Supplementary Information – Bayesian Optimization for Chemical Reactions

Jeff Guo[+1,2], Bojana Ranković[+1,2], and Philippe Schwaller[1,2]

[1]Laboratory of Artificial Chemical Intelligence (LIAC), Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland
[2]National Centre of Competence in Research (NCCR) Catalysis, Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland

1 Bayesian Optimization

In this section, we provide the mathematical foundation for Bayesian optimization (BO) to promote its practical application for chemical reactions. BO is a strategy to maximize (or minimize) a 'black-box function', defined here as an oracle which is expensive to query (Equation 1):

$$x^* = \arg\max_{x \in X} or \min f(x)$$

where x^* is the input which optimizes the oracle evaluation, $f(x)$. In this work, the oracle evaluation refers to the reaction outcome and the objective being optimized, e.g., yield. Sequential decision-making in BO relies on the *prior* and the *posterior.* The former is the initial belief in how the reaction behaves, i.e., the prior knowledge, and provides a starting point to propose more optimal input, x. For example, one may hypothesize that electron withdrawing groups on the substrate will promote the reaction, thus constituting prior knowledge. The posterior model represents our model of the reaction landscape *after* making new observations. For example, it may turn out that it is the pKa of the base which is most important. Correspondingly, BO iteratively proposes new x, i.e., reaction conditions conditioned on previous observations and until convergence or the experimental budget is exceeded. For a detailed discussion of BO, we refer the reader to a review by Shahriari et al.[1]

2 Surrogate

The surrogate model takes input x, i.e., descriptors for a reaction, and outputs $f(x)$, e.g., the yield, stereoselectivity, or both. The choice of the surrogate model represents the function form used to model our belief in how the reaction outcome behaves. Generally, there are three criteria for a surrogate model:

- The function should be flexible enough to model the potential reaction landscape. As an extreme example, linear regression is not used for BO as it imposes a linear relationship between the descriptors and the reaction outcome which is likely not the case.
- The model should output an associated uncertainty with its prediction.
- Training and querying the model should be much less resource intensive than querying the oracle. In the case of reaction optimization, this should be automatically satisfied as surely, performing and analyzing the reaction will be more expensive than model inference.

The most common surrogate used in BO is a Gaussian Process (GP). GPs output a prediction and an associated uncertainty that follows a Gaussian distribution which allows one to obtain an analytical expression for popular acquisition functions, i.e., the heuristic used to *decide* which new experiment to perform. In the following sections, we provide a more formal background on GPs in addition to Neural network (NN) surrogate models.

[+]These authors contributed equally to this work

2.1 Gaussian Process

Gaussian Process (GP) models random variables as multivariate Gaussian distributions.[1,2] Concretely, for any finite set of inputs, $x_1, x_2, ...x_n$, there is a corresponding multivariate Gaussian distribution which models the probability distribution of the reaction outcome. GPs output predictions and uncertainties using the mean function, $m(x)$, and the covariance function, $\kappa(x, x^{'})$, respectively:

$$f(x) \sim GP(m(x), \kappa(x, x^{'}))$$

Since Gaussian distributions are completely characterized by its mean and variance, for any input, x, GPs predict the expected reaction outcome, $f(x)$ as the peak of the Gaussian and the uncertainty is quantified by the width of the distribution.

Moreover, the kernel function quantifies the relationship between data points and thus encodes inductive bias into the model. For example, a popular kernel is the squared exponential (SE, also known as the radial basis function, RBF):

$$K_{SE}(x, x^{'}) = e^{(-\frac{(x-x')^2}{2})}$$

here, the simplest form of the SE kernel is shown without any hyperparameters. The SE kernel is used for continuous functions and returns a scalar whose magnitude depends on the difference between x and x'. In particular, a larger difference yields a larger scalar and vice versa. The inductive bias encoded here is that similar (in input space) data points should yield similar GP predictions. For example, performing a reaction at 100 degrees should yield a similar outcome to 101 degrees. Finally, with a kernel function selected and given a query data point, the following equations return the mean and standard deviation of the Gaussian distribution, respectively:

$$\mu(x^{'}) = K(x, x^{'})K(x, x)^{-1}f$$

$$\sigma(x^{'}) = K(x', x^{'}) - K(x, x^{'})^T K(x, x^{'})^{-1} K(x, x^{'})$$

where $x^{'}$ and $\sigma(x^{'})$ are the mean and standard deviation of the Gaussian distribution of the query data point, $x^{'}$. For details on the derivation, we refer the reader to Williams and Rasmussen.[2]

2.2 Neural Network

Neural Networks (NNs) are not often used in BO applications for chemical reaction due to relatively small datasets as NNs typically require at least hundreds of data points, i.e., reaction outcomes to train. Nonetheless, examples exist in which feed-forward neural networks (FFNNs) have demonstrated success in predicting reaction outcomes.[3] The standard computations performed by FFNN are:

$$H = \phi(XW_1 + b_1)$$

$$O = HW_2 + b_2$$

where ϕ is the activation function that confers non-linearity, X is the input matrix, W is the weight matrix, b is the bias, H is the hidden layer, and O is the output layer. Model inference simply involves forward propagation of an input, i.e., a series of matrix multiplications.

