Machine Learning based Translation Initiation Rate predictor

Zhang M., Holowko M. B., Hayman Zumpe H., Ong, C. S.

November 25, 2020

1 Introduction

One of the main tenets of synthetic biology is design, evaluation and standardisation of genetic parts [**Brophy2014**, **Canton2008**, **Stanton2014**]. This is usually done in terms of the Design-Build-Test-Learn (DBTL) cycle, where the given genetic part or organism are continually improved by going through a number of turns of the said cycle. This normally involves designing the DNA sequence in Computer Aided Design (CAD) software and then physically testing it in a laboratory. Additionally, computer modelling and prediction of part behaviour based on the designed DNA sequence or design of DNA sequence based on expected function can be used[**Yeoh2019**, **Nielsen2016**]. Most of these models are based on either the thermodynamic properties of the involved molecules (DNA, RNA, proteins, etc.) or empirically obtained values describing a relevant to a given design property, like Translation Initiation Rate (TRI) in case of Ribosome Binding Sites (RBS) [**Xia1998**, **Chen2013**, **Reeve2014**]. The biggest limitation for this approach currently is the Learn part of the cycle - there is very limited access to methods and software that can improve designs based on experimental results.

According to Reeve *et al.* there are three main RBS calculators, all predicting the TRI based on the thermodynamic properties of the RBS and the ribosome [**Seo2013**, **Na2010**, **Salis2009**]. Predictions from all of these models are relatively good ($R^2 > 0.8$), but they come with a number of caveats: i) they rely on calculations of free energies that can be hard to estimate with high precision ii) in general, one of the best ways to improve the models' accuracy is by increasing the number of phenomenons taken into account, but this can lead to paradoxically decreased model accuracy due to accumulation of errors [**EspahBorujeni2016**] and iii) by using deterministic coefficients to calculate energies one disregards often stochastic nature of processes in the cells which again increases perceived prediction error [**Goss1998**].

Synthetic biology is currently going through a phase of exponential increase in volume of data produced during experiments [**Freemont2019**]. New experimental methods heavily relying on advances in automation and microfludics allow unprecedented precision and throughput in data generation. These new data-sets can be combined with data reliant machine learning algorithms to generate new models and predictors for use in synthetic biology, vastly improving the DBTL cycle's performance [**Camacho2018**]. In the past few years there was a significant uptake of Machine Learning based approaches in synthetic biology. Jervis *et al.* used support vector machine and neural network to optimise production of monoterpenoid in *Esherichia coli* [**Jervis2019**]. Similarly, Costello *et al.* have used a number of machine learning approaches to analyse time-series multiomics data to predict metabolic pathway behaviour [**Costello2018**]. There were also successful attempts at using deep learning techniques for analysis of big data-sets [**Alipanahi2015**, **Angermueller2016**]. However, the use of machine learning in synthetic biology is still in its infancy and will require additional research to show its full potential.

Here we present how machine learning algorithms can be used as part of the DBTL cycle to predict (Learn) and recommend (Design) variants of RBS with goal of optimization of protein level expression. RBS being one of the key genetic elements controlling protein expression and at he same time having a relatively short sequence is a perfect target for establishing workflows that can be later translated to more complicated systems. We have used Gaussian Process-Upper Confidence Bound and multi-armed Bandits algorithms for prediction and recommendation respectively to analyse and optimise the initiation rates of the designed RBS. Our overall experimental goal was to maximise the Translation Initiation Rate (TIR) by identifying the set of RBS sequences with top TIR scores while minimising the number of DBTL cycle turns that we had to do. We did this by designing a sequential experimental workflow, where we start with either randomised RBS sequences designed to explore the experimental space (designated round zero since only preexisting, literature data is used for its design) or with RBS sequences recommended by the algorithm (subsequent rounds). The designs were then physically constructed in batches of 90 to fit our

automated process (see **Methods** section). After constructing, the plasmids harbouring the new genetic devices are tested in microplate reader and flow cytometer. The results are then fed back to the algorithm for it to recommend the next round of designs.

2 Methods

2.1 Laboratory experimental design

2.1.1 BUILD: Construction of genetic devices

Plasmid Design. We have used the pBbB6c-GFP plasmid for all our designs. This plasmid comes with GFP mut3b CDS inducible with addition of Isopropyl β-D-1-thiogalactopyranoside (IPTG). The original RBS for the GFP CDS was replaced with combination of PCR and isothermal assembly. Primers and the assembly strategy have been generated using the Teselagen DESIGN software (Teselagen Biotechnology).

PCR. PCR amplification of the cloning inserts was done using Q5 High-Fidelity 2X Master Mix (NEB, catalogue no. M0492L). 20 μL reactions were prepared by dispensing each of the 10 μM reverse primers into a well of a 96-well PCR plate using the Labcyte Echo Liquid Handler. A mastermix consisting of polymerase premix, plasmid DNA template, and the single 10 forward primer was prepared by and dispensed by hand. Reactions were run using Touchdown PCR or standard PCR cycling methods in BioRad C1000 thermal cyclers. Then, samples were incubated at 37°C for 60 minutes, followed by a 20-minute heat inactivation step at 80°C. Capillary electrophoresis of PCR products was performed using the Agilent Technologies ZAG DNA Analyzer system. 2μL of each PCR reaction was electrophoresed using the ZAG 130 dsDNA Kit (75-20000bp) or ZAG 110 dsDNA Kit (35-5000bp) (Agilent Technologies, catalogue no. ZAG-110-5000; ZAG-130-5000). ProSize Data Analysis Software (Agilent Technologies) was used to generated gel images from the sample chromatograms and sizes were estimated by reference to the upper and lower DNA markers spiked into each sample and a DNA ladder run in well H12 of each sample plate.

Isothermal DNA Assembly. Constructs were assembled using NEBuilder HiFi DNA Assembly Master Mix (NEB, catalogue no. E2621L). Samples were incubated at 37°C for 60 minutes, followed by a 20-minute heat inactivation step at 80°C. Reactions consisting of the common fragment and the variable fragment were prepared using the Echo acoustic liquid handler, to a final volume of 5 or 10μL. Assemblies were run in the thermal cycler for 1 hour at 50°C, followed by an infinite hold step at 4°C.

***E. coli* transformation.** The DH5α cell line (Thermo Fisher Scientific, catalogue no. 18265017) was made chemically competent using the Mix & Go *E. coli* Transformation Kit & Buffer Set (Zymo Research, catalogue no. T3001). 20μL of cells was aliquoted into each well of a cold 96-well PCR plate and stored at -80°C for later use. Plates of cells were thawed on a -20°C cold block before 3μL of the assembly product was added and mixed using the CyBio FeliX liquid handler. Cells were incubated on a cold block for 2-5 minutes before being plated in a 96 square grid on Omnitrays containing LB (BD, catalogue no. ***) with 34μg/mL chloramphenicol (Sigma, catalogue no. ***). Multiple dilutions of cells in LB were prepared and plated in parallel on Omnitrays in 96 square grid. Plates were incubated overnight at 37°C.

Automated colony picking and culturing. A Singer Instruments PIXL colony picker was used to select individual colonies from the transformation plates. Each selected colony was used to inoculate 1mL of selective medium in a 2mL square well 96 plate. They were then cultured overnight in 37°C with shaking (300rpm).

Glycerol stock preparation. 100μL of sterile 80% (v/v) glycerol and 100μL of overnight culture were combined in the wells of a 96 deep (2mL) round well plate using the CyBio Felix liquid handler. They were then sealed with a 96-well silicon sealing mat and transferred to a -80°C freezer.

