Conditioning by adaptive sampling for robust design

David H. Brookes [1] **Hahnbeom Park** [2 3] **Jennifer Listgarten** [4]

Abstract

We present a method for design problems wherein the goal is to maximize or specify the value of one or more properties of interest (*e. g.,* maximizing the fluorescence of a protein). We assume access to black box, stochastic "oracle" predictive functions, each of which maps from design space to a distribution over properties of interest. Because many state-of-the-art predictive models are known to suffer from pathologies, especially for data far from the training distribution, the design problem is different from directly optimizing the oracles. Herein, we propose a method to solve this problem that uses model-based adaptive sampling to estimate a distribution over the design space, conditioned on the desired properties.

1. Predictive-model based design

The design of molecules and proteins to achieve desired properties, such as binding affinity to a target, has a long history in chemistry and bioengineering. Historically, design has been performed through a time-consuming and costly iterative experimental process, often dependent on domain experts using a combination of intuition and trial and error. For example, a popular technique is *Directed Evolution* (Chen & Arnold, 1991), where first random variations in a parent population of proteins are induced; next, for each of these variants, the property of interest or a proxy to it, is measured; finally, the procedure is repeated on the top-performing variants and the process is iterated. This process is dependent on expensive and time-consuming laboratory measurements. Moreover, it is performing a hybrid greedy-random (uniform) walk through the protein design space—a strategy that is not likely to be particularly efficient in finding the best protein.

[1]Biophysics Graduate Group, UC Berkeley, CA [2]Department of Biochemistry, University of Washington, Seattle, WA [3]Institute for Protein Design, University of Washington, Seattle, WA [4]EECS Department, UC Berkeley, CA. Correspondence to: David Brookes, Jennifer Listgarten <{david.brookes, jennl}@berkeley.edu>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

Advances in biotechnology, chemistry and machine learning allow for the possibility to improve such design cycles with computational approaches (*e. g.,* Schneider & Wrede (1994); Schneider et al. (1998); Gómez-Bombarelli et al. (2018); Killoran et al. (2017); Brookes & Listgarten (2018); Gupta & Zou (2019); Yang et al. (2018)). Where exactly should computational methods be used in such a setting? Within the aforementioned *Directed Evolution*, the obvious place to employ machine learning is to replace the laboratory-based property measurements with a property 'oracle'—a regression model that bypasses costly and time-consuming laboratory experiments. However, a potentially higher impact innovation is to also replace the greedy-random search by a more effective search. In particular, given a probabilistic oracle for a property of interest, one can achieve more effective search by employing state-of-the-art optimization algorithms over the inputs (*e. g.,* protein sequence) of the oracle regression model, ideally while accounting for uncertainty in the oracle (*e. g.,* Brookes & Listgarten (2018)).

In the framing of the problem just described, there is an implicit assumption that the regression oracle is well-behaved, in the sense that it is trustworthy not only in and near the regime of inputs where it was trained, but also beyond. However, it is now well-known that many state-of-the-art predictive models suffer from pathological behaviour, especially in regimes far from the training data (Szegedy et al., 2014; Nguyen et al., 2015). Methods that optimize these predictive functions directly may be led astray into areas of state space where the predictions are unreliable and the corresponding suggested sequences are unrealistic (*e. g.,* correspond to proteins that will not fold). Therefore, we must modulate the optimization with prior information to avoid this problem. There are two related viewpoints on what this prior information represents: either as encoding knowledge about the regions where the oracle is expected to be accurate (*e. g.,* by representing information about the distribution of training inputs), or as encoding knowledge about what constitutes a 'realistic' input (*e. g.,* by representing proteins known to stably fold). Herein we focus on the former viewpoint, and thus assume that we have access to the input distribution of our oracle training data. However, the latter viewpoint is required when such data is not available.

How then should one use such prior knowledge? Formally,

for inputs, $\mathbf{x}$ and property of interest y, we should model the joint probability $p(\mathbf{x}, y) = p(y|\mathbf{x})p(\mathbf{x})$, and then perform design (*i. e.,* obtain one or more sequences with the desired properties) by sampling from the conditional distribution. For example, in the case of maximizing one property oracle, we should sample from $p(\mathbf{x}|y \geq y_{\text{max}})$ to obtain our desired designed sequences. More generally, one conditions on the appropriate desired event $p(\mathbf{x}|S)$, where S is the conditioning event). To achieve this conditioning, we will assume that we have access to a property oracle, $p(y|\mathbf{x})$, which may be a black box function. We also assume that our prior knowledge has been encoded in $p(\mathbf{x})$. The prior density, $p(\mathbf{x})$, can be modelled by training an appropriate generative model, such as a Variational Auto-Encoder (VAE) (Kingma & Welling, 2014), Real NVP (Dinh et al., 2017), an HMM (Baum & Petrie, 1966) or a Transformer (Vaswani et al., 2017; Rives et al., 2019), on the chosen set of 'realistic' examples. Recently some generative models have themselves been shown to exhibit pathologies (Nalisnick et al., 2019), however we have not observed any such phenomenon in our setting.

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

2. Related work

The problem set-up just described has a strong similarity to that of *activation-maximization* (AM) with a Generative Adversarial Network (GAN) prior (Goodfellow et al., 2014; Simonyan et al., 2013; Nguyen et al., 2016), which is typically used to visualize what a neural network has learned, or to generate new images. Nguyen et al. (2017) use approximate Langevin sampling from the conditional $p(\mathbf{x}|y = c)$, where c is be the event that $\mathbf{x}$ is an image of a particular class, such as a penguin. There are two main differences between our problem setting and that of AM. The first is that AM is conditioning on a discrete class being achieved, whereas our oracles are typically regression models. The second is that our design space is discrete, whereas in AM it is real-valued. Aside from the fact that in any case AM requires a differentiable oracle, these two differences pose significant challenges. The first difference makes it unclear how to specify the desired event to condition on, since the event may be that involving an unknown maximum. Moreover, even if one knew or could approximate this maximum, such an event would be by definition rare or never-seen, causing the relevant conditional probability to yield vanishing gradients from the start. Second, the issue of back-propagating through to a discrete input space is inherently difficult, although attempts have been made to use annealed relaxations (Jang et al., 2017). In fact, Killoran et al. (2017) adapt the AM-GAN procedure to side-step these two issues for protein design. Thus in our experiments, we compare to their variation of the AM approach.

Gómez-Bombarelli et al. (2018) tackle a chemistry design problem much like our own. Their approach is to (1) learn a neural-network-supervised variational auto-encoder (VAE) latent space so as to order the latent space by the property of interest, (2) build a (Gaussian Process) GP regression model from the latent space to the supervised property labels, (3) perform gradient-based maximisation of the GP over the latent space, (4) decode the optimal solution point(s) using the VAE decoder. Effectively they are approximately modelling the joint probability $p(\mathbf{x}, \mathbf{z}, y) = p(y|\mathbf{z})p(\mathbf{x}|\mathbf{z})p(\mathbf{z})$, for latent representation $\mathbf{z}$, and then finding the argument, $\mathbf{z}$, that maximizes $E[p(y|\mathbf{z})]$ using gradient descent. Similarly to AM, this approach does not satisfy our black box oracle *desideratum*. This approach in turn has a resemblance to Engel et al. (2018), wherein the goal is image generation, and a GAN objective is placed on top of a VAE in order to learn latent constraints. Because this latter approach was designed for (real-valued) images and for classification labels, we compare only to Gómez-Bombarelli et al. (2018).

Gupta & Zou (2019) offer a solution to our problem, including the ability to use a non-differentiable oracle. They propose to first train a GAN on the initial set of realistic examples and then iterate the following procedure: (1) create a sample set by generating samples from the GAN, (2) use an oracle regression model to make a point prediction for sample as whether or not a protein achieved some desired property, (3) update the sample set by replacing the oldest n samples with the n samples from step 2 that exceed a user-specified threshold that remains fixed throughout the algorithm (and where n at each iteration is determined by this threshold), (4) retrain the GAN on the updated set of samples from step 3 for one epoch. The intended goal is that as iterations proceed, the GAN will tend to produce samples which better maximize the property. They argue that because they only replace n samples at a time, the shift in the GAN distribution is slow, enabling them to implicitly stay near the training data if not run for too long. Their procedure does not arise from any particular formalism. As such, it is not clear what objective function is being optimized, nor how to choose an appropriate stopping point to balance progress and staying near the original realistic examples.

Our approach, *Conditioning by Adaptive Sampling* (*CbAS*), offers several advantages over these methods. Our approach is grounded in a coherent statistical framework, with a clear objective, and explicit use of prior information. It does not require a differentiable oracle, which has the added benefit of side-stepping the need to back-propagate through to discrete inputs, or of needing to anneal an approximate representation of the inputs. Our approach is based on parametric conditional density estimation. As such, it resembles a number of model-based optimization schemes, such as Evolutionary Distribution Algorithm (EDA) and Information Geometric Optimization (IGO) approaches (Hansen, 2006; Ollivier et al., 2017), both of which have shown to have good practical performance on a wide range of optimization problems. We additionally make use of ideas from Cross Entropy Methods (CEM) for estimating rare events, and their optimization counterparts (Rubinstein, 1997; 1999), which allows us to robustly condition on rare events, such as maximization events. In the case where one can be certain of a perfectly unbiased oracle, *DbAS*, which uses no prior information, should suffice; this method in turn is related to some earlier work in protein design (*e. g.,* Schneider & Wrede (1994); Schneider et al. (1998)).

