Forecasting labels under distribution-shift for machine-guided sequence design

Anonymous Author(s)
Affiliation
Address
email

Abstract

The ability to design and optimize biological sequences with specific functionalities would unlock enormous value in technology and healthcare. In recent years, machine learning-guided sequence design has progressed this goal significantly, though validating designed sequences in the lab or clinic takes many months and substantial labor. It is therefore valuable to assess the likelihood that a designed set contains sequences of the desired quality (which often lies outside the label distribution in our training data) before committing resources to an experiment. Forecasting, a prominent concept in many domains where feedback can be delayed (e.g. elections), has not been used or studied in the context of sequence design. Here we propose a method to guide decision-making that forecasts the performance of high-throughput libraries (e.g. containing 10^5 unique variants) based on estimates provided by models, providing a posterior for the distribution of labels in the library. We show that our method outperforms baselines that naively use model scores to estimate library performance, which are the only tool available today for this purpose.

1 Introduction

Biological sequence design has long been of interest to practitioners in many domains, from agriculture to therapeutics. For decades, sequences were designed through two means (i) Labor-intensive rational design where expert human knowledge would generate a handful of candidate sequences [1], (ii) High-throughput directed evolution approaches that utilize biological evolution to optimize sequences towards a desired property [2]. Recently, the ability to synthesize DNA in high-throughput, together with the wide adoption of high-capacity of machine learning models, has opened a new path that can combine the benefits of rational design (high quality), and directed evolution (high throughput) [3, 4, 5]. In this setting, libraries of up to size 10^5 sequences are designed using machine learning algorithms. Machine learning methods are used to score, optimize, and filter sequences before committing to experiments [6, 4, 7]. In recent years, increasingly nuanced perspectives on how to improve our trust in the output of machine learning models and paired optimization procedures have evolved [7, 8, 9]. Using these methods, sequences can be synthesized and used to produce target sequences (e.g. transcription factor binding sequences [10]) in a library that can be measured in the desired context. However, especially with products or traits of high complexity (e.g. *in-vivo* studies of proteins [11]), the overall cost required to validate designs can be prohibitive. Therefore, even with model evaluations and calibration of uncertainty around samples, there remains a gap in our ability to *forecast* the probability of success: be it reaching a certain maximum performance, or finding a certain number of variants above a minimum desired performance. Forecasts can help us decide whether to commit to a certain design and can save large costs. Forecasts can also inform other decisions such as deciding whether to repeat the design procedure for a library, deciding among libraries designed for different targets, or estimating the final price of developing a drug.

Preprint. Under review.

Forecasting is ubiquitous in domains with delayed feedback such as elections [12, 13]. The related topic of label shift [14, 15] classically relies on the "anticausal" assumption that the distribution of inputs given labels is constant across training and test sets - an assumption that is invalid in the case of design. More generally, domain adaptation has been studied in biological sequence design [12, 16] but does not directly address forecasting and calibrating distributions under covariate and label shift. Recent work in conformal prediction directly tackles the problem of the kind of covariate shift that arises in sequence design settings [9], but its use requires known probability distributions over training and test sets, and nonzero prior probabilities on the entire support, meaning it cannot be applied to most libraries that were not designed with this approach in mind. To our knowledge, there are currently no methods that are suitable for forecasting library performance in the sequence design setting. This setting presents an interesting and somewhat unique challenge. For every designed sequence we can obtain scores for the expected performance, possibly from multiple models. However we are often aiming to make sequences that have a significantly higher score than anything observed in our training data, i.e. distribution shift is by design. Our challenge is to find the right balance between trusting our models' predictions out-of-distribution and betting that our new designs would provide us with better-than-observed sequences.

2 Forecasting method

We start with labeled training data $(\mathbf{S}^0, \mathbf{Y}^0)$, where $\mathbf{S}^0 = \{s_i^0\}$ is a set of biological sequences and $\mathbf{Y}^0 = \{y_i^0\}$ is a set of continuous-valued labels, generally a fitness measurement in the sequence design setting such as packaging or transduction efficiency rates. Our goal is to forecast a distribution of labels $\mathbf{Y}^1$ for an unlabeled set $\mathbf{S}^1$. That is, we are not concerned with the accuracy of each pair (s_i^1, y_i^1), but only the overall distribution of $\mathbf{Y}^1$, and in particular, in the right tail of $\mathbf{Y}^1$, which indicates the maximum quality achieved by the set of sequences. To create our forecast, we have at our disposal a set of J regression models trained on $(\mathbf{S}^0, \mathbf{Y}^0)$, which produce test set predictions m_{ij} for sequence i by model j.