Sequencing Sequences that showed TIR on the level of at least 85% of TIR of the consensus sequence were sequenced using Sanger sequencing either at the Australian Genome Research Facility (Brisbane, QLD, Australia) or at Macrogen (Seoul, Republic of Korea).

2.1.2 TEST: Culture analysis

Test strain culture. Overnight cultures were started by inoculating 1mL of LB medium supplemented with 34μg/mL chloramphenicol with 2μL of the glycerol stock in a 96 deep (2mL) round well plate. Cultures were incubated at 37°C with shaking (300rpm) for 17 hours. The following morning, 20μL of overnight culture was

added to 980μL of fresh selection medium and precultures were grown at 37°C with shaking in 2mL round well 96 plate. After 90 minutes, two parallel cultures were prepared in flat-bottom clear polystyrene 96-well plates and were induced with 500μM IPTG – one plate of 300μL cultures for flow cytometry analysis (induced with 1.5μL of 0.1M IPTG) and one plate of 200μL cultures for plate reader analysis (induced with 1.0μL of 0.1M IPTG).

Microplate spectrophotometry. The plates were tested in Cytation5 microplate reader. Cytation 5 acquisition and incubation/shaking settings

Flow cytometry. The plates were tested in Beckman Coulter CytoFLEX S flowcytometer. CytoFLEX S acquisition settings

2.2 Machine learning experimental design

We show the flowchart of machine learning based experimental design in Figure 1. To automatically design the RBS sequences in batch using machine learning, we can logically divided the workflow into two parts: 1) **LEARN**: A regression algorithm which takes the RBS sequences as input features and TIR scores as labels, trains on sequences with known labels and returns the predicted TIR scores and the respective confidence intervals. 2) **DESIGN**: An online learning approach which recommends the RBS sequences based on the predicted TIR scores and confidence intervals.

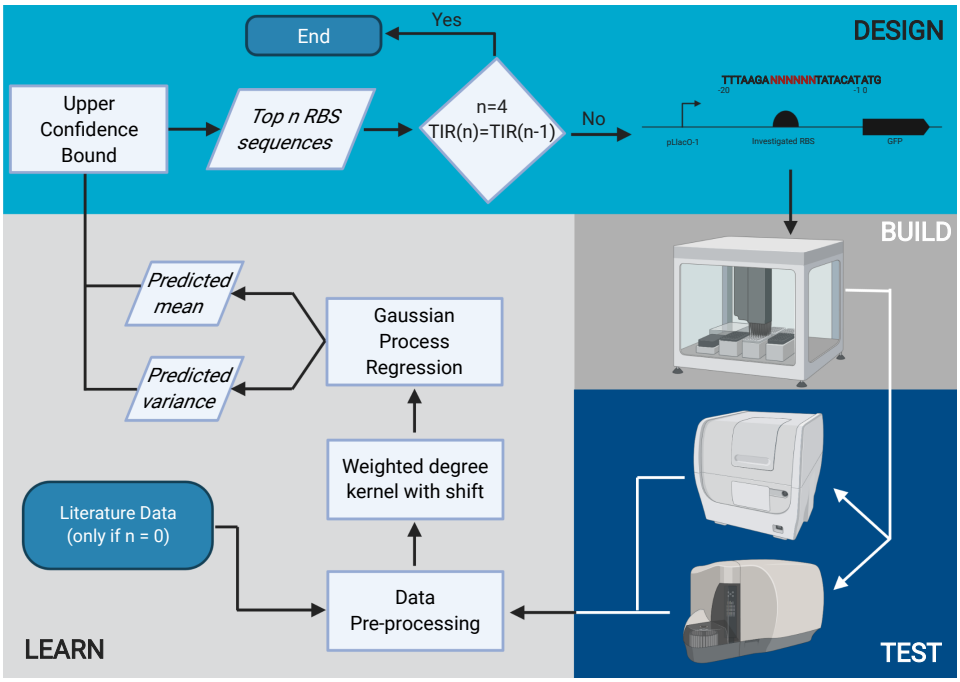


Figure 1: Flowchart of machine learning based experimental design.

2.2.1 LEARN: Gaussian Process Regression with Weighted Degree Kernel

To find RBS sequences with the highest possible TIR score after a total number of rounds N, we consider our experimental design problem as sequential optimisation of an unknown reward function $f : \mathcal{D} \to \mathbb{R}$, where $\mathcal{D}$ is the set containing all RBS sequence points, and $f(\mathbf{x})$ is the TIR score at $\mathbf{x}$. In each round t, we choose a set of m points $\mathcal{S}_t \subset \mathcal{D}$ and observe the function values at each point in the selected set $\mathcal{S}_t$, i.e. $y_i = f(\mathbf{x}_i) + \epsilon_i$, for all $i \in \mathcal{S}$, where ϵ_i is the noise (we assume that the noise is following Gaussian distribution with unknown mean and variance). This noise is influenced by the accuracy of the RBS predictor and other experimental sources of interference (e.g. time, temperature, operator, etc.).

For regression model, we have used the *Gaussian Process Regression (GPR)*. A Gaussian process regression model [**Rasmussen2004**] is a Bayesian approach which provides uncertainty measurements on predictions. We model f as a sample from a *Gaussian process* $\mathcal{GP}(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$, which is specified by the mean function $\mu(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$ and

the kernel (or covariance) function $k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - \mu(\mathbf{x}))(f(\mathbf{x}') - \mu(\mathbf{x}'))]$. GPR can predict both the posterior mean and posterior variance. The posterior variance represents the level of uncertainty for the prediction.

The choice of covariance (kernel) function is critical for accurate predictions, since it controls smoothness and amplitude of the function we model. To represent the RBS sequences and formulate the similarity between sequences, we use the *weighted degree kernel with shift* (WDS) [**ratsch˙rase˙2005**] to specify the kernel function of GP. WDS is a type of a string kernel, which takes two sequences (strings) as inputs and outputs a scalar value which represents the similarities between the two sequences. WDS kernel does this by counting the matches of substrings of a certain length (i.e. kmers) that constitute the sequence. The maximum substring length is specified by ℓ. The WDS takes into account the positional information by counting substrings starting from different positions, where the start position is specified by l. Additionally, the WDS kernel considers the shifting of substrings, with the maximum shift specified by s. For example, for two core sequences $ACCTGA$ and $CCTGAA$, there is a common part $CCTGA$ which is begins at the 2nd nucleotide in sequence 1 and at the 1st nucleotide in sequence 2, hence the calculated shift would be 1.

Let $\mathbb{I}$(A) is the indicator function, which equals 1 if A is true and 0 otherwise. Then $\mathbb{I}(\mathbf{x}_{[l+s:l+s+d]} = \mathbf{x}'_{[l:l+d]})$ counts the matches of substrings of length d between $\mathbf{x}$ starting from position $l+s$ and $\mathbf{x}'$ starting from position l. This is similarly done for $\mathbb{I}(\mathbf{x}_{[l:l+d]} = \mathbf{x}'_{[l+s:l+s+d]})$. By having these two terms considering substrings of two sequences with starting positions differing by s characters, the WDS can measure shifted positional information. When $s = 0$, the kernel function counts the matches with no shift between sequences. Let $\mathbf{x}, \mathbf{x}'$ be two RBS sequences with length L, the WDS kernel is defined as

$$k_\ell^{WDS}(\mathbf{x}, \mathbf{x}') = \sum_{d=1}^{\ell} \beta_d \sum_{l=1}^{L-d+1} \gamma_l \sum_{s=0, s+l \le L}^{S(l)} \delta_s \left(\mathbb{I}(\mathbf{x}_{[l+s:l+s+d]} = \mathbf{x}'_{[l:l+d]}) + (\mathbb{I}(\mathbf{x}_{[l:l+d]} = \mathbf{x}'_{[l+s:l+s+d]}) \right), \tag{1}$$

where $\beta_d = \frac{2(\ell-d+1)}{\ell(\ell+1)}, \delta_s = \frac{1}{2(s+1)}$, γ_l is a weighting parameter over the position in the sequence, where we choose to use a uniform weighting over the sequences, i.e. $\gamma_l = 1/L$. $S(l)$ determines the shift range at position l.