Uncertainty quantification adds slight complexity compared to GPs and RFs. The most straightforward method would be to the train the FFNN to output both a prediction and an uncertainty. However, this requires the training data to also report uncertainty which is often unavailable. The brute-force solution is to train an ensemble of FFNNs and take the variance in the predictions to be the uncertainty, but this can impose a significant computational overhead as an N-size ensemble would require N networks to be trained. An alternative method is Monte Carlo dropout (MC dropout). FFNNs can be trained with dropout, i.e., deactivate neurons with a probability, which mitigates overfitting. During inference, one can keep dropout such that repeated queries to the model yield different results. Thus, only one model needs to be trained and the prediction and uncertainty are the average and variance of the results, respectively.[4]

3 Acquisition Function

The acquisition function (AF) is the final ingredient for BO and represents the heuristic used to *propose* the next reaction conditions to try. There have been many AFs proposed, each showing improved performance in certain problem set-ups. Following the sentiment from Shahriari et al. [1], AFs are less important than the choice of surrogate model for BO performance. Correspondingly, we do not aim to give an exhaustive list of AFs. Instead, we introduce the exploration-exploitation balance and present the most popular AFs used for reaction optimization.

3.1 Random Acquisition

One can randomly choose the next reaction condition to try. This is usually the baseline comparison in BO applications as one would hope that a meaningful AF is better than just random choice. We include this as random acquisition represents a purely explorative AF and also shows that *any* heuristic can be considered an AF.

3.2 Greedy

In contrast to random acquisition, greedy acquisition is a purely exploitative AF where the most optimal prediction is chosen as the next reaction condition to try. In a scenario where one wants to optimize yield but has only performed experiments changing the temperature, greedy acquisition will likely *only* exploit this knowledge and propose reaction conditions that change the temperature. Consequently, greedy acquisition does not *explore* reaction conditions for which the surrogate is uncertain.

3.3 Probability of Improvement

The Probability of Improvement (PI) [5] measures the probability that a new reaction condition improves upon the previous best condition. In GPs, the assumption that the reaction landscape is modelled by Gaussian distributions makes PI meaningful and analytically tractable to compute. PI is expressed as:

$$Improvement(x') = max(f(x) - f(x'), 0)$$

Recall that every input to the GP has a Gaussian distribution outcome. The area under any probability distribution is 1. Intuitively, PI measures how much area under the Gaussian distribution associated with the query x is above (if we are trying to maximize an objective such as reaction yield) the current optimal point. Since we are working with Gaussian distributions, there is a closed-form solution given by:

$$PI(x') = \Phi(\frac{\mu - f(x')}{\sigma(x)})$$

where Φ is the Cumulative Distribution Function (CDF) and PI values are bound between 0 and 1. The same expression above can be used with Random Forest (RF) and Neural Network (NN) surrogate models by simply replacing $f(x')$ with the prediction by the corresponding model. In practice, PI can be too exploitative, leading to sub-optimal performance.

3.4 Expected Improvement

The Expected Improvement (EI) [6] attempts to quantify *how much* better a new reaction condition will be. The starting expression for EI is:

$$EI(x') = \int_{-\infty}^{\infty} Improvement(x')Probability(z)dz$$

the *expected* improvement is the improvement (same expression as in PI) multiplied by the probability function. After performing some operations, we arrive at a closed-form solution:

$$EI(x') = (\mu - f(x'))\Phi(\frac{\mu - f(x')}{\sigma(x)}) + \sigma\phi(\frac{\mu - f(x')}{\sigma(x)})$$

EI is the most widely used AF in the BO for chemical reactions literature. [7] A variant of EI to facilitate batch acquisition is Batch Expected Improvement (qEI). [8,9] In qEI, the model makes a prediction on the best expected improvement, the prediction is taken by faith and is used to re-train the surrogate[10], the model is used to make a new prediction, and this process is repeated until the batch number of reaction conditions has been acquired.[7] The same expression above can again be used in random forest (RF) and NN models.

3.5 Upper-Confidence Bound

The Upper-Confidence Bound (UCB) is a linear combination of the prediction and uncertainty:

$$UCB(x') = \mu(x') + \beta\sigma(x')$$

where β is a hyperparameter that controls how much exploration we want to encode in the AF. UCB takes the prediction by the surrogate and adds the standard deviation scaled by β. If β is 0, we recover the Greedy AF which simply chooses the optimal predicted reaction condition. Larger β would encourage the model to explore reaction conditions for which it is more uncertain about. UCB has been used for reaction optimization, though less frequently than EI.[11,12]

References

[1] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas. *Proceedings of the IEEE*, **2015**. 104, 1, 148.

[2] C. K. Williams and C. E. Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, **2006**.

[3] N. S. Eyke, W. H. Green, and K. F. Jensen. *Reaction Chemistry & Engineering*, **2020**. 5, 10, 1963.

[4] Y. Gal and Z. Ghahramani. In *international conference on machine learning*. PMLR, pages 1050–1059.

[5] H. J. Kushner. **1964**.

[6] J. Mockus, V. Tiesis, and A. Zilinskas. *Towards global optimization*, **1978**. 2, 117-129, 2.

[7] B. J. Shields, J. Stevens, J. Li, M. Parasram, F. Damani, J. I. M. Alvarado, J. M. Janey, R. P. Adams, and A. G. Doyle. *Nature*, **2021**. 590, 7844, 89.

[8] D. Ginsbourger, J. Janusevskis, and R. Le Riche. *Dealing with asynchronicity in parallel Gaussian process based global optimization*. Ph.D. thesis, Mines Saint-Etienne, **2011**.

[9] J. Snoek, H. Larochelle, and R. P. Adams. *Advances in neural information processing systems*, **2012**. 25.

[10] D. Ginsbourger, R. L. Riche, and L. Carraro. In *Computational intelligence in expensive optimization problems*. Springer, **2010**. pages 131–162.

[11] B. Burger, P. M. Maffettone, V. V. Gusev, C. M. Aitchison, Y. Bai, X. Wang, X. Li, B. M. Alston, B. Li, R. Clowes, et al. *Nature*, **2020**. 583, 7815, 237.

[12] Y.-S. Choi, Y. Kwon, D. Lee, J. W. Kim, and S. Kim. **2022**.