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 $\forall i \in [1 \ldots M]$.

```
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 $Q^{\text{th}}$ 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:

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

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

- 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 80th 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.

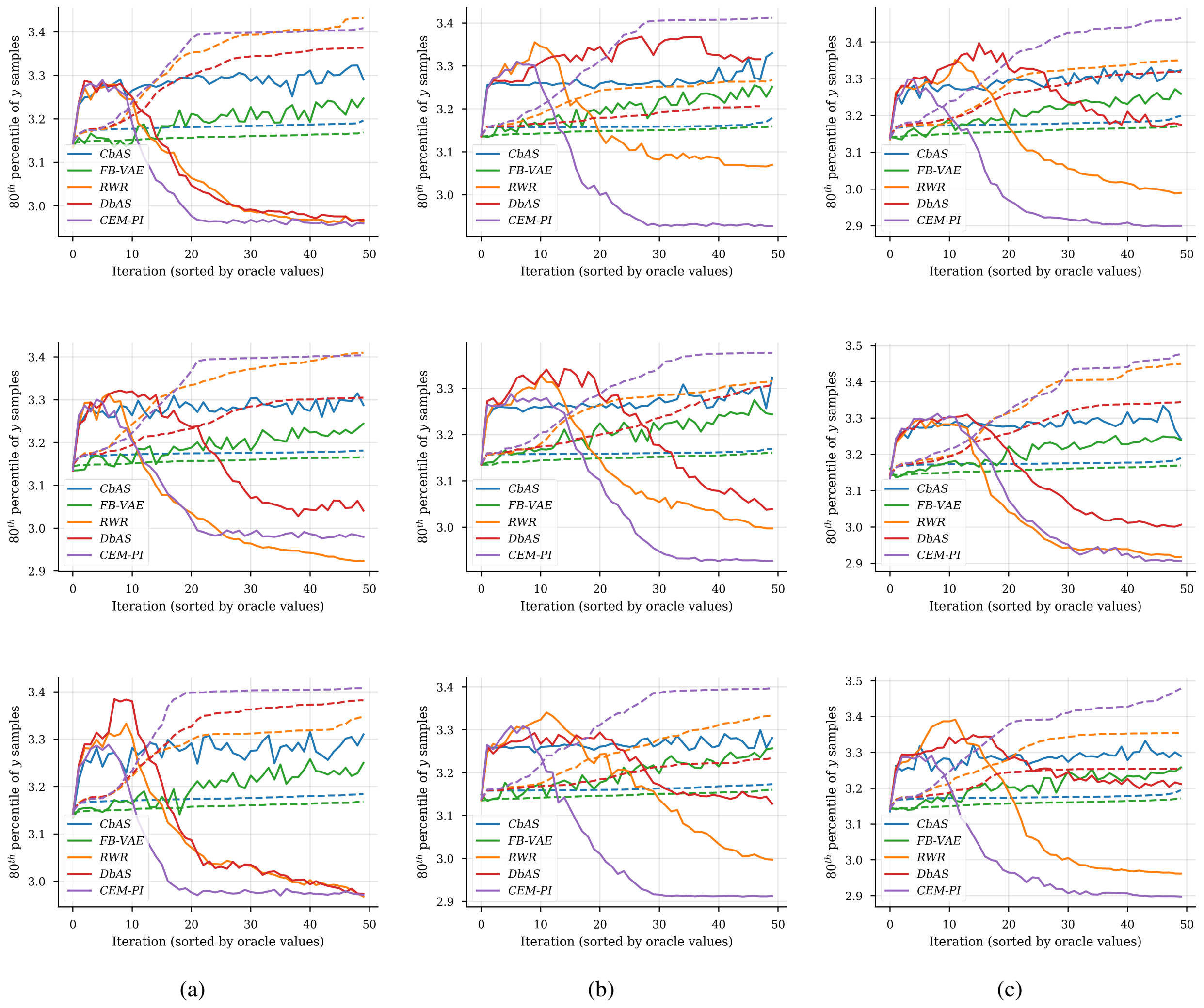
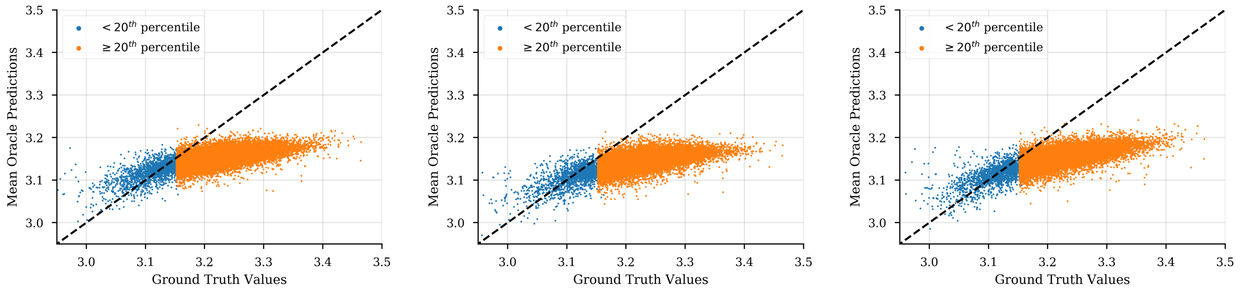


Figure S1. Trajectory plots of the same type as Figure 2b (main paper) for all runs of the methods reported in Figure 