2.2.2 DESIGN: Batch UCB

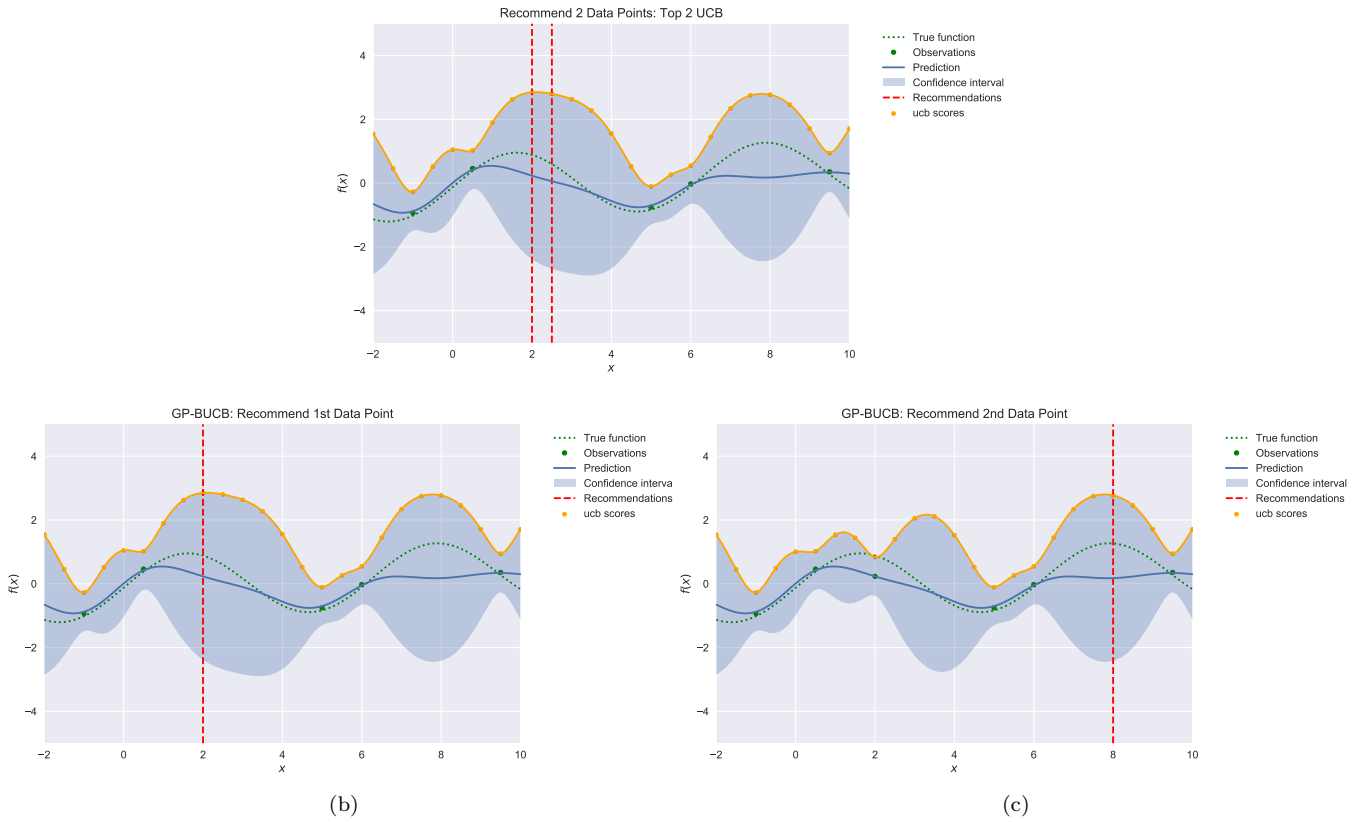
For recommendations of RBS sequences that should be experimentally labeled next, we have used the *Upper Confidence Bound (UCB)* algorithm. On one hand, we want to exploit the function in terms of the design space, that is to pinpoint sequences that are believed to have high labels (i.e. high predicted mean); on the other hand, we also want to explore the design space where we have little information and sequences have a chance to have high labels (i.e. high predicted SD). The UCB algorithm provides such *exploitation-exploration balance* by balancing the predicted mean and SD. More precisely, UCB algorithm selects RBS sequences with the maximum upper confidence bound at round t, i.e.

$$\text{argmax}_{\mathbf{x}_i \in \mathcal{D}} \left(\mu_{t-1}(\mathbf{x}_i) + \beta_t \sigma_{t-1}(\mathbf{x}_i) \right), \tag{2}$$

where β_t is a hyperparmeter balancing the exploitation and exploration, $\mu_t(\mathbf{x}_i), \sigma_t(\mathbf{x}_i)$ are the predicted mean and standard deviation (SD) at round t for the sequence $\mathbf{x}_i$. We call $\mu_{t-1}(\mathbf{x}_i) + \beta_t \sigma_{t-1}(\mathbf{x}_i)$ the *UCB score* of sequence $\mathbf{x}_i$ at round t.

Since experimentally labelling sequences is time-consuming, it is unrealistic to recommend sequence sequentially (i.e. one-by-one) and then wait for the label to be tested and used to improve the model. Instead, we can recommend RBS sequences in a batch of size n. One naive approach is to recommend sequences in design space with top n UCB scores, as shown in Figure 2(a) ($n = 2$). However, this approach may end up recommending similar sequences in the same local maximum (e.g. $x = 2, x = 2.5$ in this example). However, since we assume similar sequences would have similar labels (e.g. by knowing $x = 2$ we can gain information of $x = 2.5$ as well), we prefer to not waste time and money on labelling sequences with high similarities in the same batch.

A key property of Gaussian Process regression is that the predictive variance depends only on observed points (i.e. features), but not on the labels of those observed points. We can make use of this property to design batch upper confidence bound (BUCB) algorithm [**desautels2012parallelizing**]. That is, we can recommend sequences sequentially by updating the UCB score with the updated predicted SD with the the previously recommended points. First, GP-BUCB recommends the data point with maximum UCB score based on the predictions over initial 5 observations as shown in Figure 2(b). Then we add the recommended data point ($x = 2$) into the training data



(b) (c)

Figure 2: Batch Recommendation. We use the batch size of 2, with 5 initial observations. The design (recommendation) space is 24 uniformly distributed points in the range [-2,10], i.e. -2, -1.5, -1, ..., 9.5, 10. The confidence interval are shown with predicted mean of ± 1.96 standard deviation. (a) Top UCB recommendations. The recommendations are 2 data points with top UCB scores, constructed with GP predictions. (b)(c) GP-BUCB recommendations. (b) shows the first recommended sequence, (c) shows the new predicted confidence interval and the second recommendation based on that.

set with the predicted mean of that point as label (note it is not the true label, i.e. observation), and update the predicted variance and then we finally update the UCB scores. We then recommend the second data point based on the new UCB scores. As we can see in Figure 2(c), since we assume we have observed $x = 2$, then the new predicted variance of the data points in design space around $x = 2$ decreases, so instead of recommending a similar data point $x = 2.5$, we recommend the data point which is in a different maximum area with potentially high labels ($x = 8$). By using GP-BUCB, we can ensure that the sequences with high similarities will not be recommended in the same batch hence increasing the exploration efficiency.