Naively, we could form ensembled point estimates $\hat{y}_i = \sum_j m_{ij}/J$ for each sequence and predict the distribution of $\mathbf{Y}^1$ to be the distribution of $\hat{y}_i$. There are three main disadvantages to this approach, which inspire different aspects of our forecasting method. We address them briefly below, and give a more thorough treatment with a complete algorithm in Section 5.1.

1) An implicit unimodal Gaussian assumption Empirically, model ensembles tend towards unimodal Gaussian score distributions which do not empirically fit experimental data from designed sequences well. At several points in the experimental pipeline variants may "drop out", failing to produce enough signal to reliably approximate a label (for example, due to failure of a protein to fold). This results in a multimodal distribution at both the population level, and implicitly, at the level of each sequence's posterior. Thus, we seek to model each sequence as a bimodal Gaussian Mixture Model (GMM), and learn the parameters for each sequence's posterior from its model scores. Explicitly, given an i.i.d. set of model predictions m_{ij} for sequence i, we seek to infer a probability p_i that distinguishes between the two distributions in the Gaussian mixture, as well as a mean and variance parameter for each Gaussian mode. Moreover, while we use a GMM, our method could in theory be applied using a range of more complex distributions, with the only constraint being our ability to sample from them.

2) Distribution and label shift Typically, the sequence set $\mathbf{S}^1$ is designed with model-guided exploration strategies informed by $(\mathbf{S}^0, \mathbf{Y}^0)$, with the objective of producing sequences that outperform the best sequences in $\mathbf{S}^0$. This results in both significant distribution (covariate) shift, because the sequences $\mathbf{S}^1$ are reaching into untested areas of sequence space, and label shift, since we anticipate that $\mathbf{Y}^1$ will dominate $\mathbf{Y}^0$, both on average and among each sets top-performers. (While there has been some important recent work on prediction in this design setting [9], this work assumes a shift in distribution within a consistent domain between $\mathbf{S}^0$ and $\mathbf{S}^1$ and no label shift.) To address distribution and label shift, we start by applying non-parametric non-linear calibration techniques to produce a "conservative" forecast that still allows for some label shift due to model uncertainty. We then consider scenarios with some trust placed in raw model scores to allow for some amount of extrapolation to regions further from our training set.

3) Point estimates to posteriors The point estimates that arise from model ensembles do not provide a posterior for $\mathbf{Y}^1$ (nor do the model score variances, directly), and consequently these

tend to underestimate the frequency of events that are rare at the sequence level, but common at the population level, such as the occurrence of high-valued sequences in the library. In our method, we simulate draws of the entire library from the sequence-level posteriors to produce both expected distributions as well as the frequency of rare events, which we interpret as posterior probabilities.

3 Experimental results

We validate our method by conducting experiments on four datasets - a set of simulated RNA binding landscapes that allow us to access ground truth values for every sequence (and repeat multiple experiments) as well as three experimentally-measured assays of protein fitness landscapes. These are a viral protein packaging landscape, an experimentally measured IgG-Fc binding dataset for Protein G's GB1 domain, and an experimentally measured GFP fluorescence dataset. Of the experimentally measured landscapes, the viral protein assay is precisely conducted in the manner that forecast is intended to be used. The GFP and GB1 landscapes are not conducted in this way, but we still expect (and observed) forecast to outperform model point estimates. (See Section 5.2 for descriptions of these datasets.)

For each of these experiments, we generated training and test sets $\mathbf{S}^0$, $\mathbf{Y}^0$ and $\mathbf{S}^1$, $\mathbf{Y}^1$, and models $\mathbf{M}$. We used the training set, models, and unlabeled test data to generate forecasts, and evaluated them against realized distributions of $\mathbf{Y}^1$ using two key tools: a 2-sample Kolmogorov-Smirnov statistic measuring distribution fit of top percentiles, and confidence interval coverage for counts of points measured above a fixed threshold value. Both evaluate fit in the right tail of the distribution, which is both the most challenging region to predict, and the one that is most critical to sequence design applications. We present results of forecasts on all four datasets in Figure 1, with one dataset per row of figures (RNA, AAV, GB1, and GFP respectively).

In the left column of figures we report on the Kolmogorov-Smirnov fit between the true distribution of scores $\mathbf{Y}^1$ and either the forecast or the set of point estimates μ_i from model ensemble scores. In figure 1a we plot the mean and 95% confidence interval across landscapes and starts (see Sec 5.2.1 for details), while the other experiments have a single landscape each. Since we are primarily concerned with fit in the right tail, we limited each distribution to values above a percentile threshold, and varied that threshold between 50% and 95%. Across this range, the forecasting method improved distributional fit compared to ensemble point estimates, and while point estimates typically decayed towards 1.0 (the theoretical worst-case upper bound of the statistic), the forecast consistently maintained some predictive power even in the top 5th percentile. A sharp covariate shift in the AAV capsid design sets $\mathbf{S}^0$ and $\mathbf{S}^1$ account for the problem difficulty in figures 1e and 1f, though even in this problem our method directionally improves upon model estimates.

