Conformal Prediction for the Design Problem

Clara Fannjiang[a], Stephen Bates[a,b], Anastasios Angelopoulos[a], Jennifer Listgarten[a,c], and Michael I. Jordan[a,b]

[a]Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, USA
[b]Department of Statistics, University of California, Berkeley, USA
[c]Center for Computational Biology, University of California, Berkeley, USA

Abstract

In many real-world deployments of machine learning, we use a prediction algorithm to choose what data to test next. For example, in the protein design problem, we have a regression model that predicts some real-valued property of a protein sequence, which we use to propose new sequences believed to exhibit higher property values than observed in the training data. Since validating designed sequences in the wet lab is typically costly, it is important to know how much we can trust the model's predictions. In such settings, however, there is a distinct type of distribution shift between the training and test data: one where the training and test data are statistically dependent, as the latter is chosen based on the former. Consequently, the model's error on the test data—that is, the designed sequences—has some non-trivial relationship with its error on the training data. Herein, we introduce a method to quantify predictive uncertainty in such settings. We do so by constructing confidence sets for predictions that account for the dependence between the training and test data. The confidence sets we construct have finite-sample guarantees that hold for any prediction algorithm, even when a trained model chooses the test-time input distribution. As a motivating use case, we demonstrate how our method quantifies uncertainty for the predicted fitness of designed protein using several real data sets.

1 Introduction

Imagine you are a protein engineer. You are interested in designing a protein with high *fitness*—some real-valued measure of its desirability, such as fluorescence or catalytic efficiency. You have a data set of different protein sequences, denoted X_i, paired 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, you train a regression model on the data set, then identify a novel sequence that the model predicts to be more fit than the training sequences. Can you 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 make an important step toward this goal by enabling uncertainty quantification with finite-sample statistical guarantees 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, X, does not change between the training and test data distributions. For example, returning to the example of protein design,

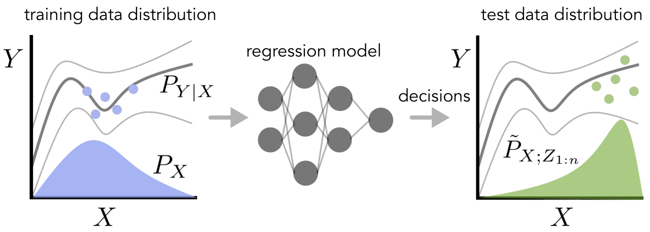


Figure 1: Illustration of feedback covariate shift. In the left graph, the blue distribution represents the training input distribution, P_X. The dark gray line sandwiched by lighter gray lines represents the mean ± the standard deviation of $P_{Y|X}$, the conditional distribution of the label given 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, adaptive experimental design, and Bayesian optimization, then give rise to a new test-time input distribution, $P_{X;Z_{1:n}}$ (right graph, green distribution). The green dots represent test data.

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 desirable proteins, small molecules, and materials, and more generally machine learning-guided scientific discovery; active learning, adaptive experimental design, and Bayesian optimization.
- **Algorithms that use predictive models to perform actions that change a system's state**, such as autonomous driving algorithms that use computer vision systems.

To make our exposition concrete, we will anchor our discussion and experiments in protein design problems. However, the methods and insights developed herein are applicable to any of the listed examples.

1.1 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 an input 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 a frequentist statistical property called *coverage*.

Definition 1. *Consider data points from some joint distribution, $(Z_1, \ldots, Z_n, Z_{\text{test}}) \sim \mathcal{P}$. 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*

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

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

There are three important parts of this definition we call attention to. First, coverage is with respect to a particular joint distribution of the training and test data, $\mathcal{P}$, as the probability statement in Eq. (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

[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.

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. (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 $\mathbb{P}(Y_{\text{test}} \in C(X_{\text{test}}) \mid X_{\text{test}} = x)$ for a particular value of interest, $x \in \mathcal{X}$. 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 independently and identically distributed from some distribution, P, *conformal prediction* provides valid confidence sets for any P and for any regression model class [63, 64, 34]. Though recent work has extended the methodology to certain forms of distribution shift [60, 15, 20, 43, 45], 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.

1.2 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 operate on. Second, we introduce a generalization of conformal prediction that produces valid confidence sets under feedback covariate shift for arbitrary regression models. 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.

1.3 Prior work

Our study investigates uncertainty quantification in a setting that brings together the well-studied concept of covariate shift [52, 58, 59, 49] with feedback between learned models and data distributions, a widespread phenomenon in real-world deployments of machine learning [23, 44]. 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. (1) for arbitrary model classes [19, 64, 3]. Though originally based on the premise of independently and identically distributed (and more broadly, exchangeable) training and test data, the framework has since been generalized to handle various forms of distribution shift, including covariate shift [60, 43], label shift [45], arbitrary distribution shifts in an online setting [20], and test distributions that are nearby the training distribution [15]. Conformal approaches have also been used to detect distribution shift [62, 28, 38, 8, 4, 46, 30].

We call particular attention to the work of Tibshirani et al. [60] 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 to distinguish it from our setting. 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—for example, 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 [12, 17]. Gaussian process regression (GPR) models are consequently popular tools for these problems, as algorithms leveraging their posterior predictive variance [6, 56] have been used to design enzymes with enhanced thermostability and catalytic activity [50, 22], and to select chemical compounds with increased binding affinity to a target [25]. 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 [33], which are increasingly being used to quantify uncertainty for deep neural networks [12, 71, 17, 37], as well as uncertainty estimates that are explicitly learned by deep models [57] 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.

2 Conformal prediction under feedback covariate shift

2.1 Feedback covariate shift

We begin by formalizing feedback covariate shift (FCS), which describes a setting where 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 are under FCS when it can be generated with 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. After we have the training data in hand, it induces the particular distribution of test inputs, $\tilde{P}_{X;Z_{1:n}}$, that the model will encounter 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 smoothness properties, for example. In particular, FCS encapsulates any design problem where we deploy an algorithm that calls a regression model fit to the training data, $Z_{1:n}$, in order to propose designed inputs.

2.2 Conformal prediction in the i.i.d. setting

To understand how we will construct valid confidence sets under FCS, we first walk through the intuition behind conformal prediction in the setting of i.i.d. training and test data, then present the adaptation to accommodate FCS.

Score function. First, we introduce the key concept 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 ideally 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 could not easily predict, which suggests it was was atypical with respect to 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 [50, 25, 22], the variance of the predictions from an ensemble of neural networks [33, 71, 37, 3], uncertainty estimates learned by deep models [2], or even the outputs of other calibration procedures [32]. Irrespective of the choice of the score function, conformal prediction will construct valid confidence sets; however, the 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 i.i.d scores. At a high level, the operating insight of conformal prediction is that when the training and test data are i.i.d, their scores are also i.i.d. from some distribution. 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 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 then compute the score using a regression model trained on the remaining n data points; the resulting $n + 1$ scores are i.i.d.

In reality, of course, we do not know the true label of the test input. However, this key property—that the scores of i.i.d. data yield i.i.d. 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 i.i.d. 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.

2.3 Conformal prediction under FCS

Conformal prediction is based on the observation that the scores of i.i.d. training and test data points are also i.i.d. When the training and test data are under FCS, the scores will no longer be i.i.d., because the training and test inputs are neither independent nor from the same distribution. Intuitively, 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 now 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 sufficiently 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

$$v(X; D) = \frac{\tilde{p}_{X;D}(X)}{p_X(X)}, \tag{2}$$

where lowercase $\tilde{p}_{X;D}$ and p_X denote the densities of the test and training input distributions, respectively, where 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 adapt various statistical procedures to standard covariate shift [58, 59, 60]. However, what distinguishes its use here will be that the particular likelihood ratio function that we call, which is indexed by a multiset, will depend on which data point is being evaluated as well as the candidate label, as will become clear shortly.

[2] Not to be confused with other uses of the term *score function* in statistics. Also note that, as 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.

Now 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 β-*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.

Using these notions of a dataset-dependent likelihood ratio function and β-quantile, we now define a 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:

$$C_\alpha(X_{\text{test}}) = \left\{y \in \mathbb{R} : S_{n+1}(X_{\text{test}}, y) \leq \textsc{Quantile}_{1-\alpha}\left(\sum_{i=1}^{n+1} w_i^y(X_{\text{test}}) \, \delta_{S_i(X_{\text{test}}, y)}\right)\right\}, \tag{3}$$

where

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

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

$$\begin{aligned} w_i^y(X_{\text{test}}) &\propto v(X_i; Z_{-i} \cup \{(X_{\text{test}}, y)\}), \; i = 1, \ldots, n, \\ w_{n+1}^y(X_{\text{test}}) &\propto v(X_{\text{test}}; Z_{1:n}), \end{aligned} \tag{4}$$

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) 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, C_α, provides coverage under FCS (see Appendix A1.1 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 confidence set, C_α, in Eq.* (3) *satisfies the coverage property in Eq.* (1)*, namely, $\mathbb{P}(Y_{\text{test}} \in C_\alpha(X_{\text{test}})) \geq 1-\alpha$.*

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) as calibrating the provided score function in order to guarantee coverage. That is, this conformal approach can complement any existing uncertainty quantification method by endowing it with finite-sample 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 in the Appendix for details.

Estimating confidence sets in practice. In practice, it is not feasible to check all candidate labels, $y \in \mathbb{R}$, in order 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) for all $y \in \mathcal{Y}$. Algorithm 1 outlines a generic structure for computing $C_\alpha(X_{\text{test}})$ for a given test input; see Section 2.4 for important special cases in which $C_\alpha(X_{\text{test}})$ can be computed more efficiently.