3 Results

Figure 1 shows the DBTL cycle with the machine learning process emphasised. Our machine learning workflow encompasses two phases of the cycle, namely the LEARN and DESIGN phase. In LEARN phase we use the Gaussian Process algorithm to predict the behaviour of different RBS sequences and in the DESIGN phase we use the multi-armed Bandit algorithm to recommend sequences to be Built and Tested next. Conventionally, the Design phase is considered the beginning of the cycle, however we will start description of our results from the learn phase, which we find a more natural starting point for a machine learning centred workflow.

3.1 LEARN - prediction of RBS performance

First problem to be tackled was how to embed the sequence to give our RBS sequences numerical form. As we will be using a Gaussian Process for regression we have investigated use of different types of kernels (also called covariances) for embedding [**Ben-Hur2008**]. The additional, and positive, effect of using kernels is that our data will be moved to a higher-dimensional space, which makes the regression process easier. We compared performance of Dot Product, RBF and a number of string kernels: spectrum, mixed spectrum, weighted degree and weighted degree with shifting Figure Xa. Since we found that the spectrum kernel performed the best, we have used it in subsequent studies. More specifically, we used a summary of three kernels: spectrum kernel to process the core 6bp and dot product kernel to process the 7bp flanking sequences both upstream and downstream of the core sequence. This approach allowed for good balance between computational complexity and performance.
The predictions were made using the Gaussian Process regression (GPR) algorithm. In essence, Gaussian Process is a stochastic (Bayesian) predictor of the shape of a function, which is built based on perceived similarities between data points (kernels), where not only a mean value, but also probability distribution of the mean is calculated. Such an approach makes GPR well suited to predict biological phenomena which are also highly stochastic.
For the zeroth round, since we don't have access to any prior data fitting our design we approximated the predictions using the data from **jervis2018machine** and the Gaussian Process regression method (see **Methods** section) [**srinivas2012information**]. This set contains 113 non-repeated records for 56 unique RBS sequences with the respective TIR. The label values are between 0 - 100,000 and skewed, which is shown in Figure xx. First, we have normalised the label to 0 - 1 using the min-max normalisation. The predictions are shown in Figure Xb
The predictions for next rounds were predicted similarly, but with use of data obtained in previous rounds, visualisation of prediction for the last round of designs is shown in Figure Xc

3.2 DESIGN of the genetic device

There is a number of factors that impact the protein expression rate, many of them concerned with how the ribosome recognises and binds to the RBS sequence [**Chen1994**, **Vellanoweth1992**]. In *E. coli* the RBS is usually located in the 20 bases upstream of the start codon. The RBS usually has a distinguishable, consensus, core sequence called the Shine-Dalgarno sequence, which in *E. coli* is AGGAGG. Here, we put that 20 bp long sequence into focus with main emphasis being put on the 6bp core region (Figure 3).

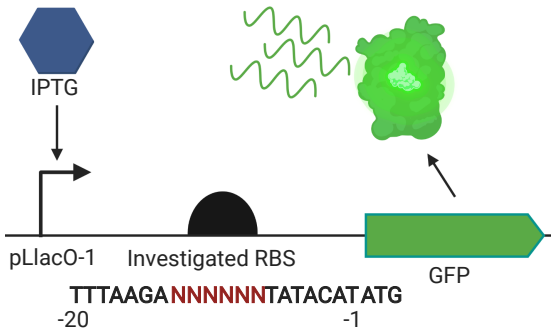


Figure 3: **Diagram of the investigated sequence.** The sequence of the investigated RBS is shown with the randomised core sequence in red. The start codon of the GFP coding sequence is also shown with numbers showing relative nucleotide positions.

In our genetic design, the investigated RBS controls expression of the Green Fluorescent Protein (GFP) in its mut3b variant. By controlling expression of a fluorescent protein with the RBS we can quickly assess the perceived TIR by measuring fluorescence of cells harbouring plasmid with the device. Finally, the mRNA is transcribed from an IPTG-inducible promoter pLlacO-1. By making the whole device inducible we can synchronise the start of the expression of the GFP in all the cultures by inducing them at the same optical density (OD_{600}) with addition of IPTG.

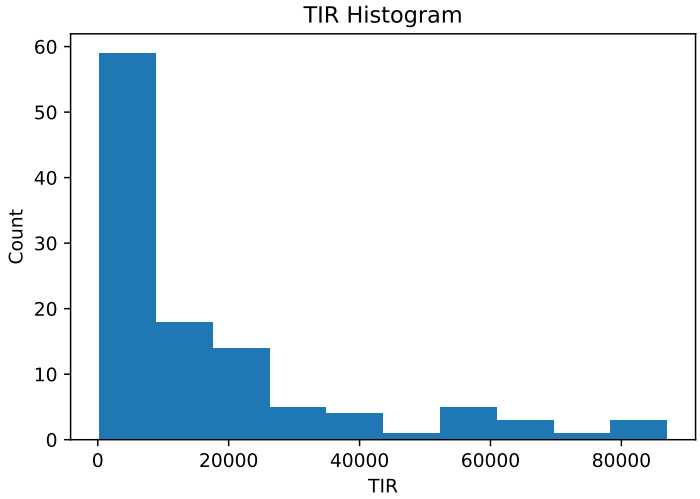


Figure 4: TIR Histogram.

The investigated RBS sequence is 20 bps long with the sequence TTTAAGAAGGAGATATACAT, which is a known high TIR RBS that comes with the pBb series plasmids [**Lee2011**]. In our design we focus on randomising of the core -8 to -13 (relative to the GFP) nucleotides of the RBS and fix others to be the same as the consensus sequence, i.e. TTTAAGA + NNNNNN + TATACAT. Since for each of the 6 position there are 4 possibilities: A, C, G, T the total variant space is $4^6 = 4096$.
Since there was no prior data that we could have used to guide our design for the zeroth round of experiments, we have designed 180 (two times 90) RBS sequences based on the consensus sequence that we expected to give good cover of the experimental space:

1. 60 RBS sequences which are subsequent single nucleotide changes of all 20 nucleotides of the original, consensus sequence. This batch is designed to show us influence of such single nucleotide changes on the overall performance of the RBS and the potential impact of changes made beyond the core part.

2. 30 RBS sequences that were fully and uniformly randomised (equal probability of choosing either nucleotide for each position)

3. 30 RBS sequences randomised based on the position probability matrix (PPM) we need a citation and better explanation for this

4. 60 RBS sequences recommended by our multi armed bandit recommendation algorithm based on the initial literature, as described below.