Finally, our problem can loosely be seen to be related to a flavor of policy learning in Reinforcement Learning (RL) called Reward Weighted Regression (RWR) (Peters & Schaal, 2007). In particular, if one ignores the states, then RWR can be viewed as an (Estimation of Distribution Algorithm) EDA (Bengoetxea et al., 2001); as such, without the use of any prior to modulate exploration through the policy space, although at times, RWR imposes a per-iteration constraint of movement by way of a Kullback-Leibler (KL)-divergence term (Peters et al., 2010)—note that this offers no global constraint in the sense of a prior, and could be viewed as a formalization of the retaining of old samples in Gupta & Zou (2019).

Note that our problem statement is different from that of Bayesian Optimization (Snoek et al., 2012) where the goal at each iteration is to decide where to acquire *new* ground truth data labels. We are not acquiring any new labels, but our method could be used at the end of Bayesian Optimization (BO) for pure exploitation. Also, a baseline method we introduce, *CEM-PI*, could be used to perform optimization within standard BO.

3. Methods

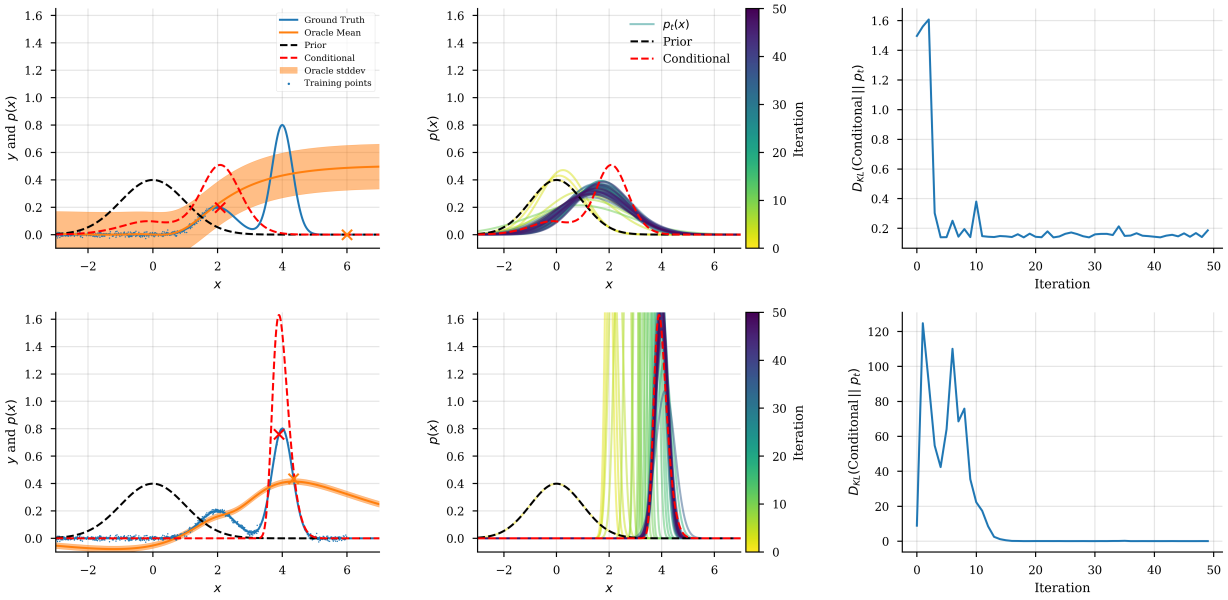
Preamble Our problem can be described as follows. We seek to find settings of the L-dimensional random vector, X (*e. g.,* representative of DNA sequences), that have high probability of satisfying some property *desideratum*. For example, we may want to design a protein that is maximally fluorescent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

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

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

Our ultimate aim is to condition this prior on our desired property values, S, and sample from the resulting distribution, thereby generating realizations of $\mathbf{x}$ that are likely to have low error in the predictive model or are realistic (*i. e.,* are drawn from the underlying data distribution); and have high probability of satisfying our *desideratum* encoded in S. Toward this end, we will assume that the desired conditional can be well-approximated with a sufficiently rich generative model, $q(\mathbf{x}|\boldsymbol{\phi})$, which need not be of the same parametric form as the prior. As our algorithm iterates, the parameter, $\boldsymbol{\phi}$, will slowly move toward a value that best approximates our desired conditional density. Our approach has a similar flavor to that of variational inference (Blei et al., 2017), but with a critical difference of needing to handle conditioning on rare events.

Below we outline our approach in the case of maximization



(a) (b) (c)

Figure 1. An illustrative example. (a) Relevant distributions and functions for two oracles (mean and standard deviation shown in orange). The oracle in the top plot was given training data corresponding to only half the domain, while the bottom one was given training data covering the whole domain. The prior conditioned on the property that the oracle is greater than its maximum is in red. The value of the ground truth at the mode of the conditional and the maximum of the oracle are shown as red and orange X's, demonstrating that the mode of the conditional distribution corresponds to a higher value of the ground truth than the maximum of the oracle (b) Evolution ('static animation') of the estimated conditional distribution as *CbAS* iterates; the exact distribution is shown in red in panel a. (c) KL divergence between the conditional and search distributions shown in (b), showing that our final approximate conditional is close to the real one.

of a single property. Details of how to readily generalize this to the specification problem and to more than one property, including a mix of maximization and specification, are in the Supplementary Information.

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

$$p(\mathbf{x}|S,\boldsymbol{\theta}^{(0)}) = \frac{P(S|\mathbf{x})p(\mathbf{x}|\boldsymbol{\theta}^{(0)})}{P(S|\boldsymbol{\theta}^{(0)})}, \tag{1}$$

where $P(S|\boldsymbol{\theta}^{(0)}) = \int d\mathbf{x}\ P(S|\mathbf{x})p(\mathbf{x}|\boldsymbol{\theta}^{(0)})$. In general, there will be no closed-form solution to this conditional density; hence we require a technique with which to approximate it, developed herein. The problem is also harder than it may seem at first because S is in general a rare event (*e. g.*, the occurrence of large property value we have never seen). To find the parameters of the search model, $q(\mathbf{x}|\boldsymbol{\phi})$, we will minimize the KL divergence between the target conditional, (1), and the search model,

$$\begin{aligned}
\boldsymbol{\phi}^* &= \operatorname*{argmin}_{\boldsymbol{\phi}} D_{KL}\left(p(\mathbf{x}|S,\boldsymbol{\theta}^{(0)})||q(\mathbf{x}|\boldsymbol{\phi})\right) && (2)\\
&= \operatorname*{argmin}_{\boldsymbol{\phi}} -\mathbb{E}_{p(\mathbf{x}|S,\boldsymbol{\theta}^{(0)})}[\log q(\mathbf{x}|\boldsymbol{\phi})] - H_0 && (3)\\
&= \operatorname*{argmax}_{\boldsymbol{\phi}} \frac{1}{P(S|\boldsymbol{\theta}^{(0)})}\mathbb{E}_{p(\mathbf{x}|\boldsymbol{\theta}^{(0)})}[P(S|\mathbf{x})\log q(\mathbf{x}|\boldsymbol{\phi})] && (4)\\
&= \operatorname*{argmax}_{\boldsymbol{\phi}} \mathbb{E}_{p(\mathbf{x}|\boldsymbol{\theta}^{(0)})}[P(S|\mathbf{x})\log q(\mathbf{x}|\boldsymbol{\phi})], && (5)
\end{aligned}$$

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

The objective in (5) may seem readily solvable by drawing samples, $\mathbf{x}_i$, from the prior, $p(\mathbf{x}|\boldsymbol{\theta}^{(0)})$, to obtain a Monte Carlo (MC) approximation to the expectation in (5); this results in a simple weighted maximum likelihood problem. However, for most interesting design problems, the desired set S will be exceedingly rare,[1] and consequently $P(S|\mathbf{x})$ will be vanishingly small for most $\mathbf{x}$ sampled from the prior.

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

Thus in such cases, an MC approximation to (5) will exhibit high variance and require an arbitrarily large number of samples to calculate accurately. Any gradient-based approach, such as reparameterization or the log-derivative trick (Kingma & Welling, 2014; Rezende et al., 2014), to try to solve (5) directly will suffer from a similar problem.

To overcome this problem of rare events, we draw inspiration from CEM (Rubinstein, 1997; 1999) and *DbAS* (Brookes & Listgarten, 2018) to propose an iterative, adaptive, importance sampling-based estimation method.