Algorithm 1 Pseudocode for approximately computing $C_\alpha(X_{\text{test}})$

Input: Training data, $(Z_1, \ldots, Z_n)$ 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 $i = 1, \ldots, n$ 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 $i = 1, \ldots, n+1$ do
        Compute $w_i^y(X_{\text{test}})$ according to Eq. (4)
    end for
    $q_y \leftarrow \textsc{Quantile}_{1-\alpha}\left(\sum_{i=1}^{n+1} w_i^y(X_{\text{test}})\, \delta_{S_i(X_{\text{test}},y)}\right)$
    if $S_{n+1}(X_{\text{test}}, y) \leq q_y$ then
        $C_\alpha(X_{\text{test}}) \leftarrow C_\alpha(X_{\text{test}}) \cup \{y\}$
    end if
end for
```

Relationship with i.i.d. and standard covariate shift settings. The weights assigned to each score, $w_i^y(X_{\text{test}})$ in Eq. (4), are the distinguishing factor between the confidence sets constructed by conformal approaches for the i.i.d., standard covariate shift, and FCS settings. When the training and test data are i.i.d., these weights are simply $1/n$. 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 [60]. 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. (4), $v(\cdot; D)$, is different for each of the $n+1$ training and candidate test data points and for each candidate label, $y \in \mathbb{R}$: for X_i, we evaluate the likelihood ratio where the test input distribution is the one induced by $Z_{-i} \cup \{(X_{\text{test}}, y)\}$. 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 where 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 $i = 1, \ldots, n$ 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. Similarly, as the number of training data points, n, grows, in general the two types of weights should become more similar.

Input distributions are known in the design problem. The design problem is a unique setting in which we have control over the test input distribution, 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 trained regression model [10], or whose parameters are set by solving an optimization problem (e.g., to train a generative model) [48, 29, 12, 17, 51, 68, 24, 53, 72]. 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 [31, 21, 36, 55, 7, 14]. 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 is also often explicitly known. In protein design problems, for example, training sequences are often generated by introducing random substitutions to a single wild type sequence [12, 10, 14], by recombining segments of several "parent" sequences [35, 50, 9, 22], or by independently sampling the amino acid at each position from a known distribution [72, 65]. Conveniently, we can then compute the weights in Eq. (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) 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.

2.4 Efficient computation of confidence sets under feedback covariate shift

Using Algorithm 1 to construct the 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. However, 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. 3 in the Appendix 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 value, y, for all i—for example, a linear function of y. Next we describe two such cases in more 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 greater catalytic efficiencies or thermostabilities than observed in the training data [10, 35, 18], using linear models with one-hot encodings of the protein sequence [35, 18] or embeddings thereof [10].

In the ridge regression setting, the quantity $\mu_{-i}^y(X_i)$—the prediction for the i-th training input, using the regression model fit to the remaining training data combined with the candidate test data point—can be written in closed form as

$$\begin{aligned} \mu_{-i}^y(X_i) &= \left[\left(\mathbf{X}_{-i}^T \mathbf{X}_{-i} + \gamma I \right)^{-1} \mathbf{X}_{-i}^T Y_{-i}^y \right]^T X_i \qquad (5) \\ &= \left(\sum_{j=1}^{n-1} Y_{-i;j} \mathbf{A}_{-i;j} \right)^T X_i + (\mathbf{A}_{-i;n}^T X_i) y, \end{aligned}$$

where the rows of the matrix $\mathbf{X}_{-i} \in \mathbb{R}^{n \times p}$ are the input vectors in Z_{-i}^y, $Y_{-i}^y = (Y_{-i}, y) \in \mathbb{R}^n$ contains the labels in Z_{-i}^y, the matrix $\mathbf{A}_{-i} \in \mathbb{R}^{n \times p}$ is defined as $\mathbf{A}_{-i} = \left(\mathbf{X}_{-i}^T \mathbf{X}_{-i} + \gamma I \right)^{-1} \mathbf{X}_{-i}^T$, $\mathbf{A}_{-i;j}$ 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. (5) is a linear function of the candidate label, y. Consequently, Alg. 3 in the Appendix details how we can 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. 2 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 as many of the acquisition functions used in Bayesian optimization. Gaussian process regression has been used in this manner to design desirable proteins [50, 9, 22], small molecules [25], and beyond.

For a linear kernel, the expression for the mean prediction, $\mu^{y}_{-i}(X_i)$, is the same as for ridge regression (Eq. (5)). For arbitrary kernels, the expression can be generalized and remains a linear function of y (see Appendix A2.2 for details). We can therefore mimic the computations described for the ridge regression case to compute the scores and weights efficiently.

2.5 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 in an practical manner. To do so, we first randomly partition all of our 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). That is, data splitting returns us to the setting of standard covariate shift, under which we can use the data splitting approach in [60] to construct valid confidence intervals (Theorem 4 in the Appendix).

We also observe that randomized data splitting approaches yield confidence sets with exact coverage; see Appendix A1.4 for details.

3 Experiments on protein design

To demonstrate practical uses of our theoretical results and accompanying algorithms, we now use our approach to quantify uncertainty for designed proteins. Given some fitness 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 so far 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. [70], Sinai and Kelsic [54], Hie and Yang [26], and Wu et al. [69] 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 new proteins [18, 12, 50, 9, 67, 5, 10, 36, 22, 66, 72]. Special attention has been given to the *single-shot* case, where the goal is to design fitter proteins 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, designing inputs based on a trained regression model creates statistical dependence between the designed input and training data, leading to a distribution shift in which the training

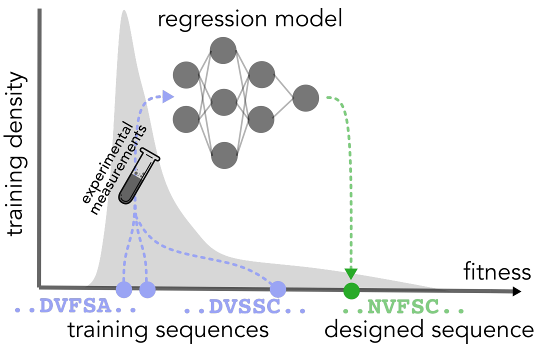


Figure 2: Single-shot protein design. The gray distribution represents the distribution of fitnesses under the training sequence distribution. The blue circles represent the fitnesses of three sequences drawn from this distribution, and the goal is to propose a sequence with higher fitness than any of these. To that end, we fit a regression model to the training sequences paired with experimental measurements of their fitnesses, then use some design procedure that leverages that trained model to propose a new sequence that we believe to have a higher fitness (green circle).

and test data are no longer independent.[3] Instead, the data fall under feedback covariate shift (FCS): since the true fitness is always some quantity dictated by nature, the conditional distribution of fitness given any sequence stays fixed, but the distribution of test sequences is chosen based on a trained regression model.

In the experiments presented here, our goal will be as follows. Given a training data set consisting of protein sequences labeled with measurements of some fitness, we will fit a regression model, then sample test sequences (representing designed proteins) according to design procedures used in recent work [10, 72] (Fig. 2). We will then construct confidence sets with guaranteed coverage for the designed proteins, and examine various characteristics of those sets to understand the utility of our approach.

3.1 Design experiments using combinatorially complete fluorescence data sets

The challenge when evaluating *in silico* design approaches is that in general, we do not have labels for the designed sequences. One workaround, which we take here, is to make use of *combinatorially complete* protein data sets [47, 67, 66, 13], in which a small number of fixed positions (typically less than twenty) are selected from some wild type sequence, and all possible variants of the wild type that vary in those selected positions are experimentally measured. Such data sets enable us to simulate design problems in which we always have labels for the designed sequences. In particular, we can use a small subset of the data for training, then run a design procedure that proposes novel proteins (restricted to being variants of the wild type at the selected positions), for which we always labels.

We use data of this kind from Poelwijk et al. [47], 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 both parents) are experimentally measured for fluorescence at both a “red” wavelength and a “blue” wavelength, resulting in combinatorially complete datasets for two different fitnesses.

3.1.1 Design experiments

Our training data sets consist of n data points, $Z_{1:n}$, sampled uniformly at random from a combinatorially complete data set. We use $n \in \{96, 192, 384\}$ as is typical of realistic scenarios [50, 9, 67, 10, 66]. We represent each training sequence with indicator features for all first- and second-order interactions between the thirteen variable sites, and fit a ridge regression model, $\mu_{Z_{1:n}}(x)$, to the training data. Linear models of interaction features between sequence positions have been observed to be both theoretically justified and

[3] 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.

empirically useful models of protein fitness [47, 27, 11] and thus may be particularly useful for protein design, particularly with small amounts of training data. Furthermore, ridge regularization endows linear models of interaction features with certain desirable properties when generalizing to amino acids not seen in the training data [27].

Following ideas in [10, 72], we design a protein by sampling from a covariate distribution whose log-likelihood is proportional to the prediction of the regression model:

$$\tilde{p}_{X;Z_{1:n}}(X_{\text{test}}) \propto \exp(\lambda \cdot \mu_{Z_{1:n}}(X_{\text{test}})), \tag{6}$$

where $\lambda > 0$, or the *inverse temperature*, is a hyperparameter that controls the entropy of the distribution. Larger values of λ result in lower-entropy distributions of designed sequences that are more likely to have high predicted fitness 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 [10, 51, 39, 72]. We take $\lambda \in \{0, 2, 4, 6\}$ to investigate how the behavior of our confidence sets varies along this trade-off.

