A Bayesian Model for Experiment Choice in Synthetic Biology

Robert P. Goldman[1] Puja Trivedi[1,3]
Daniel Bryce[1] Matthew DeHaven[1] Alex Plotnick[1]
Peter L. Lee[2] Joshua Nowak[2] Vanessa M. Biggers[2]
Trissha R. Higa[2] Jeremy P. Hunt[2]
[1] SIFT, LLC, Minneapolis, MN USA
[2]Strateos, Inc. Menlo Park, CA USA
[3]University of Michigan Ann Arbor, MI USA

Abstract

We are working on an experiment planning system for synthetic biology. As part of this effort, we wish to identify the most informative experiments to perform, based on our current state of knowledge about some design or designs. To do so, we introduce a hierarchical Bayesian model of genetic circuits, assess its predictive adequacy, and explain how it is used to score candidate experiments. We conclude with a discussion of future research directions, including model extensions, improvements to the inference methods, and more ambitious uses of the predictive model.

Introduction

In many emerging fields like Synthetic Biology, engineering design is closely allied to scientific investigation, because in many cases our ability to generate artifacts exceeds our ability to predict those artifacts' characteristics from their design. In synthetic biology, there is a high capability to engineer organisms' genomes, but much less ability to predict the implications of genomic changes. At the same time, with increasing automation comes the ability to perform many more experiments – design constructions followed by measurements of the resulting artifact. Even so, the design spaces are enormous, so we need techniques to better predict performance from design.

The intended use of the model is shown in Figure 1: we construct the model, and train it using whatever training data are available. Then we use the model to generate *posterior predictive* samples ("Projected results") – hypothetical experimental results – from the trained (posterior) distribution for various experimental configurations. We then apply one or more scoring methods on the sets of posterior predictive samples to evaluate the experimental configurations. This information is used to choose new experiments, whose outcomes can be fed back into the model ("Model Update"), which is further trained. This process repeats indefinitely.

Our paper contributes a computable, adaptive Bayesian method for choosing experiments in synthetic biology, and in other related fields. We show how to construct an effectively computable model of the output of genetic circuits, and use it to score parts of the experimental space. We present a cross-validation assessment of the model, and describe the next steps in our research.

Yeast Gate Design

The test case for our work is investigation of two-input logic gates implemented in yeast cells, as developed by Gander et al. (2017). These circuits take *promoters* which they treat as input signals, either high or low, depending on concentration, compute the boolean function through a sequence of genetic operations, and then signal their response through fluorescence. This output fluorescence signal is read by flow cytometry. An example circuit diagram is given in Figure 2.

This set of experiments aims to 1. replicate the original results; 2. identify factors that account for (in)correct functioning of the gates and 3. characterize the operational envelope of the circuits: specifically the growth conditions under which the circuit will (not) function correctly. "Correctness" of circuit function is defined in terms of the proportion of cells that exhibit high (low) output by fluorescence above (below) a defined threshold. Growth conditions explored in these experiments include choice of growth medium, incubation temperature, optical density (the density of the cell populations), etc.

The experimental data we have used in the work described in this paper have been run in an commercial automated wet lab owned and operated by Strateos (formerly Transcriptic).[1] Their laboratory accepts experimental requests via the internet, using a programmatic API, executes them robotically according to parameterized protocols, and then uploads the resulting data sets.

Our intent is to use Bayesian probabilistic predictive models to support automatic or semi-automatic experiment planning: here what cell strains and growth conditions should be run to gain the most information. While this *specific* model aims at predicting the results of flow cytometry measurements of these yeast gates, our overarching aim is to develop a general methodology for experiment planning.