First we introduce an importance sampling distribution, $r(\mathbf{x})$, and rewrite the objective function in (5) as

$$\mathbb{E}_{p(\mathbf{x}|\boldsymbol{\theta}^{(0)})}[P(S|\mathbf{x}) \log q(\mathbf{x}|\boldsymbol{\phi})] \tag{6}$$

$$= \mathbb{E}_{r(\mathbf{x})}\left[\frac{p(\mathbf{x}|\boldsymbol{\theta}^{(0)})}{r(\mathbf{x})} P(S|\mathbf{x}) \log q(\mathbf{x}|\boldsymbol{\phi})\right], \tag{7}$$

to mitigate the problem that the expectation of $P(S|\mathbf{x})$ over our prior in (5) is likely to be vanishingly small. Now the question remains of how to find a good proposal distribution, $r(\mathbf{x})$. Rather than finding a single proposal distribution, we will construct a series relaxed conditions, $S^{(t)}$, and corresponding importance sampling distributions, $r^{(t)}(\mathbf{x})$, such that (a) $E_{r^{(t)}(\mathbf{x})}[P(S^{(t)}|\mathbf{x})]$ is non-vanishing, and (b) $S^{(t)} \supset S^{(t+1)} \supset S$, for all t. The first condition implies that we can draw samples, $\mathbf{x}$, from $r^{(t)}(\mathbf{x})$ that have reasonably high values of $P(S^{(t)}|\mathbf{x})$. The second condition ensures that we slowly move toward our desired property condition; that is, it ensures that $S^{(t)}$ approaches S as t grows large. (In practice, we choose to use the less stringent condition $S^{(t)} \supseteq S^{(t+1)} \supseteq S$, but the stricter condition can trivially be achieved with a minor change to our algorithm, in which case one should be careful to ensure that the variance of the expectation argument does not grow too large).

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

Consider the first condition, which is that $E_{r^{(t)}(\mathbf{x})}[P(S^{(t)}|\mathbf{x})]$ is non-vanishing. To achieve this, we could (1) set $S^{(t)}$ to be the relaxed condition that $p(y \geq \gamma^{(0)}|\mathbf{x})$ where $\gamma^{(0)}$ is the Q^{th} percentile of property values predicted for those samples used to construct the prior, and (2) set the proposal distribution to the prior, $r^{(t)}(\mathbf{x}) = p(\mathbf{x}|\boldsymbol{\theta}^{(0)})$. For Q small enough, $E_{q(\mathbf{x}|\boldsymbol{\phi}^{(t)})}[P(S^{(t)}|\mathbf{x})]$ will be non-vanishing by definition and condition (a) will be satisfied. Thus, by construction, we can now reasonably perform maximization of the objective in (7) instantiated with $S^{(t)}$ because the rare event is no longer rare. In fact, this is how we set the first tuple of the required sequence, $(S^{(0)}, r^{(t)})$. After that first iteration, we will then have solved for our approximate conditional, under a relaxed version of our property *desideratum* to obtain $q(\mathbf{x}|\boldsymbol{\phi}^{(0)})$. Then, at each iteration, we set $r^{(t)} = q(\mathbf{x}|\boldsymbol{\phi}^{(t-1)})$, and $\gamma^{(t)}$ to the Q^{th} percentile of property values predicted from the samples obtained in the $(t-1)^{\text{th}}$ iteration. By the same arguments made for the initial tuple of the sequence, we will have achieved condition (a), and condition (b) can trivially be achieved by disallowing $\gamma^{(t)}$ from decreasing from the previous iteration (*i. e.,* , set it to the same value as the previous iteration if necessary).

Altogether now, (7) becomes

$$\mathbb{E}_{q(\mathbf{x}|\boldsymbol{\phi}^{(t)})}\left[\frac{p(\mathbf{x}|\boldsymbol{\theta}^{(0)})}{q(\mathbf{x}|\boldsymbol{\phi}^{(t)})} P(S^{(t)}|\mathbf{x}) \log q(\mathbf{x}|\boldsymbol{\phi})\right], \tag{8}$$

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

$$\boldsymbol{\phi}^{(t+1)} = \underset{\boldsymbol{\phi}}{\operatorname{argmax}} \sum_{i=1}^{M} \frac{p(\mathbf{x}_i^{(t)}|\boldsymbol{\theta}^{(0)})}{q(\mathbf{x}_i^{(t)}|\boldsymbol{\phi}^{(t)})} P(S^{(t)}|\mathbf{x}_i^{(t)}) \log q(\mathbf{x}_i^{(t)}|\boldsymbol{\phi}). \tag{9}$$

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

It is clear from (9) that the variance of the MC estimate not only relies on $P(S^{(t)}|\mathbf{x})$ being non-vanishing for many of the samples but also that the density ratio, $\frac{p(\mathbf{x}|\boldsymbol{\theta}^{(0)})}{q(\mathbf{x}|\boldsymbol{\phi}^{(t)})}$ is similarly well-behaved. Fortunately, this is enforced by the weighting scheme itself, which encourages the density of the search model to remain close to the prior distribution. This is intuitively satisfying, as it shows that minimizing the KL divergence between the search distribution and the target conditional requires balancing the maximization the probability of $S^{(t)}$ with adherence to the prior distribution.

Extension to intractable latent variable models Our final objective in (9) requires us to reliably evaluate the densities of the prior and search models for a given input $\mathbf{x}$. This is often not possible, particularly for latent variable models where the marginalization over the latent space is intractable. Although one might consider using an Evidence Lower Bound (ELBO) (Blei et al., 2017) approximation

to the required densities, one can exploit the structure of our objective to exactly derive the needed quantity. In particular, we can maintain exactness of our objective for prior and search model densities where one can only calculate the joint densities of the observed and latent variables, namely $p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta}^{(0)})$ and $q(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta}^{(t)})$, which can be achieved only if both model's densities are defined on the same latent variable space, $\mathcal{Z}$.

This extension relies on the fact that an expectation over a marginal distribution is equal to the same expectation over an augmented joint distribution, for instance $\mathbb{E}_{p(x)}[f(x)] = \mathbb{E}_{p(x,y)}[f(x)]$. Starting with (6) and using this fact, we arrive at an equivalent objective function,

$$\mathbb{E}_{p(\mathbf{x}|\boldsymbol{\theta}^{(0)})}[P(S^{(t)}|\mathbf{x}) \log q(\mathbf{x}|\boldsymbol{\phi})] \tag{10}$$

$$= \mathbb{E}_{p(\mathbf{x},\mathbf{z}|\boldsymbol{\theta}^{(0)})}[P(S^{(t)}|\mathbf{x}) \log q(\mathbf{x}|\boldsymbol{\phi})] \tag{11}$$

$$= \mathbb{E}_{q(\mathbf{x},\mathbf{z}|\boldsymbol{\phi}^{(t)})}\left[\frac{p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta}^{(0)})}{q(\mathbf{x}, \mathbf{z}|\boldsymbol{\phi}^{(t)})} P(S^{(t)}|\mathbf{x}) \log q(\mathbf{x}|\boldsymbol{\phi})\right]. \tag{12}$$

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

Note that in the common case where both models are constructed such that they have the same density over $\mathcal{Z}$, $p(\mathbf{z})$, then (12) can be further simplified using $\frac{p(\mathbf{x}|\mathbf{z},\boldsymbol{\theta}^{(0)})p(\mathbf{z})}{q(\mathbf{x}|\mathbf{z},\boldsymbol{\phi}^{(t)})p(\mathbf{z})} = \frac{p(\mathbf{x}|\mathbf{z},\boldsymbol{\theta}^{(0)})}{q(\mathbf{x}|\mathbf{z},\boldsymbol{\phi}^{(t)})}$.

Practical Considerations In practice, we often use the same parametric form for the search model and the prior density. This allows us to simply initialize the search distribution parameters as $\boldsymbol{\phi}^{(1)} = \boldsymbol{\theta}^{(0)}$. Additionally, in practice we cache the search model parameters $\boldsymbol{\phi}^{(t)}$ and use these to initialize the parameters at the next time step in order to reduce the computational cost of the training procedure at each step.

In Algorithm 1 in the Supplemental Information, we outline our complete procedure when the prior and generative model are both latent variable models of the same parametric form (*e. g.,* both a VAE).

4. Experiments

We perform two main sets of experiments. In the first, we use simple, one-dimensional, toy examples to demonstrate some of the main points about our approach. In the second set of experiments, we ground ourselves in a real protein fluorescence data set, conducting extensive simulations to compare methods.

4.1. An illustrative example

We first wanted to investigate the properties of our approach with a simple, visual example to understand how the prior influences the solutions found. In particular we wanted to see how an oracle could readily become untrustworthy away from the training data, and how use of the prior might alleviate this. We also wanted to see that even when the oracle is trustworthy, that our approach still yields sensible results. Finally, we wanted to ensure that our approach does indeed approximate the desired conditional distribution satisfactorily for simple cases that we can see. The summary results of this experimentation are shown in Figure 1.

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

Next we created two different oracles, by giving one access only to data that covered half of the domain, and the other by giving it data that covered the entire domain (including all those in the first data set). These training data were ground truth observations corrupted with zero-mean Gaussian noise with variance of 0.01. Then two oracles were trained, one for each data set, and of the form, $p(y|x) = \mathcal{N}(\mu(x), \sigma^2)$ where $\mu(x)$ is a two hidden-layer neural network fit to one of the training sets and σ^2 was set to the mean squared error between the ground truth and $\mu(x)$ on a hold-out set.

The resulting oracles, and the underlying ground truth are shown in Figure 1a. The oracle trained with the smaller data set suffers from a serious pathology—it continues to increase in regions where the ground truth function rapidly decreases to zero. This is exactly the type of pathology that *CbAS* aims to overcome by estimating the conditional density of the prior rather than directly optimizing the objective. The second oracle does not suffer from as serious a pathology, and serves to show that *CbAS* can still perform well in the case that the oracle is rather accurate (*i. e.,* the prior does not overly constrain the search). As with any Bayesian method with a prior, there may be settings where the prior could lead one astray, and this example is simply meant to convey some intuition. The next set of experiments, grounded in a protein design problem, suggest that the prior we constructed, used within *CbAS*, works well in practice.