1. RMSE of predictions.

2. Similarity of recommendations.

The bandit algorithm aims at maximising the reward (output) from testing a limited number of instances from a big pool which cannot be wholly tested due to limited resources (time, computational power, capital). As such, it is very useful in solving problems like the one presented here - a big pool of potential designs, but only limited time and money that can be used for finding the optimal one.
In short, the multi-armed bandit algorithm is a stochastic method of probing of the experimental space. In our case we use the Upper Confidence Bound version of the algorithm, which focuses its recommendations on sequences that should give highest TIR based on the probabilities computer by the prediction algorithm (GPR). Another feature of the bandits algorithms is that it balances two approaches: exploration and exploitation. Exploration makes the

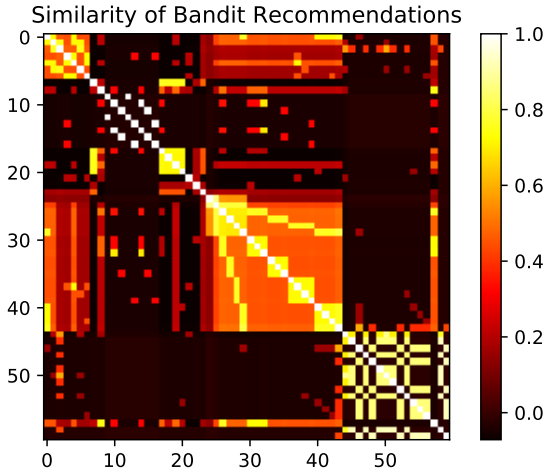


Figure 5: TIR Histogram.

algorithm to recommend designs that will improve the predictions better, whereas exploitation will recommend designs that focus on delivering the most efficient design the fastest. The two approaches can be controlled with the β parameter. We have decided that in the first iterations of the cycle it would be beneficial to skew the algorithm towards the exploration with exploitation taking increasing role in later iterations. One thing of note is that the bandit algorithm is stochastic, that is it exploits the probabilities of given event occurring (in this case RBS having a specific TIR). As such, it pairs naturally with our prediction algorithm, the Gaussian Process, which provides probability based function regression.

For rounds beyond the initial, zeroth one, the designs were recommended only by the bandit algorithm based on the data obtained in the previous one, without adding any random designs. In total, three more rounds beyond the initial one have been performed, each with designs recommended by the algorithm.

3.3 BUILD and TEST

For the machine learning to work effectively the analysed data-set needs to show two qualities: high relative volume and high quality of data. These two elements don't have a specific definitions, but in general the data-set have to be going into at least hundreds possible data points and one have to cover 5-10% of that space. And since quality of the obtained data has a direct and strong correlation with quality of the predictions and in effect - recommendations, one need to ensure that the obtained results represent the real value as close as possible.
To help us obtain reliable and reproducible results we have employed automation-heavy workflow for our experiments. This way we hope to eliminate a big part of sample-to-sample variation as well as human-introduced variation. Additionally, performing all the procedures directly in 96-well microplate format enabled us to significantly cut down time to prepare our variants.
In short, the genetic variations of the RBS were introduced to the plasmids with combination of PCR and isothermal assembly. The plasmids were then transformed and the resulting transformants were tested both using microplate reader and flow cytometry.
Figure X shows the results of the zeroth and all the subsequent rounds.

3.4 EXIT

There are two potential points where the exit from the cycle should be considered: i) the optimum solution has been found or ii) depletion of available resources (time and/or money). In our case we have performed a total of four

rounds, with performance of the RBSes increasing as seen in Figure X.

4 Summary

5 Contributions

Zhang M. and Ong C. S. designed and implemented the machine learning algorithms and workflow. Holowko M. B. and Hayman Zumpe H. have designed and performed the laboratory experiments. All authors contributed to and reviewed the manuscript.

Appendix

A Reproducible Plots

In this section, we put reproducible plots and will update the plots when we get full results.

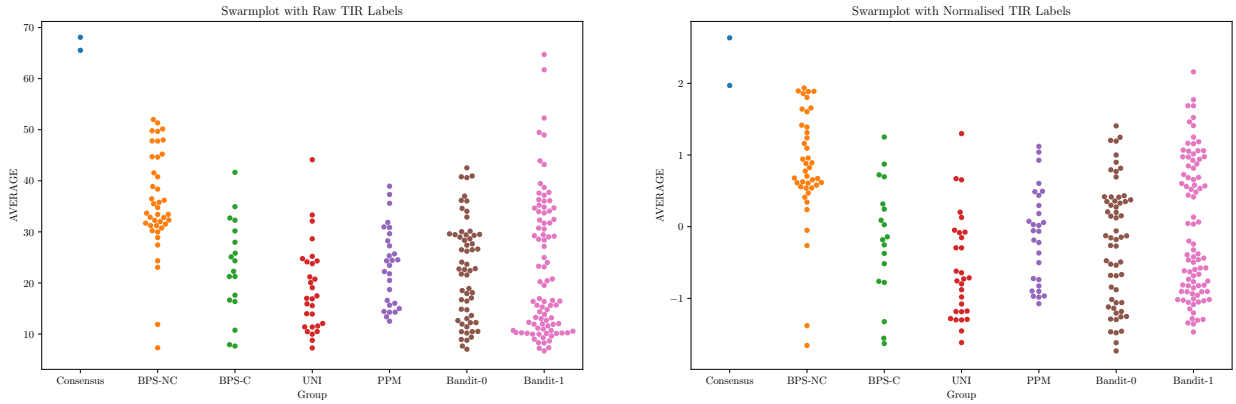


Figure 6: Swarmplots for different groups, with raw (left) and normalised (right) TIR labels (averaged over 6 replicates). Group names represent: Consensus (consensus sequence tested in different round); BPS-NC (bps noncore); BPS-C (bps core); UNI (uniformly random); PPM (position-based probability matrix); Bandit-0 (bandit design for round 0); Bandit-1 (bandit design for round 1).

B Gaussian Process Regression

A *Gaussian process* is a collection of random variables, any finite number of which have a joint Gaussian distribution. We define mean function $\mu(\mathbf{x})$ and covariance function $k(\mathbf{x}, \mathbf{x}')$ of a real process $f(\mathbf{x})$ as

$$\mu(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})] \tag{3}$$

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - \mu(\mathbf{x}))(f(\mathbf{x}') - \mu(\mathbf{x}'))]. \tag{4}$$

A Gaussian process is specified by its mean function and convariance function as $f(\mathbf{x}) \sim \mathcal{GP}\left(\mu(\mathbf{x}), k\left(\mathbf{x}, \mathbf{x}'\right)\right)$. We consider the case where the observations are noisy, i.e. $\{(\mathbf{x}_i, y_i) | i = 1, \ldots, n\}$, where $y_i = f(\mathbf{x}_i) + \epsilon$ with $\epsilon \sim \mathcal{N}(0, \alpha^2)$. The Gaussian noise is independent identically distributed, and the prior on the noisy observations is then $\operatorname{cov}\left(y_p, y_q\right) = k\left(\mathbf{x}_p, \mathbf{x}_q\right) + \alpha^2 \delta_{pq}$, where δ_{pq} is a Kronecker delta which is one if $p = q$ and zero otherwise. It is equivalent to a diagonal matrix $\alpha^2 I$ on the kernel matrix evaluated on the training points.

