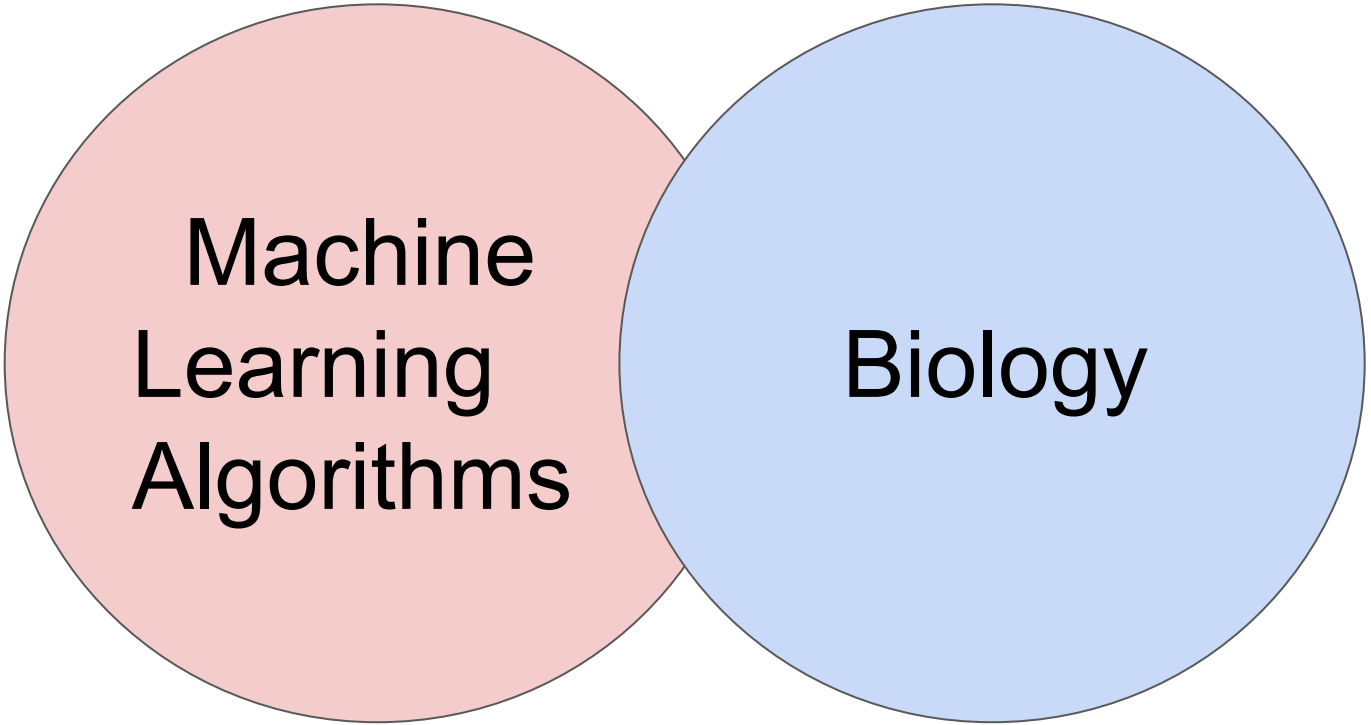
Bayesian Optimization and its applications in biology

