zero set of (x, y) pairs, then the problem is again nondegenerate.

Defining the rectifier. Following the outline in Section 4.1, the first step in prediction-powered inference is to define the rectifier, which captures a estimand-specific notion of prediction error. For nondegenerate convex estimation problems, the rectifier is the bias of the subgradient g_θ when evaluated using the predictions:

$$\mathbf{\Delta}^f(\theta) = \mathbb{E}\left[g_\theta(X_1, Y_1) - g_\theta(X_1, f_1)\right]. \tag{4.2}$$

Rectifier confidence set. The second step is to construct a confidence set for the rectifier, $\mathcal{R}_\delta(\theta)$, that satisfies

$$P\left(\mathbf{\Delta}^f(\theta) \in \mathcal{R}_\delta(\theta)\right) \geq 1 - \delta.$$

Because the rectifier is an expectation for each θ, $\mathcal{R}_\delta(\theta)$ can be constructed using off-the-shelf tools for building confidence intervals for means.

Prediction-powered confidence set. The final step is to form a confidence set for $\theta^\star$. We do so by combining $\mathcal{R}_\delta(\theta)$ with a term that accounts for finite-sample fluctuations due to having N samples. In particular, for every θ, we want a confidence set $\mathcal{T}_{\alpha-\delta}(\theta)$ for $\mathbb{E}[g_\theta(X_1, f_1)]$, that satisfies

$$P\left(\mathbb{E}[g_\theta(X_1, f_1)] \in \mathcal{T}_{\alpha-\delta}(\theta)\right) \geq 1 - (\alpha - \delta).$$

Again, since $\mathbb{E}[g_\theta(X_1, f_1)]$ is a mean, constructing $\mathcal{T}_{\alpha-\delta}(\theta)$ can be easily done with off-the-shelf tools.

We put all the steps together in Theorem 5.

Theorem 5 (Convex estimation)**.** *Suppose we have a nondegenerate convex estimation problem as in Eq.* (4.1)*. Fix $\alpha \in (0,1)$ and $\delta \in (0, \alpha)$. Suppose that, for any $\theta \in \mathbb{R}^p$, we can construct $\mathcal{R}_\delta(\theta)$ and $\mathcal{T}_{\alpha-\delta}(\theta)$ that satisfy*

$$P\left(\mathbf{\Delta}^f(\theta) \in \mathcal{R}_\delta(\theta)\right) \geq 1 - \delta; \quad P\left(\mathbb{E}[g_\theta(X_1, f_1)] \in \mathcal{T}_{\alpha-\delta}(\theta)\right) \geq 1 - (\alpha - \delta).$$

Let $\mathcal{C}_\alpha^{\mathrm{PP}} = \{\theta : 0 \in \mathcal{R}_\delta(\theta) + \mathcal{T}_{\alpha-\delta}(\theta)\}$, where $+$ denotes the Minkowski sum.[1] *Then,*

$$P(\theta^\star \in \mathcal{C}_\alpha^{\mathrm{PP}}) \geq 1 - \alpha.$$

This result means that we can construct a valid confidence set for $\theta^\star$ using the predictions, without any assumptions about the data distribution or the machine-learning model, for any nondegenerate convex estimation problem. We instantiate some common examples in Section 4.3.

The general principle of prediction-powered inference described in Section 4.1 can also be applied more broadly, such as to minimizers of any nonconvex objective function, as described next.

[1] The Minkowski sum of two sets A and B is equal to $\{a + b : a \in A, b \in B\}$.

General risk minimizers

The tools developed in Section 4.2 were tailored to unconstrained convex optimization problems. More generally, estimands can be expressed as minimizers of nonconvex objective functions, possibly also with convex or nonconvex constraints, in which case we cannot expect the condition Eq. (4.1) to hold. In this section, we generalize prediction-powered inference to a broad class of risk minimizers:

$$\theta^\star = \arg\min_{\theta\in\Theta} \mathbb{E}[\ell_\theta(X_1, Y_1)], \tag{4.3}$$

where $\ell_\theta : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ is a possibly nonconvex objective function and Θ is an arbitrary parameter set. As before, if $\theta^\star$ is not a unique minimizer, our method will return a confidence set that contains all minimizers.

In this section, we present a prediction-powered inference solution for any estimand that can be expressed as in Eq. (4.3). Note, however, that this solution does not reduce to the one in Section 4.2 when the objective ℓ_θ is convex and subdifferentiable and $\Theta = \mathbb{R}^p$ for some p, in which case $\theta^\star$ can be equivalently characterized by Eq. (4.1). We expect the method from Section 4.2 to be more powerful for nondegenerate convex estimation problems, particularly when p is small.

We use the following rectifier:

$$\boldsymbol{\Delta}^f(\theta) = \mathbb{E}\left[\ell_\theta(X_1, Y_1) - \ell_\theta(X_1, f_1)\right]. \tag{4.4}$$

Notice that the rectifier Eq. (4.4) is one-dimensional, while the rectifier Eq. (4.2) for nondegenerate convex estimation problems is p-dimensional.

A key distinction from the approach in Section 4.2 is the inclusion of an additional step involving data splitting. This additional step is necessary because, unlike in nondegenerate convex estimation where we know that $\mathbb{E}[g_{\theta^\star}(X_1, Y_1)] = 0$, in general we do not know the value of $\mathbb{E}[\ell_{\theta^\star}(X_1, Y_1)]$. To address this challenge, we estimate $\mathbb{E}[\ell_{\theta^\star}(X_1, Y_1)]$ by approximating $\theta^\star$ with an prediction-based estimate using the first $N/2$ unlabeled data points, assuming N is even for simplicity. To state the main result, we define

$$\tilde{\theta}^f = \arg\min_{\theta\in\Theta} \frac{2}{N}\sum_{i=1}^{N/2} \ell_\theta(\tilde{X}_i, \tilde{f}_i), \quad \tilde{L}^f(\theta) := \frac{2}{N}\sum_{i=N/2+1}^{N} \ell_\theta(\tilde{X}_i, \tilde{f}_i).$$

Theorem 6 (General risk minimization)**.** *Fix $\alpha \in (0,1)$ and $\delta \in (0,\alpha)$. Suppose that, for any $\theta \in \Theta$, we can construct $\left(\mathcal{R}^l_{\delta/2}(\theta), \mathcal{R}^u_{\delta/2}(\theta)\right)$ and $\left(\mathcal{T}^l_{\frac{\alpha-\delta}{2}}(\theta), \mathcal{T}^u_{\frac{\alpha-\delta}{2}}(\theta)\right)$ such that*

$$\begin{aligned}
P\left(\boldsymbol{\Delta}^f(\theta) \le \mathcal{R}^u_{\delta/2}(\theta)\right) &\ge 1 - \delta/2,\\
P\left(\boldsymbol{\Delta}^f(\theta) \ge \mathcal{R}^l_{\delta/2}(\theta)\right) &\ge 1 - \delta/2,\\
P\left(\tilde{L}^f(\theta) - \mathbb{E}[\ell_\theta(X_1, f_1)] \le \mathcal{T}^u_{\frac{\alpha-\delta}{2}}(\theta)\right) &\ge 1 - \frac{\alpha-\delta}{2},\\
P\left(\tilde{L}^f(\theta) - \mathbb{E}[\ell_\theta(X_1, f_1)] \ge \mathcal{T}^l_{\frac{\alpha-\delta}{2}}(\theta)\right) &\ge 1 - \frac{\alpha-\delta}{2}.
\end{aligned}$$

Let

$$\mathcal{C}_\alpha^{\mathrm{PP}} = \left\{\theta \in \Theta : \tilde{L}^f(\theta) \leq \tilde{L}^f(\tilde{\theta}^f) - \mathcal{R}^l_{\delta/2}(\theta) + \mathcal{R}^u_{\delta/2}(\tilde{\theta}^f) + \mathcal{T}^u_{\frac{\alpha-\delta}{2}}(\theta) - \mathcal{T}^l_{\frac{\alpha-\delta}{2}}(\tilde{\theta}^f)\right\}.$$

Then, we have

$$P\left(\theta^\star \in \mathcal{C}_\alpha^{\mathrm{PP}}\right) \geq 1 - \alpha.$$

For example, if the loss $\ell_\theta(x, y)$ takes values in $[0, B]$ for all x, y, then we can set $\mathcal{T}_{\alpha-\delta}(\theta) = B\sqrt{\frac{\log(1/(\alpha-\delta))}{N}}$, which is valid by Hoeffding's inequality.

Mode estimation. A common estimand that cannot be described by nondegenerate convex estimation is the mode of the label distribution. We can handle the setting where the label takes values in a discrete set, Θ, by using the objective function $\ell_\theta(y) = \mathbb{1}\left\{y \neq \theta\right\}, \theta \in \Theta$. A generalization of this approach to continuous label distributions is obtained by defining the objective $\ell_\theta(y) = \mathbb{1}\left\{|y - \theta| > \eta\right\}$, for some width parameter $\eta > 0$. The estimand is thus the point $\theta \in \mathbb{R}$ that has the greatest probability mass in its η-neighborhood, $\theta^\star = \arg\min_{\theta\in\mathbb{R}} P\left(|Y_1 - \theta| > \eta\right)$. Theorem 6 applies directly in both the discrete and continuous cases.