Framing the Problem

The model aims to identify the most interesting strains (in this case a "strain" is an organism/gate paired with

[1] `https://strateos.com/`

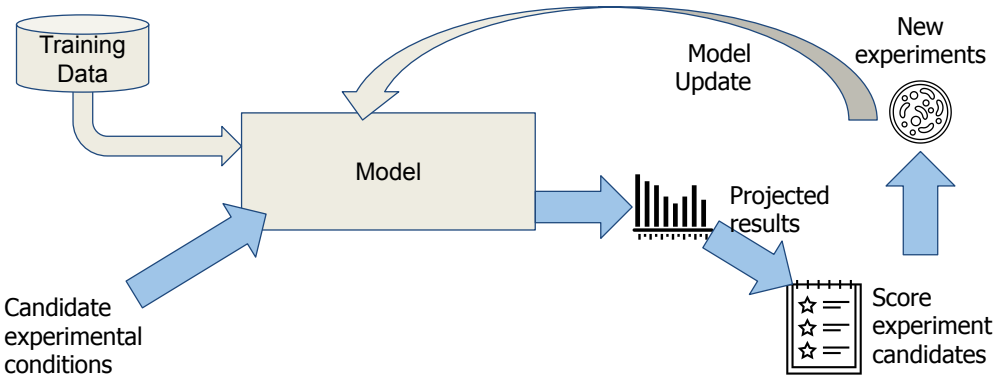


Figure 1: Intended application of the model.

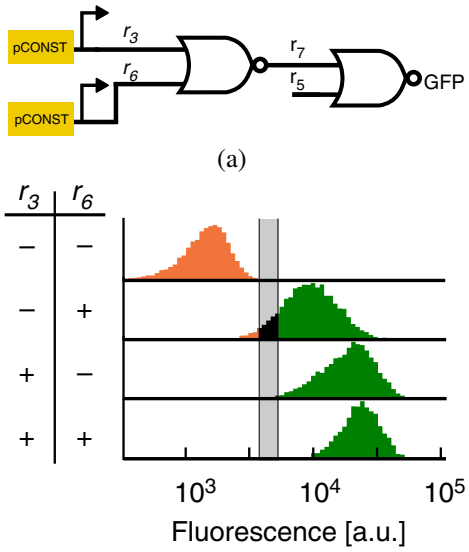


Figure 2: Example yeast gate circuit: (a) circuit diagram; (b) output levels. Gander et al. (2017).

an input) and growth conditions to experiment with. We assess these experiment candidates by measuring them in terms of *entropy* – where entropy is the inverse of information, a measure of our *ignorance* about a random variable. Entropy ($H(P)$) is defined as

$$H(P(X)) = \sum_x P(X = x) \log_2 P(X = x) \quad (1)$$

For a Bernoulli variable, entropy varies between 0 and 1 (see Figure). Note that this is Shannon's definition of entropy, which is defined over discrete distributions, rather than thermodynamic entropy. We return to this topic later.

Entropy is a common proxy utility measure used in decision-theoretic approaches to information-gathering where information gathering is done for its own sake, rather than in service of a decision with defined costs

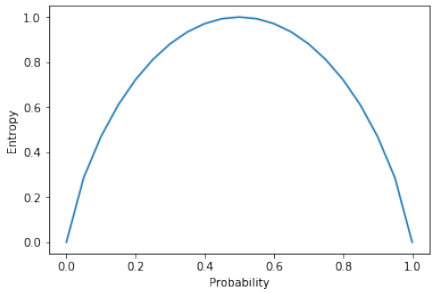


Figure 3: Entropy of Bernoulli random variable.

and benefits.

We wish to know under what circumstances (what growth conditions) these circuits will perform correctly. So we operationalize "perform correctly," and then we use the method of *posterior predictive sampling* to predict when the circuits will and will not work correctly. We do not simply use "works correctly" as the measure, however – instead we ask what is the *entropy* of the question "does this circuit work correctly?" Entropy will be low if we are certain that the circuit will function correctly or if we are certain that it will *not* function correctly. Intuitively, this means that situations where we have little information are the most interesting.