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

conditional lies near a local maximum and the conditional assigns little density to regions where the oracle is highly biased.

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

4.2. Application to protein fluorescence

Fluorescent proteins are a workhouse of modern molecular biology. Improving them has a long history, but also serves as a good test bed for protein optimization. Thus we here focus on the problem of maximizing protein fluorescence. In particular, we perform a systematic comparison of our method, *CbAS*, to seven other approaches, described below. We anchored our experiments on a real protein fluorescence data set (Sarkisyan et al., 2016) (see Supplementary Information).

Methods considered We compared our approach to other competing methods, including, for completeness, those which can only work with differentiable oracles. For models that originally used a GAN prior, we instead use a VAE prior so as to make the comparisons meaningful.[2] We compare our method, *CbAS* against the following methods: (1) *AM-VAE*—the activation-maximization method of Killoran et al. (2017). This method requires a differentiable oracle. (2) *FB-VAE*—the method of Gupta & Zou (2019). This method does not require a differentiable oracle. (3) *GB-NO*—the approach described by Gómez-Bombarelli et al. (2018). This method requires a differentiable oracle. (4) *GB*—the approach implemented by Gómez-Bombarelli et al. (2018) which has some additional optimization constraints placed on the latent space that were not reported in the paper but were used in their code. This method requires a differentiable oracle. (5) *DbAS*—a method similar to *CbAS*, but which assumes an unbiased oracle and hence has no need for or ability to incorporate a prior on the input design space. (6) *RWR*—Reward Weighted Regression (Peters & Schaal, 2007), which is similar to *DbAS*, but without taking into account the oracle uncertainty. This method does not require a differentiable oracle. (7) *CEM-PI*—use of the Cross Entropy Method to maximize the Probability of Improvement (Snoek et al., 2012), an acquisition function often used in Bayesian Optimization; this approach does not make use of a prior on the input design space. This method does not require a differentiable oracle. Implementation details for each of these methods can be found in the Supplementary Information.

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

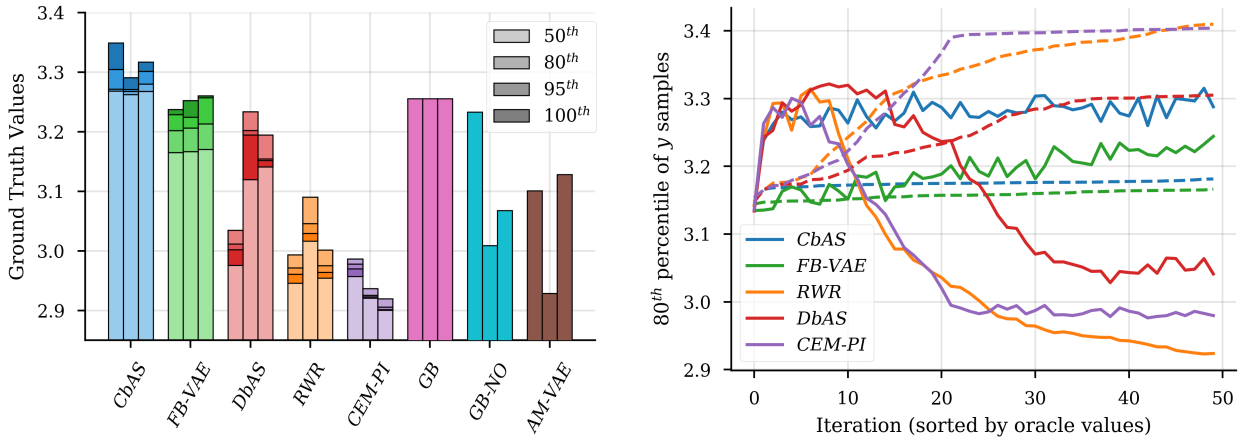
Simulations In order to compare methods it was necessary to first simulate the real data with a known ground truth because evaluating hold out data from a real data set is not feasible in our problem setting. To obtain a ground truth model we trained a GP regression (GPR) model on the protein fluorescence data, with a protein-specific kernel (Shen et al., 2014), whose feature space was augmented by adding a bias feature and exponentiating to obtain a second order polynomial kernel. Our ground truth is the mean of this GPR model, which, notably, is a different model class than the oracle, as we expect in practice.

Next, for each protein sequence in the original data set we compute its ground truth fluorescence. Then we take those sequences that lie in the bottom 20^{th} percentile of ground truth fluorescence, choose 5,000 of them at random, and use these to train our oracles using maximum likelihood. We wanted to investigate different kinds of oracles with different properties, with a particular focus on investigating different uncertainty estimates. Specifically, we considered three types of oracle (details are in the Supplementary Information). The first was a single-layer neural network model with homoscedastic, Gaussian noise in its predictions. The second was an ensemble of five of these models, each with heteroscedastic noise, as in Lakshminarayanan et al. (2017). The third was the same as the second, but with an ensemble of 20 neural networks. Performance of these models on a test set are shown in the Supplemental Information.

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

To fairly compare the methods we keep the total number of samples considered during each run, N, constant. We call this the sequence budget. This sequence budget corresponds to limiting the total number samples drawn from the gener-

[2] As shown in Brookes & Listgarten (2018), the GAN and VAE appear to yield roughly similar results in this problem setting.



(a) (b)

Figure 2. Design for maximization of protein fluorescence. (a) For sampling-based methods, namely *CbAS*, *FB-VAE*, *DbAS*, *RWR* and *CEM-PI*, shown are the mean values of the ground truth evaluated at samples coming from different percentiles (50^{th}, 80^{th}, 95^{th}, and 100^{th}) of oracle predictions over all iterations. The gradient-based methods, *GB*, *GB-NO* and *AM-VAE*, yield only a single protein at each iteration and converge rapidly; thus only the ground truth value of the final single protein for each of these methods is used. For sampling-based methods, an optimal method has darkest bars at the top, followed by progressively less dark bars, indicating that the method has successfully avoided untrustworthy regions of the space. Methods that have the darkness out of this order are being led astray into untrustworthy oracle regions. The height of a bar shows how well the ground truth fluorescence is maximized, where higher is better. For *CbAS*, the height is highest, and the darkness ordering, correct—unsurprisingly, as avoiding oracle pathologies should help achieve higher ground truth during maximization. This trend of better darkness ordering yielding high property values holds for all sampling based methods. The sets of three bars indicate the three different oracles (each bar was averaged over three random runs). (b) For one representative run in panel a), a trajectory is shown for each method (other runs are in the Supplemental Information). Each point on a dashed lines shows the 80^{th} percentile of oracle evaluations of the samples at that iteration. The corresponding point on a solid line shows the mean ground truth value of those same samples. The last point on a curve shows what final protein sequence would be used from each method (*i. e.,* that with the highest oracle value seen, as the ground truth would be unknown)—high dashed values and low solid ones are methods that have been led astray into pathological regions of the oracle.

ative model in *CbAS*, *DbAS*, *FB-VAE*, *RWR* and *CEM-PI*; and limiting the number of total gradient step updates performed in the *AM-VAE*, *GB* and *GB-NO* methods. For the latter class of methods, if the method converged but had not used up its sequence budget, then the method was re-started with a new, random initialization and executed until the sequence budget was exhausted, or a new initialization was needed. For convergence we used the built-in method that comes with *GB* and *GB-NO*, and for *AM-VAE*, convergence is defined as that the maximum value did not improve in 100 iterations.

For each of the three oracle models, we ran three randomly initialized runs of each method with a sequence budget of 10,000. Figure 2a shows the results from this experiment, averaged over the three separate runs. Methods that have no notion of a prior (*DbAS*, *RWR*, *CEM-PI*) are clearly led astray, as they only optimize the oracle, finding themselves in untrustworthy regions of oracle space. This suggests that our simulation settings are in the regime of the illustrative example shown in Figure 1a (top).

To further sanity check that these models are being led astray, we also roughly estimated the protein stability (Park et al., 2016) of the final selected sequences for each method, finding that that prior-free models can easily be led into parts of the space that are not realistic for exiting proteins because they are unstable (see Supplementary Information).

5. Discussion

Our main contributions have been the following. (1) We introduced a new method that enables one to solve design problems in discrete space, and with a potentially non-differentiable oracle model, (2) we introduced a new way to perform approximate, conditional density modelling for rich classes of models, and importantly, in the presence of rare events, (3) we showed how to leverage the structure of latent variable models to achieve exact importance sampling in the presence of models whose density cannot be computed exactly. Finally, we showed that compared to other alternative solutions to this problem, even those which require the oracle to be differential, our approach yields competitive results.

Acknowledgements

The authors thank Sergey Levine, Kevin Murphy, Petros Koumoutsakos, Gisbert Schneider, David Duvenaud, Yisong Yue and Ben Recht for helpful discussion and pointers; and Benjamin Sanchez-Lengeling for providing python scripts to reproduce the method in Gómez-Bombarelli et al. (2018).