Prediction-powered p-values

The duality between confidence intervals and p-values enables us to repurpose the presented theory for valid prediction-powered p-values.

To formalize this, suppose we want to test the hull hypothesis $H_0 : \theta^\star \in \Theta_0$ for some set $\Theta_0 \in \mathbb{R}^p$. For example, a common choice when $p = 1$ is $\Theta_0 = \mathbb{R}_{\leq 0}$. Let $\mathcal{C}_\alpha$ be a valid confidence interval. Then, we can construct a valid p-value as

$$P = \inf\{\alpha : \theta_0 \notin \mathcal{C}_\alpha, \forall \theta_0 \in \Theta_0\}. \tag{4.5}$$

A p-value P is valid if it is super-uniform under the null, meaning $P(P \leq u) \leq u$ for all $u \in [0, 1]$. This is indeed the case for the p-value defined in Eq. (4.5), because when $\theta^\star \in \Theta_0$, we have

$$P(P \leq u) \leq P(\theta^\star \notin \mathcal{C}_u) \leq u.$$

The first inequality follows from the definition of P and the fact that $\theta^\star \in \Theta_0$, and the second inequality follows by the validity of $\mathcal{C}_u$ at level $1 - u$. We implicitly use the fact that $\mathcal{C}_u \subseteq \mathcal{C}_{u'}$ when $u \geq u'$.

The above derivation is a general recipe for deriving p-values from confidence intervals. For the prediction-powered confidence interval stated in Theorem 5, the corresponding prediction-powered p-value is given by:

$$P^{\mathrm{PP}} = \inf\left\{\alpha : |m_{\theta_0} + \boldsymbol{\Delta}^f_{\theta_0}| > w_{\theta_0}(\alpha),\ \forall \theta_0 \in \Theta_0\right\}.$$

4.3 Algorithms

In this section, we present nonasymptotically-valid prediction-powered algorithms for several canonical estimands: mean estimation, quantile estimation, and logistic regression, which all correspond to nondegenerate convex estimation problems. The algorithms follow the abstract recipe in Theorem 5, and proofs of their validity are in Section 4.5.

The algorithms rely on any off-the-shelf method for computing nonasymptotically-valid confidence intervals for the mean. For concreteness, we instantiate these algorithms with a variance-adaptive confidence interval for the mean by Waudby-Smith and Ramdas [148] (Algorithm 10), due to its strong performance in practice. The only assumption required to apply Algorithm 10 is that the data are almost surely bounded within a known interval.

Mean estimation. We begin by returning to the problem of mean estimation:

$$\theta^\star = \mathbb{E}[Y_1]. \tag{4.6}$$

The mean can alternatively be expressed as the minimizer of the expected squared difference from the label, a convex objective:

$$\theta^\star = \arg\min_{\theta \in \mathbb{R}} \mathbb{E}[\ell_\theta(Y_1)] = \arg\min_{\theta \in \mathbb{R}} \mathbb{E}\left[\frac{1}{2}(Y_1 - \theta)^2\right].$$

The squared difference $\ell_\theta(y)$ is differentiable with the gradient $g_\theta(y) = \theta - y$. Applying this to the definition of the rectifier Eq. (4.2), we get

$$\boldsymbol{\Delta}^f(\theta) \equiv \boldsymbol{\Delta}^f = \mathbb{E}[f_1 - Y_1].$$

Note that this rectifier has no dependence on θ. We provide an explicit algorithm for prediction-powered mean estimation and its guarantee in Algorithm 7 and Corollary 2, respectively.

Algorithm 7 Prediction-powered mean estimation

```
Input: labeled data $(X, Y)$, unlabeled features $\widetilde{X}$, prediction rule $f$, error levels $\alpha, \delta \in (0,1)$, bound $B$
$(f^l_{\alpha-\delta}, f^u_{\alpha-\delta}) \leftarrow \texttt{MeanCI}\Big(\{\tilde{f}_i\}_{i=1}^N, \mathrm{err} = \alpha - \delta, \mathrm{range} = [0, B]\Big)$
$(\mathcal{R}^l_\delta, \mathcal{R}^u_\delta) \leftarrow \texttt{MeanCI}\Big(\{f_i - Y_i\}_{i=1}^n, \mathrm{err} = \delta, \mathrm{range} = [-B, B]\Big)$
Output: prediction-powered confidence set $\mathcal{C}^{\mathrm{PP}}_\alpha = \big(f^l_{\alpha-\delta} - \mathcal{R}^u_\delta, f^u_{\alpha-\delta} - \mathcal{R}^l_\delta\big)$
```

Corollary 2 (Mean estimation)**.** *Let $\theta^\star$ be the mean outcome Eq. (4.6). Suppose that $Y_1, f_1 \in [0, B]$ almost surely. Then, the prediction-powered confidence set in Algorithm 7 has valid coverage: $P(\theta^\star \in \mathcal{C}^{\mathrm{PP}}_\alpha) \geq 1 - \alpha$.*

Quantile estimation. We now turn to quantile estimation. For a pre-specified $q \in (0,1)$, we wish to estimate the q-quantile of the label distribution:

$$\theta^\star = \min\left\{\theta : P\left(Y_1 \le \theta\right) \ge q\right\}. \tag{4.7}$$

To simplify the exposition, we assume that the distribution of Y_1 does not have point masses, which ensures that the problem is nondegenerate Eq. (4.1). It is well known [2] that the q-quantile can be expressed in variational form as

$$\begin{aligned}\theta^\star &= \underset{\theta\in\mathbb{R}}{\arg\min}\ \mathbb{E}\left[\ell_\theta(Y_1)\right]\\ &= \underset{\theta\in\mathbb{R}}{\arg\min}\ \mathbb{E}\left[q(Y_1-\theta)\mathbb{1}\left\{Y_1 > \theta\right\} + (1-q)(\theta - Y_1)\mathbb{1}\left\{Y_1 \le \theta\right\}\right],\end{aligned}$$

where ℓ_θ is called the quantile or "pinball" loss. The quantile loss has the subgradient $g_\theta(y) = -q\mathbb{1}\left\{y > \theta\right\} + (1-q)\mathbb{1}\left\{y \le \theta\right\} = -q + \mathbb{1}\left\{y \le \theta\right\}$. Plugging the expression for $g_\theta(y)$ into the definition Eq. (4.2), we get the relevant rectifier:

$$\boldsymbol{\Delta}^f(\theta) = P(Y_1 \le \theta) - P(f_1 \le \theta) = \mathbb{E}\left[\mathbb{1}\left\{Y_1 \le \theta\right\} - \mathbb{1}\left\{f_1 \le \theta\right\}\right].$$

We give an algorithm for prediction-powered quantile estimation in Algorithm 8; see Corollary 3 for the corresponding guarantee of validity.

Algorithm 8 Prediction-powered quantile estimation

```
Input: labeled data $(X, Y)$, unlabeled features $\widetilde{X}$, predictor $f$, quantile $q \in (0,1)$, error levels $\alpha, \delta \in (0,1)$
Construct fine grid $\Theta_{\text{grid}}$ between $\min_{i\in[N]} \tilde{f}_i$ and $\max_{i\in[N]} \tilde{f}_i$
for $\theta \in \Theta_{\text{grid}}$ do
    $(\mathcal{R}^l_\delta(\theta), \mathcal{R}^u_\delta(\theta)) \leftarrow \texttt{MeanCI}\Big(\left\{\mathbb{1}\left\{Y_i \le \theta\right\} - \mathbb{1}\left\{f_i \le \theta\right\}\right\}_{i=1}^n, \text{err} = \delta, \text{range} = [-1, 1]\Big)$
    $(\hat{F}^l_{\alpha-\delta}(\theta), \hat{F}^u_{\alpha-\delta}(\theta)) \leftarrow \texttt{MeanCI}\Big(\left\{\mathbb{1}\left\{\tilde{f}_i \le \theta\right\}\right\}_{i=1}^N, \text{err} = \alpha - \delta, \text{range} = [0, 1]\Big)$
end for
Output: prediction-powered confidence set
```

$$\mathcal{C}^{\text{PP}}_\alpha = \left\{\theta \in \Theta_{\text{grid}} : q \in \left(\hat{F}^l_{\alpha-\delta}(\theta) + \mathcal{R}^l_\delta(\theta), \hat{F}^u_{\alpha-\delta}(\theta) + \mathcal{R}^u_\delta(\theta)\right)\right\}$$

Corollary 3 (Quantile estimation). *Let $\theta^\star$ be the q-quantile Eq. (4.7). Then, the prediction-powered confidence set in Algorithm 8 has valid coverage: $P(\theta^\star \in \mathcal{C}^{\text{PP}}_\alpha) \ge 1 - \alpha$.*

Logistic regression. In logistic regression, the estimand is the parameter vector $\theta^\star$ defined by

$$\begin{aligned}\theta^\star &= \underset{\theta\in\mathbb{R}^d}{\arg\min}\,\mathbb{E}[\ell_\theta(X_1, Y_1)] \qquad (4.8)\\ &= \underset{\theta\in\mathbb{R}^d}{\arg\min}\,\mathbb{E}\left[-Y_1\theta^\top X_1 + \log(1 + \exp(\theta^\top X_1))\right],\end{aligned}$$

where $Y_1 \in \{0,1\}$. The logistic loss is convex and differentiable, which ensures the optimality condition Eq. (4.1). Its gradient is equal to

$$g_\theta(x,y) = -xy + x\mu_\theta(x), \tag{4.9}$$

where $\mu_\theta(x) = 1/(1+\exp(-x^\top\theta))$ is the predicted mean for $x \in \mathcal{X}$ based on parameters θ. Plugging this into the definition of the rectifier for nondegenerate convex estimation problems, Eq. (4.2), we derive a rectifier that is constant for all θ and equal to

$$\boldsymbol{\Delta}^f(\theta) \equiv \boldsymbol{\Delta}^f = \mathbb{E}\left[X_1(f_1 - Y_1)\right].$$

Algorithm 9 gives the corresponding method for prediction-powered logistic regression, and Corollary 4 states its validity guarantee. Note that we use $X_{i,j}$ to denote the j-th coordinate of X_i.