Model Design

We have created a hierarchical Bayesian model that predicts the mean fluorescence output of experimental wells of strains of yeast, as measured by flow cytometry, under varying growth conditions. A hierarchical Bayesian model involves not just priors over the model's basic parameters – in this case a strain's ideal output, the effect of incubation temperature on output, etc. – but also *hyperparameters*. These hyperparameters provide the priors for the basic parameters, and also support *generalization*. For example, we have a hyperparameter that is the prior mean of the effect of a particular growth medium on the output of all strains, and a parameter that is the mean effect on a single strain. As we train the model, we update the parameter, and

through the parameter, the corresponding hyperparameter. In turn, that hyperparameter provides updated information to other strains. The variance hyperparameter learns how far strains tend to differ from each other in the effect of this particular growth medium. Specific priors and qualitative structure were chosen through discussion with biologists, and interactively debugged through examining predictions (samples from the prior) for reasonableness, and through fixing issues in sampling. Betancourt (Betancourt 2017) points out that issues in HMC sampling (see below), are diagnostic of specific issues in model design and parameters.

The structure of our model is given in Figure 4. This illustration suppresses many details: we show only the hyperparameters (light blue), and the variables specific to the AND gate. The box in the lower right indicates the presence of models for the other 5 gates, which are identical in structure to that of the AND gate. In this figure, the light blue nodes are the hyperparameters; the green nodes are the AND gate parameters; the gray nodes are model inputs, both predictors (at the top) and observed outputs (at the bottom).

The hierarchical model structure follows two core design principles: first, that experimental factors, such as optical density and output mean, have the same prior but varying effects across *gates*; second, that within individual gates, the factors have different effects across *strains/inputs*. For example, $\beta_1(\mathrm{od})$ and $\beta_2(\mathrm{od})$ are the hyperparameters for cross-gate effects of optical density. For the `AND` gate, the corresponding parameters are: $\beta_1(\mathrm{od}, \texttt{AND})$ and $\beta_2(\mathrm{od}, \texttt{AND})$, which get their prior means from the corresponding hyperparameters. Note that each of the β parameters for `AND` is a vector of four values, one for each strain (input). All of these are vectors with one entry per input value.

Ideal output	$\mathrm{Out}(g)$
Medium influence	$\delta(g)$
OD influence	$\beta_1(\mathrm{od})$ and $\beta_2(\mathrm{od})$
Temperature influence	$\beta_1(\mathrm{temp})$ and $\beta_2(\mathrm{temp})$

Table 1: Gate parameters.

The experimental factors in our model correspond to growth conditions, and each has a nominal value. For the temperature and optical density, our priors treat the nominal value as a maximum, at an influence of zero, with negative effects above and below. Note that this does *not* prevent our model from learning that some other value may be optimal. To capture this, we model the effects of OD and temperature with quadratic functions. For example, for temperature:

$$t_{\mathrm{eff}}(t) = \beta_1(\mathrm{temp})t - \beta_1(\mathrm{temp})t^2 + err$$

The equation for the effect of optical density is similar. For growth media, we "peg" the effect of the standard medium at zero, and give uninformative priors, with a mean of zero to the others. This has the beneficial effect of removing pointless variation from the search space.

The remaining random variable is the "ideal output" of the strain, under nominal conditions. The hyperparameter for the ideal outputs gives the "ideal" high and low fluorescence values, based on the mean over all high and low outputs. The corresponding parameter for each strain takes its prior from the ideal high or low output, depending on the gate's truth table.

Our model is implemented using the PyMC3 library (Salvatier, Wiecki, and Fonnesbeck 2016), for probabilistic programming in Python.

Our priors were weakly informative, based on the assumption that the yeast gates essentially worked, along the lines laid out in the original paper (Gander et al. 2017). Figure 5 shows samples from the prior distribution for the OR gate, under normal growth conditions, with actuals superimposed; it can be seen that this gate does not, in fact, work very well for single input high cases.