References

Baum, L. E. and Petrie, T. Statistical inference for probabilistic functions of finite state markov chains. *Ann. Math. Statist.*, 37(6):1554–1563, 12 1966. doi: 10.1214/aoms/1177699147.

Bengoetxea, E., Larrañaga, P., Bloch, I., and Perchant, A. Estimation of distribution algorithms: A new evolutionary computation approach for graph matching problems. In Figueiredo, M., Zerubia, J., and Jain, A. K. (eds.), *Energy Minimization Methods in Computer Vision and Pattern Recognition*, pp. 454–469, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg. ISBN 978-3-540-44745-0.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.

Brookes, D. H. and Listgarten, J. Design by adaptive sampling. *arXiv*, abs/1810.03714, 2018.

Chen, K. and Arnold, F. H. Enzyme Engineering for Nonaqueous Solvents: Random Mutagenesis to Enhance Activity of Subtilisin E in Polar Organic Media. *Bio/Technology*, 9(11):1073–1077, 1991.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using Real NVP. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

Engel, J., Hoffman, M., and Roberts, A. Latent constraints: Learning to generate conditionally from unconditional generative models. In *International Conference on Learning Representations (ICLR)*, 2018.

Gómez-Bombarelli, R., Wei, J. N., Duvenaud, D., Hernández-Lobato, J. M., Sánchez-Lengeling, B., Sheberla, D., Aguilera-Iparraguirre, J., Hirzel, T. D., Adams, R. P., and Aspuru-Guzik, A. Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules. *ACS Central Science*, 4(2):268–276, 2018.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 27*, pp. 2672–2680. Curran Associates, Inc., 2014.

Gupta, A. and Zou, J. Feedback GAN for DNA optimizes protein functions. *Nat. Mach. Intell.*, 1(2):105–111, 2019. ISSN 2522-5839. doi: 10.1038/s42256-019-0017-4.

Hansen, N. The CMA evolution strategy: A tutorial. *arXiv*, 1412.6980, 2006.

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with Gumbel-softmax. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.

Killoran, N., Lee, L. J., Delong, A., Duvenaud, D., and Frey, B. J. Generating and designing DNA with deep generative models. *arXiv*, 1712.06148, 2017.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 6402–6413. 2017.

Nalisnick, E., Matsukawa, A., Teh, Y. W., Gorur, D., and Lakshminarayanan, B. Do deep generative models know what they don't know? In *International Conference on Learning Representations*, 2019.

Nguyen, A., Clune, J., Bengio, Y., Dosovitskiy, A., and Yosinski, J. Plug & play generative networks: Conditional iterative generation of images in latent space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2017.

Nguyen, A. M., Yosinski, J., and Clune, J. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pp. 427–436, 2015. doi: 10.1109/CVPR.2015.7298640.

Nguyen, A. M., Dosovitskiy, A., Yosinski, J., Brox, T., and Clune, J. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. *arXiv*, 1605.09304, 2016.

Ollivier, Y., Arnold, L., Auger, A., and Hansen, N. Information-geometric optimization algorithms: A unifying picture via invariance principles. *Journal of Machine Learning Research*, 18(18):1–65, 2017.

Park, H., Bradley, P., Greisen, P., Liu, Y., Mulligan, V. K., Kim, D. E., Baker, D., and DiMaio, F. Simultaneous Optimization of Biomolecular Energy Functions on Features from Small Molecules and Macromolecules. *Journal of Chemical Theory and Computation*, 12(12):6201–6212, 2016. doi: 10.1021/acs.jctc.6b00819.

Peshkin, L. and Shelton, C. R. Learning from scarce experience. *arXiv*, cs.AI/0204043, 2002.

Peters, J. and Schaal, S. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pp. 745–750, New York, NY, USA, 2007. ACM.

Peters, J., Mülling, K., and Altun, Y. Relative entropy policy search. *AAAI 2010*, 2010.

Precup, D., Sutton, R. S., and Dasgupta, S. Off-policy temporal difference learning with function approximation. In *ICML*, 2001.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, volume 32 of *JMLR Workshop and Conference Proceedings*, pp. 1278–1286, 2014.

Rives, A., Goyal, S., Meier, J., Guo, D., Ott, M., Zitnick, C. L., Ma, J., and Fergus, R. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *bioRxiv*, 2019. doi: 10.1101/622803.

Rubinstein, R. The Cross-Entropy Method for Combinatorial and Continuous Optimization. *Methodology And Computing In Applied Probability*, 1(2):127–190, 1999.

Rubinstein, R. Y. Optimization of computer simulation models with rare events. *European Journal of Operational Research*, 99(1):89–112, 1997.

Sarkisyan, K. S., Bolotin, D. A., Meer, M. V., Usmanova, D. R., Mishin, A. S., Sharonov, G. V., Ivankov, D. N., Bozhanova, N. G., Baranov, M. S., Soylemez, O., Bogatyreva, N. S., Vlasov, P. K., Egorov, E. S., Logacheva, M. D., Kondrashov, A. S., Chudakov, D. M., Putintseva, E. V., Mamedov, I. Z., Tawfik, D. S., Lukyanov, K. A., and Kondrashov, F. A. Local fitness landscape of the green fluorescent protein. *Nature*, 533:397, 2016.

Schneider, G. and Wrede, P. The rational design of amino acid sequences by artificial neural networks and simulated molecular evolution: de novo design of an idealized leader peptidase cleavage site. *Biophysical Journal*, 66(2 Pt 1):335–44, feb 1994. ISSN 0006-3495.

Schneider, G., Schrödl, W., Wallukat, G., Müller, J., Nissen, E., Rönspeck, W., Wrede, P., and Kunze, R. Peptide design by artificial neural networks and computer-based evolutionary search. *Proceedings of the National Academy of Sciences of the United States of America*, 95 (21):12179–84, oct 1998. ISSN 0027-8424.

Shen, W.-J., Wong, H.-S., Xiao, Q.-W., Guo, X., and Smale, S. Introduction to the Peptide Binding Problem of Computational Immunology: New Results. *Foundations of Computational Mathematics*, 14(5):951–984, oct 2014. ISSN 1615-3375. doi: 10.1007/s10208-013-9173-9.

Simonyan, K., Vedaldi, A., and Zisserman, A. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv*, 1312.6034, 2013.

Snoek, J., Larochelle, H., and Adams, R. P. Practical bayesian optimization of machine learning algorithms. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 25*, pp. 2951–2959. Curran Associates, Inc., 2012.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.

Tang, J. and Abbeel, P. On a connection between importance sampling and the likelihood ratio policy gradient. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 1*, NIPS'10, pp. 1000–1008, USA, 2010.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 5998–6008. Curran Associates, Inc., 2017.

Yang, K. K., Wu, Z., and Arnold, F. H. Machine learning in protein engineering. *arXiv*, nov 2018.

Supplementary Information: Conditioning by adaptive sampling for robust design

S1. Algorithm

Algorithm 1. **Maximization of a single, continuous property.** $h_{\text{oracle}}(\mathbf{x}_i)$ is a function returning the expected value of the property oracle. $\text{CDF}_{\text{oracle}}(\mathbf{x}, \gamma)$ is a function to compute the CDF of the oracle predictive model for threshold γ. $\text{GenProb}(\mathbf{x}_i, \mathbf{z}_i, \boldsymbol{\theta})$ is a method that evaluates the probability of an input in a generative model with parameters $\boldsymbol{\theta}$. $\text{GenTrain}(\{(\mathbf{x}_i, w_i)\}$ is a procedure to take weighted training data $\{(\mathbf{x}_i, w_i)\}$, and return the parameters of the trained model. Q is a parameter that determines the next iteration's relaxation threshold; M is the number of samples to generate at each iteration. See main text for convergence criteria. $\{\mathbf{x}_{\text{init}}\}$ is an initial set of samples with which to train the prior. Any line with an i subscript implicitly denotes $\forall i \in [1 \dots M]$.

procedure $CbAS(h_{\text{oracle}}(\mathbf{x}), \text{CDF}_{\text{oracle}}(\mathbf{x}, \gamma), \text{GenTrain}(\{\mathbf{x}_i\, w_i\}), \text{GenProb}(\mathbf{x}_i), [Q = 0.9], [M = 100])$

- $\boldsymbol{\theta}^{(0)} \leftarrow \text{GenTrain}(\{\mathbf{x}_{\text{init}}\}, w_i = 1)$
- $\boldsymbol{\phi}^{(1)} \leftarrow \boldsymbol{\theta}^{(0)}$
- $t \leftarrow 1$
- **while** not converged **do**
  - $set_i \leftarrow \mathbf{x}_i, \mathbf{z}_i \sim G(\boldsymbol{\phi}^{(t)})$
  - $scores_i \leftarrow h_{\text{oracle}}(\mathbf{x}_i)$
  - $q \leftarrow$ index of Q^{th} percentile of $scores$
  - $\gamma^{(t)} \leftarrow scores_{\text{q}}$
  - $weights_i \leftarrow \dfrac{\text{GenProb}(set_i, \boldsymbol{\theta}^{(0)})}{\text{GenProb}(set_i, \boldsymbol{\phi}^{(t)})}$
  - $weights_i \leftarrow weights_i * (1 - CDF_{\text{oracle}}(\mathbf{x}_i, \gamma^{(t)}))$
  - $\boldsymbol{\phi}^{(t)} =\leftarrow \text{GenTrain}(set, weights)$
  - $t \leftarrow t + 1$
- **return** $set, weights$

S2. Generalization to specification

The *CbAS* procedure can be easily extended to perform specification of a property value rather than maximization. In this case the target set is an infinitesimally small range around the target, *i. e.,* $S = [y_0 - \epsilon, y_0 + \epsilon]$ for a target value y_0 and a small $\epsilon > 0$. The *CbAS* procedure remains mostly identical to that of maximization case, except in this case the intermediate sets $S^{(t)} = [y_0 - \gamma^{(t)}, y_0 + \gamma^{(t)}]$ are centered on the specified value and have an update-able width, $\gamma^{(t)}$. The $\gamma^{(t)}$ values are then updated analogously to the thresholds in the maximization case, *i. e.,* $\gamma^{(t)}$ is set to the Q^{th} percentile of $|y_i - y_0|$ values, for $i = 1, ..., M$, where y_i are the expected property values according to the sample $\mathbf{x}_i \sim p(\mathbf{x}|\boldsymbol{\theta}^{(t)})$.