Other generalized linear models also have the optimality condition Eq. (4.1) with gradients of the same form as in Eq. (4.9), but with different predicted means $\mu_\theta(x)$; see Chapter 3 of Efron [130]). For example, Poisson regression uses $\mu_\theta(x) = \exp(x^\top\theta)$. Consequently, by changing the instantiation of $\mu_\theta(x)$ in Line 4 of Algorithm 9, we can handle other generalized linear models in essentially the same way.

Algorithm 9 Prediction-powered logistic regression

```
Input: labeled data $(X,Y)$, unlabeled features $\widetilde{X}$, predictor $f$, error levels $\alpha,\delta \in (0,1)$,
    bounds $\mathbf{B} = (B_j)_{j=1}^d$
1: Construct fine grid $\Theta_{\mathrm{grid}} \subset \mathbb{R}^d$ of possible coefficients
2: $(\mathcal{R}^l_{\delta,j}, \mathcal{R}^u_{\delta,j}) \leftarrow \texttt{MeanCI}\left(\{X_{i,j}(f_i - Y_i)\}_{i=1}^n, \mathrm{err} = \delta, \mathrm{range} = [-B_j, B_j]\right), \quad j \in [d]$
3: for $\theta \in \Theta_{\mathrm{grid}}$ do
4:     Define $\mu_\theta(x) = \frac{1}{1+\exp(-x^\top\theta)}$
5:     for $g \in [d]$ do
6:         $I \leftarrow \texttt{MeanCI}\left(\left\{\widetilde{X}_{i,j}\left(\mu_\theta(\widetilde{X}_i) - \tilde{f}_i\right)\right\}_{i=1}^N, \mathrm{err} = \frac{\alpha-\delta}{d}, \mathrm{range} = [-B_j, B_j]\right)$
7:         $(g^l_{\alpha-\delta,j}(\theta), g^u_{\alpha-\delta,j}(\theta)) \leftarrow (\min I, \max I)$
8:     end for
9: end for
Output: prediction-powered confidence set,
```

$$\mathcal{C}^{\mathrm{PP}}_\alpha = \left\{\theta \in \Theta_{\mathrm{grid}} : 0 \in \left[g^l_{\alpha-\delta,j}(\theta) + \mathcal{R}^l_{\delta,j}, g^u_{\alpha-\delta,j}(\theta) + \mathcal{R}^u_{\delta,j}\right], \forall j \in [d]\right\}$$

Corollary 4 (Logistic regression). *Let $\theta^\star$ be the logistic regression coefficients defined in Eq. (4.8). Suppose that $|X_{1,j}| \le B_j$ and $Y_1, f_1 \in [0,1]$ almost surely. Then, the prediction-powered confidence set in Algorithm 9 has valid coverage: $P(\theta^\star \in \mathcal{C}^{\mathrm{PP}}_\alpha) \ge 1 - \alpha$.*

Algorithm 10 `MeanCI` (from Theorem 3 by Waudby-Smith and Ramdas [148])

```
Input: data points $\{Z_1, \dots, Z_n\}$, error level $\alpha \in (0,1)$, range $[L, U]$ s.t. $Z_i \in [L, U]$
For all $i \in [n]$, let $Z_i \leftarrow (Z_i - L)/(U - L)$        ▷ normalize data to interval $[0,1]$
Construct fine grid $M_{\text{grid}}$ of interval $[0,1]$
Initialize active set $\mathcal{A} = M_{\text{grid}}$
for $t \in 1, \dots, n$ do
    Set $\hat{\mu}_t \leftarrow \frac{0.5+\sum_{j=1}^{t} Z_j}{t+1}$, $\hat{\sigma}_t^2 \leftarrow \frac{0.25+\sum_{j=1}^{t}(Z_j-\hat{\mu}_t)^2}{t+1}$, $\lambda_t \leftarrow \sqrt{\frac{2\log(2/\alpha)}{n\hat{\sigma}_{t-1}^2}}$
    for $m \in \mathcal{A}$ do
        $M_t^+(m) \leftarrow \left(1 + \min\left(\lambda_t, \frac{0.5}{m}\right)(Z_t - m)\right) M_{t-1}^+(m)$
        $M_t^-(m) \leftarrow \left(1 - \min\left(\lambda_t, \frac{0.5}{1-m}\right)(Z_t - m)\right) M_{t-1}^-(m)$
        $M_t(m) \leftarrow \frac{1}{2}\max\left\{M_t^+(m),\ M_t^-(m)\right\}$        ▷ construct test martingale for $m \in [0,1]$
        if $M_t(m) \geq 1/\alpha$ then
            $\mathcal{A} \leftarrow \mathcal{A} \setminus \{m\}$        ▷ Remove $m$ from active set
        end if
    end for
end for
Output: Confidence set for the mean $\mathcal{C}_\alpha = \{m(U-L) + L : m \in \mathcal{A}\}$
```

4.4 Applications in proteomics and genomics

In the following two applications, we compute the prediction-powered confidence interval for an estimand of interest and compare it to two alternatives: the classical interval, which uses only the gold-standard data, and the imputation interval, which uses only the predictions on the unlabeled data, by treating it as gold-standard data. We show that the imputation interval, which does not account for prediction error, does not contain the true value of the estimand. For the two intervals that are guaranteed to be valid—prediction-powered and classical—we compare their widths as a function of n, the amount of labeled data used.

Code is available at this link.

Relating protein structure and post-translational modifications

We demonstrate how prediction-powered confidence intervals for the mean can be used to construct confidence intervals for more elaborate estimands, such as the odds ratio, which is commonly used to quantify associations between binary random variables.

The goal in this application is to characterize the structural context of post-translational modifications (PTMs), which are biochemical modifications of specific positions of a protein sequence that play regulatory roles. One question of interest is whether PTMs occur more frequently in intrinsically disordered regions (IDRs), segments of a protein that do not abide in a fixed three-dimensional structure. Recently, Bludau et al. [127] studied this relationship on an unprecedented proteome-wide scale by using AlphaFold-predicted structures [116] to predict IDRs, in contrast to previous work which considered far fewer experimentally derived structures.

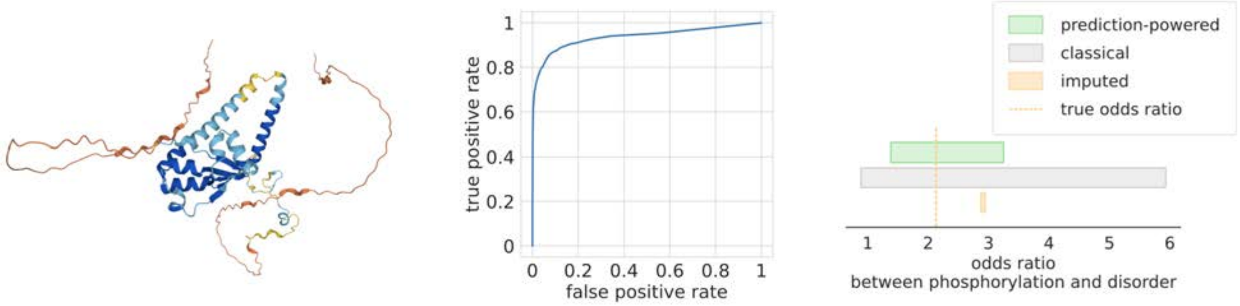


Figure 4.1: **AlphaFold-based prediction of intrinsic disorder.** Left: predicted disorder for one example protein (UniProt S5FZ81), colored by predicted probability of disorder per position. Middle: ROC curve of disorder prediction based on AlphaFold structure. Right: confidence interval for the odds ratio between disorder and phosphorylation (a type of post-translational modification) produced by prediction-powered inference and the classical and imputation baselines, when $n = 571$. Unlike the classical interval, the prediction-powered interval excludes the value of one and thus the direction of the association is unambiguous.

Let $Y_i \in \{1, 0\}$ denote the gold-standard label of whether or not a position is in an IDR, and let $Z_i \in \{1, 0\}$ denote whether or not a position has a PTM. Following Bludau et al. [127], we obtain a prediction for Y_i, denoted $f_i \in \{1, 0\}$, from the AlphaFold-predicted structure (see Section 4.6 for details). To quantify the association between PTMs and IDRs, the authors computed the odds ratio between f_i and Z_i on a data set of hundreds of thousands of protein sequence positions. Though some of the data points also contained a gold-standard label, Y_i, Bludau et al. [127] did not use these labels in their analyses to avoid dealing with conflicts between labels and predictions. Here, we show how to use both the gold-standard labels and the predictions to give confidence intervals for the odds ratio that are valid, in contrast to the imputation interval, and smaller than the classical interval.