For n_* test points X_*, we assume the prior over the functions values as a random Gaussian vector $\mathbf{f}_* \sim \mathcal{N}(\mathbf{0}, K(X_*, X_*))$. Then the joint distribution of the observed target values and the function values at the test points under the prior as

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X, X) + \alpha^2 I & K\left(X, X_*\right) \\ K\left(X_*, X\right) & K\left(X_*, X_*\right) \end{bmatrix}\right) \tag{5}$$

where $K(X, X_*)$ denotes the $n \times n_*$ covariance/Kernel matrix evaluated at all pairs of training and testing points, similarly for other kernel matrices. Then the posterior of the test points (i.e. predictive distributions) is given by the conditional distribution $\mathbf{f}_* | X, \mathbf{y}, X_* \sim \mathcal{N}(\bar{\mathbf{f}}_*, cov(\mathbf{f}_*))$, where

$$\bar{\mathbf{f}}_* \triangleq \mathbb{E}\left[\mathbf{f}_* \mid X, \mathbf{y}, X_*\right] = K\left(X_*, X\right)\left[K(X, X) + \alpha^2 I\right]^{-1} \mathbf{y} \tag{6}$$

$$\operatorname{cov}\left(\mathbf{f}_*\right) = K\left(X_*, X_*\right) - K\left(X_*, X\right)\left[K(X, X) + \alpha^2 I\right]^{-1} K\left(X, X_*\right) \tag{7}$$

For noisy test targets $\mathbf{y}_*$, we can compute the predictive distribution by adding $\alpha^2 I$ to the variance term $cov(\mathbf{f}_*)$ in Eq. (7).

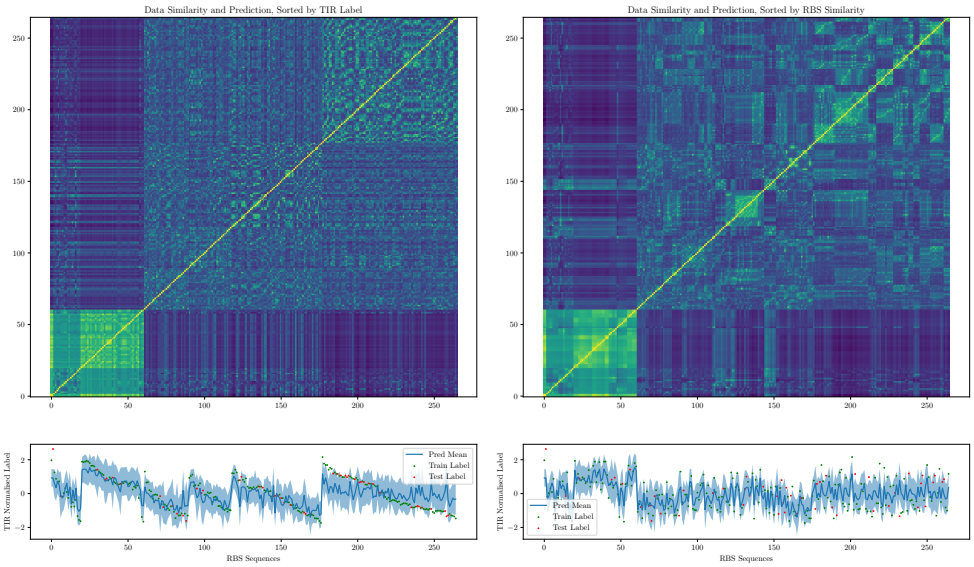


Figure 7: Kernel Heatmap and Predictions. Sequences are grouped as Consensus (1-2), BPS-C (3-20), BPS-NC (21-61), UNI (62-90), PPM (91-118), Bandit-0 (119-177), Bandit-1 (178-265). Inside of each group, sequences are clustered and sorted in terms of TIR labels (left) or RBS similarity (right). The first row shows the similarity measured by weighted degree kernel with shift, the second shows the predicted mean and uncertainty (1.95 standard deviation).

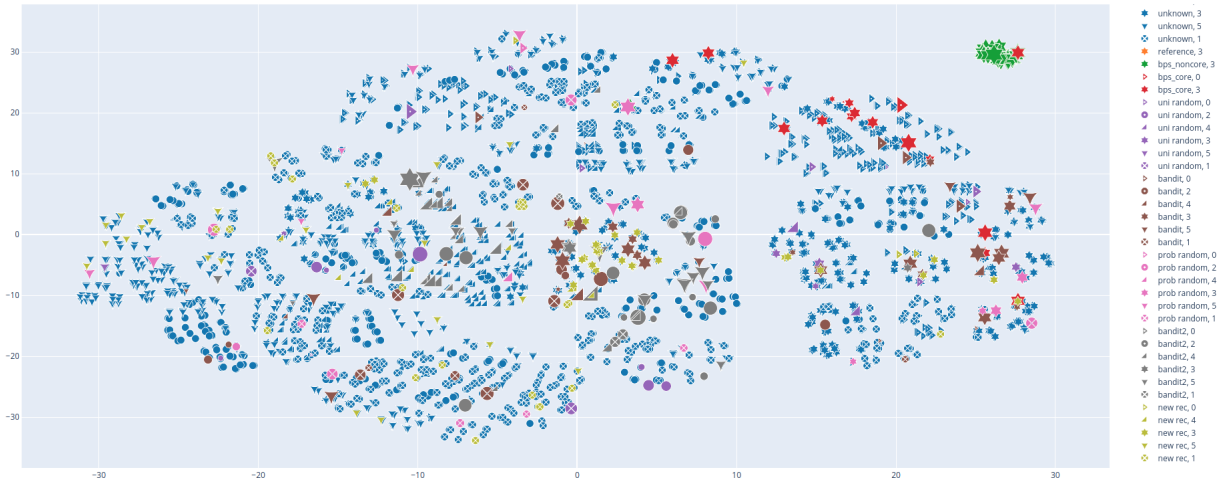


Figure 8: TSNE of RBS sequences in design space with clustering. The distance is calculated based on the weighted degree kernel on RBS sequences. Colours indicate different groups, shapes indicate different clusters.

C Choices of Kernels

The choice of covariance function is critical for the performance of Gaussian process regression, we show a number of different string kernels tested in this study below:

- *Spectrum Kernel.*

$$k_\ell^{\mathrm{Spec}}(X, X') = \left\langle \phi_\ell^{\mathrm{Spec}}(\mathbf{x}), \phi_\ell^{\mathrm{Spec}}(\mathbf{x}') \right\rangle = \phi_\ell^{\mathrm{Spec}}(\mathbf{x})^T \phi_\ell^{\mathrm{Spec}}(\mathbf{x}') \,. \tag{8}$$

  where $\mathbf{x}, \mathbf{x}'$ are two RBS sequences in $\mathcal{D}$ over an alphabet Σ. We denote the number of letters in the alphabet as $|\Sigma|$. $\phi_\ell^{\mathrm{spec}}(\mathbf{x})$ maps the sequence X into a $|\Sigma|^\ell$ dimensional feature space, where each dimension is the count of the number of one of the $|\Sigma|^\ell$ possible strings s of length ℓ. Let X, X' be two metrics which include n sequences, and $\Phi_d^{Spec}(X) \in \mathbb{R}^{n \times |\Sigma|^\ell}$, then the spectrum kernel over metrics is

$$K_\ell^{\mathrm{Spec}}(X, X') = \Phi_\ell^{\mathrm{Spec}}(X) \Phi_\ell^{\mathrm{Spec}}(X')^T \,. \tag{9}$$

- *Sum of Spectrum Kernel,* considers weighted sum over different parts of the string.