S3. Generalization to multiple properties

Additionally, *CbAS* can be extended to handle multiple properties $Y_1, ..., Y_K$ with corresponding desired sets $S_1, ..., S_K$. We only require that these properties are conditionally independent given a realization of X. In this case,

$$P(S_1, ..., S_K|\mathbf{x}) = \prod_{i=1}^{K} P(S_i|\mathbf{x}) \tag{S1}$$

where now each Y_i has an independent oracle. This distribution, and the corresponding marginal distribution $P(S_1, ..., S_K|\boldsymbol{\theta}) = \int d\mathbf{x}\, p(\mathbf{x}|\boldsymbol{\theta}) \prod_{i=1}^{K} P(S_i|\mathbf{x})$ can then be used in place of $P(S|\mathbf{x})$ and $P(S|\boldsymbol{\theta})$ in Equations (1)-(13) in the main text to recover the *CbAS* procedure for multiple properties.

S4. Extension to models not permitting MLE

Many models cannot be fit with maximum likelihood estimation, and in this case we cannot solve the *CbAS* update equation, (9), exactly. However, *CbAS* can still be used in the case when approximate inference procedures can be performed on these models, for example any model that can be fit with variational inference (Jordan et al., 1999). We derive the *CbAS* update equation in the variational inference case below, but the update equation can be modified in a corresponding way for any model that permit other forms of approximate MLE.

In variational inference specifically, the maximum likelihood is lower bounded by an alternative objective:

$$\max_{\boldsymbol{\phi}} \log q(\mathbf{x}|\boldsymbol{\phi}) \tag{S2}$$

$$= \max_{\boldsymbol{\phi}} \log \mathbb{E}_{q(\mathbf{z})}[q(\mathbf{x}|\mathbf{z}, \boldsymbol{\phi})] \tag{S3}$$

$$\geq \max_{\boldsymbol{\phi}\boldsymbol{\psi}} \mathbb{E}_{r(\mathbf{z}|\mathbf{x},\boldsymbol{\psi})} \left[\log q(\mathbf{x}|\mathbf{z}, \boldsymbol{\phi})\right] - D_{KL}\left[r(\mathbf{z}|\mathbf{x}, \boldsymbol{\psi})||q(\mathbf{z})\right] \tag{S4}$$

$$= \max_{\boldsymbol{\phi},\boldsymbol{\psi}} \mathcal{L}(\mathbf{x}, \boldsymbol{\phi}, \boldsymbol{\psi}) \tag{S5}$$

where $\mathbf{z}$ is a realization of a latent variable with prior $p(\mathbf{z})$, and $r(\mathbf{z}|\mathbf{x}, \boldsymbol{\psi})$ is an approximate posterior with parameters $\boldsymbol{\psi}$. Equation (S5) implies that we can lower bound the the argument of (9):

$$\max_{\boldsymbol{\phi}} \sum_{i=1}^{M} \frac{p(\mathbf{x}_i^{(t)}|\boldsymbol{\theta}^{(0)})}{q(\mathbf{x}_i^{(t)}|\boldsymbol{\phi}^{(t)})} P(S^{(t)}|\mathbf{x}_i^{(t)}) \log q(\mathbf{x}_i^{(t)}|\boldsymbol{\phi}) \geq \max_{\boldsymbol{\phi},\boldsymbol{\psi}} \sum_{i=1}^{M} \frac{p(\mathbf{x}_i^{(t)}|\boldsymbol{\theta}^{(0)})}{q(\mathbf{x}_i^{(t)}|\boldsymbol{\phi}^{(t)})} P(S^{(t)}|\mathbf{x}_i^{(t)}) \mathcal{L}(\mathbf{x}_i^{(t)}, \boldsymbol{\phi}, \boldsymbol{\psi}) \tag{S6}$$

which is a tight bound when the approximate posterior in the model is rich enough for the approximate posterior to exactly match the true model posterior. This suggests a new update equation, specific for models trained with variational inference:

$$\boldsymbol{\phi}^{(t+1)}, \boldsymbol{\psi}^{(t+1)} = \operatorname*{argmax}_{\boldsymbol{\phi},\boldsymbol{\psi}} \sum_{i=1}^{M} p(S^{(t)}|\mathbf{x}_i^{(t)}) \mathcal{L}(\mathbf{x}_i^{(t)}; \boldsymbol{\phi}, \boldsymbol{\psi}) \tag{S7}$$

where we now give time dependence to the approximate posterior parameters, $\boldsymbol{\psi}$. In practice, this is the update equation we use for *CbAS* variants that use VAEs as the search distribution (appropriately augmented for latent variables, as described in Section 3 of the main text). This same process can be used to derive approximate update equations for any weighted maximum likelihood method.

S5. Using Samples from Previous Iterations

When using model-based optimization methods that update the model parameters using weighted maximum likelihood updates at each iteration, one can extend such a method to make use of samples generated in previous iterations of the algorithm, in the current iteration, so as to potentially increase the effective sample size for the maximum likelihood problem at each iteration. Examples of such methods include that presented herein, *CbAS*, and also *DbAS*, *RWR* and *CEM-PI* (see Section S6.1). To achieve this goal of using samples from previous iterations, we use an importance sampling scheme to re-scale the weights, $w(\mathbf{x})$, by the ratio of likelihoods of the sample under the current search model (t^{th} iteration), to that of the search model from the earlier iteration (s^{th} iteration). This can be seen as follows:

$$\mathbb{E}_{p(\mathbf{x}|\boldsymbol{\theta}^{(t)})}[w(\mathbf{x}) \log p(\mathbf{x}|\boldsymbol{\theta})] \tag{S8}$$

$$= \frac{1}{t} \sum_{s=1}^{t} \mathbb{E}_{p(\mathbf{x}|\boldsymbol{\theta}^{(t)})}[w(\mathbf{x}) \log p(\mathbf{x}|\boldsymbol{\theta})] \tag{S9}$$

$$= \frac{1}{t} \sum_{s=1}^{t} \mathbb{E}_{p(\mathbf{x}|\boldsymbol{\theta}^{(s)})} \left[\frac{p(\mathbf{x}|\boldsymbol{\theta}^{(t)})}{p(\mathbf{x}|\boldsymbol{\theta}^{(s)})} w(\mathbf{x}) \log p(\mathbf{x}|\boldsymbol{\theta})\right], \tag{S10}$$

where (S9) uses a bookkeeping trick of averaging identical quantities in order to introduce a sum, and (S10) introduces $p(\mathbf{x}|\boldsymbol{\theta}^{(s)})$ as an importance sampling proposal density into this sum. Using samples drawn from the model at each iteration,

s, (*i. e.,* old samples) we arrive at the final objective argument sums over the current samples, and samples from all previous iterations:

$$\frac{1}{tM}\sum_{s=1}^{t}\sum_{i=1}^{M}\frac{p(\mathbf{x}_i^{(s)}|\boldsymbol{\theta}^{(t)})}{p(\mathbf{x}_i^{(s)}|\boldsymbol{\theta}^{(s)})}w(\mathbf{x}_i^{(s)})\log p(\mathbf{x}_i^{(s)}|\boldsymbol{\theta}), \tag{S11}$$

where $\mathbf{x}_i^{(s)} \sim p(\mathbf{x}|\boldsymbol{\theta}^{(s)})$ are M samples drawn at iteration s. Note that the same methods outlined in the 'Extension to intractable latent variable models' section of the main text can be used to calculate the likelihood ratio, $\frac{p(\mathbf{x}_i^{(s)}|\boldsymbol{\theta}^{(t)})}{p(\mathbf{x}_i^{(s)}|\boldsymbol{\theta}^{(s)})}$, in (S11), if the marginal likelihoods cannot be calculated exactly (*e. g.,* when an Evidence Lower Bound is used such as in the VAE).