The odds ratio between Y_i and Z_i can be written as a function of two means:

$$\theta^\star = \frac{\mu_1/(1-\mu_1)}{\mu_0/(1-\mu_0)}, \tag{4.10}$$

where $\mu_1 = P(Y = 1 \mid Z = 1)$ and $\mu_0 = P(Y = 1 \mid Z = 0)$. We therefore proceed by constructing $1 - \alpha/2$ prediction-powered confidence intervals for μ_0 and μ_1, denoted $\mathcal{C}_0^{\mathrm{PP}} = [l_0, u_0]$ and $\mathcal{C}_1^{\mathrm{PP}} = [l_1, u_1]$, respectively. We then propagate $\mathcal{C}_0^{\mathrm{PP}}$ and $\mathcal{C}_1^{\mathrm{PP}}$ through the odds-ratio formula Eq. (4.10) to get the following confidence interval:

$$\mathcal{C}^{\mathrm{PP}} = \left\{ \frac{c_1}{1-c_1} \cdot \frac{1-c_0}{c_0} : c_0 \in \mathcal{C}_0^{\mathrm{PP}}, c_1 \in \mathcal{C}_1^{\mathrm{PP}} \right\} = \left(\frac{l_1}{1-l_1} \cdot \frac{1-u_0}{u_0}, \frac{u_1}{1-u_1} \cdot \frac{1-l_0}{l_0} \right).$$

By a union bound, $\mathcal{C}^{\mathrm{PP}}$ contains $\theta^\star$ with probability at least $1 - \alpha$. We set $\alpha = 0.1$.

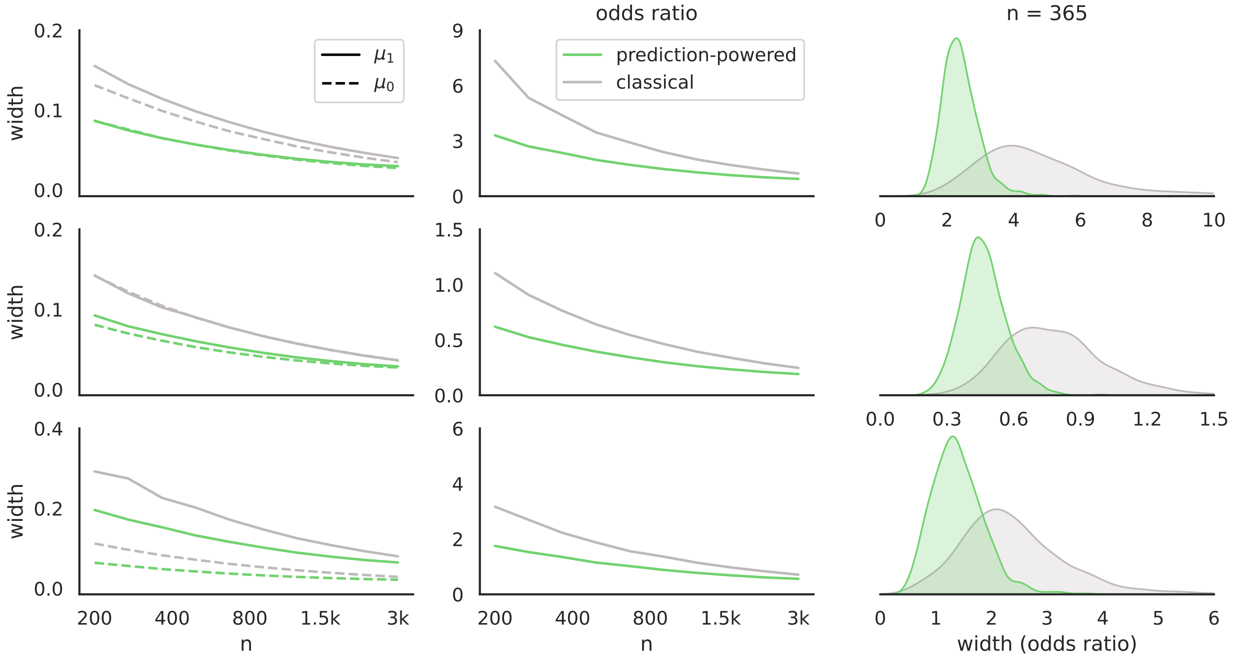


Figure 4.2: **Classical and prediction-powered confidence intervals on the odds ratio for three different post-translational modifications, phosphorylation (top row), ubiquitination (middle row), and acetylation (bottom row).** Left: widths of prediction-powered and classical confidence intervals for μ_1 (solid line) and μ_0 (dashed line). Middle: widths of prediction-powered and classical confidence intervals for the odds ratio. Right: distribution of interval widths for the odds ratio when $n = 365$.

We have $10,803$ data points from [127], from which we simulated labeled and unlabeled data sets as follows. For each of 1000 trials, we randomly sampled n points to serve as the labeled data set and used the remaining $N = 10803 - n$ points as the unlabeled data set, where we do not observe the labels. For all values of n and all three different types of PTMs that we examined, the prediction-powered confidence intervals are smaller than classical intervals (Fig. 4.2). On the other hand, the imputation confidence interval does not contain the true odds ratio (Fig. 4.1).

Distribution of gene expression levels

In this section, we demonstrate the construction of prediction-powered confidence intervals on quantiles for studying the effects of regulatory DNA on gene expression.

In particular, we aim to characterize the distribution of gene expression levels induced by a population of promoters—regulatory DNA sequences that control how frequently a gene is transcribed. Recently, Vaishnav et al. [139] trained a state-of-the-art transformer model on tens of millions of random promoter sequences, to predict the expression level of a particular gene induced by a promoter sequence (Fig. 4.3). They then used the model's predictions to study the effects of promoters—for example, by assessing how quantiles of predicted expression levels differ between different populations of promoters, and verifying those observations by experimentally measuring the expression levels of the promoters of interest.

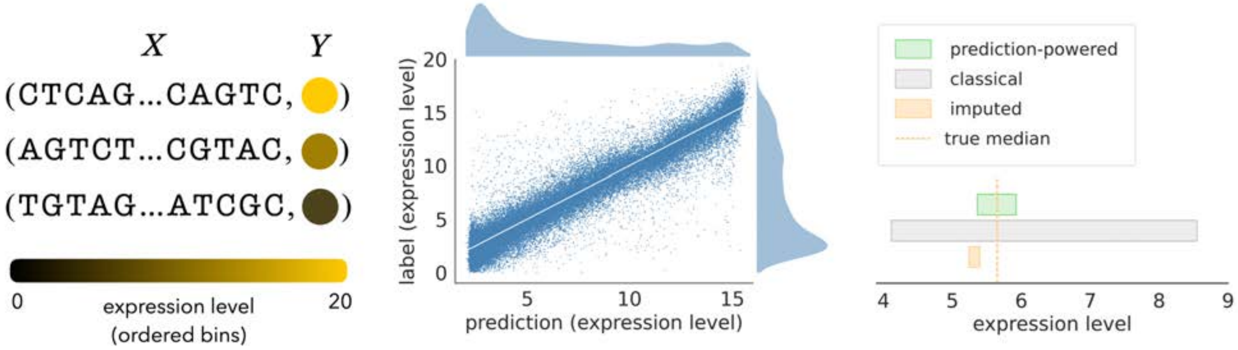


Figure 4.3: **Predicting gene expression level from promoter sequence.** Data and transformer predictive model are from Vaishnav et al. [139]. Left: each data point consists of a promoter sequence, X_i, and an expression level, Y_i. Middle: predictive performance of the transformer model on the native yeast promoters used in our experiments (RMSE 2.18, Pearson 0.963, Spearman 0.946). Right: confidence intervals for the median native yeast promoter expression level with $n = 75$ and $\alpha = 0.1$.

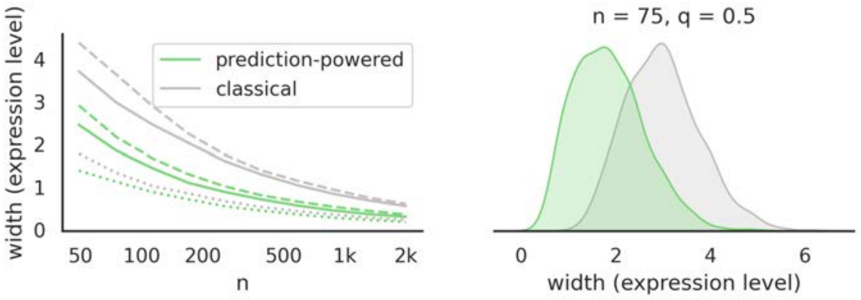


Figure 4.4: **Widths of confidence intervals for quantiles of gene expression level.** Data and transformer predictive model are from Vaishnav et al. [139]. Left: average width of prediction-powered and classical confidence intervals for the 0.25-quantile (dashed lines), 0.5-quantile (solid lines), and 0.75-quantile (dotted lines). Right: distribution of confidence interval widths for the median using $n = 75$.

Let X_i be an 80-base-pair promoter sequence for a particular gene, and let $Y_i \in [0, 20]$ denote a measurement of the expression level it causes for the gene. Furthermore, let $f_i \in [0, 20]$ denote the corresponding expression level predicted by the transformer model in [139]. We focus on estimating the 0.25-, 0.5-, and 0.75-quantiles of expression levels induced by native yeast promoters: the population of promoter sequences that are naturally found in the genomes of *S. cerevisiae*.