Training the Model

Exact inference using this complex model is infeasible, not only because of computational complexity, but because no closed form is known. Therefore, to train the model, we have used Hamiltonian Monte Carlo (HMC) (Betancourt 2017) and the No U-Turn Sampler (NUTS) (Homan and Gelman 2014). HMC with NUTS provides an effective method of quickly training probabilistic models with complex structure. Other sampling methods are available, as are variational inference (VI) approximations. We have confined ourselves to HMC so far, as it is the current best-in-class MC method. We have made some attempts to use VI, but have not found a method yet that handles our model successfully.

We have not collected detailed timing information, but on a high speed machine, a training run with a data set with approximately 7600 members, training takes between 3 and 4 hours, on a blade server with 2 x Intel® Xeon® CPUs, E5-2680 v3 with 2.50GHz clock speed and 384Gb memory (although our jobs are not especially memory-hungry). Note that each of the 7600 data points used here are aggregate metrics of as many as 30,000 individual flow cytometry measurements. While the current implementation uses multiple cores, the parallelism at the sampling level is quite limited, and does not take advantage of GPUs. We expect that the training process can be significantly sped up, and will investigate this later. However, 3-4 hours to choose experiments on a weekly rotation, *is* acceptable.

Generating Predictions and Scoring

Because our Bayesian model is generative, it can generate hypothetical samples from the trained model. Using those hypothetical samples, we can compute a large variety of predictions and scores for those predictions (probability correct, RMSE, etc.). For this application, we score regions of the experimental space based on how interesting they are, in order to choose the best next measurements. These regions are based on discretiza-

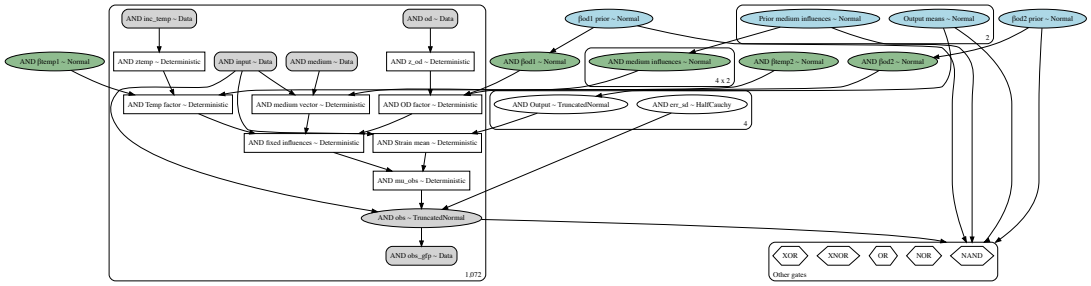


Figure 4: Outline of the model, with the `AND` gate expanded and the other gates suppressed.

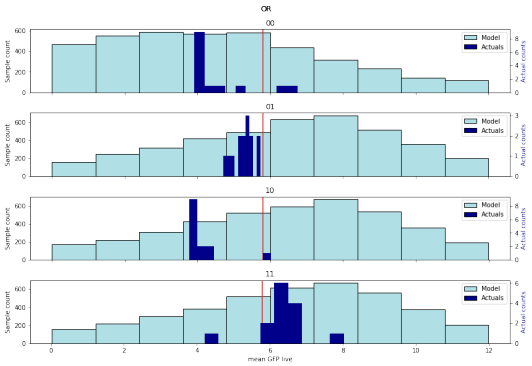


Figure 5: Prior samples for OR gate compared with actuals.