For each setting of n and λ, we generate n training data points and one test data point, $(Z_1, \ldots, Z_n, Z_{\text{test}})$, as just described, $T = 5000$ times. For each of these T trials, we use Alg. 3 in the Appendix to construct a confidence set, $C_\alpha(X_{\text{test}})$, using a grid of real values between 0 and 1.7 spaced $\Delta = 0.01$ apart as the set of candidate labels, $\mathcal{Y}$. This range contains the ranges of fitness values in both the blue and red combinatorially complete data sets, $[0.091, 1.608]$ and $[0.025, 1.692]$, respectively. [4]

We use $\alpha = 0.1$ as a representative miscoverage value, corresponding to coverage of $1 - \alpha = 0.9$. We then compute 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 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 will be 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 the intuitive notion of a confidence interval. 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, $\Delta \cdot |C_\alpha(X_{\text{test}})|$.

3.1.2 Results

Here we discuss results for the blue fluorescence data set. Analogous results for the red fluorescence data set are presented in Appendix A4.

Effect of inverse temperature. First we examined the effect of the inverse temperature, λ, on the fitnesses of designed proteins (Fig. 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 low label values less than 0.5. Recall that our goal is to find a protein with higher fitness than observed in the training data. 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 procedures [51, 10, 39, 72], and is meant to provide an intuitive interpretation of the hyperparameter to readers unfamiliar with its use in design problems.

[4] 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 value is equivalent to including the Δ-width interval centered at that value. Generally, one should therefore set Δ as small as possible, or small enough that the resolution of coverage is acceptable, subject to one's computational budget.

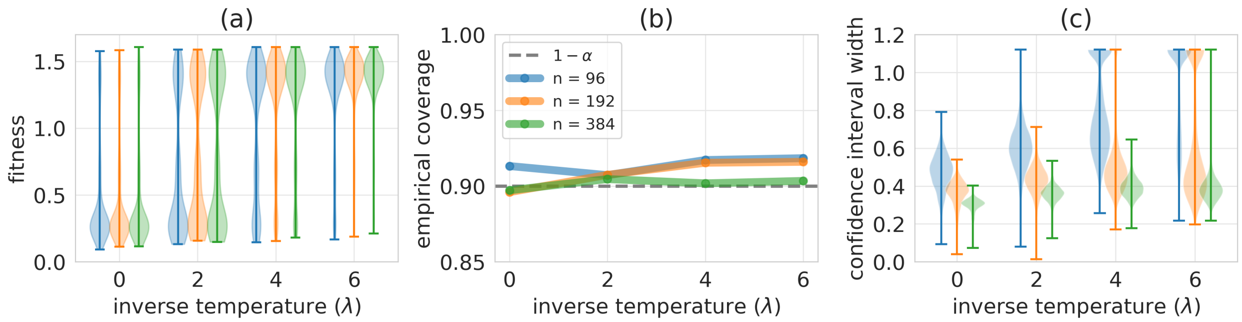


Figure 3: Quantifying the predictive uncertainty for designed proteins, using the blue fluorescence data set. (a) Distributions of fitnesses of designed proteins, (b) empirical coverage, compared to the theoretical lower bound of $1-\alpha = 0.9$ (dashed gray line), and (c) distributions of confidence interval widths for different values of the inverse temperature, λ, and different amounts of training data, n, over $T = 5000$ trials. The interval widths in (c) are reported as a fraction of the range of true fitness values in the combinatorially complete data set, $[0.091, 1.608]$; widths reported as > 1 signify confidence intervals that contain $[0.091, 1.608]$. In (a), and (c), the whiskers signify the minimum and maximum observed values.

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. 3b), reaching up to around 0.92 in some settings. Loosely speaking, this observation suggests that the confidence intervals are nearly as small, and therefore as informative, as they can be while achieving coverage. In contrast, deploying conformal prediction here with weights prescribed for standard covariate shift [60] results in overly conservative confidence sets (see Fig. A3).

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. 3c). Also, for any value of n, the intervals tend to get larger as λ increases. The first phenomenon arises from the fact that the more training data points that inform the model, the fewer the candidate labels, $y \in \mathcal{Y}$, that seem plausible for the designed protein; this 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 fitness values 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.

Using uncertainty quantification to set design procedure hyperparameters. As the inverse temperature, λ, in Eq. (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. 3a, c). As an example of how our method might be used to inform the design procedure itself, one can visualize this trade-off 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 budget of experimental resources 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 contain different amounts of measurement noise, this trade-off (and therefore, the appropriate setting of λ) will look different (Fig. 4).

Protein design experiments on the red fluorescence data set, for example, 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. 4b). We might therefore choose a smaller value of λ for the former than for 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 one of the wild type parent sequences decreases rather than increases (Fig. 4a) 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

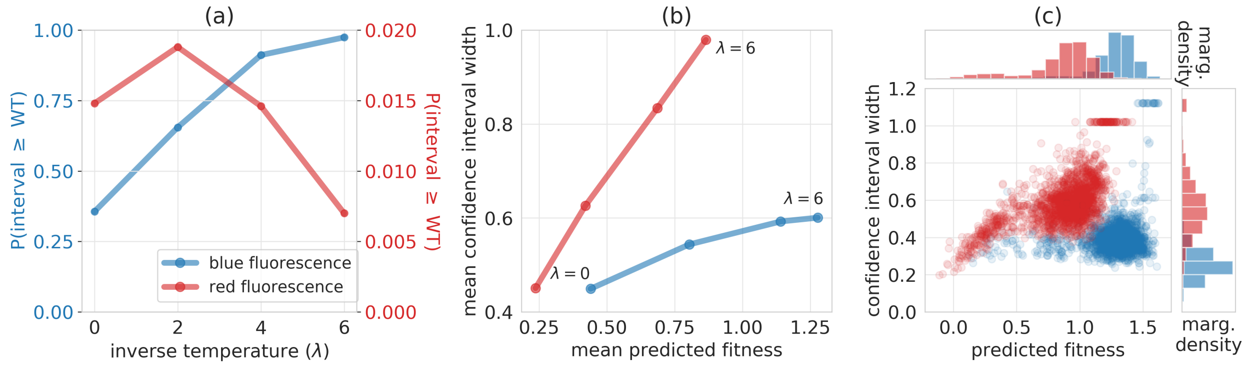


Figure 4: Comparison of trade-off between predicted fitness and predictive certainty on the red and blue fluorescence data sets. (a) Empirical probability that the smallest value in the confidence intervals of designed proteins exceeds the true fitness of one of the wild-type parent sequences, mKate2. (b) 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. (c) For $n = 384$ and $\lambda = 6$, the distributions of both confidence interval width and predicted fitnesses of designed proteins. In (b) and (c), the widths are reported as a fraction of the range of fitness values in the combinatorially complete data sets, $[0.091, 1.608]$ and $[0.025, 1.692]$ for blue and red fluorescence, respectively.

uncertainty of those predictions, it may seem that we should set λ to a higher value.

For blue fluorescence, however, we have enough predictive certainty for designed proteins that we do stand to gain from setting λ to high values, such as $\lambda = 6$. Though the mean interval width continues to grow with λ, it does so at a much slower rate than for red fluorescence; correspondingly, the empirical frequency at which the confidence interval surpasses the fitness of the wild type also continues to increase (Fig. 4a, b).

We can observe these differences in the trade-off even for a given value of λ. For $n = 384, \lambda = 6$, proteins designed for blue fluorescence with higher predicted fitness do not have much wider intervals than those with lower predicted fitness; regardless of the predicted fitness, most intervals are between roughly 0.25 and 0.5 of the total fitness range (Fig. 4c). In contrast, for red fluorescence, designed proteins with higher predicted fitness also tend to have wider confidence intervals.

3.2 Design experiments using a high-throughput fitness data set

In Section 3.1, we simulated design experiments using combinatorially comprehensive data sets so that we always had the true label of designed sequences. As an alternative approach, here we circumvent the issue of missing designed sequence labels by using a massively high-throughput data set in which not every possible sequence is labeled, but the input space is sampled densely enough that we can perform rejection sampling to sample designed sequences for which we have labels. Specifically, we use a data set with the fitnesses of millions of sequences that vary at seven selected positions, and that otherwise match a wild type. We hold out half a million of these sequences and use the remaining data for training and calibration, as detailed shortly. The input space—all 21^7 sequences that vary at those positions—is sampled sufficiently densely and uniformly by the held-out sequences that we can treat them as samples from a proposal distribution, and perform rejection sampling to get designed sequences from the test input distribution.

We now describe the particular protein design problem in more detail.

3.2.1 Design of adeno-associated virus capsids with improved packaging ability

Adeno-associated viruses (AAVs) are a class of viruses whose capsid, the protein shell that encapsulates the viral genome, holds great promise as a delivery vehicle for gene therapy. As such, the protein that constitutes the capsid has been modified to enhance various fitnesses, such as the ability to enter specific tissues and to not prompt an immune response [40, 16, 61]. Such efforts usually start by sampling millions of protein sequences from some distribution, called a *library*, then running an experiment that selects out the fittest sequences. Standard libraries used today have relatively high entropy, resulting in a highly diverse set of sequences that can yield successful outcomes for a myriad of downstream selection experiments.

However, most sequences sampled from standard libraries fail to assemble into a capsid that packages the genetic payload [1, 61, 41]—an ability called *packaging*, which is the minimum requirement of a gene therapy delivery mechanism, and therefore a prerequisite to any other desired fitness. If libraries could be developed with higher expected packaging ability, without compromising sequence diversity, the success rate of downstream selection experiments should improve. To this end, Zhu et al. [72] use regression models trained on sequence-packaging data to solve for the parameters of a library that simultaneously has high entropy and yields sequences with high predicted packaging ability.

Here, we replicate their methodology to set library parameters, then use the data splitting method described in Section 2.5 to construct confidence sets for protein sequences sampled from that library. We use the high-throughput data collected by Zhu et al. [72], which sampled millions of sequences from a common baseline called the NNK library and measured their packaging abilities, resulting in $8,552,729$ labeled sequences. We randomly select and hold out one million of these data points, for calibration and test purposes we will describe shortly, then train an ensemble of five neural networks on the remaining data to predict packaging ability from sequence.

3.2.2 Setting library parameters

In accordance with commonly used DNA synthesis protocols, the library parameters we can control are the marginal probabilities of codons at each site of a protein sequence. Let $\{q_\theta : \theta \in \Theta\}$ denote the set of all sequence distributions that are a product of independent distributions over codons at each site, where the library parameters, θ, contain the site-specific codon probabilities. Zhu et al. [72] set these parameters by solving the following optimization problem, where μ denotes the trained regression model:

$$\theta_\tau = \underset{\theta \in \Theta}{\arg\max}\, \mathbb{E}_{q_\theta(x)}\left[\mu(x)\right] + \tau \cdot H(q_\theta), \tag{7}$$

where $H(\cdot)$ denotes the entropy of a discrete distribution, and $\tau \geq 0$ is a regularization hyperparameter called the temperature. As τ increases, the library parameters that solve Eq. (7) should exhibit greater entropy, but at the cost of lower expected predicted fitness.

Following Zhu et al. [72], we solve Eq. (7) using stochastic gradient descent for a range of temperature values between 0 and 1. We calculate the entropy of each of the resulting libraries in closed form, and also use the neural network ensemble to predict the fitnesses of sequences sampled from them; refer to the Supplementary Materials and Methods in the work by Zhu et al. [72] for the full methodological details. By visualizing the trade-off between entropy and mean predicted fitness, we identify temperature values that result in libraries with both higher expected predicted fitness than the NNK library, and negligible decrease in entropy.