In the right column of figures we focus more closely on the right tail of the label distribution of $\mathbf{Y}^1$, reporting the forecast's confidence interval for the number of sequence we can expect to find above a threshold value compared to the estimate from model ensembles and the true counts. Since Figure 1b encompasses several landscapes and seeds, we set one threshold per landscape/seed at the 99th percentile, while in the remaining experiments with a single landscape we evaluate accuracy over a range of thresholds. Here we see that the forecast gives confidence intervals that include the true count of sequences above a high threshold some of the time, and always improves upon the ensemble estimates, which for most datasets severely underpredicts the prevalence of top-performers.

4 Conclusion

Here we argued for the relevance and impact of forecasting in the sequence design setting. We developed a novel approach for forecasting label distributions under covariate and label shift. Our approach can be used on any machine-guided library design for which we have regression models. We applied these methods in simulated and real-world sequence design settings and showed near-universal improvement (and never worse than the naive approach) in our ability to predict the shape of the right tail and counts of top performers. This work enables valuable estimation of the quality of designed libraries of biological sequences long before experimental results can be produced, which provides essential feedback to the designer. We hope that by defining this problem framework, and showcasing an approach to address it, we inspire further development for improving distributional forecasting in the model-guided sequence design setting.

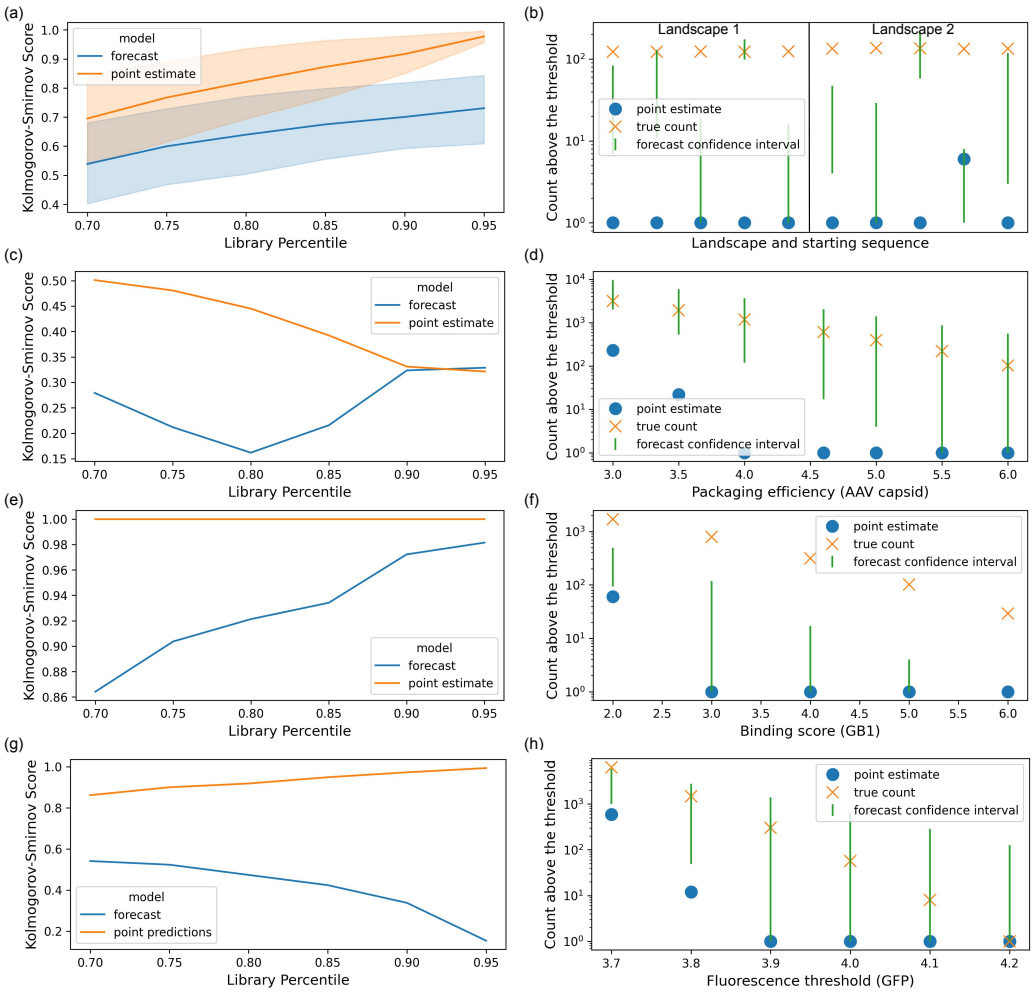

Figure 1: Comparison between forecast and point predictions in describing the right tail statistics of designed libraries. a) Distance from true (right-tail) distribution as measured by KS two-sample score for ensemble point estimate and our forecast on two RNA landscapes and five distinct designs per landscape (10 total forecasts) b) Top centile confidence interval coverage for RNA landscape 1 (14 nt) and RNA landscape 2 (50 nt) c,e,g) Distance from true distribution fit for ensemble point estimate and forecast d,f,h) Confidence interval coverage based on the number of samples above a certain measured performance c,d) For the AAV capsid design problem e,f) For the GB1 binding landscape g,h) For the GFP florescence landscape.

References

[1] HW Hellinga. Rational protein design: combining theory and experiment. *Proceedings of the National Academy of Sciences*, 94(19):10015–10017, 1997.

[2] Frances H Arnold. Design by directed evolution. *Accounts of chemical research*, 31(3):125–131, 1998.

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

[4] 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. *Nature Biotechnology*, 39(6):691–696, 2021.