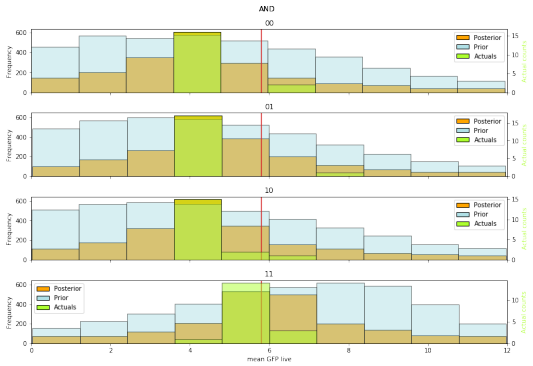


Figure 6: Priors, posterior, and actuals for the AND gate.

tions of the condition space that change based on external considerations. We first sample from the regions to choose a set of concrete conditions to input into the model, sample from the trained model based on these concrete conditions, score the results, and then aggregate the scores over the region.

We generate *posterior predictive samples* from the trained model to make predictions and to evaluate the model. Since our model is generative, we can give it hypothetical sets of predictors and extract samples from the posterior distribution conditioned on those predictors. In practice, this is implemented by taking from the posterior trace values of the parameters and hyperparameters of the model, combining them with the predictors, and then randomly generating conditional samples for the output variables. An example of posterior predictive sampling is shown in Figure 6. It can readily be seen how the posterior samples are a mixture of the prior and the actual data, and how extreme values, there to accommodate updates from the data, have been pruned away from the prior.

Recall that the intended use of our model is to evaluate possible experiments. For this purpose, the module containing the system accepts scoring queries. Other parts of the system can ask for the model's current scores for a set of experimental conditions, a *condition set*. A condition set is a bounded region of condition space, which is in turn a discretized representation of the full condition space. Recall that while some of the dimensions of the experimental space are discrete (`e.g.`, growth medium), others are continuous (`e.g.`, incubation temperature). Although continuous, and represented as such in our model, there is a limit to how precisely these conditions can be manipulated, which imposes a most refined discretization of condition space. For example, incubation temperature is discretized into degrees celsius between 30 and 40. OD is represented on a log scale. A condition set is a set of region bounds in this condition space. For example:

high Osmolarity growth medium

$$0 < OD \leq 1 \times 10^{-4}$$

$$30C \leq Temp \leq 34C$$

To generate samples from the posterior for a particular condition set, we sample values from a uniform distribution over the predictor variables in the condition set that are not given a single value. This gives us a set of random predictors taken from the condition set. For an example of such samples, see Figure 7. Note that the growth medium does not appear here, because we do not combine different growth media in a single region. We show incubation temperature, which is treated as an integer variable (owing to limits in accuracy) and OD, which is varied on a logarithmic scale.

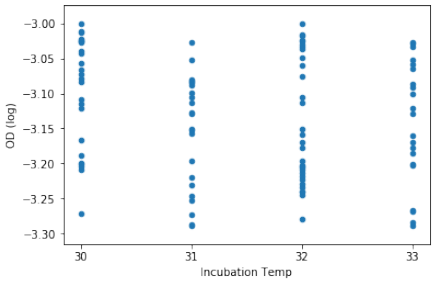


Figure 7: Samples for a condition space region.

For each element of this set, we sample a set of cell fluorescences from the trained posterior model. These provide raw materials for scoring. Currently we compute two scores: proportion correct (number of output samples on the right side of the threshold), and an entropy score for this proportion. The entropy score is computed using a histogram of values over the condition set. This means that the entropy for a condition set is a function not only of the state of information, but also of the *size* of the region.

The heatmaps in Figure 8 show the outputs of our model trained on a large dataset, contrasted with predictions from the prior. The coverage of this dataset is shown in Figure 9. In the top row of Figure 8 we see heatmaps from samples of the prior. Along the x-axis are the strains – gates and inputs, grouped by gates, in alphabetical order, so that the first four columns are for the AND gate. Along the y-axis are the growth conditions. These are three dimensional – growth medium, optical density, and incubation temperature – flattened to a single dimension. So the bottom third are the values for the standard growth medium, the middle third for slow growth medium, and the top third are for high osmolarity medium. Within these are four regions of optical density, and within *those* two regions of incubation temperature, normal and high. This dimensional flattening accounts for the horizontal banding in parts of these figures.