We have 61, 150 labeled native yeast promoter sequences from [139], from which we simulated labeled and unlabeled data sets as follows. For each of 1000 trials, we randomly sampled n points to serve as the labeled data set and used the remaining $N = 61150-n$ points as the unlabeled data set. We then constructed prediction-powered confidence intervals for the quantiles with $\alpha = 0.1$. The prediction-powered intervals for all three quantiles are smaller than the classical intervals for all values of n (Fig. 4.4). On the other hand, the imputation confidence intervals do not contain the true quantiles (Fig. 4.3 for the median).

4.5 Proofs

Proof of Theorem 5

We show that $\theta^\star \in \mathcal{C}_\alpha^{\mathrm{PP}}$ with probability at least $1-\alpha$; that is, with probability at least $1-\alpha$ it holds that

$$0 \in \mathcal{R}_\delta(\theta^\star) + \mathcal{T}_{\alpha-\delta}(\theta^\star).$$

Consider the event $E = \{\boldsymbol{\Delta}^f(\theta^\star) \in \mathcal{R}_\delta(\theta^\star)\} \cap \{\mathbb{E}[g_{\theta^\star}(X_1, f_1)] \in \mathcal{T}_{\alpha-\delta}(\theta^\star)\}$. By a union bound, $P(E) \geq 1-\alpha$. On the event E, we have that

$$\begin{aligned}\mathbb{E}[g_{\theta^\star}(X_1, Y_1)] &= \mathbb{E}[g_{\theta^\star}(X_1, Y_1)] - \mathbb{E}[g_{\theta^\star}(X_1, f_1)] + \mathbb{E}[g_{\theta^\star}(X_1, f_1)] \\ &= \boldsymbol{\Delta}^f(\theta^\star) + \mathbb{E}[g_{\theta^\star}(X_1, f_1)] \in \mathcal{R}_\delta(\theta^\star) + \mathcal{T}_{\alpha-\delta}(\theta^\star).\end{aligned}$$

The theorem follows by invoking the nondegeneracy condition, which ensures $\mathbb{E}[g_{\theta^\star}(X_1, Y_1)] = 0$, so we have shown that $0 \in \mathcal{R}_\delta(\theta^\star) + \mathcal{T}_{\alpha-\delta}(\theta^\star)$.

Proof of Corollary 2

The proof follows by instantiating the terms in Theorem 5. In particular, we have

$$\mathbb{E}[g_\theta(f_1)] = \theta - \mathbb{E}[f_1],$$

hence it is valid to construct $\mathcal{T}_{\alpha-\delta}(\theta)$ as:

$$\mathbb{E}[g_\theta(f_1)] \in \mathcal{T}_{\alpha-\delta}(\theta) = \theta - (f^l_{\alpha-\delta}, f^u_{\alpha-\delta}).$$

Therefore, the condition $0 \in \mathcal{R}_\delta + \mathcal{T}_{\alpha-\delta}(\theta)$ becomes

$$0 \in (\mathcal{R}^l_\delta, \mathcal{R}^u_\delta) + \theta - (f^l_{\alpha-\delta}, f^u_{\alpha-\delta}),$$

which, after rearranging and simplifying, is equivalent to

$$\theta \in \left(f^l_{\alpha-\delta} - \mathcal{R}^u_\delta, f^u_{\alpha-\delta} - \mathcal{R}^l_\delta\right).$$

This set exactly matches the set $\mathcal{C}_\alpha^{\mathrm{PP}}$ constructed in Algorithm 7.

Proof of Corollary 3

The proof follows by instantiating the terms in Theorem 5. First, we have

$$\mathbb{E}[g_\theta(f_1)] = -q + P(f_1 \leq \theta);$$

therefore, it is valid to construct $\mathcal{T}_{\alpha-\delta}(\theta)$ as:

$$\mathbb{E}[g_\theta(f_1)] \in \mathcal{T}_{\alpha-\delta}(\theta) = -q + \left(\hat{F}^l_{\alpha-\delta}(\theta), \hat{F}^u_{\alpha-\delta}(\theta)\right).$$

Therefore, the condition $0 \in \mathcal{R}_\delta(\theta) + \mathcal{T}_{\alpha-\delta}(\theta)$ becomes

$$q \in \left(\hat{F}^l_{\alpha-\delta}(\theta) + \mathcal{R}^l_\delta(\theta), \hat{F}^u_{\alpha-\delta}(\theta) + \mathcal{R}^u_\delta(\theta)\right),$$

which matches the condition used to form $\mathcal{C}^{\mathrm{PP}}_\alpha$ in Algorithm 8.

Proof of Corollary 4

We instantiate the relevant terms in Theorem 5. We have

$$\mathbb{E}[g_\theta(X_1, f_1)] = \mathbb{E}\left[-X_1 f_1 + X_1 \frac{1}{1+\exp(-X_1^\top \theta)}\right].$$

Note that, because X_1 is bounded coordinate-wise, and $Y_1, 1/(1+\exp(-X_1^\top\theta)) \in [0,1]$, we have $|(g_\theta(X_1, f_1))_j| \leq B_j$ almost surely. Therefore, we can construct $\mathcal{T}_{\alpha-\delta}(\theta)$ as:

$$\begin{aligned}\mathbb{E}[g_\theta(X_1, f_1)] \in \mathcal{T}_{\alpha-\delta}(\theta) &= \left(g^l_{\alpha-\delta}(\theta), g^u_{\alpha-\delta}(\theta)\right) \\ &= \left(g^l_{\alpha-\delta,1}(\theta), g^u_{\alpha-\delta,1}(\theta)\right) \times \cdots \times \left(g^l_{\alpha-\delta,d}(\theta), g^u_{\alpha-\delta,d}(\theta)\right).\end{aligned}$$

Since the rectifier has no dependence on θ, the condition $0 \in \mathcal{R}_\delta(\theta) + \mathcal{T}_{\alpha-\delta}(\theta)$ becomes

$$0 \in (\mathcal{R}^l_{\delta,j}, \mathcal{R}^u_{\delta,j}) + \left(g^l_{\alpha-\delta,j}(\theta), g^u_{\alpha-\delta,j}(\theta)\right), \quad \forall j \in [d],$$

which matches the condition defining $\mathcal{C}^{\mathrm{PP}}_\alpha$ in Algorithm 9.

4.6 Experimental details

Model for predicting intrinsically disordered regions. The predictive model f is a logistic regression model that maps relative solvent-accessible surface area (RSA) of each position, computed based on the AlphaFold-predicted structure, to a probability that the position is in an IDR. Following Bludau et al. [127], the RSA was locally smoothed with a window of $5, 10, 15, 20, 25, 30$, or 35 amino acids, and a sigmoid function was used to predict disorder from this smoothed RSA quantity. To fit the sigmoid, we used the data in [127] that had disorder labels but no PTM labels. The window size was chosen as the value that resulted in the lowest variance of the bias, $Y - f$, on this data.

Bibliography

1. Cassel, C. M., Särndal, C. E. & Wretman, J. H. Some results on generalized difference estimation and generalized regression estimation for finite populations. *Biometrika* **63,** 615–620 (1976).
2. Koenker, R. & Bassett Jr, G. Regression quantiles. *Econometrica: Journal of the Econometric Society,* 33–50 (1978).
3. Pepe, M. S. Inference using surrogate outcome data and a validation sample. *Biometrika* **79,** 355–365 (1992).
4. Särndal, C.-E., Swensson, B. & Wretman, J. *Model Assisted Survey Sampling* (Springer Science & Business Media, 1992).
5. Robins, J. M., Rotnitzky, A. & Zhao, L. P. Estimation of regression coefficients when some regressors are not always observed. *Journal of the American Statistical Association* **89,** 846–866 (1994).
6. Baluja, S. & Caruana, R. in *Machine Learning Proceedings 1995* (eds Prieditis, A. & Russell, S.) 38–46 (Morgan Kaufmann, 1995). ISBN: 978-1-55860-377-6.
7. Robins, J. M. & Rotnitzky, A. Semiparametric efficiency in multivariate regression models with missing data. *Journal of the American Statistical Association* **90,** 122–129 (1995).
8. Rubinstein, R. Y. Optimization of computer simulation models with rare events. *European Journal of Operational Research* **99,** 89–112 (1997).
9. Gammerman, A., Vovk, V. & Vapnik, V. Learning by transduction. *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence* **14,** 148–155 (1998).
10. Rubinstein, R. The Cross-Entropy Method for Combinatorial and Continuous Optimization. *Methodology And Computing In Applied Probability* **1,** 127–190 (1999).
11. Vovk, V., Gammerman, A. & Saunders, C. *Machine-Learning Applications of Algorithmic Randomness* in *Proceedings of the 16th International Conference on Machine Learning (ICML)* (Morgan Kaufmann Publishers Inc., 1999), 444–453. ISBN: 1558606122.
12. Shimodaira, H. Improving predictive inference under covariate shift by weighting the log-likelihood function. *J. Stat. Plan. Inference* **90,** 227–244 (Oct. 2000).