[5] Ethan C Alley, Grigory Khimulya, Surojit Biswas, Mohammed AlQuraishi, and George M Church. Unified rational protein engineering with sequence-based deep representation learning. *Nature methods*, 16(12):1315–1322, 2019.

[6] 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. *Nature communications*, 12(1):1–11, 2021.

[7] David H Brookes, Hahnbeom Park, and Jennifer Listgarten. Conditioning by adaptive sampling for robust design. *arXiv preprint arXiv:1901.10060*, 2019.

[8] Christof Angermueller, David Belanger, Andreea Gane, Zelda Mariet, David Dohan, Kevin Murphy, Lucy Colwell, and D Sculley. Population-based black-box optimization for biological sequence design. In *International Conference on Machine Learning*, pages 324–334. PMLR, 2020.

[9] Clara Fannjiang, Stephen Bates, Anastasios Angelopoulos, Jennifer Listgarten, and Michael I Jordan. Conformal prediction for the design problem. *arXiv preprint arXiv:2202.03613*, 2022.

[10] Luis A Barrera, Anastasia Vedenko, Jesse V Kurland, Julia M Rogers, Stephen S Gisselbrecht, Elizabeth J Rossin, Jaie Woodard, Luca Mariani, Kian Hong Kock, Sachi Inukai, et al. Survey of variation in human transcription factors reveals prevalent dna binding changes. *Science*, 351(6280):1450–1454, 2016.

[11] 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, 2019.

[12] Patrick Hummel and David Rothschild. Fundamental models for forecasting elections. *ResearchDMR. com/HummelRothschild_FundamentalModel*, 2013.

[13] Wei Wang, David Rothschild, Sharad Goel, and Andrew Gelman. Forecasting elections with non-representative polls. *International Journal of Forecasting*, 31(3):980–991, 2015.

[14] Zachary Lipton, Yu-Xiang Wang, and Alexander Smola. Detecting and correcting for label shift with black box predictors. In *International conference on machine learning*, pages 3122–3130. PMLR, 2018.

[15] Marco Saerens, Patrice Latinne, and Christine Decaestecker. Adjusting the outputs of a classifier to new a priori probabilities: a simple procedure. *Neural computation*, 14(1):21–41, 2002.

[16] Shireen Abestesh, Ilke Akartuna, Alexander Brown, Megan Cramer, Farhan Damani, Jeff Gerold, Jorma Gorns, Jeff Jones, Helene Kuchwara, Jamie Kwasnieski, et al. Efficient design of optimized AAV capsids using multi-property machine learning models trained across cells, organs and species. In *MOLECULAR THERAPY*, volume 29, pages 12–12. CELL PRESS, 2021.

[17] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*, 9(1):62–66, 1979.

[18] Jerome Friedman and Robert Tibshirani. The monotone smoothing of scatterplots. *Technometrics*, 26(3):243–250, 1984.

[19] 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. *arXiv preprint arXiv:2010.02141*, 2020.

[20] Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2):159–195, 2001.

[21] Colin A Smith and Tanja Kortemme. Predicting the tolerated sequences for proteins and protein interfaces using rosettabackrub flexible backbone design. *PloS one*, 6(7):e20451, 2011.