The bottom row of Figure 8 shows the results of samples from the posterior (trained) model. A distribution of the training data is shown in Figure 9, from which it can be seen that most of the experimental samples to date were grown in standard medium, and normal incubation temperature. NAND, NOR, XNOR and XOR were more heavily sampled than AND and OR. Comparing Figure 8 (c) and (d), it can be seen that the relationship of probability correct and entropy is not straightforward. Part of this is the non-uniform size of the different regions, especially in optical density. We can see that particular inputs for OR, NOR, XNOR, and XOR are most interesting, under all growth conditions. Interestingly, these appear to be difficult to predict under all growth conditions. We return to this in our discussion.

Evaluating the Model

We conducted a 10-way cross-validation, partitioning the existing data into 90% training and 10% test data, and assessed the model predictions in terms of predicted strain correctness and predicted output value. Figure 10 shows the accuracy with which the model predicted whether a strain would produce the correct output. For each of the test cases, we collected 2000 predictions from the trained model, computed the proportion of predictions that agreed with the (in)correctness of the point in the test data set. Figure 10 shows the proportion correct, averaged over the points in the data set, together with the standard deviation.

Interestingly, when we examine the data, we see that each of the cross-validation sets has some predictions completely wrong, and that of those, all were predictions of the OR gate with input 10. This strain clearly invites further investigation.

The model does quite well at predicting whether or not a strain will provide the right kind of output fluorescence, high or low. It also does fairly well at predicting *absolute* output fluorescence, as can be seen from the Root Mean Squared Error (RMSE) plotted in Figure 11. The RMSE is calculated for each test data point against the set of 2000 predictions, and then is averaged across all the test data points in a trial. So we have not only a mean RMSE, but also a variance, shown by the black bars. It can be seen that this variance is quite high, meaning that some predictions are much better (resp., worse) than others.

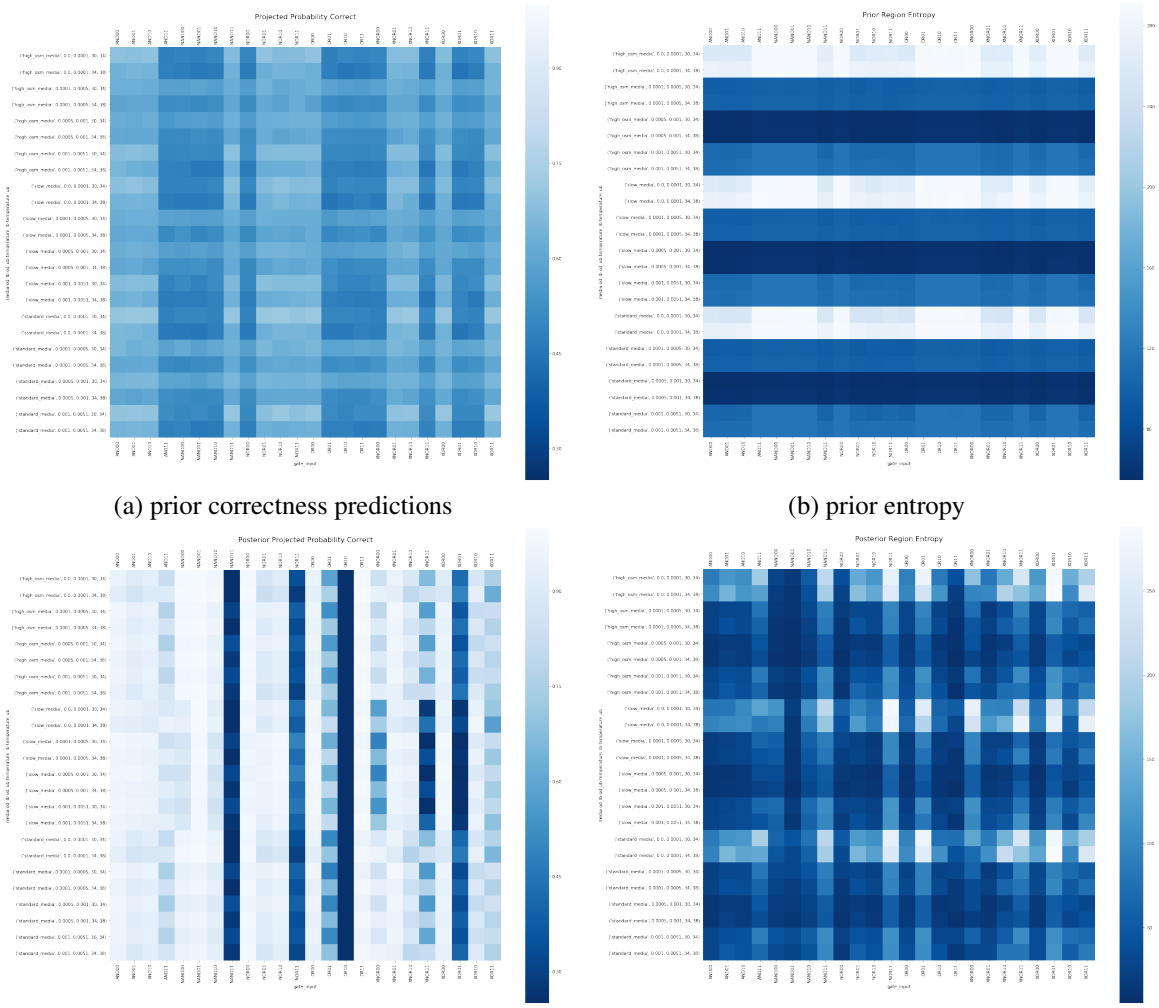
Unsurprisingly, OR10 is also among the worst-predicted strains, which are as follows:

Gate	Input	Gate	Input
AND	01	OR	10
NOR	00	OR	11
NOR	10	XNOR	10

There is no clear pattern of strains among the *bad* predictions, as opposed to the worst. A large proportion of the poorly predicted values were run at a high incubation temperature, and the less common growth media were represented, but even nominal growth conditions were represented. We conjecture that some of these distributions are problematic (see the discussion of XOR01 below). Recall, however, that it is the job of our model to determine what situations are most interesting, not to provide a biologically accurate prediction. Recall also that the entropy measure we use is based on strain correctness, rather than the absolute value of the output.

Conclusions and Future Work

We have described a Bayesian model for prioritizing experiments to assess designs in synthetic biology. Despite its complexity, this model can be trained effectively, and provides useful predictions of future measurements. We are currently working in a number of directions to improve the fidelity and coverage of the



(a) prior correctness predictions

(b) prior entropy

(c) posterior correctness predictions

(d) posterior entropy

Figure 8: Predictions from model.

model, speed inference, and refine the way we use the model's predictions for experiment planning.

Our results raise one issue with the use of entropy as a measure of "interestingness": looking at the heatmaps, we see that certain strains that function unpredictably, `e.g.`, XOR with 01 inputs, are regarded as interesting even in conditions for which we have a relatively large number of samples. Looking at this strain, we can see that it *is* interesting: Figure 12 shows that its output is bimodal. However, there is a potential problem here – if the distribution were unimodal, and predictably, but centered around the hi/lo threshold, the entropy metric would see it as interesting, when it really was not, in the sense that the current experiments would not provide a better characterization of the strain's behavior. We believe that this issue could be addressed by adding a new variable to the model the variance of the circuit output. Currently, our model predicts only the mean output of the well, and only provides a measure of the variance *between* wells, not within a well.

A second issue has to do with how we make experimental choices based on entropy. Entropy provides us with a myopic measure of potential information gain. It does not take into account the effect on other, neighboring regions, of learning about one region in the condition space, nor does it take into account the effects of a combination or sequence of experiments. We are working to combine our model with sequential decision making, to compute a better *value of information* metric.