13. Bengoetxea, E., Larrañaga, P., Bloch, I. & Perchant, A. *Estimation of Distribution Algorithms: A New Evolutionary Computation Approach for Graph Matching Problems* in *Energy Minimization Methods in Computer Vision and Pattern Recognition* (eds Figueiredo, M., Zerubia, J. & Jain, A. K.) (Springer Berlin Heidelberg, 2001), 454–469.

14. Precup, D., Sutton, R. S. & Dasgupta, S. *Off-Policy Temporal Difference Learning with Function Approximation* in *Proceedings of the 18th International Conference on Machine Learning (ICML)* (eds Brodley, C. E. & Danyluk, A. P.) (Morgan Kaufmann, 2001), 417–424.

15. Wu, C. & Sitter, R. R. A Model-Calibration Approach to Using Complete Auxiliary Information From Survey Data. *Journal of the American Statistical Association* **96,** 185–193 (2001).

16. Auer, P. Using confidence bounds for exploitation-exploration trade-offs. *J. Mach. Learn. Res.* **3** (2002).

17. Papadopoulos, H., Proedrou, K., Vovk, V. & Gammerman, A. *Inductive confidence machines for regression* in *Machine Learning: European Conference on Machine Learning* (2002), 345–356.

18. Chen, S. X., Leung, D. H. Y. & Qin, J. Information recovery in a study with surrogate endpoints. *Journal of the American Statistical Association* **98,** 1052–1062 (2003).

19. Chen, J. & Breslow, N. E. Semiparametric efficient estimation for the auxiliary outcome problem with the conditional mean model. *Canadian Journal of Statistics* **32,** 359–372 (2004).

20. Zlochin, M., Birattari, M., Meuleau, N. & Dorigo, M. Model-Based Search for Combinatorial Optimization: A Critical Survey. *Ann. Oper. Res.* **131,** 373–395 (Oct. 2004).

21. Chen, X., Hong, H. & Tamer, E. Measurement error models with auxiliary data. *The Review of Economic Studies* **72,** 343–366 (2005).

22. Sugiyama, M. & Müller, K.-R. Input-dependent Estimation of Generalization Error under Covariate Shift. *Statistics & Decisions* **23,** 249–279 (Jan. 2005).

23. Vovk, V., Gammerman, A. & Shafer, G. *Algorithmic Learning in a Random World* (Springer US, 2005).

24. Zhu, X. J. Semi-supervised learning literature survey (2005).

25. Carroll, R. J., Ruppert, D., Stefanski, L. A. & Crainiceanu, C. M. *Measurement Error in Nonlinear Models: A Modern Perspective* (Chapman and Hall/CRC, 2006).

26. Hansen, N. in *Towards a New Evolutionary Computation: Advances in the Estimation of Distribution Algorithms* (eds Lozano, J. A., Larrañaga, P., Inza, I. & Bengoetxea, E.) 75–102 (Springer Berlin Heidelberg, 2006).

27. Huang, J., Smola, A. J., Gretton, A., Borgwardt, K. M. & Schölkopf, B. *Correcting Sample Selection Bias by Unlabeled Data* in *Advances in Neural Information Processing Systems 19 (NeurIPS)* (eds Schölkopf, B., Platt, J. C. & Hofmann, T.) (MIT Press, 2006), 601–608.

28. Maheshri, N., Koerber, J. T., Kaspar, B. K. & Schaffer, D. V. Directed evolution of adeno-associated virus yields enhanced gene delivery vectors. *Nat. Biotechnol.* **24,** 198–204 (Feb. 2006).

29. Yu, M. & Nan, B. A revisit of semiparametric regression models with missing data. *Statistica Sinica,* 1193–1212 (2006).

30. Fox, R. J. *et al.* Improving catalytic function by ProSAR-driven enzyme evolution. *Nat. Biotechnol.* **25,** 338–344 (Mar. 2007).

31. Li, Y. *et al.* A diverse family of thermostable cytochrome P450s created by recombination of stabilizing fragments. *Nature Biotechnology* **25,** 1051–1056 (2007).

32. Peters, J. & Schaal, S. *Reinforcement learning by reward-weighted regression for operational space control* in *Proceedings of the 24th International Conference on Machine Learning (ICML)* (ed Ghahramani, Z.) **227** (ACM, 2007), 745–750.

33. Sugiyama, M., Krauledat, M. & Müller, K.-R. Covariate Shift Adaptation by Importance Weighted Cross Validation. *J. Mach. Learn. Res.* **8,** 985–1005 (2007).

34. Wasserman, L. & Lafferty, J. *Statistical Analysis of Semi-Supervised Regression* in *Advances in Neural Information Processing Systems* (eds Platt, J., Koller, D., Singer, Y. & Roweis, S.) **20** (Curran Associates, Inc., 2007).

35. Hafner, J. Ab-initio simulations of materials using VASP: Density-functional theory and beyond. *J. Comput. Chem.* **29,** 2044–2078 (Oct. 2008).

36. Schmidt, M. D. & Lipson, H. Coevolution of Fitness Predictors. *IEEE Trans. Evol. Comput.* **12,** 736–749 (Dec. 2008).

37. Sugiyama, M. *et al.* Direct importance estimation for covariate shift adaptation. *Ann. Inst. Stat. Math.* **60,** 699–746 (Dec. 2008).

38. Bickel, S., Bruckner, M. & Scheffer, T. Discriminative Learning Under Covariate Shift. *J. Mach. Learn. Res.* **10,** 2137–2155 (2009).

39. Gretton, A. *et al.* Covariate shift by kernel mean matching. *Dataset shift in machine learning* (2009).

40. Quiñonero-Candela, J., Sugiyama, M., Schwaighofer, A. & Lawrence, N. D. *Dataset Shift in Machine Learning* ISBN: 0262170051 (The MIT Press, 2009).

41. Romero, P. A. & Arnold, F. H. Exploring protein fitness landscapes by directed evolution. *Nat. Rev. Mol. Cell Biol.* **10,** 866–876 (Dec. 2009).

42. Zhu, X. & Goldberg, A. B. *Introduction to Semi-Supervised Learning* (Morgan & Claypool Publishers, 2009).

43. Hautier, G., Fischer, C. C., Jain, A., Mueller, T. & Ceder, G. Finding Nature's Missing Ternary Oxide Compounds Using Machine Learning and Density Functional Theory. *Chem. Mater.* **22,** 3762–3767 (June 2010).

44. Deisenroth, M. P. & Rasmussen, C. E. *PILCO: A Model-Based and Data-Efficient Approach to Policy Search* in *Proceedings of the 28th International Conference on Machine Learning (ICML)* (eds Getoor, L. & Scheffer, T.) (Omnipress, 2011), 465–472.

45. Jin, Y. Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation* **1,** 61–70 (June 2011).

46. Marks, D. S. *et al.* Protein 3D Structure Computed from Evolutionary Sequence Variation. *PLOS One* **6,** 1–20 (Dec. 2011).

47. Snoek, J., Larochelle, H. & Adams, R. P. *Practical Bayesian Optimization of Machine Learning Algorithms* in *Advances in Neural Information Processing Systems 25 (NeurIPS)* (eds Bartlett, P. L., Pereira, F. C. N., Burges, C. J. C., Bottou, L. & Weinberger, K. Q.) (2012), 2960–2968.

48. Sugiyama, M., Suzuki, T. & Kanamori, T. *Density Ratio Estimation in Machine Learning* (Cambridge University Press, Feb. 2012).

49. Dalkara, D. *et al.* In vivo-directed evolution of a new adeno-associated virus for therapeutic outer retinal gene delivery from the vitreous. *Sci. Transl. Med.* **5,** 189ra76 (June 2013).

50. Le, M. N., Ong, Y. S., Menzel, S., Jin, Y. & Sendhoff, B. Evolution by adapting surrogates. *Evol. Comput.* **21,** 313–340 (2013).

51. Owen, A. B. *Monte Carlo Theory, Methods and Examples* (2013).

52. Romero, P. A., Krause, A. & Arnold, F. H. Navigating the protein fitness landscape with Gaussian processes. *Proc. Natl. Acad. Sci. U. S. A.* **110,** E193–201 (2013).

53. Adachi, K., Enoki, T., Kawano, Y., Veraz, M. & Nakai, H. Drawing a high-resolution functional map of adeno-associated virus capsid by massively parallel sequencing. *Nat. Commun.* **5,** 3075 (2014).

54. Kingma, D. P. & Welling, M. *Auto-Encoding Variational Bayes* in *2nd International Conference on Learning Representations (ICLR)* (eds Bengio, Y. & LeCun, Y.) (2014).

55. Van Erven, T. & Harremos, P. Rényi Divergence and Kullback-Leibler Divergence. *IEEE Trans. Inf. Theory* **60,** 3797–3820 (July 2014).

56. Kingma, D. P. & Ba, J. *Adam: A Method for Stochastic Optimization* in *3rd International Conference on Learning Representations (ICLR)* (eds Bengio, Y. & LeCun, Y.) (2015).

57. Nguyen, A. M., Yosinski, J. & Clune, J. *Deep neural networks are easily fooled: High confidence predictions for unrecognizable images* in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE Computer Society, 2015), 427–436.

58. Hardt, M., Megiddo, N., Papadimitriou, C. H. & Wootters, M. *Strategic Classification* in *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science* (ed Sudan, M.) (ACM, 2016), 111–122.