- *Mixed Spectrum Kernel,* considers weighted sum over different substring length, with $\beta_d = \frac{2(\ell-d+1)}{\ell(\ell+1)}$,

$$k_\ell^{MixedSpec}(\mathbf{x}, \mathbf{x}') = \sum_{d=1}^{\ell} \beta_d k_d^{Spec}(\mathbf{x}, \mathbf{x}') \tag{10}$$

- *Weighted Degree Kernel,* considers positional information. WD kernel counts the match of kmers at corresponding positions in two sequences. For sequences with fixed length L and weighted degree kernel considers substrings starting at each position $l = 1, ..., L$, with $\beta_d = \frac{2(\ell-d+1)}{\ell(\ell+1)}$,

$$k_\ell^{WD}(\mathbf{x}, \mathbf{x}') = \sum_{d=1}^{\ell} \beta_d \sum_{l=1}^{L-d+1} \gamma_l k_d^{Spec}(\mathbf{x}_{[l:l+d]}, \mathbf{x}'_{[l:l+d]}) \tag{11}$$

$$= \sum_{d=1}^{\ell} \beta_d \sum_{l=1}^{L-d+1} \gamma_l \phi_d^{Spec}(\mathbf{x}_{[l:l+d]})^T \phi_d^{Spec}(\mathbf{x}'_{[l:l+d]}) \tag{12}$$

$$= \sum_{d=1}^{\ell} \beta_d \sum_{l=1}^{L-d+1} \gamma_l \mathbb{I}(\mathbf{x}_{[l:l+d]} = \mathbf{x}'_{[l:l+d]}), \tag{13}$$

  where $\mathbb{I}(\text{true}) = 1$ and 0 otherwise.

- *Weighted Degree Kernel With Shift.*

$$k_\ell^{WDS}(\mathbf{x}, \mathbf{x}') = \sum_{d=1}^{\ell} \beta_d \sum_{l=1}^{L-d+1} \gamma_l \sum_{s=0, s+l \le L}^{S(l)} \delta_s \left(k_d^{Spec}(\mathbf{x}_{[l+s:l+s+d]}, \mathbf{x}'_{[l:l+d]}) + (k_d^{Spec}(\mathbf{x}_{[l:l+d]}, \mathbf{x}'_{[l+s:l+s+d]}) \right) \tag{14}$$

$$= \sum_{d=1}^{\ell} \beta_d \sum_{l=1}^{L-d+1} \gamma_l \sum_{s=0, s+l \le L}^{S(l)} \delta_s \left(\mathbb{I}(\mathbf{x}_{[l+s:l+s+d]} = \mathbf{x}'_{[l:l+d]}) + (\mathbb{I}(\mathbf{x}_{[l:l+d]} = \mathbf{x}'_{[l+s:l+s+d]}) \right), \tag{15}$$

  where $\beta_d = \frac{2(\ell-d+1)}{\ell(\ell+1)}, \delta_s = \frac{1}{2(s+1)}$, γ_l is a weighting over the position in the sequence, where we choose to use a uniform weighting over the sequences, i.e. $\gamma_l = 1/L$. $S(l)$ determines the shift range at position l.

From kernel to distance:

$$d(\mathbf{x}, \mathbf{x}') = \sqrt{k(\mathbf{x}, \mathbf{x}) + k(\mathbf{x}', \mathbf{x}') - 2k(\mathbf{x}, \mathbf{x}')}$$

C.1 Normalisation of Kernel

As part of data pre-processing, the range of all features should be normalised so that each feature contributes approximately proportionately to the predictive model. The kernel matrix is represented by the inner product of the underlying feature vectors, it needs to be normalised before being used in the downstream regression models. Up-scaling (down-scaling) features can be understood as down-scaling (up-scaling) regularizers such that they penalise the features less (more).

Here we consider two approaches for kernel normalisation: centering and unit norm. We will show how to convert the normalisation in terms of feature vectors to normalisation in terms of kernel matrices. As defined before, consider $\mathbf{x}, \mathbf{x}'$ are two RBS sequences in $\mathcal{D}$ over an alphabet Σ. We denote $\phi(\mathbf{x}_i)$ as a column feature vector of sequence $\mathbf{x}_i$, where a feature function $\phi : \mathbf{x} \to \mathbb{R}^d$. Assume there is total of n sequences in the data X (n' sequences in the data X'). We illustrate centering and unit norm normalisation below.

- Centering. Defining the mean vector as $\bar{\Phi}(X) = \frac{1}{n}\sum_{s=1}^{n}\phi(\mathbf{x}_s) \in \mathbb{R}^d$, the centered feature vector $\phi^C(\mathbf{x}_i) \in \mathbb{R}^d$ of $\mathbf{x}_i$ is

$$\phi^C(\mathbf{x}_i) = \phi(\mathbf{x}_i) - \bar{\Phi}(X) = \phi(\mathbf{x}_i) - \frac{1}{n'}\sum_{s=1}^{n'}\phi(\mathbf{x}_s). \tag{16}$$

  The corresponding centering kernel value between $\mathbf{x}_i$ and $\mathbf{x}_j$ is then

$$k^C(\mathbf{x}_i, \mathbf{x}_j) =< \phi^C(\mathbf{x}_i), \phi^C(\mathbf{x}_j) > \tag{17}$$

$$= \left(\phi(\mathbf{x}_i) - \frac{1}{n}\sum_{s=1}^{n}\phi(\mathbf{x}_s)\right)^T \left(\phi(\mathbf{x}_j) - \frac{1}{n'}\sum_{s'=1}^{n'}\phi(\mathbf{x}_{s'})\right) \tag{18}$$

$$= \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j) - \left(\frac{1}{n}\sum_{s=1}^{n}\phi(\mathbf{x}_s)\right)^T \phi(\mathbf{x}_j) - \phi(\mathbf{x}_i)^T \left(\frac{1}{n}\sum_{s'=1}^{n'}\phi(\mathbf{x}_{s'})\right) + \left(\frac{1}{n}\sum_{s=1}^{n}\phi(\mathbf{x}_s)\right)^T \left(\frac{1}{n'}\sum_{s'=1}^{n'}\phi(\mathbf{x}_{s'})\right) \tag{19}$$

$$= k(\mathbf{x}_i, \mathbf{x}_j) - \frac{1}{n}\sum_{s=1}^{n}k(\mathbf{x}_s, \mathbf{x}_j) - \frac{1}{n'}\sum_{s'=1}^{n'}k(\mathbf{x}_i, \mathbf{x}_{s'}) + \frac{1}{n^2}\sum_{s=1}^{n}\sum_{s'=1}^{n'}k(\mathbf{x}_s, \mathbf{x}_{s'}) \tag{20}$$

- Unit Norm. Define the (l_2) norm of a feature vector as $||\phi(\mathbf{x})|| = \sqrt{\sum_{m=1}^{d}\phi_d(\mathbf{x})^2} = \sqrt{k(\mathbf{x}, \mathbf{x})} \in \mathbb{R}^+$, then the unit norm feature vector $\phi^{UN}(\mathbf{x}_i) \in \mathbb{R}^d$ of $\mathbf{x}_i$ is

$$\phi^{UN}(\mathbf{x}_i) = \frac{\phi(\mathbf{x}_i)}{||\phi(\mathbf{x}_i)||}. \tag{21}$$

  The corresponding unit norm kernel value between $\mathbf{x}_i$ and $\mathbf{x}_j$ is then

$$k^{UN}(\mathbf{x}_i, \mathbf{x}_j) =< \frac{\phi(\mathbf{x}_i)}{||\phi(\mathbf{x}_i)||}, \frac{\phi(\mathbf{x}_j)}{||\phi(\mathbf{x}_j)||} > \tag{22}$$

$$= \frac{\phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)}{||\phi(\mathbf{x}_i)|| \times ||\phi(\mathbf{x}_j)||} \tag{23}$$

$$= \frac{k(\mathbf{x}_i, \mathbf{x}_j)}{\sqrt{k(\mathbf{x}_i, \mathbf{x}_i)k(\mathbf{x}_j, \mathbf{x}_j)}} \tag{24}$$