3.2.3 Sampling and constructing confidence sets for designed sequences

For each of these libraries, we sample designed sequences from the library using rejection sampling, where we treat 500,000 of the held-out sequences as samples from a proposal distribution (i.e., the NNK library). Then, using the remaining 500,000 held-out data points as calibration data, we employ the data splitting method described in Section 2.5 to construct confidence sets for the test sequences.

3.2.4 Results

Varying the temperature hyperparameter exposes a trade-off between mean predicted fitness and entropy, compared to which the NNK library, whose parameter values are intended to roughly maximize entropy while minimizing the probability of generating a stop codon, exhibits the highest entropy but also the lowest mean predicted fitness by a considerable margin (Fig. 5a). Temperature values roughly between 0.3 and 0.6 yield libraries with similar entropy to the NNK library, but much higher mean predicted fitness. As expected, the library parameters—site-specific probabilities of the twenty amino acids, as well as a stop codon—resulting

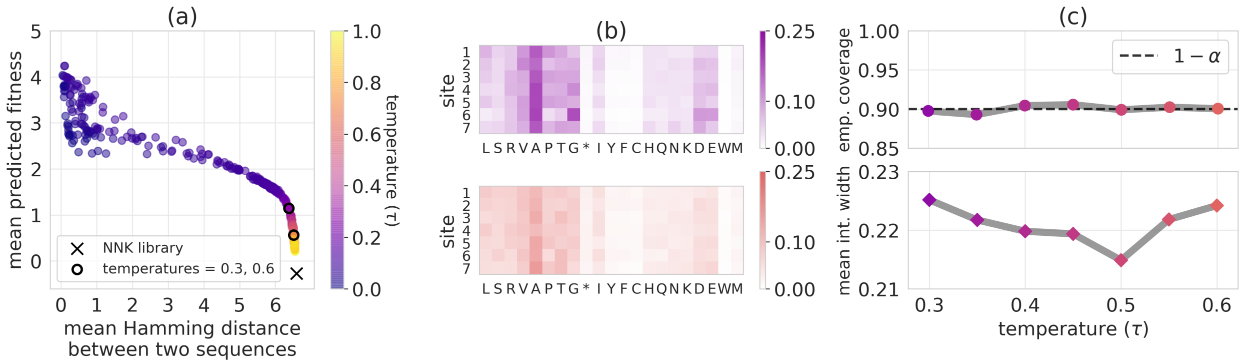


Figure 5: Quantifying uncertainty of predicted fitness of designed adeno-associated viral capsid proteins. (a) Trade-off curve between mean predicted fitness and entropy of the library that solves Eq. (7), for different values of the temperature, τ. As a more interpretable proxy of entropy, we instead report the mean Hamming distance between two sequences sampled from the library, where larger values correspond to greater entropy. The + marker is the baseline NNK library, from which training sequences were sampled, and the open circles demarcate the endpoints of the range of temperature values for which we perform uncertainty quantification, $\tau \in [0.3, 0.6]$. (b) Heatmaps depicting the library parameters corresponding to $\tau = 0.3$ (top) and $\tau = 0.6$ (bottom), namely, seven site-specific categorical distributions over the twenty amino acids plus the stop codon. The color intensity depicts the probability of each amino acid at each site. (c) Empirical coverage (top) and mean width (bottom) of confidence intervals constructed using the data splitting approach in Appendix A1.3, for the range of of temperature values demarcated in the left subplot. The interval width is reported as a fraction of the entire range of fitness values in the labeled data, $[-7.32, 8.86]$, and the dashed black line is the theoretical lower bound of $1 - \alpha = 0.9$.

from $\tau = 0.6$ appear more homogeneous, corresponding to a higher-entropy sequence distribution, than those resulting from $\tau = 0.3$ (Fig. 5b).

For seven libraries with temperature values ranging between 0.3 and 0.6, the empirical coverage resulting from the confidence sets is very close to $1 - \alpha = 0.9$, which indicates that the sets are not overly conservative (Fig. 5c). The mean confidence interval widths are also a reasonably small fraction of the total range of fitness values, $[-7.32, 8.86]$ (Fig. 5c).

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 marks an important step toward enabling practitioners to trust the predictions of learned models in such settings. In particular, we hope our examples of quantifying the predictive uncertainty of designed proteins demonstrate the broad applicability of our approach in the context of design problems. Now that we have demonstrated the theoretical and empirical validity of our uncertainty quantification approach, in terms of the frequentist statistical notion of coverage, it will be of interest to investigate whether and how design procedures can leverage these uncertainty estimates, as touched on in Section 3.1.2.

Looking beyond the design problem, the formalism of feedback covariate shift (FCS) introduced here describes a myriad of 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.

5 Acknowledgments

We are grateful to Danqing Zhu, David H. Brookes, Akosua Busia, and David Schaffer for generously allowing us to work with the AAV capsid packaging data. We also thank Hunter Nisonoff and Tijana Zrnic for insightful discussions. C.F. was supported by a National Science Foundation Graduate Research Fellowship under Grant DGE 2146752.

References

[1] Kei Adachi, Tatsuji Enoki, Yasuhiro Kawano, Michael Veraz, and Hiroyuki Nakai. Drawing a high-resolution functional map of adeno-associated virus capsid by massively parallel sequencing. *Nat. Commun.*, 5:3075, 2014.

[2] Alexander Amini, Wilko Schwarting, Ava Soleimany, and Daniela Rus. Deep evidential regression. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 14927–14937. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/aab085461de182608ee9f607f3f7d18f-Paper.pdf`.

[3] Anastasios N Angelopoulos and Stephen Bates. A gentle introduction to conformal prediction and distribution-free uncertainty quantification. *arXiv preprint arXiv:2107.07511*, 2021.

[4] Anastasios N Angelopoulos, Stephen Bates, Emmanuel J Candès, Michael I Jordan, and Lihua Lei. Learn then test: Calibrating predictive algorithms to achieve risk control. *arXiv preprint arXiv:2110.01052*, 2021.

[5] Christof Angermueller, David Dohan, David Belanger, Ramya Deshpande, Kevin Murphy, and Lucy Colwell. Model-based reinforcement learning for biological sequence design. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2019.

[6] P Auer. Using confidence bounds for exploitation-exploration trade-offs. *J. Mach. Learn. Res.*, 2002.

[7] Ali Bashir, Qin Yang, Jinpeng Wang, Stephan Hoyer, Wenchuan Chou, Cory McLean, Geoff Davis, Qiang Gong, Zan Armstrong, Junghoon Jang, Hui Kang, Annalisa Pawlosky, Alexander Scott, George E Dahl, Marc Berndl, Michelle Dimon, and B Scott Ferguson. Machine learning guided aptamer refinement and discovery. *Nat. Commun.*, 12(1):2366, April 2021.

[8] Stephen Bates, Emmanuel Candès, Lihua Lei, Yaniv Romano, and Matteo Sesia. Testing for outliers with conformal p-values. *arXiv preprint arXiv:2104.08279*, 2021.

[9] Claire N Bedbrook, Kevin K Yang, J Elliott Robinson, Elisha D Mackey, Viviana Gradinaru, and Frances H Arnold. Machine learning-guided channelrhodopsin engineering enables minimally invasive optogenetics. *Nat. Methods*, 16(11):1176–1184, 2019.

[10] Surojit Biswas, Grigory Khimulya, Ethan C Alley, Kevin M Esvelt, and George M Church. Low-N protein engineering with data-efficient deep learning. *Nature Methods*, 18(4):389–396, 2021.

[11] David Brookes, Amirali Aghazadeh, and Jennifer Listgarten. On the sparsity of fitness functions and implications for learning. *Proc. of the Natl. Acad. Sci.*, 2021.

[12] David H. Brookes, Hahnbeom Park, and Jennifer Listgarten. Conditioning by adaptive sampling for robust design. In *Proc. of the International Conference on Machine Learning (ICML)*, 2019.

[13] David H Brookes, Amirali Aghazadeh, and Jennifer Listgarten. On the sparsity of fitness functions and implications for learning. *Proc. Natl. Acad. Sci. U. S. A.*, 119(1), January 2022.

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

[15] Maxime Cauchois, Suyash Gupta, Alnur Ali, and John C Duchi. Robust validation: Confident predictions even when distributions shift. *arXiv preprint arXiv:2008.04267*, 2020.

[16] Deniz Dalkara, Leah C Byrne, Ryan R Klimczak, Meike Visel, Lu Yin, William H Merigan, John G Flannery, and David V Schaffer. In vivo-directed evolution of a new adeno-associated virus for therapeutic outer retinal gene delivery from the vitreous. *Sci. Transl. Med.*, 5(189):189ra76, June 2013.

[17] Clara Fannjiang and Jennifer Listgarten. Autofocused oracles for model-based design. In *Advances in Neural Information Processing Systems 33*, 2020.

[18] Richard J Fox, S Christopher Davis, Emily C Mundorff, Lisa M Newman, Vesna Gavrilovic, Steven K Ma, Loleta M Chung, Charlene Ching, Sarena Tam, Sheela Muley, John Grate, John Gruber, John C Whitman, Roger A Sheldon, and Gjalt W Huisman. Improving catalytic function by ProSAR-driven enzyme evolution. *Nat. Biotechnol.*, 25(3):338–344, March 2007.

[19] Alex Gammerman, Volodya Vovk, and Vladimir Vapnik. Learning by transduction. *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, 14:148–155, 1998.

[20] Isaac Gibbs and Emmanuel Candès. Adaptive conformal inference under distribution shift. *arXiv preprint arXiv:2106.00170*, 2021.

[21] Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a Data-Driven continuous representation of molecules. *ACS Cent Sci*, 4(2):268–276, February 2018.

[22] Jonathan C Greenhalgh, Sarah A Fahlberg, Brian F Pfleger, and Philip A Romero. Machine learning-guided acyl-ACP reductase engineering for improved in vivo fatty alcohol production. *Nat. Commun.*, 12(1):5825, October 2021.

[23] Moritz Hardt, Nimrod Megiddo, Christos Papadimitriou, and Mary Wootters. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, 2016.

[24] Alex Hawkins-Hooker, Florence Depardieu, Sebastien Baur, Guillaume Couairon, Arthur Chen, and David Bikard. Generating functional protein variants with variational autoencoders. *PLoS Comput. Biol.*, 17(2):e1008736, February 2021.

[25] Brian Hie, Bryan D Bryson, and Bonnie Berger. Leveraging uncertainty in machine learning accelerates biological discovery and design. *Cell Syst*, 11(5):461–477.e9, November 2020.

[26] Brian L Hie and Kevin K Yang. Adaptive machine learning for protein engineering. *Curr. Opin. Struct. Biol.*, 72:145–152, December 2021.

[27] Chloe Hsu, Hunter Nisonoff, Clara Fannjiang, and Jennifer Listgarten. Combining evolutionary and assay-labelled data for protein fitness prediction. *Nat. Biotech.*, 2021.

[28] Xiaoyu Hu and Jing Lei. A distribution-free test of covariate shift using conformal prediction. *arXiv preprint arXiv:2010.07147*, 2020.

[29] Seokho Kang and Kyunghyun Cho. Conditional molecular design with deep generative models. *J. Chem. Inf. Model.*, 59(1):43–52, January 2019.

[30] Ramneet Kaur, Susmit Jha, Anirban Roy, Sangdon Park, Edgar Dobriban, Oleg Sokolsky, and Insup Lee. iDECODe: In-distribution equivariance for conformal out-of-distribution detection. *arXiv preprint arXiv:2201.02331*, 2022.

[31] Nathan Killoran, Leo J Lee, Andrew Delong, David Duvenaud, and Brendan J Frey. Generating and designing DNA with deep generative models. In *Neural Information Processing Systems (NeurIPS) Computational Biology Workshop*, 2017.

[32] Volodymyr Kuleshov, Nathan Fenner, and Stefano Ermon. Accurate uncertainties for deep learning using calibrated regression. 2018. arXiv:1807.00263.

[33] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, pages 6402–6413, 2017.

[34] Jing Lei, Max G'Sell, Alessandro Rinaldo, Ryan J. Tibshirani, and Larry Wasserman. Distribution-free predictive inference for regression. *Journal of the American Statistical Association*, 113(523):1094–1111, 2018.