Note that as the difference between t and s increases (*i. e.,* we start to consider older and older samples), it may be the case that the likelihood ratios, $\frac{p(\mathbf{x}_i^{(s)}|\boldsymbol{\theta}^{(t)})}{p(\mathbf{x}_i^{(s)}|\boldsymbol{\theta}^{(s)})}$, in (S11) become extremely small. In such cases, the effective sample size (*i. e.,* $\sum_s\sum_i \frac{p(\mathbf{x}_i^{(s)}|\boldsymbol{\theta}^{(t)})}{p(\mathbf{x}_i^{(s)}|\boldsymbol{\theta}^{(s)})}w(\mathbf{x}_i^{(s)})$) of the MC estimate (S11) may not be much more than that from an iteration that does not use older samples, $\sum_{i=1}^{M} w(\mathbf{x}_i^{(s)})$. This potentially limits the utility of keeping old samples for many weighting schemes, such as in *DbAS*, *RWR* and *CEM-PI*. However, in *CbAS*, where $w(\mathbf{x}) = \frac{p(\mathbf{x}|\boldsymbol{\theta}^{(0)})}{p(\mathbf{x}|\boldsymbol{\theta}^{(t)})}P(S^{(t)}|\mathbf{x})$ at iteration t, (S11) becomes:

$$\frac{1}{tM}\sum_{s=1}^{t}\sum_{i=1}^{M}\frac{p(\mathbf{x}_i^{(s)}|\boldsymbol{\theta}^{(t)})}{p(\mathbf{x}_i^{(s)}|\boldsymbol{\theta}^{(s)})}\frac{p(\mathbf{x}_i^{(s)}|\boldsymbol{\theta}^{(0)})}{p(\mathbf{x}_i^{(s)}|\boldsymbol{\theta}^{(t)})}P(S^{(t)}|\mathbf{x}_i^{(s)})\log p(\mathbf{x}_i^{(s)}|\boldsymbol{\theta}) \tag{S12}$$

$$=\frac{1}{tM}\sum_{s=1}^{t}\sum_{i=1}^{M}\frac{p(\mathbf{x}_i^{(s)}|\boldsymbol{\theta}^{(0)})}{p(\mathbf{x}_i^{(s)}|\boldsymbol{\theta}^{(s)})}P(S^{(t)}|\mathbf{x}_i^{(s)})\log p(\mathbf{x}_i^{(s)}|\boldsymbol{\theta}). \tag{S13}$$

In this case, the likelihood ratio uses the prior rather than the current search model density as in (S11). Consequently, the old samples get used in the same way as they would have at their own iteration, and their impact does not diminish as iterations proceed, other than by virtue of the factor $P(S^{(t)}|\mathbf{x}_i^{(s)})$. In particular, the only change in the weight of a sample as it gets used in later iterations arises from the updating of the relaxed desired property, $S^{(t)}$. This suggests that it may be more useful to keep old samples in *CbAS* than in other methods that update with a weighted ML objective.

It's interesting to note that this correct use of previous samples, by virtue of an importance sampling weight, is conceptually similar to some off-policy procedures in reinforcement learning (Precup et al., 2001; Peshkin & Shelton, 2002; Tang & Abbeel, 2010).

S6. Experimental Details

Here we provide the necessary details to run the experiments described in the main text. In what follows, when we specify model architectures we use the notation `LayerType(OutputShape)` to describe layers, and the notation `Layer1(Out1)` $\rightarrow$ `Layers2(Out2)` to denote that `Out1` is given as the input to `Layer2`.

S6.1. Methods details

Weighted Maximum Likelihood methods Similar to *CbAS* the methods *RWR*, *DbAS*, and *CEM-PI* are weighted maximum likelihood methods that update the parameters of a generative model by taking samples from the model, $\mathbf{x}_i \sim q(x|\boldsymbol{\phi}^{(t)})$ and optimizing the objective:

$$\boldsymbol{\phi}^{(t+1)} = \underset{\boldsymbol{\phi}}{\text{argmax}}\sum_{i=1}^{M} w(\mathbf{x}_i)\log q(\mathbf{x}_i^{(t)}|\boldsymbol{\phi}). \tag{S14}$$

The methods differ in the definition of these weights:

- In *CbAS*, $w(\mathbf{x}_i) = \frac{p(\mathbf{x}|\boldsymbol{\theta}^{(0)})}{p(\mathbf{x}|\boldsymbol{\theta}^{(t)})}P(S^{(t)}|\mathbf{x}_i)$

- In *DbAS*, $w(\mathbf{x}_i) = P(S^{(t)}|\mathbf{x}_i)$
- In *RWR*, $w(\mathbf{x}_i) = \frac{e^{\alpha \mathbb{E}_{p(y|\mathbf{x}_i)}[y]}}{\sum_{i=1}^{M} e^{\alpha \mathbb{E}_{p(y|\mathbf{x}_i)}[y]}}$, where $\alpha = 50$ for our experiments (chosen based on a grid search to best optimize the oracle expectation).
- In *CEM-PI*, $w(\mathbf{x}_i) = \mathbb{1}_{\{PI(X) \geq \beta^{(t)}\}}(\mathbf{x}_i)$ where $PI(\mathbf{x})$ is the probability of improvement function and $\beta^{(t)}$ is adjusted according to the methods of CEM (there is a parameter in CEM that corresponds to our Q, which we set to 0.8 for *CEM-PI*).

VAE architecture The following VAE architecture was used for all experiments. The VAE encoder architecture is `Input(L, 20)` $\rightarrow$ `Flatten(L*20)` $\rightarrow$ `Dense(50)` $\rightarrow$ `Dense(40)`. The final output is split into two vectors of length 20, which represent the mean and log-variance of the latent variable, respectively. The decoder architecture is `Input(20)` $\rightarrow$ `Dense(50)` $\rightarrow$ `Dense(L*20)` $\rightarrow$ `Reshape(L, 20)` $\rightarrow$ `ColumnSoftmax(L, 20)`. Note that the 20 comes from both the number of amino acids, which encode proteins, and the fact that we set the dimensionality of our latent space to be 20.

For the Gomez-Bombarelli methods, this is augmented with a predictive network from the latent space to the property space `Input(20)` $\rightarrow$ `Dense(50)` $\rightarrow$ `Dense(1)`

FB-VAE **parameter settings** A major implementation choice in *FB-VAE* is the value of the threshold used to decide whether to give 0/1 weights to samples. We found that setting the threshold to the 80^{th} percentile of the property values in the initial training set gave the best performance, and used that setting for all tests presented here.

FB-VAE **implementation** We note that a minor modification to the *FB-GAN* framework was required to accommodate a VAE generator instead of a GAN. Specifically we must have the method sample from the distribution output by the VAE decoder in order to get sequence realizations, rather than taking the argmax of the Gumbel-Softmax approximation output by the WGAN.

Oracle Details The oracles for the GFP fluorescence task are neural network ensembles trained according to the method in (Lakshminarayanan et al., 2017) (without adversarial examples), where each model has the architecture `Input(L, 20)` $\rightarrow$ `Flatten(L*20)` $\rightarrow$ `Dense(20)` $\rightarrow$ `Dense(2)`.

S7. Fluorescence data set

These data consisted of 50,000 protein sequences each with fluorescent readout. These data showed a clear bimodal distribution of fluorescence values; we retained only the top 34,256 fluorescent proteins (corresponding to the mode with higher fluorescence values) in order to simplify our simulations., consisting of 50,000 protein sequences each with fluorescent readout. These data showed a clear bimodal distribution of fluorescence values; we retained only the top 34,256 fluorescent proteins (corresponding to the mode with higher fluorescence values) in order to simplify our simulations. Also see Figure S2

S8. Protein stability analysis

To estimate protein stability for each sequence, initially, the wild-type protein is relaxed in Cartesian space using the Rosetta FastRelax protocol. Then, the best rotameric side-chain conformations for wild-type sequence and mutation are determined, and is followed by FastRelax in Cartesian space allowing movements in the backbones of a three-residue window around all of each mutated residues and all the sidechains in the protein. This allows more degrees of freemdom to move compared to the original implementation(Park et al., 2016), better accounting for larger structural changes anticipated from more than one mutation in the sequence.

We used this approximate stability estimation method to analyze the sequences chosen by each method (*i. e.,* , those with the highest oracle values from the tests described in Section 4.2 of the main text). Specifically, we calculated the stabilities for each of the nine runs (3 oracles times 3 randomly-initialized runs for each oracle $= 9$ total sequences) for each method. The average stabilities found from these 9 runs, reported as the difference in free energy, $\Delta\Delta G$, between the wild type

GFP sequence and the tested sequence, are shown in Table S1. Negative or small positive values of $\Delta\Delta G$ indicate that the tested sequence is more or only slightly less stable than the wild type GFP, while large positive values indicate that it is substantially less stable, and may not even fold properly. We additionally report the average number of mutations away from the wild type of the tested sequences. The main point here is that methods which do not effectively make use of a prior can be taken to non-sensible parts of the space. For example, *CEM-PI* and *RWR* both have rather low stability (scores greater than 5.0).

Method	$\Delta\Delta G$ (kcal/mol)	Number of Mutations
AM-VAE	-1.8060	17.0
DbAS	0.0805	13.55
GB	1.1970	2.0
CbAS	1.5804	2.55
FBVAE	1.8571	4.33
GB-NO	2.7433	8.66
RWR	6.4537	21.77
CEM-PI	11.4296	28.4

Table S1. Results of protein stability analysis. The first column reports the method, the second, the average stability score, and the final column reports the average number of mutations of each sequence from the wild type GFP sequence.