2a (main paper). Each column shows results for different oracle, namely (a) the ensemble-of-one oracle, (b) the ensemble-of-five oracle, and (c) the ensemble-of-20 oracle. The three columns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

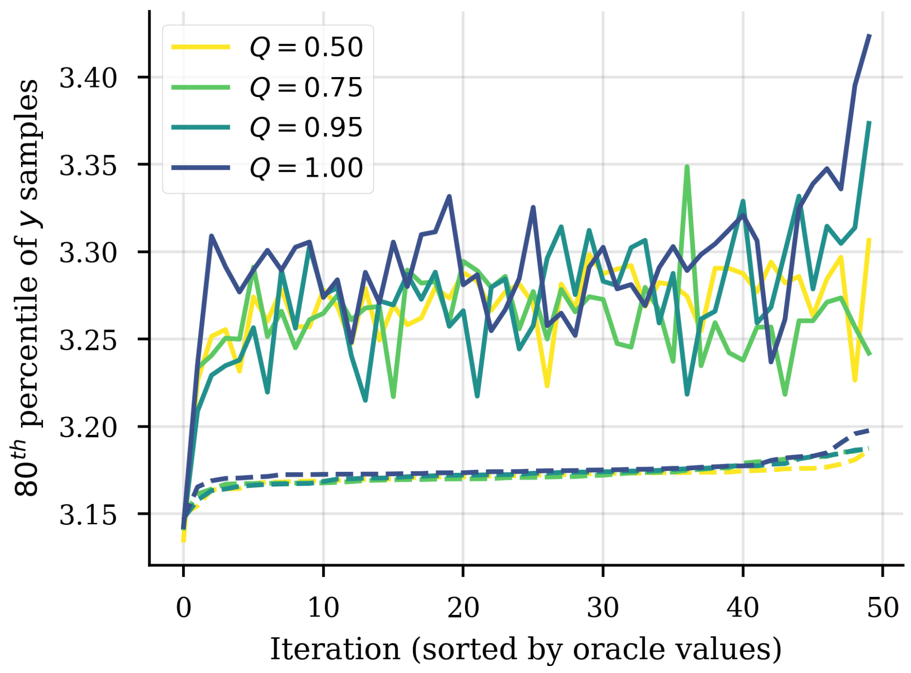


Figure S3. A comparison of *CbAS* trajectories for different value of Q (the quantile threshold parameter described in the main text). Each of these trajectories was generated using the ensemble-of-one for the oracle. We see that is relatively insensitive to the setting of Q; that is, in this example,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