[35] Yougen Li, D Allan Drummond, Andrew M Sawayama, Christopher D Snow, Jesse D Bloom, and Frances H Arnold. A diverse family of thermostable cytochrome p450s created by recombination of stabilizing fragments. *Nature Biotechnology*, 25(9):1051–1056, 2007.

[36] Johannes Linder, Nicholas Bogard, Alexander B Rosenberg, and Georg Seelig. A generative neural network for maximizing fitness and diversity of synthetic DNA and protein sequences. *Cell Syst*, 11(1): 49–62.e16, July 2020.

[37] Ge Liu, Haoyang Zeng, Jonas Mueller, Brandon Carter, Ziheng Wang, Jonas Schilz, Geraldine Horny, Michael E Birnbaum, Stefan Ewert, and David K Gifford. Antibody complementarity determining region design using high-capacity machine learning. *Bioinformatics*, 36(7):2126–2133, April 2020.

[38] Rachel Luo, Shengjia Zhao, Jonathan Kuck, Boris Ivanovic, Silvio Savarese, Edward Schmerling, and Marco Pavone. Sample-efficient safety assurances using conformal prediction. *arXiv preprint arXiv:2109.14082*, 2021.

[39] Ali Madani, Ben Krause, Eric R Greene, Subu Subramanian, Benjamin P Mohr, James M Holton, Jose Luis Olmos, Caiming Xiong, Zachary Z Sun, Richard Socher, James S Fraser, and Nikhil Naik. Deep neural language modeling enables functional protein generation across families. July 2021.

[40] Narendra Maheshri, James T Koerber, Brian K Kaspar, and David V Schaffer. Directed evolution of adeno-associated virus yields enhanced gene delivery vectors. *Nat. Biotechnol.*, 24(2):198–204, February 2006.

[41] Pierce J Ogden, Eric D Kelsic, Sam Sinai, and George M Church. Comprehensive AAV capsid fitness landscape reveals a viral gene and enables machine-guided design. *Science*, 366(6469):1139–1143, November 2019.

[42] Harris Papadopoulos, Kostas Proedrou, Vladimir Vovk, and Alex Gammerman. Inductive confidence machines for regression. In *Machine Learning: European Conference on Machine Learning*, pages 345–356, 2002. doi: https://doi.org/10.1007/3-540-36755-1_29.

[43] Sangdon Park, Shuo Li, Osbert Bastani, and Insup Lee. PAC confidence predictions for deep neural network classifiers. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=Qk-Wq5AIjpq`.

[44] Juan Perdomo, Tijana Zrnic, Celestine Mendler-Dünner, and Moritz Hardt. Performative prediction. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 7599–7609. PMLR, 13–18 Jul 2020. URL `https://proceedings.mlr.press/v119/perdomo20a.html`.

[45] Aleksandr Podkopaev and Aaditya Ramdas. Distribution-free uncertainty quantification for classification under label shift. *arXiv preprint arXiv:2103.03323*, 2021.

[46] Aleksandr Podkopaev and Aaditya Ramdas. Tracking the risk of a deployed model and detecting harmful distribution shifts. *arXiv preprint arXiv:2110.06177*, 2021.

[47] Frank J Poelwijk, Michael Socolich, and Rama Ranganathan. Learning the pattern of epistasis linking genotype and phenotype in a protein. *Nat. Commun.*, 10(1):4213, 2019.

[48] Mariya Popova, Olexandr Isayev, and Alexander Tropsha. Deep reinforcement learning for de novo drug design. *Sci Adv*, 4(7):eaap7885, July 2018.

[49] Joaquin Quiñonero Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D. Lawrence. *Dataset Shift in Machine Learning.* The MIT Press, 2009. ISBN 0262170051.

[50] Philip A Romero, Andreas Krause, and Frances H Arnold. Navigating the protein fitness landscape with gaussian processes. *Proc. Natl. Acad. Sci. U. S. A.*, 110(3):E193–201, 2013.

[51] William P Russ, Matteo Figliuzzi, Christian Stocker, Pierre Barrat-Charlaix, Michael Socolich, Peter Kast, Donald Hilvert, Remi Monasson, Simona Cocco, Martin Weigt, and Rama Ranganathan. An evolution-based model for designing chorismate mutase enzymes. *Science*, 369(6502):440–445, July 2020.

[52] Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *J. Stat. Plan. Inference*, 90(2):227–244, October 2000.

[53] Jung-Eun Shin, Adam J Riesselman, Aaron W Kollasch, Conor McMahon, Elana Simon, Chris Sander, Aashish Manglik, Andrew C Kruse, and Debora S Marks. Protein design and variant prediction using autoregressive generative models. *Nat. Commun.*, 12(1):2403, April 2021.

[54] Sam Sinai and Eric D Kelsic. A primer on model-guided exploration of fitness landscapes for biological sequence design. October 2020.

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

[56] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In F Pereira, C J C Burges, L Bottou, and K Q Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[57] Ava P Soleimany, Alexander Amini, Samuel Goldman, Daniela Rus, Sangeeta N Bhatia, and Connor W Coley. Evidential deep learning for guided molecular property prediction and discovery. *ACS Cent Sci*, 7(8):1356–1367, August 2021.

[58] Masashi Sugiyama and Klaus-Robert Müller. Input-dependent estimation of generalization error under covariate shift. *Statistics & Decisions*, 23:249–279, 01 2005. doi: 10.1524/stnd.2005.23.4.249.

[59] Masashi Sugiyama, Matthias Krauledat, and Klaus-Robert Müller. Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research*, 8(35):985–1005, 2007. URL `http://jmlr.org/papers/v8/sugiyama07a.html`.

[60] Ryan J Tibshirani, Rina Foygel Barber, Emmanuel Candes, and Aaditya Ramdas. Conformal prediction under covariate shift. In *Advances in Neural Information Processing Systems*, volume 32, pages 2530–2540. 2019. URL `https://proceedings.neurips.cc/paper/2019/file/8fb21ee7a2207526da55a679f0332de2-Paper.pdf`.

[61] Longping Victor Tse, Kelli A Klinc, Victoria J Madigan, Ruth M Castellanos Rivera, Lindsey F Wells, L Patrick Havlik, J Kennon Smith, Mavis Agbandje-McKenna, and Aravind Asokan. Structure-guided evolution of antigenically distinct adeno-associated virus variants for immune evasion. *Proc. Natl. Acad. Sci. U. S. A.*, 114(24):E4812–E4821, June 2017.

[62] Vladimir Vovk. Testing for concept shift online. *arXiv preprint arXiv:2012.14246*, 2020.

[63] Vladimir Vovk, Alexander Gammerman, and Craig Saunders. Machine-learning applications of algorithmic randomness. In *International Conference on Machine Learning*, pages 444–453, 1999.

[64] Vladimir Vovk, Alex Gammerman, and Glenn Shafer. *Algorithmic Learning in a Random World.* Springer, New York, NY, USA, 2005.

[65] Eli N Weinstein, Alan N Amin, Will Grathwohl, Daniel Kassler, Jean Disset, and Debora S Marks. Optimal design of stochastic DNA synthesis protocols based on generative sequence models. October 2021.

[66] Bruce J Wittmann, Yisong Yue, and Frances H Arnold. Informed training set design enables efficient machine learning-assisted directed protein evolution. *Cell Syst*, 12(11):1026–1045.e7, 2021.

[67] Zachary Wu, S B Jennifer Kan, Russell D Lewis, Bruce J Wittmann, and Frances H Arnold. Machine learning-assisted directed protein evolution with combinatorial libraries. *Proc. Natl. Acad. Sci. U. S. A.*, 116(18):8852–8858, 2019.

[68] Zachary Wu, Kevin K Yang, Michael J Liszka, Alycia Lee, Alina Batzilla, David Wernick, David P Weiner, and Frances H Arnold. Signal peptides generated by Attention-Based neural networks. *ACS Synth. Biol.*, 9(8):2154–2161, August 2020.

[69] Zachary Wu, Kadina E Johnston, Frances H Arnold, and Kevin K Yang. Protein sequence design with deep generative models. *Curr. Opin. Chem. Biol.*, 65:18–27, December 2021.

[70] Kevin K Yang, Zachary Wu, and Frances H Arnold. Machine-learning-guided directed evolution for protein engineering. *Nat. Methods*, 16(8):687–694, August 2019.

[71] Haoyang Zeng and David K Gifford. Quantification of uncertainty in Peptide-MHC binding prediction improves High-Affinity peptide selection for therapeutic design. *Cell Syst*, 9(2):159–166.e3, August 2019.

[72] Danqing Zhu, David H Brookes, Akosua Busia, Ana Carneiro, Clara Fannjiang, Galina Popova, David Shin, Edward F Chang, Tomasz J Nowakowski, Jennifer Listgarten, and David V Schaffer. Machine learning-based library design improves packaging and diversity of adeno-associated virus (AAV) libraries. 2021.

A1 Proofs

A1.1 Proof of Theorem 1

Data from feedback covariate shift are a special case of what we call *pseudo-exchangeable*[5] random variables.

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

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

where $v_{-i} = v_{1:(n+1)} \setminus v_i$, 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. Our technical development here builds upon the work of Tibshirani et al. [60], who generalize conformal prediction to handle "weighted exchangeable" random variables, including data under standard covariate shift. We further generalize conformal prediction to account for pseudo-exchangeable random variables, and show that data generated under feedback covariate shift are pseudo-exchangeable. A straightforward application of Theorem 2 then yields Theorem 1 in the main text as a corollary.

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 [60].

Proposition 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 exchangeable 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 $i = 1, \dots, n+1$. Define*

$$w_i(z_1, \dots, z_{n+1}) \equiv \frac{\sum_{\sigma : \sigma(n+1) = i} \prod_{j=1}^{n+1} g_j(z_{\sigma(j)};\, z_{-\sigma(j)})}{\sum_{\sigma} \prod_{j=1}^{n+1} g_j(z_{\sigma(j)};\, z_{-\sigma(j)})}, \quad i = 1, \dots, n+1, \tag{A1}$$

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 taken on 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

[5] 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, $v_1, \dots, v_{i-1}, v_{i+1}, \dots, v_n$, whereas each factor in the pseudo-likelihood approximation also depends on the identities (i.e., the ordering) of the other variables.

the general case, but the notation that accommodates duplicate values is cumbersome. For $i = 1, \ldots, n+1$,

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

□

Lemma 1. *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 exchangeable 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 $i = 1, \ldots, n+1$. For any $\beta \in (0,1)$,*

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

where $w_i(z_1, \ldots, z_{n+1})$ is defined in Eq. (A1).

Proof. Due to the invariance of the factor functions, $g(\cdot; s)$, to the ordering of the values in s, the arguments that prove Lemma 3 in [60] also hold for pseudo-exchangeable random variables, as follows.

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

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

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 $Z_i = (X_i, Y_i) \in \mathcal{X} \times \mathbb{R}$, $i = 1, \ldots, n+1$ 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 $x \in \mathcal{X}$:*

$$C_\alpha(x) = \left\{ y \in \mathbb{R} : S_{n+1}(x,y) \le \textsc{Quantile}_{1-\alpha}\left(\sum_{i=1}^{n+1} w_i(Z_1, \ldots, Z_n, (x,y))\, \delta_{S_i(x,y)} \right) \right\}, \tag{A2}$$

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. (A1)*. Then C_α satisfies*

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

where the probability is over all $n+1$ data points, $(X_1, Y_1), \ldots, (X_{n+1}, Y_{n+1})$.