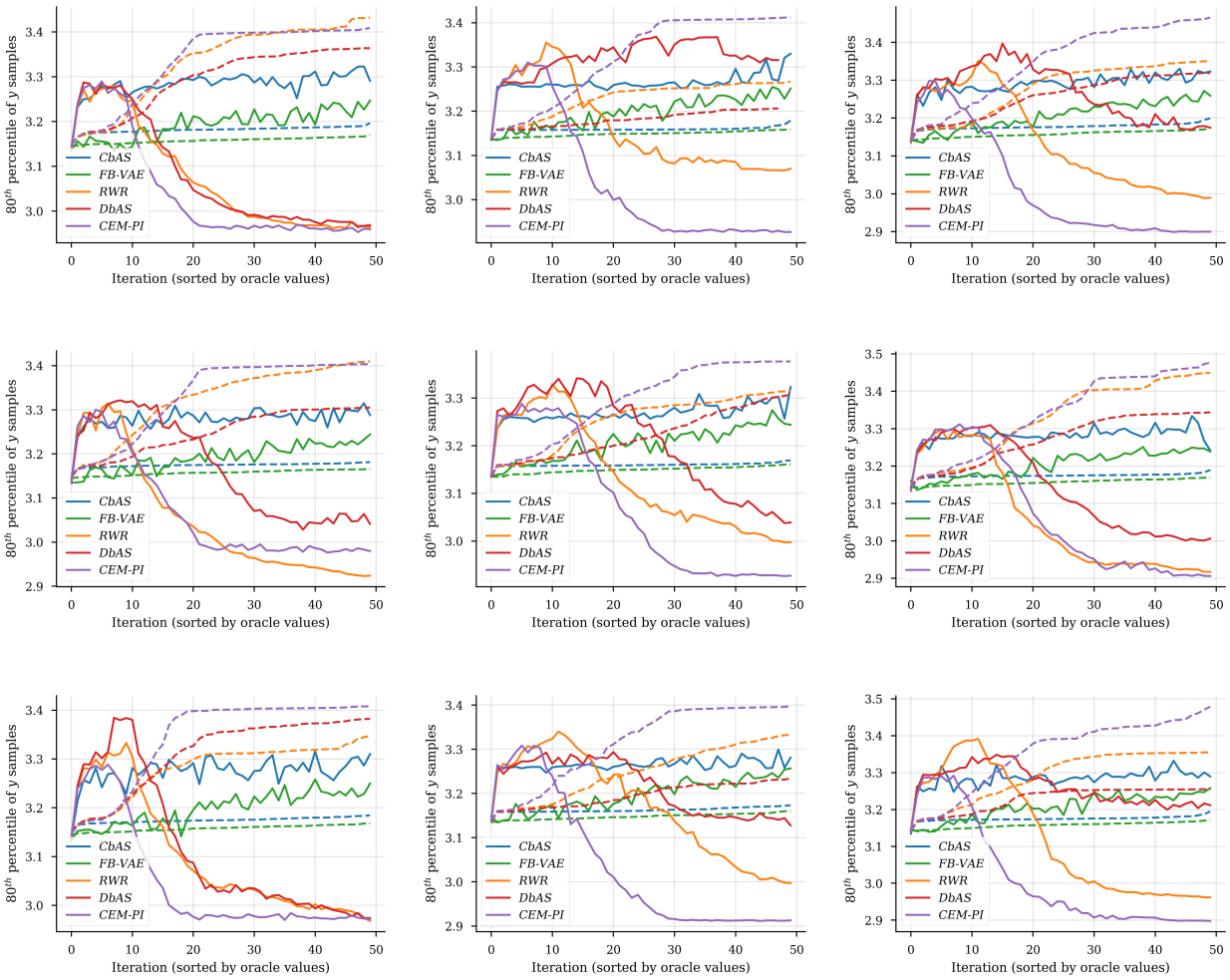
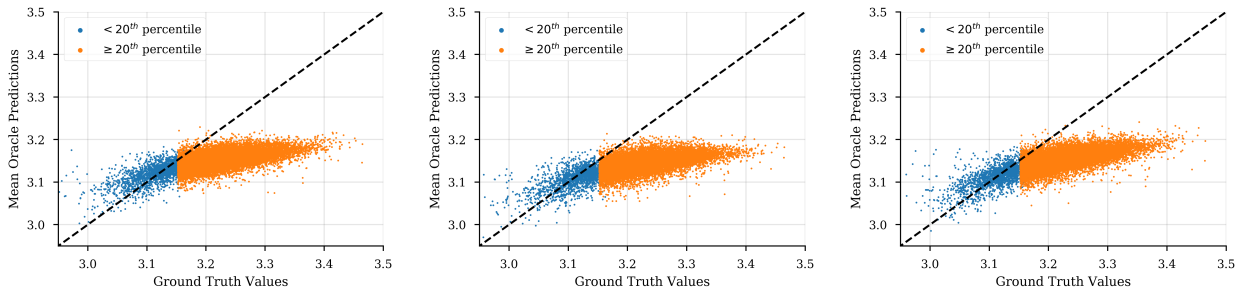


Figure S1. Trajectory plots of the same type as Figure 2b (main paper) for all runs of the methods reported in Figure 2a (main paper). Each column shows results for different oracle, namely (a) the ensemble-of-one oracle, (b) the ensemble-of-five oracle, and (c) the ensemble-of-20 oracle. The three columns show three random initializations for each oracle.



(a) (b) (c)

Figure S2. Paired plots for each of the three oracles described in Section 4.2 of the main text. Each point corresponds to a GFP sequence from one of two test sets, described next. The horizontal axis reports the ground truth fluorescence values of the sequence and the vertical axis represents the mean prediction by the oracle for the sequence. Recall that our ground truth data had 6,851 sequences with fluorescence values below the 20$^{\text{th}}$ percentile. Five thousand of these were used to train each oracle, and the remaining 1,851 of them are the test dots shown in blue. The orange dots are all 27,404 sequences with fluorescence equal to or above the 20$^{\text{th}}$ percentile (oracles were not trained on data in this range). The plots correspond to (a) the ensemble-of-one oracle, (b) the ensemble-of-five oracle, and (c) the ensemble-of-20 oracle. We can see that all oracles are severely biased outside of the training distribution, which is one of the pathologies that *CbAS* is designed to avoid.

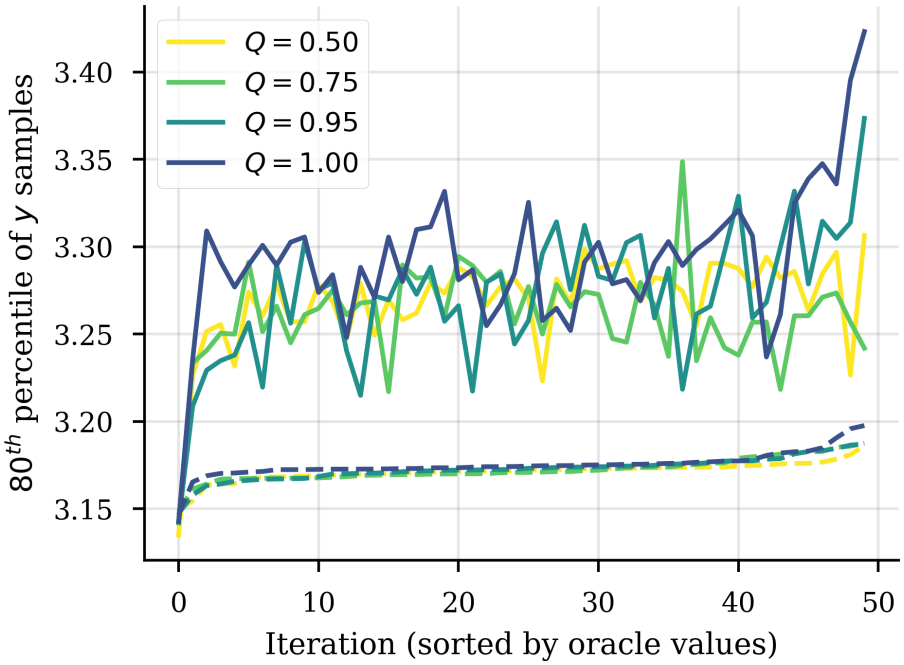


Figure S3. A comparison of *CbAS* trajectories for different value of Q (the quantile threshold parameter described in the main text). Each of these trajectories was generated using the ensemble-of-one for the oracle. We see that *CbAS* is relatively insensitive to the setting of Q; that is, in this example, it would be sufficient to use anything in the range $[0.5, 1.0]$.

Supplementary References

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 6402–6413. 2017.

Park, H., Bradley, P., Greisen, P., Liu, Y., Mulligan, V. K., Kim, D. E., Baker, D., and DiMaio, F. Simultaneous Optimization of Biomolecular Energy Functions on Features from Small Molecules and Macromolecules. *Journal of Chemical Theory and Computation*, 12(12):6201–6212, 2016. doi: 10.1021/acs.jctc.6b00819.

Peshkin, L. and Shelton, C. R. Learning from scarce experience. *arXiv*, cs.AI/0204043, 2002.

Precup, D., Sutton, R. S., and Dasgupta, S. Off-policy temporal difference learning with function approximation. In *ICML*, 2001.

Tang, J. and Abbeel, P. On a connection between importance sampling and the likelihood ratio policy gradient. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 1*, NIPS'10, pp. 1000–1008, USA, 2010.