We are working to make the training and scoring processes faster. PyMC3's implementation of posterior predictive sampling is incompletely vectorized. We are working with the community on a fix. We are also working on incremental training (Angelino, Johnson, and Adams 2016).

We will soon apply this technique to a different problem to evaluate the paradigm's generality.

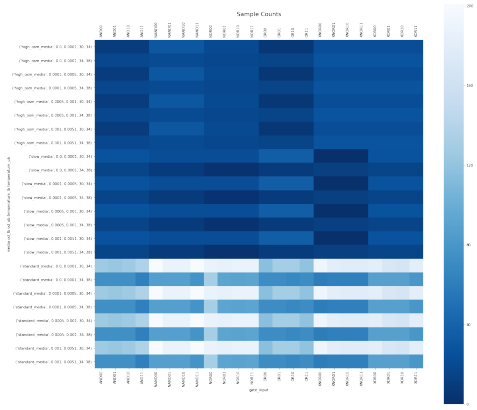


Figure 9: Distribution of samples over strains and condition space.

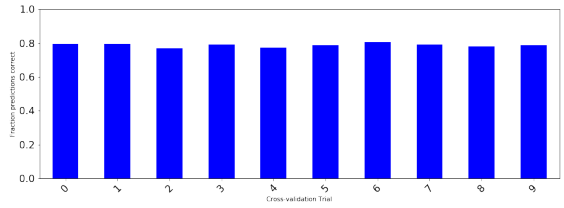


Figure 10: Proportion of correctness predicted correctly.

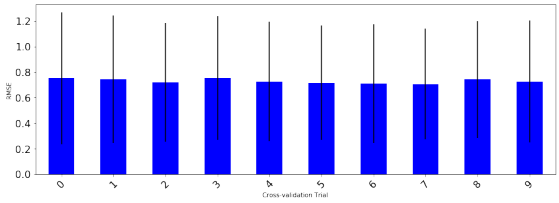


Figure 11: RMSE of the fluorescence predictions.

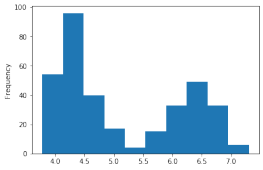


Figure 12: Observations of XOR with input 01.

Acknowledgements

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory under Contract No. FA8750-17-C-0184. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not reflect the views of DARPA, the Dept. of Defense, or the U.S. Government.

Thanks to members of the DARPA SD2 consortium whose advice and explanations helped us formulate the model: Jacob Beal, Nic Roehner, Steven Haase, Devin Strickland, Bree Cummins, Ulli Schächtle, and Vikash Mansinghka. We are also deeply indebted to members of the PyMC3 developer community for advice on how to reformulate the model so that it could be effectively trained using HMC. Particularly helpful were Junpeng Lao, Luciano Paz, and Chris Hartl.

References

[Angelino, Johnson, and Adams 2016] Angelino, E.; Johnson, M. J.; and Adams, R. P. 2016. Patterns of Scalable Bayesian Inference. *arXiv:1602.05221*.

[Betancourt 2017] Betancourt, M. 2017. A Conceptual Introduction to Hamiltonian Monte Carlo. *arXiv:1701.02434 [stat]*.

[Gander et al. 2017] Gander, M. W.; Vrana, J. D.; Voje, W. E.; Carothers, J. M.; and Klavins, E. 2017. Digital logic circuits in yeast with CRISPR-dCas9 NOR gates. *Nature Communications* 8:15459.

[Homan and Gelman 2014] Homan, M. D., and Gelman, A. 2014. The No-U-turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. *J. Mach. Learn. Res.* 15(1):1593–1623.

[Salvatier, Wiecki, and Fonnesbeck 2016] Salvatier, J.; Wiecki, T. V.; and Fonnesbeck, C. 2016. Probabilistic programming in Python using PyMC3. *PeerJ Computer Science* 2:e55.