Proof. By construction, we have

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

Applying Lemma 1 gives the result. □

Finally, Theorem 1 in the main text 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 feedback covariate shift are pseudo-exchangeable with the exchangeable 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

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

for the likelihood ratio function, $v(\cdot;\cdot)$, defined in Eq. (2). The weights, $w_i(z_1, \ldots, z_{n+1})$, in Eq. (A1) 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, used to define the confidence set in Eq. (3) in the main text: $w_i^y(x) = w_i(Z_1, \ldots, Z_n, (x,y))$ for $i = 1, \ldots, n+1$. That is, Eq. (3) gives the confidence set defined in Eq. (A2) for data under feedback covariate shift. Applying Theorem 2 then yields Theorem 1.

A1.2 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 $w = (w_1, \dots, w_{n+1})$ on points $s = (s_1, \dots, s_{n+1})$, respectively, 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* as

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

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

$$\textsc{RandomizedQuantile}_\beta(s, w) = \begin{cases} \textsc{QuantileLB}_\beta(s, w) & \text{w. p. } \frac{\mathrm{QF}_\beta(s,w) - \beta}{\mathrm{QF}_\beta(s,w) - \mathrm{LF}_\beta(s,w)}, \\ \textsc{Quantile}_\beta(s, w) & \text{w. p. } 1 - \frac{\mathrm{QF}_\beta(s,w) - \beta}{\mathrm{QF}_\beta(s,w) - \mathrm{LF}_\beta(s,w)}. \end{cases} \tag{A3}$$

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:

$$C_\alpha^{\text{rand}}(x) = \left\{ y \in \mathbb{R} : S((x, y), Z_{1:n}) \leq \textsc{RandomizedQuantile}_{1-\alpha}(s(Z_1, \dots, Z_n, (x, y)), w(Z_1, \dots, Z_n, (x, y)) \right\}, \tag{A4}$$

where $s(Z_1, \dots, Z_n, (x, y)) = (S_1, \dots, S_n, S((x, y), Z_{1:n}), S_i = S(Z_i, Z_{-i} \cup \{(x, y)\})$ 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. (4). Note that for each value of $y \in \mathbb{R}$, 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 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 confidence set C_α^{rand} in Eq.* (A4) *satisfies the exact coverage property:*

$$\mathbb{P}(Y_{\text{test}} \in C_\alpha^{\text{rand}}(X_{\text{test}})) = 1 - \alpha, \tag{A5}$$

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

Proof. Denote $Z_{n+1} = Z_{\text{test}}$ 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:

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

where $s(z_1, \dots, z_{n+1}) = (S(z_1, z_{-1}), \dots, S(z_{n+1}, z_{-(n+1)}))$. 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 1

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

and consequently

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

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. In fact, this approach is how we randomize a data splitting approach to achieve exact coverage; see Appendix A1.4.

A1.3 Data splitting

In general, computing the confidence set, $C_\alpha(x)$, using Alg. 1 requires fitting $(n + 1) \times |\mathcal{Y}|$ regression models. A much more computationally attractive alternative is called a *data splitting* or *split conformal* approach [42, 34], in which we 1) randomly partition the labeled data into disjoint training and *calibration* data sets, 2) fit a single regression model to the training data, and 3) 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 for low-n settings such as many protein design problems.

Here, we show how a data splitting approach for feedback covariate shift simplifies our setting to standard covariate shift. We can then use the data splitting method from Tibshirani et al. [60] to produce confidence sets with coverage; the subsequent two subsections show how to introduce randomization to achieve exact coverage.

To begin, we recall the standard covariate shift model [52, 58, 59]. Each training data point, $Z_i = (X_i, Y_i)$ for $i = 1, \ldots, n$, is drawn independently and identically from some distribution: $X_i \sim P_X, Y_i \sim P_{Y|X_i}$. 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 feedback covariate shift, here the test data point cannot be chosen in a way that depends on the training data.

Returning to feedback covariate shift, suppose we randomly partition all of our labeled data into two disjoint sets: a training data set and a calibration data set. Let μ denote the regression model fit to the training data; we henceforth consider μ as a fixed predictor, and make no further use of the training data. That is, we are conditioning on the training data. As such, without loss of generality we will use $Z_1, \ldots, Z_m$ to refer to the calibration data and Z_{m+1} to refer to the test data point. Now suppose the test input distribution is induced by the trained regression model, μ; we will write $\tilde{P}_{X;\mu}$. 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 has returned us to standard covariate shift.

To construct a confidence set under standard covariate shift, define the following likelihood ratio function:

$$v(x) = \frac{\tilde{p}_{X;\mu}(x)}{p_X(x)}, \tag{A6}$$

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 [3]:

$$S(x, y) = \frac{|y - \mu(x)|}{u(x)}. \tag{A7}$$

where u is any heuristic, nonnegative notion of uncertainty; one can also set $u(x) = 1$ to use the residual as the score. 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

$$\begin{aligned} C_\alpha^{\text{split}}(x) &= \mu(x) \pm q \cdot u(x), \\ q &= \textsc{Quantile}_{1-\alpha}\left(\sum_{i=1}^{m} w_i(x)\, \delta_{S_i} + w_{n+1}(x)\, \delta_\infty\right), \end{aligned} \tag{A8}$$

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

$$\begin{aligned} w_i(x) &= \frac{v(X_i)}{\sum_{j=1}^m v(X_j) + v(x)}, \quad i = 1, \ldots, m, \\ w_{m+1}(x) &= \frac{v(x)}{\sum_{j=1}^m v(X_j) + v(x)}. \end{aligned} \tag{A9}$$

For data under standard covariate shift, the split conformal confidence set achieves coverage, as first shown by Tibshirani et al. [60].

Theorem 4 (Corollary 1 in [60]). *Suppose calibration and test data 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.* (A7)*, and any miscoverage level, $\alpha \in (0, 1)$, the split conformal confidence set, $C_\alpha^{\text{split}}(x)$, in Eq.* (A8) *satisfies the coverage property in Eq.* (1).

Note that since standard covariate shift is a special case of feedback covariate shift, this result also follows as a corollary of Theorem 2.

In settings under feedback covariate shift with abundant data, Theorem 4 allows us to use the split conformal confidence set in Eq. (A8) to achieve coverage. To achieve exact coverage, we can introduce randomization, as we discuss next.

A1.4 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. (A7), any miscoverage level, $\alpha \in (0,1)$, the *randomized split conformal* confidence set is the following random variable for $x \in \mathcal{X}$:

$$C_\alpha^{\text{rand,split}}(x) = \left\{ y \in \mathbb{R} : S(x,y) \le \textsc{RandomizedQuantile}_{1-\alpha}\left((S_1, \dots, S_m, S(x,y)), (w_1(x), \dots, w_{m+1}(x))\right)\right\}, \tag{A10}$$

where the randomized β-quantile, $\textsc{RandomizedQuantile}_\beta$ is defined in Eq. (A3), $S_i = S(X_i, Y_i)$ for $i = 1, \dots, m$, and $w_i(\cdot)$ for $i = 1, \dots, m+1$ is defined in Eq. (A9). Observe that for each value $y \in \mathbb{R}$, 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 follows as a consequence of Theorem 3.

Corollary 1. *Suppose calibration data, $Z_1, \dots, Z_m$, and a test data point, Z_{m+1}, are drawn under standard covariate shift, and assume $\tilde{P}_X$ is absolutely continuous with respect to P_X. For score functions of the form in Eq.* (A7)*, and any miscoverage level, $\alpha \in (0,1)$, the randomized split conformal confidence set, $C_\alpha^{\text{rand,split}}(x)$, in Eq.* (A10) *satisfies the exact coverage property in Eq.* (A5).

Proof. Since standard covariate shift is a special case of feedback covariate shift, the calibration and test data can be described by feedback covariate shift where $\tilde{P}_{X;D} = \tilde{P}_{X;\mu}$. 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. (A4), 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 $\tilde{P}_{X;D} = \tilde{P}_{X;\mu}$. The result then follows from Theorem 3. □

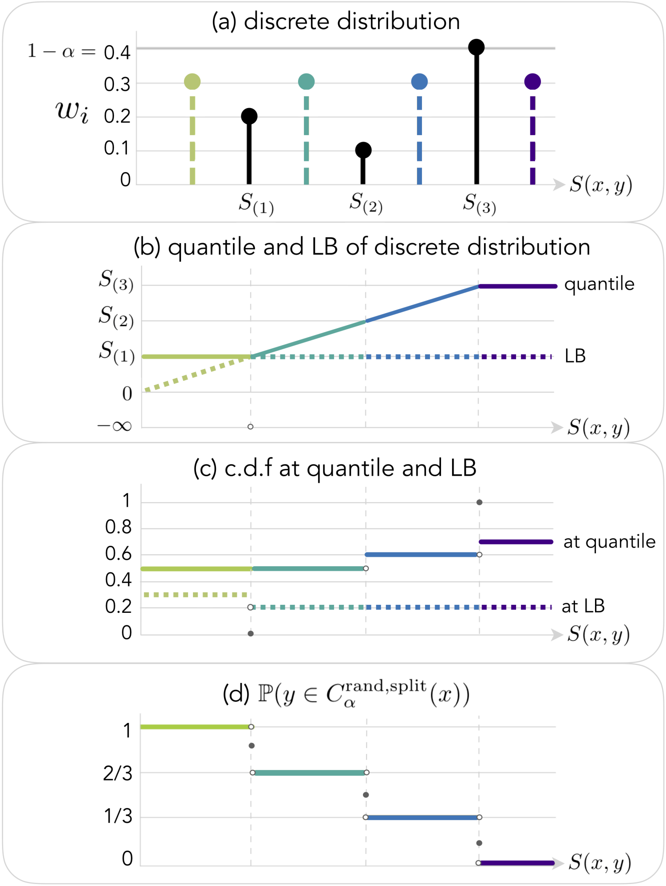
While we only need to fit a single regression model, μ, used to compute the scores, naively it might seem that in practice, we can only approximate $C_\alpha^{\text{rand,split}}(x)$ by introducing a discrete grid of candidate values, $\mathcal{Y} \subset \mathbb{R}$, and computing a randomized β-quantile 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. 2).

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 with the same value of $\mathbb{P}(y \in C_\alpha^{\text{rand,split}}(x))$.

Fig. A1 illustrates this observation, which we now explain. First, the discrete distribution in Eq. (A10) has probability masses $w_1(x), \dots, w_{m+1}(x)$ at the points $S_1, \dots, S_m, S(x,y)$. Given the values of the m calibration data points and the test input, x, all of these quantities are fixed—except for the score $S(x,y)$. 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, $S_1, \dots, S_m$, in sorted 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: $S(x,y) \le \beta$-quantile lower bound (of the discrete distribution with probability masses $w_1, \dots, w_{m+1}$ at support points $S_1, \dots, S_m, S(x,y)$, respectively), $S(x,y) = \beta$-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. A1), while an interval that belongs to the last category is deterministically excluded (color-coded purple in Fig. A1). 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. (A3), that the randomized β-quantile equals the β-quantile. Consequently, we can 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. (A3)). As depicted in Fig. A1 (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



color legend:

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

Figure A1: Depiction of how the probability $\mathbb{P}(y \in C_\alpha^{\text{rand,split}}(x))$ is a piecewise constant function of y. (a) Given the values of the calibration data and test input, the scores $S_1, \dots, S_m$ and corresponding probability masses $w_1, \dots, w_m$ (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. (A3) 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 for the plots (calculations for $S(x,y) = S_{(i)}$ omitted).

at the calibration scores, $S_1, \dots, S_m$ (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 $\hat{\mu}(x)$ and decreases as y moves further

Algorithm 2 Randomized staircase confidence set

Input: Miscoverage level, $\alpha \in (0,1)$; calibration data, $(Z_1, \dots, Z_m)$; test input, X_{m+1}; subroutine for likelihood ratio function, $v(\cdot)$, defined in Eq. (A6); subroutine for uncertainty heuristic, $u(\cdot)$; subroutine for regression model prediction, $\mu(\cdot)$.
Output: Randomized staircase confidence set, $C = C_\alpha^{\text{staircase}}(X_{m+1})$.