59. Thomas, P. S. & Brunskill, E. *Data-Efficient Off-Policy Policy Evaluation for Reinforcement Learning* in *Proceedings of the 33rd International Conference on Machine Learning (ICML)* (eds Balcan, M. & Weinberger, K. Q.) **48** (JMLR.org, 2016), 2139–2148.

60. Breidt, F. J. & Opsomer, J. D. Model-assisted survey estimation with modern prediction techniques. *Statistical Science* **32,** 190–205 (2017).

61. Killoran, N., Lee, L. J., Delong, A., Duvenaud, D. & Frey, B. J. Generating and designing DNA with deep generative models. arXiv: `1712.06148 [cs.LG]` (Dec. 2017).

62. Lakshminarayanan, B., Pritzel, A. & Blundell, C. *Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles* in *Advances in Neural Information Processing Systems 30 (NeurIPS)* (eds Guyon, I. *et al.*) (2017), 6402–6413.

63. Olivecrona, M., Blaschke, T., Engkvist, O. & Chen, H. Molecular de-novo design through deep reinforcement learning. *J. Cheminform.* **9,** 48 (Sept. 2017).

64. Tse, L. V. *et al.* Structure-guided evolution of antigenically distinct adeno-associated virus variants for immune evasion. *Proc. Natl. Acad. Sci. U. S. A.* **114,** E4812–E4821 (June 2017).

65. Arnold, F. H. Directed Evolution: Bringing New Chemistry to Life. *Angew. Chem.* **57,** 4143–4148 (Apr. 2018).

66. Chakrabortty, A. & Cai, T. Efficient and adaptive linear regression in semi-supervised settings. *Annals of Statistics* **46,** 1541–1572. ISSN: 00905364. arXiv: `1701.04889` (2018).

67. Chua, K., Calandra, R., McAllister, R. & Levine, S. *Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models* in *Advances in Neural Information Processing Systems 31 (NeurIPS 2018)* (eds Bengio, S. *et al.*) (2018), 4759–4770.

68. Gómez-Bombarelli, R. *et al.* Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules. *ACS Central Science* **4,** 268–276 (2018).

69. Hamidieh, K. A data-driven statistical model for predicting the critical temperature of a superconductor. *Comput. Mater. Sci.* **154,** 346–354 (Nov. 2018).

70. Kuleshov, V., Fenner, N. & Ermon, S. *Accurate Uncertainties for Deep Learning Using Calibrated Regression* in *Proceedings of the 35th International Conference on Machine Learning (ICML)* (eds Dy, J. G. & Krause, A.) **80** (PMLR, 2018), 2801–2809.

71. Lei, J., G'Sell, M., Rinaldo, A., Tibshirani, R. J. & Wasserman, L. Distribution-Free Predictive Inference for Regression. *Journal of the American Statistical Association* **113,** 1094–1111 (2018).

72. Mansouri Tehrani, A. *et al.* Machine Learning Directed Search for Ultraincompressible, Superhard Materials. *J. Am. Chem. Soc.* **140,** 9844–9853 (Aug. 2018).

73. Popova, M., Isayev, O. & Tropsha, A. Deep reinforcement learning for de novo drug design. *Sci Adv* **4,** eaap7885 (July 2018).

74. Stanev, V. *et al.* Machine learning modeling of superconducting critical temperature. *npj Computational Materials* **4,** 29 (June 2018).

75. Bedbrook, C. N. *et al.* Machine learning-guided channelrhodopsin engineering enables minimally invasive optogenetics. *Nat. Methods* **16,** 1176–1184 (Nov. 2019).

76. Brookes, D. H., Park, H. & Listgarten, J. *Conditioning by adaptive sampling for robust design* in *Proceedings of the 36th International Conference on Machine Learning (ICML)* (eds Chaudhuri, K. & Salakhutdinov, R.) **97** (PMLR, 2019), 773–782.

77. De Boer, C. G. *et al.* Deciphering eukaryotic gene-regulatory logic with 100 million random promoters. *Nat. Biotechnol.* (Dec. 2019).

78. Grover, A. *et al. Bias Correction of Learned Generative Models using Likelihood-Free Importance Weighting* in *Advances in Neural Information Processing Systems 32 (NeurIPS)* (eds Wallach, H. M. *et al.*) (2019), 11056–11068.

79. Gupta, A. & Zou, J. Feedback GAN for DNA optimizes protein functions. *Nature Machine Intelligence* **1,** 105–111 (Feb. 2019).

80. Kang, S. & Cho, K. Conditional Molecular Design with Deep Generative Models. *J. Chem. Inf. Model.* **59,** 43–52 (Jan. 2019).

81. Little, R. J. & Rubin, D. B. *Statistical Analysis with Missing Data* (John Wiley & Sons, 2019).

82. Ogden, P. J., Kelsic, E. D., Sinai, S. & Church, G. M. Comprehensive AAV capsid fitness landscape reveals a viral gene and enables machine-guided design. *Science* **366,** 1139–1143 (Nov. 2019).

83. Poelwijk, F. J., Socolich, M. & Ranganathan, R. Learning the pattern of epistasis linking genotype and phenotype in a protein. *Nat. Commun.* **10,** 4213 (2019).

84. Tibshirani, R. J., Barber, R. F., Candès, E. J. & Ramdas, A. *Conformal Prediction Under Covariate Shift* in *Advances in Neural Information Processing Systems 32 (NeurIPS)* (eds Wallach, H. M. *et al.*) (2019), 2526–2536.

85. Wu, Z., Kan, S. B. J., Lewis, R. D., Wittmann, B. J. & Arnold, F. H. Machine learning-assisted directed protein evolution with combinatorial libraries. *Proc. Natl. Acad. Sci. U. S. A.* **116,** 8852–8858 (Apr. 2019).

86. Yang, K. K., Wu, Z. & Arnold, F. H. Machine-learning-guided directed evolution for protein engineering. *Nat. Methods* **16,** 687–694 (Aug. 2019).

87. Zeng, H. & Gifford, D. K. Quantification of Uncertainty in Peptide-MHC Binding Prediction Improves High-Affinity Peptide Selection for Therapeutic Design. *Cell Syst* **9,** 159–166.e3 (Aug. 2019).

88. Amini, A., Schwarting, W., Soleimany, A. & Rus, D. *Deep Evidential Regression* in *Advances in Neural Information Processing Systems* (eds Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F. & Lin, H.) **33** (Curran Associates, Inc., 2020), 14927–14937.

89. Angermüller, C. *et al. Model-based reinforcement learning for biological sequence design* in *8th International Conference on Learning Representations (ICLR)* (OpenReview.net, 2020).

90. Brookes, D. H., Busia, A., Fannjiang, C., Murphy, K. & Listgarten, J. *A view of estimation of distribution algorithms through the lens of expectation-maximization* in *GECCO '20: Genetic and Evolutionary Computation Conference* (ed Coello, C. A. C.) (ACM, 2020), 189–190.

91. Cauchois, M., Gupta, S., Ali, A. & Duchi, J. C. Robust Validation: Confident Predictions Even When Distributions Shift. arXiv: `2008.04267 [stat.ML]` (Aug. 2020).

92. Fannjiang, C. & Listgarten, J. *Autofocused oracles for model-based design* in *Advances in Neural Information Processing Systems* (eds Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F. & Lin, H.) **33** (Curran Associates, Inc., 2020), 12945–12956.

93. Hu, X. & Lei, J. *A distribution-free test of covariate shift using conformal prediction* arXiv preprint 2010.07147. 2020.

94. Kallus, N. & Mao, X. On the role of surrogates in the efficient estimation of treatment effects with limited outcome data. *arXiv preprint arXiv:2003.12408* (2020).

95. Kumar, A. & Levine, S. *Model Inversion Networks for Model-Based Optimization* in *Advances in Neural Information Processing Systems 33 (NeurIPS)* (eds Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. & Lin, H.) (2020).

96. Levine, S., Kumar, A., Tucker, G. & Fu, J. Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems. *arXiv.* eprint: `2005.01643` (cs.LG) (May 2020).

97. Linder, J., Bogard, N., Rosenberg, A. B. & Seelig, G. A Generative Neural Network for Maximizing Fitness and Diversity of Synthetic DNA and Protein Sequences. *Cell Syst* **11,** 49–62.e16 (July 2020).

98. Liu, G. *et al.* Antibody complementarity determining region design using high-capacity machine learning. *Bioinformatics* **36,** 2126–2133 (Apr. 2020).

99. Perdomo, J. C., Zrnic, T., Mendler-Dünner, C. & Hardt, M. *Performative Prediction* in *Proceedings of the 37th International Conference on Machine Learning (ICML)* **119** (PMLR, 2020), 7599–7609.

100. Rhinehart, N., McAllister, R. & Levine, S. *Deep Imitative Models for Flexible Inference, Planning, and Control* in *8th International Conference on Learning Representations (ICLR)* (OpenReview.net, 2020).

101. Russ, W. P. *et al.* An evolution-based model for designing chorismate mutase enzymes. *Science* **369,** 440–445 (July 2020).

102. Sinai, S. & Kelsic, E. D. *A primer on model-guided exploration of fitness landscapes for biological sequence design* arXiv preprint 2010.10614. Oct. 2020. eprint: `2010.10614` (q-bio.QM).