- Unit Variance. After the centering and unit norm normalisation, the kernel matrix is unit variance as well. In the following, we show transformations of the unit variance (with centering) normalisation. Define the variance vector $Var(\Phi(X)) = \frac{1}{n}\sum_{s=1}^{n}||\phi(\mathbf{x}_s) - \bar{\Phi}(X)||^2 = \frac{1}{n}\sum_{s=1}^{n}||\phi(\mathbf{x}_s) - \sum_{s'=1}^{n}(\phi(\mathbf{x}'_s))||^2 = \frac{1}{n}\sum_{s=1}^{n}k^C(\mathbf{x}_s, \mathbf{x}_s) \in \mathbb{R}$, the unit variance feature vector $\phi^{UV}(\mathbf{x}_i) \in \mathbb{R}^d$ of $\mathbf{x}_i$ is

$$\phi^{UV}(\mathbf{x}_i) = \frac{\phi(\mathbf{x}_i)}{\sqrt{Var(\Phi(X))}}. \tag{25}$$

The corresponding kernel representation is

$$k^{UV}(\mathbf{x}_i, \mathbf{x}_j) = < \frac{\phi(\mathbf{x}_i)}{\sqrt{Var(\Phi(X))}}, \frac{\phi(\mathbf{x}_j)}{\sqrt{Var(\Phi(X'))}} > \tag{26}$$

$$= \frac{\phi(\mathbf{x}_i)^T \mathbf{x}_j}{\sqrt{Var(\Phi(X))Var(\Phi(\mathbf{X}'))}} \tag{27}$$

$$= \frac{k(\mathbf{x}_i, \mathbf{x}_j)}{\sqrt{\frac{1}{n}\sum_{s=1}^{n} k^C(\mathbf{x}_s, \mathbf{x}_s) \frac{1}{n}\sum_{s'=1}^{n'} k^C(\mathbf{x}_{s'}, \mathbf{x}_{s'})}} \tag{28}$$

After centering and unit norm, $\frac{1}{n}\sum_{s=1}^{n} k^C(\mathbf{x}_s, \mathbf{x}_s) = k(\mathbf{x}_i, \mathbf{x}_i)$, which implies that after centering and unit norm, the kernel matrix is already unit variance normalised.

For the Gaussian Process regression, we make of use of two kernel matrices: the kernel function between the training data itself, i.e. $K(X_{train}, X_{train})$; and the kernel function taking the training data and testing data as inputs, i.e. $K(X_{test}, X_{train})$. We will state two ways of normalisation those two kind of matrices:

- Normalise training and testing data separately. This approach is preferred for most of the machine learning algorithms since it follows the rule that we have no information about testing data while training. Then for centering, one should subtract the mean vector over the training data for both kinds of matrices. For unit norm normalisation, when one calculates $K^{UN}(X_{test}, X_{train})$, the two terms inside of square root: $k(\mathbf{x}_i, \mathbf{x}_i)$ is taken from $K(X_{test}, X_{test})[i, i]$, and $k(\mathbf{x}_j, \mathbf{x}_j)$ is taken from $K(X_{train}, X_{train})[j, j]$.
- Normalise training and testing data together, i.e. normalise $K(X_{train+test}, X_{train+test})$, then extra the parts we need from the normalised matrix. This approach is suitable in a case where one already knows the whole of testing features. For centering, one should subtract the mean vector over the whole matrix $\Phi(X_{train+test})$. The unit norm normalisation is the same as in the previous case.

For our experiment, we fix the design space before training, i.e. the testing features are already known before testing. So we choose to normalise the kernel matrix over the training and testing data together, by first applying centering and then unit norm normalisation.

D Batch Recommendation

For recommending RBS sequences to label, we consider the Upper Confidence Bound (UCB) algorithm, selecting RBS sequences with the maximum upper confidence bound at round t, i.e.

$$\text{argmax}_{\mathbf{x}_i \in \mathcal{D}} \left(\mu_{t-1}(\mathbf{x}_i) + \beta_t \sigma_{t-1}(\mathbf{x}_i)\right), \tag{29}$$

where β_t is a hyperparmeter balancing the exploitation and exploration, $\mu_t(\mathbf{x}_i), \sigma_t(\mathbf{x}_i)$ are the predicted mean and standard deviation at round t for the sequence $\mathbf{x}_i$.

Since labelling sequences is time-consuming, it is unrealistic to recommend sequence sequentially (i.e. one-by-one) and waiting for the label after each prediction. Therefore we consider recommending sequences in batch and using Gaussian Process Batch Upper Confidence Bound (GP-BUCB) algorithm [**desautels2012parallelizing**]. With batches of size B, the feedback mapping $fb[t] = \lfloor (t-1)/B \rfloor B$, i.e.

$$\text{fb}[t] = \begin{cases} 0 & : t \in \{1, \ldots, B\} \\ B & : t \in \{B+1, \ldots, 2B\} \\ 2B & : t \in \{2B+1, \ldots, 3B\} \\ \vdots & \end{cases} \tag{30}$$

A key property of Gaussian Process regression is that the predictive variance in Eq. (7) only depends on observed points (i.e. features), but not on the labels of these observed points. So one can compute the posterior variance without actually observing the labels. The GP-BUCB policy is to select sequences that

$$\text{argmax}_{\mathbf{x}_i \in \mathcal{D}} \left(\mu_{fb[t]}(\mathbf{x}_i) + \beta_t \sigma_{t-1}(\mathbf{x}_i)\right). \tag{31}$$

And only update $y_{t'} = f(\boldsymbol{x}_{t'}) + \varepsilon_{t'}$ for $t' \in \{\text{fb}[t]+1, \ldots, \text{fb}[t+1]\}$ at the end of each batch ($\text{fb}[t] < \text{fb}[t+1]$). This is equivalent to sequential GP-UCB with *hallucinated observations* $\boldsymbol{y}_{\text{fb}[t]+1:t-1} = \left[\mu_{\text{fb}[t]}\left(\boldsymbol{x}_{\text{fb}[t]+1}\right), \ldots, \mu_{\text{fb}[t]}\left(\boldsymbol{x}_{t-1}\right)\right]$, while the posterior variance decreases.

E Design Pipeline

The n + 1 round design is based on the nth round result, where each sequence has 6 replicates with TIR labels. We pre-processed the data by taking a logarithm transformation and standardisation of the raw TIR label for each replicates respectively, where TIR is calculated as a derivative of GFP fluoresence divided by OD600 of culture over 4h counting from the start of log phase of growth. After normalisation, each replicate has zero mean and unit variance.

For prediction, we use Gaussian process regression, with training on all normalised replicates and predicting on the design space (6-base core part design) except known sequences. We assume the observation are noisy, where the noise is under centered normal distribution with standard deviation α. We model the covariance matrix using the weighted degree kernel with shift. We normalise the kernel with centering and unit norm in terms of the whole kernel constructed by both first round result and design space. The hyperparameter for kernel, including maximum substring length l, maximum shift length s, and the noise standard deviation α of Gaussian process model are choose based on 10-repeat 5-fold cross validation. We choose $l = 6, s = 1, \alpha = 2$ for the second round design.

For recommendation, we use batch upper confidence bound introduced by GP-BUCB algorithm [**desautels2012parallelizing**] The upper confidence bound is constructed by predicted mean plus 2 predicted standard deviation. We recommend 90 sequences from the design space.

F Intuition behind UCB and visualisation

- Exploitation and exploration explanation.
- Visualise coverage by clustering plot.
- Table for in-clustering mean and variance.

G Result analysis

- violinplot.
- regression performance plot, table.
- kernel matrix plot.

H Figures