```
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
$v_{m+1} \leftarrow v(X_{m+1})$
for $i = 1, \dots, m+1$ do                                      ▷ Compute calibration and test weights
    $w_i \leftarrow \frac{v_i}{\sum_{j=1}^{m+1} v_j}$
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 $\sum_{j: S_j \leq S_{(i)}} w_j + w_{m+1} < 1 - \alpha$ then     ▷ $S(x,y) \leq \beta$-quantile lower bound, so include deterministically
        $C = C \cup \big[\mu(x) + S_{(i)} \cdot u(x), \mu(x) + S_{(i+1)} \cdot u(x)\big] \cup \big[\mu(x) - S_{(i+1)} \cdot u(x), \mu(x) - S_{(i)} \cdot u(x)\big]$
    else if $\sum_{j: S_j \leq S_{(i)}} w_j + w_{m+1} \geq 1 - \alpha$ and $\sum_{j: S_j \leq S_{(i)}} w_j < 1 - \alpha$ then ▷ $S(x,y) = \beta$-quantile, so randomize inclusion
        if LowerBoundIsSet = False then
            LowerBoundIsSet ← True                              ▷ Set $\beta$-quantile lower bound
            $LF = \sum_{j: S_j \leq S_{(i)}} w_j$
        end if
        $F \leftarrow \frac{\sum_{j: S_j \leq S_{(i)}} w_j + w_{m+1} - (1-\alpha)}{\sum_{j: S_j \leq S_{(i)}} w_j + w_{m+1} - LF}$
        $b \sim \text{Bernoulli}(1 - F)$
        if $b$ then
            $C = C \cup \big[\mu(x) + S_{(i)} \cdot u(x), \mu(x) + S_{(i+1)} \cdot u(x)\big] \cup \big[\mu(x) - S_{(i+1)} \cdot u(x), \mu(x) - S_{(i)} \cdot u(x)\big]$
        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
        $C = C \cup \big[\mu(x) + S_{(m)} \cdot u(x), \infty\big] \cup \big[-\infty, \mu(x) - S_{(m)} \cdot u(x)\big]$
    end if
end if
```

away from it, resembling a staircase, as depicted in Fig. A1 (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: $[0, S_{(1)}), (S_{(1)}, S_{(2)}), \dots, (S_{(m-1)}, S_{(m)}), (S_{(m)}, \infty]$, 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. (A7), this set comprises two equal-length intervals on both sides of $\mu(x)$: $(\hat{\mu}(x) - S_{(i+1)}, \hat{\mu}(x) - S_{(i)}) \cup (\hat{\mu}(x) + S_{(i+1)}, \hat{\mu}(x) + S_{(i)})$.

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 $i = 1, \dots, m$, does not affect the exact coverage property. For simplicity, Alg. 2 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. (A7) only at the end of the argument, two paragraphs ago. We can relax this assumption as follows. For any continuous score function, consider the preimage of the intervals $[0, S_{(1)}), (S_{(1)}, S_{(2)}), \dots, (S_{(m-1)}, S_{(m)}), (S_{(m)}, \infty]$ under the function $S(x_{m+1}, \cdot)$ (a function of the second argument with x_{m+1} held fixed), rather than the intervals given explicitly in Lines 14, 23, and 34 of Alg. 2. This algorithm then gives exact coverage for any continuous score function, although it will only be computationally feasible when the preimages can be computed efficiently.

A2 Efficient computation for ridge regression and Gaussian process regression

A2.1 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 constructing the 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. A construction that achieves this is presented in Alg. 3.

As an example of the TestCovariateLikelihood subroutine, for the protein design experiments we computed the likelihood in Eq. (6) in Lines 11 and 15 of Alg. 3 as

$$\begin{aligned} v(X_i; Z_{-i,y}) &\leftarrow \frac{\exp(\lambda \cdot (a_i + b_i y))}{p_X(X_i) \cdot \sum_{x \in \mathcal{X}} \exp(\lambda \cdot (C_i + y\mathbf{A}_{-i,n})^T x)}, \\ v(X_{n+1}; Z_{1:n}) &\leftarrow \frac{\exp(\lambda \cdot a_{n+1})}{p_X(X_{n+1}) \cdot \sum_{x \in \mathcal{X}} \exp(\lambda \cdot \beta^T x)}, \end{aligned} \tag{A11}$$

respectively, where p_X is the likelihood under the training input distribution, and the domain $\mathcal{X}$ is the combinatorially complete set of the protein sequences under consideration (a total of 8192 sequences for the data set used in Section 3.1).

Computing these likelihoods is 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.

Algorithm 3 Efficient computation of scores and weights for ridge regression-based feedback covariate shift

Input: training data, $\{(X_i, y_i)\}_{i=1}^n$; test input, X_{n+1}; grid of candidate label values, $\mathcal{Y} \subset \mathbb{R}$; subroutine, TestCovariateLikelihood$(\cdot)$, that takes in an input outputs its likelihood under the test input distribution.

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