[22] Sebastian Kmiecik and Andrzej Kolinski. Folding pathway of the b1 domain of protein g explored by multiscale modeling. *Biophysical journal*, 94(3):726–736, 2008.

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

[24] Karen S Sarkisyan, Dmitry A Bolotin, Margarita V Meer, Dinara R Usmanova, Alexander S Mishin, George V Sharonov, Dmitry N Ivankov, Nina G Bozhanova, Mikhail S Baranov, Onuralp Soylemez, et al. Local fitness landscape of the green fluorescent protein. *Nature*, 533(7603):397–401, 2016.

5 Supplemental Material

5.1 Detailed description of the forecasting method

To generate forecasts, we first transform model predictions for each sequence into parameters for their posterior distributions. We then draw from those distributions to form simulations of the library. Our objective is to infer these posterior parameters from model predictions in a way that is both well-calibrated to the training data and allows for test set performance of the sequences to differ from or exceed training set performance due to distribution shift.

5.1.1 Fitting sequence label posteriors

While this forecasting framework can be applied to learn posteriors for many distributions families, informed by empirical library label distributions from historical experiments, we believe the natural distribution for labels in our data is a Gaussian mixture model (GMM) with two modes: one for "functional" sequences and one for "non-functional" or "broken" sequences. That is, given some sequence s_i, we assume there is a probability of functionality p_i, a mean and variance in the functional mode $\left(\mu_i^+, (\sigma_i^+)^2\right)$, and a mean and variance in the non-functional mode $\left(\mu_i^-, (\sigma_i^-)^2\right)$ that parameterize normal distributions $\mathcal{N}$ so that

$$Y_i \sim p_i \mathcal{N}(\mu_i^+, \sigma_i^+) + (1 - p_i)\mathcal{N}(\mu_i^-, \sigma_i^-). \tag{1}$$

In contrast, our predictive models only provide point estimates m_{ij} for each sequence. We assume that the true, multimodal distribution of each sequence can be summarized with two degrees of freedom (a mean μ_i and standard deviation σ_i) and that these two parameters independently generate model scores, mixture model parameters, and measurement values. Explicitly, we assume the model predictions $m_{ij} \sim \mathcal{N}(\mu_i, \sigma_i)$, so that, given a set of model predictions, we infer μ_i and σ_i to be the models' sample mean and variance ($\mu_i = 1/J \sum_j m_{ij}$ and $\sigma_i^2 = 1/J \sum_j (m_{ij} - \mu_i)^2$). We further assume there are independent relationships between μ_i and the set (p_i, μ_i^+, μ_i^-), and between σ_i and the pair σ_i^+ and σ_i^-.

Since the GMM parameters $(p_i, \mu_i^+, \mu_i^-, \sigma_i^+, \sigma_i^-)$ are unique to each sequence, we cannot infer them in the usual manner using $\mathbf{S}^0$ as a training set. Instead, we need to further model and learn the relationship between the pair μ_i, σ_i and the GMM parameters. Specifically, we start by identifying the value y_{mid} that we use to separate the two modes of $\mathbf{Y}$. This value can either be set manually, using expert knowledge, or automatically by analyzing the distribution $\mathbf{Y}^0$. We found that Otsu's method [17], which finds the separating point that minimizes intra-class variance, provided robust values of y_{mid} on our data. We can then divide our set $\mathbf{S}^0$ into two halves across the boundary: $\mathbf{S}^{0+} = \{s_i \mid y_i^0 \geq y_{mid}\}$ and $\mathbf{S}^{0-} = \{s_i \mid y_i^0 < y_{mid}\}$. This provides us with separate training sets for the functional parameters μ_i^+, σ_i^+ and the non-functional parameters μ_i^-, σ_i^-.

5.1.2 About isotonic regression

We can use isotonic regression to find the best monotonic piece-wise linear fit to this data. Explicitly, isotonic regression operates on a dataset of pairs of scalars $\{x_i, y_i\}$ and produces a non-parametric model represented by data-prediction pairs $(x_i, \hat{y}_i)$ that seek to minimize the least squares error

$\sum_i(\hat{y}_i - y_i)^2$ subject to a monotonicity constraint $y_i \leq y_j \ \forall i, j$ s.t. $x_i < x_j$. This results in a quadratic program, though it is easily solvable exactly by sorting y_i and iteratively averaging pairs of "violators" of monotonicity, making training efficient and deterministic [18].