103. Sinai, S. *et al. AdaLead: A simple and robust adaptive greedy search algorithm for sequence design* arXiv preprint 2010.02141. Oct. 2020. eprint: `2010.02141` (cs.LG).

104. Sun, Y. *et al. Test-Time Training with Self-Supervision for Generalization under Distribution Shifts* in *Proceedings of the 37th International Conference on Machine Learning (ICML)* **119** (PMLR, 2020), 9229–9248.

105. Vovk, V. *Testing for concept shift online* arXiv preprint 2012.14246. 2020.

106. Wang, S., McCormick, T. H. & Leek, J. T. Methods for correcting inference based on outcomes predicted by machine learning. *Proceedings of the National Academy of Sciences* **117,** 30266–30275 (2020).

107. Wu, Z. *et al.* Signal Peptides Generated by Attention-Based Neural Networks. *ACS Synth. Biol.* **9,** 2154–2161 (Aug. 2020).

108. Angelopoulos, A. N., Bates, S., Candès, E. J., Jordan, M. I. & Lei, L. *Learn then Test: Calibrating Predictive Algorithms to Achieve Risk Control* arXiv preprint 2110.01052. 2021.

109. Bashir, A. *et al.* Machine learning guided aptamer refinement and discovery. *Nat. Commun.* **12,** 2366 (Apr. 2021).

110. Biswas, S., Khimulya, G., Alley, E. C., Esvelt, K. M. & Church, G. M. Low-N protein engineering with data-efficient deep learning. *Nature Methods* **18,** 389–396 (2021).

111. Bryant, D. H. *et al.* Deep diversification of an AAV capsid protein by machine learning. *Nat. Biotechnol.* **39** (Feb. 2021).

112. Gibbs, I. & Candès, E. J. *Adaptive Conformal Inference Under Distribution Shift* in *Advances in Neural Information Processing Systems 34 (NeurIPS)* (eds Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P. & Vaughan, J. W.) (2021), 1660–1672.

113. Greenhalgh, J. C., Fahlberg, S. A., Pfleger, B. F. & Romero, f. A. Machine learning-guided acyl-ACP reductase engineering for improved in vivo fatty alcohol production. *Nat. Commun.* **12,** 5825 (Oct. 2021).

114. Hawkins-Hooker, A. *et al.* Generating functional protein variants with variational autoencoders. *PLoS Comput. Biol.* **17,** e1008736 (Feb. 2021).

115. Hou, J., Guo, Z. & Cai, T. Surrogate assisted semi-supervised inference for high dimensional risk prediction. *arXiv preprint arXiv:2105.01264* (2021).

116. Jumper, J. *et al.* Highly accurate protein structure prediction with AlphaFold. *Nature* (July 2021).

117. Meier, J. *et al. Language models enable zero-shot prediction of the effects of mutations on protein function* in *Advances in Neural Information Processing Systems 34 (NeurIPS)* (eds Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P. & Vaughan, J. W.) (2021), 29287–29303.

118. Park, S., Li, S., Bastani, O. & Lee, I. *PAC Confidence Predictions for Deep Neural Network Classifiers* in *Proc. of the Ninth International Conference on Learning Representations (ICLR)* (2021).

119. Podkopaev, A. & Ramdas, A. *Distribution-free uncertainty quantification for classification under label shift* in *Proceedings of the 37th Conference on Uncertainty in Artificial Intelligence* (eds de Campos, C. & Maathuis, M. H.) **161** (PMLR, 2021), 844–853.

120. Shin, J.-E. *et al.* Protein design and variant prediction using autoregressive generative models. *Nat. Commun.* **12,** 2403 (Apr. 2021).

121. Soleimany, A. P. *et al.* Evidential Deep Learning for Guided Molecular Property Prediction and Discovery. *ACS Cent Sci* **7,** 1356–1367 (Aug. 2021).

122. Wittmann, B. J., Yue, Y. & Arnold, F. H. Informed training set design enables efficient machine learning-assisted directed protein evolution. *Cell Syst* **12,** 1026–1045.e7 (Nov. 2021).

123. Wu, Z., Johnston, K. E., Arnold, F. H. & Yang, K. K. Protein sequence design with deep generative models. *Curr. Opin. Chem. Biol.* **65,** 18–27 (Dec. 2021).

124. Zhu, D. *et al. Optimal trade-off control in machine learning-based library design, with application to adeno-associated virus (AAV) for gene therapy* bioRxiv preprint 2021.11.02.467003. 2021.

125. Azriel, D. *et al.* Semi-Supervised Linear Regression. *Journal of the American Statistical Association* **117,** 2238–2251 (2022).

126. Barber, R. F., Candes, E. J., Ramdas, A. & Tibshirani, R. J. Conformal prediction beyond exchangeability. arXiv: `2202.13415 [stat.ME]` (Feb. 2022).

127. Bludau, I. *et al.* The structural context of posttranslational modifications at a proteome-wide scale. *PLoS Biol.* **20,** e3001636 (May 2022).

128. Brookes, D. H., Aghazadeh, A. & Listgarten, J. On the sparsity of fitness functions and implications for learning. *Proc. Natl. Acad. Sci. U. S. A.* **119** (Jan. 2022).

129. Chakrabortty, A., Dai, G. & Carroll, R. J. Semi-Supervised Quantile Estimation: Robust and Efficient Inference in High Dimensional Settings. *arXiv preprint arXiv:2201.10208* (2022).

130. Efron, B. *Exponential Families in Theory and Practice* (Cambridge University Press, 2022).

131. Fannjiang, C., Bates, S., Angelopoulos, A. N., Listgarten, J. & Jordan, M. I. Conformal prediction under feedback covariate shift for biomolecular design. *Proc. Natl. Acad. Sci. U. S. A.* **119,** e2204569119 (Oct. 2022).

132. Fricke, E. C. *et al.* Collapse of terrestrial mammal food webs since the Late Pleistocene. *Science* **377,** 1008–1011 (Aug. 2022).

133. Hsu, C., Nisonoff, H., Fannjiang, C. & Listgarten, J. Learning protein fitness models from evolutionary and assay-labeled data. *Nat. Biotechnol.,* 1–9 (Jan. 2022).

134. Kaur, R. *et al. iDECODe: In-Distribution Equivariance for Conformal Out-of-Distribution Detection* in *Thirty-Sixth AAAI Conference on Artificial Intelligence* (AAAI Press, 2022), 7104–7114.

135. Luo, R. *et al. Sample-Efficient Safety Assurances Using Conformal Prediction* in *Algorithmic Foundations of Robotics XV: Proceedings of the Fifteenth Workshop on the Algorithmic Foundations of Robotics (WAFR)* (eds LaValle, S. M., O'Kane, J. M., Otte, M. W., Sadigh, D. & Tokekar, P.) **25** (Springer, 2022), 149–169.

136. Nijkamp, E., Ruffolo, J., Weinstein, E. N., Naik, N. & Madani, A. ProGen2: Exploring the Boundaries of Protein Language Models. arXiv: `2206.13517 [cs.LG]` (June 2022).

137. Notin, P. *et al. Tranception: Protein Fitness Prediction with Autoregressive Transformers and Inference-time Retrieval* in *Proceedings of the 39th International Conference on Machine Learning (ICML)* (eds Chaudhuri, K. *et al.*) **162** (PMLR, 2022), 16990–17017.

138. Podkopaev, A. & Ramdas, A. *Tracking the risk of a deployed model and detecting harmful distribution shifts* in *The Tenth International Conference on Learning Representations (ICLR)* (OpenReview.net, 2022).

139. Vaishnav, E. D. *et al.* The evolution, evolvability and engineering of gene regulatory DNA. *Nature* **603,** 455–463 (Mar. 2022).

140. Weinstein, E. N. *et al. Optimal Design of Stochastic DNA Synthesis Protocols based on Generative Sequence Models* in *International Conference on Artificial Intelligence and Statistics (AISTATS)* (eds Camps-Valls, G., Ruiz, F. J. R. & Valera, I.) **151** (PMLR, 2022), 7450–7482.

141. Zhang, Y. & Bradic, J. High-dimensional semi-supervised learning: in search of optimal inference of the mean. *Biometrika* **109,** 387–403 (2022).

142. Zhou, Y. *et al.* Molecular landscapes of human hippocampal immature neurons across lifespan. *Nature* **607,** 527–533 (July 2022).

143. Angelopoulos, A. N. & Bates, S. Conformal Prediction: A Gentle Introduction. *Foundations and Trends® in Machine Learning* **16,** 494–591 (2023).

144. Angelopoulos, A. N., Bates, S., Fannjiang, C., Jordan, M. I. & Zrnic, T. Prediction-Powered Inference. arXiv: `2301.09633 [stat.ML]` (Jan. 2023).

145. Barrio-Hernandez, I. *et al. Clustering predicted structures at the scale of the known protein universe* bioRxiv preprint 2023.03.09.531927. Mar. 2023.

146. Bates, S., Candès, E., Lei, L., Romano, Y. & Sesia, M. Testing for outliers with conformal p-values. *The Annals of Statistics* **51,** 149–178 (2023).

147. Madani, A. *et al.* Large language models generate functional protein sequences across diverse families. *Nat. Biotechnol.* (Jan. 2023).

148. Waudby-Smith, I. & Ramdas, A. Estimating means of bounded random variables by betting. *J. R. Stat. Soc. Series B Stat. Methodol.* (Feb. 2023).