```
 1: for $i = 1, \ldots, n$ do
 2:     $C_i \leftarrow \sum_{j=1}^{n-1} Y_{-i;j} \mathbf{A}_{-i;j}$
 3:     $a_i \leftarrow C_i^T X_i$
 4:     $b_i \leftarrow \mathbf{A}_{-i;n}^T X_i$
 5: end for
 6: $\beta \leftarrow (\mathbf{X}^T\mathbf{X} + \gamma I)^{-1}\mathbf{X}^T Y$
 7: $a_{n+1} \leftarrow \beta^T X_{n+1}$
 8: for $i = 1, \ldots, n$ do
 9:     for $y \in \mathcal{Y}$ do
10:         $S_i(X_{n+1}, y) \leftarrow |Y_i - (a_i + b_i y)|$      ▷ Can vectorize via outer product between $(b_1, \ldots, b_n)$ and vector of all $y \in \mathcal{Y}$.
11:         $v(X_i; Z_{-i,y}) \leftarrow$ TestCovariateLikelihood$(a_i + b_i y)$      ▷ Can vectorize (see commentary on Eq. (A11)).
12:     end for
13: end for
14: $S_{n+1}(X_{n+1}, y) \leftarrow |y - a_{n+1}|$
15: $v(X_{n+1}; Z_{1:n}) \leftarrow$ TestCovariateLikelihood$(a_{n+1})$
```

A2.2 Gaussian process regression

Here we describe how the scores and weights for constructing the confidence set in Eq. (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

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

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. (5); furthermore, the predictive variance is constant in y. Therefore, we can mimic Alg. 3 to efficiently compute scores and weights by training just $n+1$ rather than $(n+1) \times |\mathcal{Y}|$ models.

A3 Details for design experiments using AAV capsid packaging data

A4 Additional results on protein design using combinatorially complete fluorescence data sets

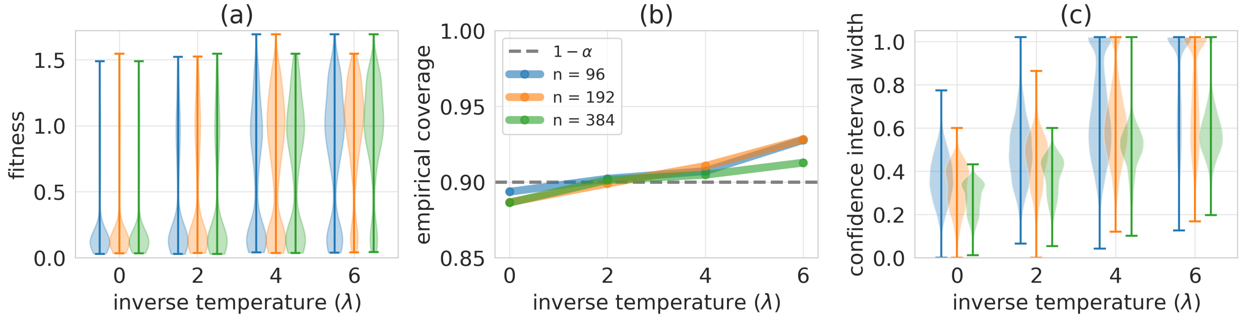


Figure A2: Quantifying the predictive uncertainty for designed proteins, using the red fluorescence data set. (a) Distributions of fitnesses of designed proteins, (b) empirical coverage, compared to the theoretical lower bound of $1 - \alpha = 0.9$ (dashed gray line), and (c) distributions of confidence interval widths for different values of the inverse temperature, λ, and different amounts of training data, n, over $T = 5000$ trials. The interval widths in (c) are reported as a fraction of the range of true fitnesses in the combinatorially complete data set, $[0.025, 1.692]$; widths reported as > 1 signify confidence intervals that contain $[0.025, 1.692]$. In (a), and (c), the whiskers signify the minimum and maximum observed values.

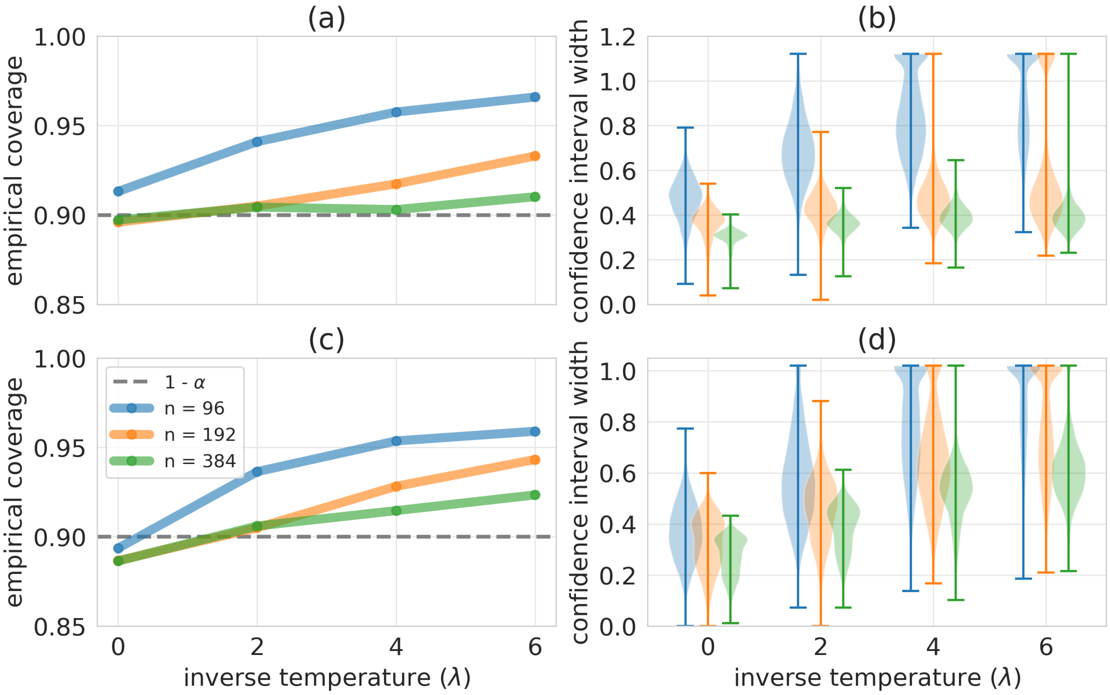


Figure A3: Using conformal prediction for standard covariate shift [60] to quantify predictive uncertainty for designed proteins. (a) Empirical coverage, compared to target coverage of $1-\alpha = 0.9$ (dashed gray line), and (b) distributions of confidence interval widths for different values of the inverse temperature, λ, and different amounts of training data, n, over $T = 5000$ trials for the blue fluorescence data set. Interval widths in (b) are reported as a fraction of the range of true fitnesses in the combinatorially complete data set, $[0.091, 1.608]$, and whiskers signify the minimum and maximum widths. (c, d) Analogous to (a, b), respectively, for the red fluorescence data set. Interval widths in (d) are reported as a fraction of the range of true fitness values in the combinatorially complete data set, $[0.025, 1.692]$, and whiskers signify the minimum and maximum widths.