We will use the abbreviation IR_y to refer to an isotonic regression model trained to predict y defined by the pairs $(x_i, \hat{y}_i)$, and $IR_y(x'_i)$ to be the prediction of this model given input x'_j. To compute $IR_y(x'_j)$ on a new data point, first we check to see if $x'_j = x_i$ for some i and if so we predict the corresponding $IR_y(x'_j) = \hat{y}_i$. Otherwise we sort x_i and find a consecutive pair such that $x_i < x'_j < x_{i+1}$ and predict $IR_y(x'_j)$ by linearly interpolating between $\hat{y}_i$ and $\hat{y}_{i+1}$. If x'_j is less than all x_i, or more than all x_i, $IR_y(x'_j)$ is set to the min and max values of $\hat{y}_i$ respectively. (We note here that this necessarily means that isotonic regression will never produce labels outside the range of the training labels - we will address this in a few ways in the coming sections.)

5.1.3 Inferring parameters with isotonic regression

We assume a non-linear monotonic relationship between the model ensemble mean for a sequence $\mu_i = 1/J \sum_j \mu_{ij}$ and the likelihood of that sequence functioning p_i. To infer p_i, we train an isotonic regression model that, given μ_i, aims to predict the indicator I_i^+ which is 1 if $s_i > y_{mid}$ and 0 otherwise. Effectively, given a new input μ_i, this model returns the fraction of sequences that are functional out of the training samples with similar mean values, and interprets this rate as the probability that the sequence i will be functional.

Inferring the mean parameters μ_i^+ and μ_i^- is more straight-forward: we build isotonic regression models to predict y_i from μ_i, but restrict the training set to $\mathbf{S}^{0+}$ for the μ_i^+ model and to $\mathbf{S}^{0-}$ for the μ_i^- model. This gives us calibration to the conditional distributions for being functional and non-functional respectively.

To infer the variances σ_i^{2+} and σ_i^{2-}, we first form squared residuals of labels given model ensemble means $res_i = (y_i - \mu_i)^2$ and build isotonic models relating the model variance to these residuals res_i. As with μ_i^+ and μ_i^-, we compute σ_i^{2+} and σ_i^{2-} by training models on the disjoint training sets $\mathbf{S}^{0+}$ and $\mathbf{S}^{0-}$ respectively. The complete algorithm for inferring the GMM parameters is described in Algorithm 1.

5.1.4 Simulating the posterior distribution

Given the parameters generated by Algorithm 1, we can draw samples $\hat{y}_i^1$ for each sequence, and aggregate them into draws for the entire distribution $\hat{\mathbf{Y}}^1$. We can then treat the set of simulated values of $\hat{\mathbf{Y}}^1$ as a posterior distribution and query this distribution to determine the frequency of distribution-level events. By computing metrics on $\hat{\mathbf{Y}}^1$ and considering their distributions across simulations, we can arrive at empirical confidence intervals for metrics such as the count of sequences that perform above some threshold value, as we see in Figures 1b,d,f,h.

5.1.5 Tuning the forecast from conservative to optimistic

We can further refine this basic algorithm using additional techniques that allow us to diversify our approach over degrees of trust in our training set.

Semi-calibrated regression Our main calibration tool, isotonic regression, aggressively limits predicted labels to be within the range of training values. To allow for some distribution shift, we can gradually transition from calibrated predictions towards the center of the distribution of $\mathbf{S}^1$ towards uncalibrated, out-of-distribution values towards the limits of the distribution, in a technique we call "semi-calibration."

Let $P_Y(y)$ be the percentile of the value y from among the empirical distribution Y. That is, $P(y)$ is the fraction of $y \in Y$ with $y_i < y$. In our case, we consider the distribution of model ensemble means on our training set $\mathbf{S}^0$, that is, the set $M = \{1/J \sum_j m_{ij} \mid s_i \in \mathbf{S}^0\}$. Then given a new sequence s_i we can compute its model ensemble prediction μ_i as well as its functional isotonically calibrated mean μ_i^+, and evaluate where its model ensemble falls relative to the training distribution by computing the percentile $P_M(\mu_i)$. Finally, for any temperature-like coefficient $0 < q \leq 1$, we

Algorithm 1 Inferring Gaussian mixture model parameters from a set of normally distributed model scores

Input: a training set ($\mathbf{S}^0$, $\mathbf{Y}^0$) and test set ($\mathbf{S}^1$) with model values m_{ij} for each $s_i \in \mathbf{S}^0 \cup \mathbf{S}^1$.
Returns: $(p_i, \mu_i^+, \mu_i^-, \sigma_i^+, \sigma_i^-)$ for each $i \in \mathbf{S}^1$
Learn cutoff value y_{mid} from $\mathbf{Y}^0$ using Otsu's method
for s_i **in** S^0 **do**
  Compute $\mu_i = \frac{\sum_j m_{ij}}{J}$ (model ensemble means)
  Compute $\sigma_i^2 = \frac{\sum_j (m_{ij} - \mu_i)^2}{J}$ (model ensemble variance)
  Compute $res_i^2 = (y_i - \mu_i)^2$ (squared residuals of model ensemble means)
  Compute $I_i^+ = 1$ if $Y_i^0 < y_{mid}$ and 0 otherwise
end for
Define $S^{0+} = \{i \mid Y_i^0 \geq y_{mid}\}$ (training subset for "functional" sequences)
Define $S^{0-} = \{i \mid Y_i^0 < y_{mid}\}$ (training subset for "broken" sequences)
Train isotonic model IR_p on pairs (μ_i, I_i^+) for $i \in S^0$
Train isotonic model IR_{μ^+} on (μ_i, y_i) for $i \in S^{0+}$
Train isotonic model IR_{μ^-} on (μ_i, y_i) for $i \in S^{0-}$
Train isotonic model IR_{σ^+} on (σ_i^2, res_i^2) for $i \in S^{0+}$
Train isotonic model IR_{σ^-} on (σ_i^2, res_i^2) for $i \in S^{0-}$
for s_i **in** S^1 **do**
  Compute $\mu_i = \frac{\sum_j m_{ij}}{J}$ (model ensemble means)
  Compute $\sigma_i^2 = \frac{\sum_j (m_{ij} - \mu_i)^2}{J}$ (model ensemble variance)
  Compute $(p_i, \mu_i^+, \mu_i^-, \sigma_i^+, \sigma_i^-) = (IR_p(\mu_i), IR_{\mu^+}(\mu_i), IR_{\mu^-}(\mu_i), IR_{\sigma^+}(\sigma_i), IR_{\sigma^-}(\sigma_i))$
end for
return $\{(p_i, \mu_i^+, \mu_i^-, \sigma_i^+, \sigma_i^-)\}$

define our semi-calibrated mean $\tilde{\mu}_i(q)$ to be

$$\tilde{\mu}_i(q) = (qP_M(\mu_i))(\mu_i) + (1 - qP_M(\mu_i))\,(IR_{\mu_i^+}(\mu_i)). \tag{2}$$

Thus, lower values will be completely calibrated to the training set, while higher values will be a mix of calibrated and uncalibrated values. Note that we only produce this correction for functional mean values μ_i^+, as we expect non-functional values to be fully in the training set distribution. This leads to an update to the model from Equation 1:

$$\hat{y}_i \sim p_i \mathcal{N}(\tilde{\mu}_i(q), \sigma_i^+) + (1 - p_i)\mathcal{N}(\mu_i^-, \sigma_i^-). \tag{3}$$

Correcting for covariate shift In addition to model score distribution shift, we also see covariate shift that creates model score bias. In our context, we consider edit distance to wild type the primary such covariate. To correct for this shift, we form *signed* residuals from the training set between the *calibrated* μ_i^+ values and the true values y_i (i.e. $(\mu_i^+ - y_i)$. (If we are also applying the semi-calibration technique from the last paragraph, we use $(\tilde{\mu}_i - y_i)$ instead.) We can regress those residuals on edit distance (ED_i) using either isotonic or linear regression, and apply this correction back to the mean prediction μ_i^+. We can also apply this approach to adjust the probability parameter p_i, encoding the understanding that sequences are less likely to be functional at higher distances from the wild-type. That is, we compute:

$$\begin{aligned} res_i &= \mu_i^+ - y_i \\ IR^{ED} &= \text{ IR model trained on } \{(ED_i, res_i)\} \\ \mu_i^{ED} &= \mu_i^+ - IR^{ED}(ED_i) \\ \hat{y}_i &\sim p_i \mathcal{N}(\mu_i^{ED}, \sigma_i^+) + (1 - p_i)\mathcal{N}(\mu_i^-, \sigma_i^-). \end{aligned} \tag{4}$$

Optimistic model de-ensembling So far, we have assumed model scores m_{ij} are drawn from a Gaussian distributions parameterized by μ_i, σ_i. Alternatively, we could assume that each model j represents a distinct distribution, and that μ_i are drawn from these distributions with equal probability.

Using this approach, in each simulation we first randomly select one model independently for each sequence and use that model's prediction as the sequence's expected value: $\mu_i \, m_{ij}$. This can result in more optimistic forecasts when scores have high inter-model variance.

Hedging against calibration assumptions Together, these calibration techniques create a menu of options that allow us to build forecasts that range from conservative to optimistic given input data. Given a set of calibration strategies, we can simulate instances of $\mathbf{Y}^1$, and by aggregating simulations across frameworks, we can form a posterior for the distribution of $\mathbf{Y}^1$ that captures our uncertainties at the sequence level, the model level, and the overall forecasting approach level.

5.2 Descriptions of experimental data

5.2.1 Simulated RNA landscapes

Our first set of experiments investigates the performance of forecasting approach using FLEXS [19], a simulation environment for sequence design which gives access to ground-truth and model-approximated fitness landscapes. We study design problems on two RNA landscapes with a hidden binding target of size 14 and 50 nucleotides. Each training set $\mathbf{S}^0$ was constructed by mutating a sequence from a starting seed (5 seeds for each landscape) with between 1 and 3 mutations per sequence on average. We trained four kinds of predictive models on one-hot encoded sequences: linear regressions, convolutional neural networks, multi-layer perceptions, and random forest models. We used four exploration algorithms to design sequences using these models $\mathbf{S}^1$: CMA-ES [20], CbAS [7],Adalead [19], and random sampling.

5.2.2 In-vitro AAV packaging assay

Bryant et al. [4] quantitatively assay the packaging efficiency of 200K viral capsid variants, modified in a 28 amino-acid region of the protein. The experiment was designed in two steps, where first a smaller set of training examples were assayed and used to train *classification* models. Then these models were used to design a set of variants that were optimized for the probability of packaging. The paper uses classification models and greedy optimization to generate the second batch. The existence of this first training set, and the distribution-shifted designed set is exactly the setting we have devised the our forecasting method for, and where its performance can be most meaningfully evaluated.

For this data set we retrain *regression* models on the first set of sequences ($\mathbf{S}^0$) using five independently seeded convolutional neural networks, and five recurrent neural networks. We used these models to generate ensembled point estimates as well as forecast the distribution of packaging efficiencies on the model-designed set of sequences ($\mathbf{S}^1$).

5.2.3 Protein G GB1 IgG-Fc binding domain

A region of four amino acids in Protein G (GB1) is known to be critical for IgG-Fc binding and has been used extensively as a tool for evaluating sequence landscape prediction and design tasks [21, 22]. Our data is sourced from experiments by Wu et al. [23]. We created $\mathbf{S}^0$ by selecting sequences with performance below that of the wild-type, combined with a small fraction of sequences between wild-type and the median performance in the set, leaving all other sequences for $\mathbf{S}^1$. We trained five independently seeded random forest models and five multi-layer perceptrons on $\mathbf{S}^0$ (forgoing more sophisticated models due to the short length of the variable sequence), and used these to generate point estimates and forecast predictions for $\mathbf{S}^1$.

5.2.4 Green Fluorescent Protein

The green fluorescent protein of Aequorea victoria (avGFP) provides an additional fitness landscape for studying sequence prediction and design. We used a dataset of 540,250 protein variants and their associated fluorescence level [24]. We followed the same procedure as GB1 for splitting $\mathbf{S}^0$ and $\mathbf{S}^1$, putting sequences with fluorescence below 3x log-fluorescence of the wild-type and a small portion of sequences up to the median fluorescence above this threshold into $\mathbf{S}^0$, and the rest into $\mathbf{S}^1$. We then trained the same models as in our AAV experiments - five independently seeded convolutional neural networks, and five recurrent neural networks, and generated point estimates and forecasts for $\mathbf{S}^1$ using these models.

5.3 Computational details

5.3.1 Compute

All of our experiments were run using a single server with a single GPU running in GCP (Google Cloud Platform). We used an Nvidia V100 for training models on the GFP landscape and an Nvidia K80 for the other three experiments' model training.

5.3.2 Hyperparameters

Across all of our experiments, we used five model architectures: convolutional neural networks (CNNs), recurrent neural networks (RNNs), multi-layer perceptrons (MLPs), linear models, and random forests. Linear models and random forests were initialized with default parameters using the sklearn library. CNNs used 32 filters, 64 filters, and 256 filters with 1, 2, and 2 convolutional layers followed by 1, 2, and 2 hidden layers of width 32, 64, and 64 for the AAV, RNA, and GFP experiments respectively. RNNs used embeddings of size 32 combined with 1 and 2 recurrent layers, then followed by 1 hidden layer of size 56 and 128 for AAV and GFP respectively. MLPs used 1 and 3 hidden layers of width 50 and 32 for GB1 and RNA experiments respectively. All three model architectures were trained used Adam with a learning rate of 1e-3 across experiments.

5.3.3 Licenses

FLEXS is open source and Apache licensed. All other code was written for this project in python using common packages that use BSD, PSFL, Apache, and MIT licenses.