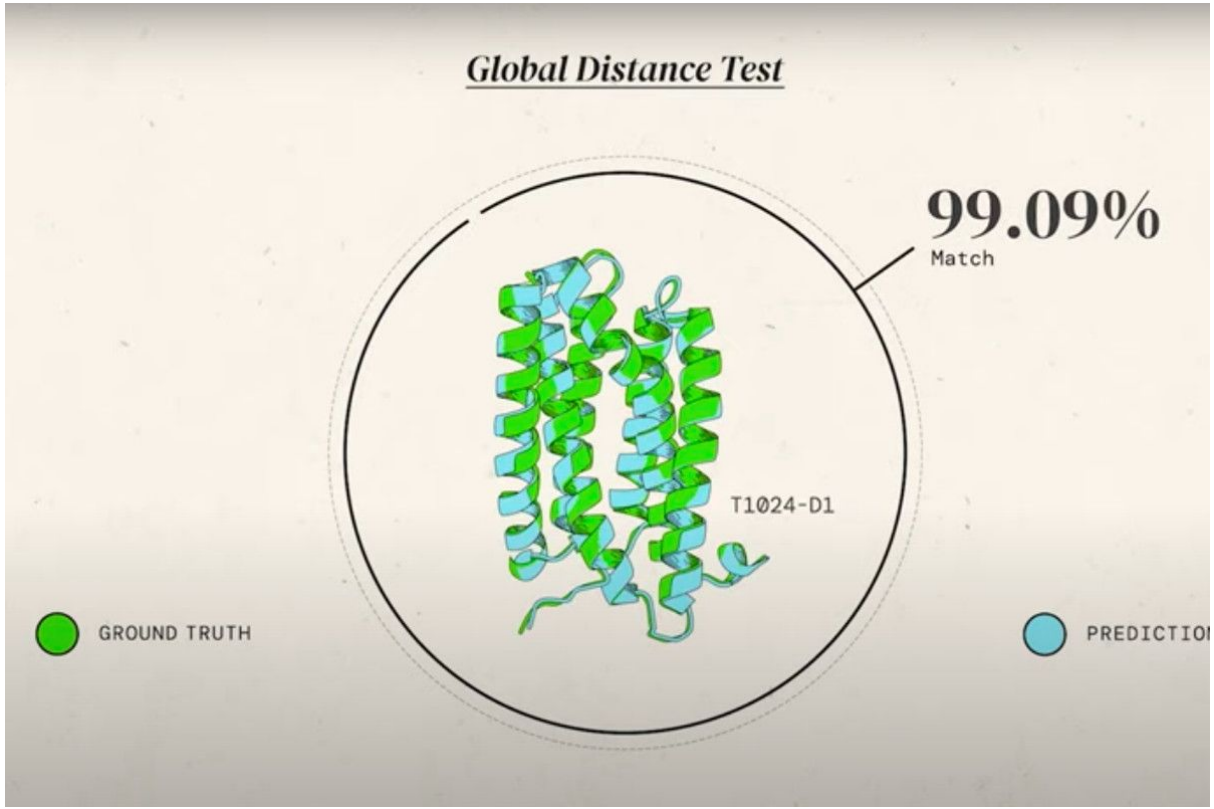
Seong Han

seonghan@seas.upenn.edu

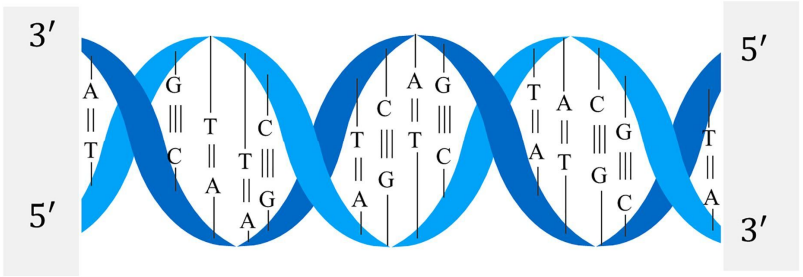
March 24, 2022

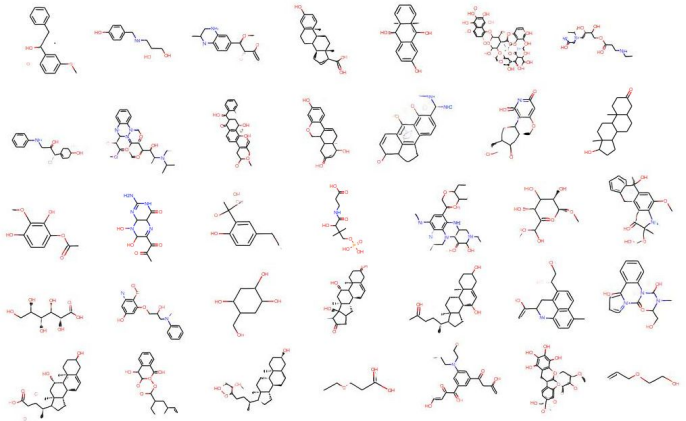

Machine
Learning
Algorithms
Biology

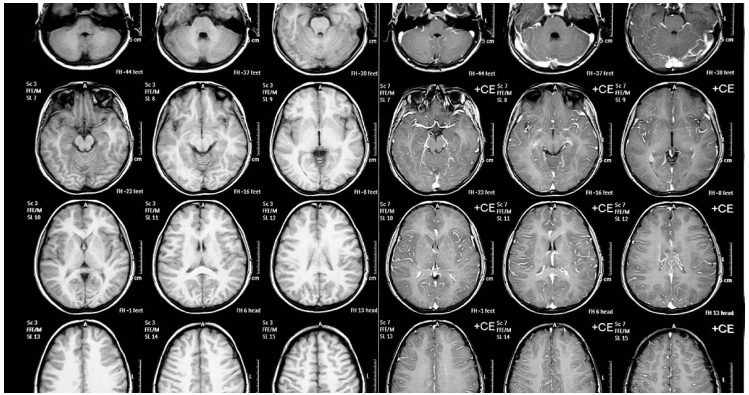
Alphafold


Global Distance Test
99.09%
Match
T1024-D1
GROUND TRUTH
PREDICTIO

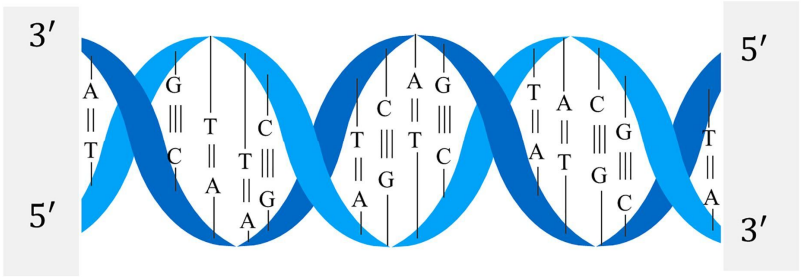
Examples

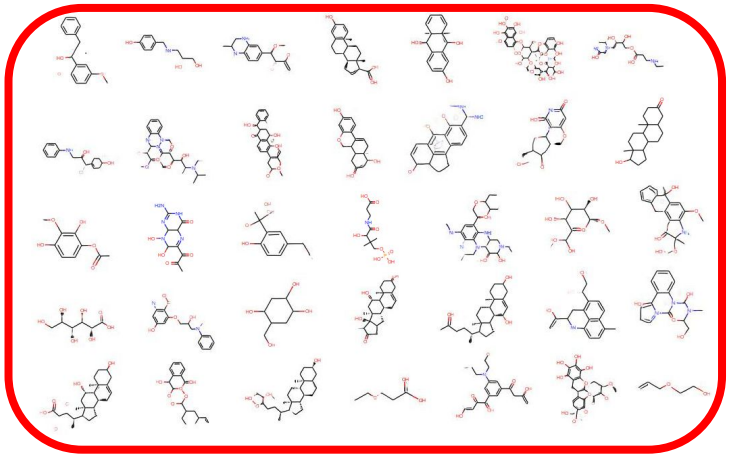


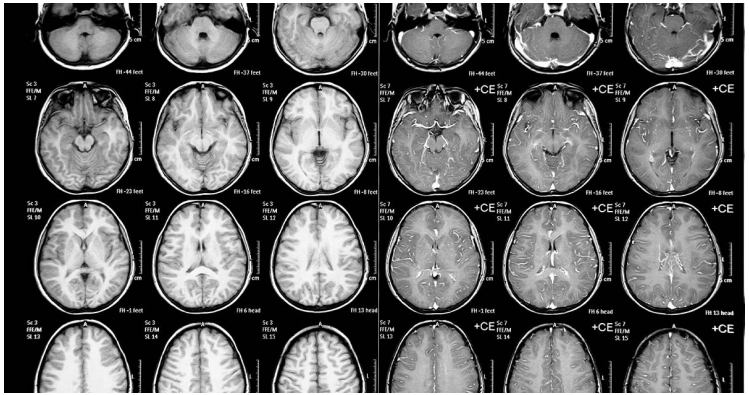




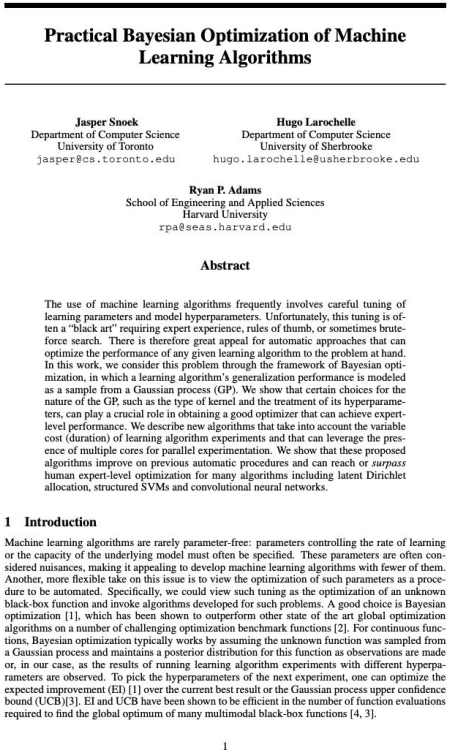
Examples







How did I get interested in Bayesian Optimization



Practical Bayesian Optimization of Machine Learning Algorithms

Jasper Snoek
Department of Computer Science
University of Toronto
jasper@cs.toronto.edu

Hugo Larochelle
Department of Computer Science
University of Sherbrooke
hugo.larochelle@usherbrooke.edu

Ryan P. Adams
School of Engineering and Applied Sciences
Harvard University
rpa@seas.harvard.edu

Abstract

The use of machine learning algorithms frequently involves careful tuning of learning parameters and model hyperparameters. Unfortunately, this tuning is often a "black art" requiring expert experience, rules of thumb, or sometimes brute-force search. There is therefore great appeal for automatic approaches that can optimize the performance of any given learning algorithm to the problem at hand. In this work, we consider this problem through the framework of Bayesian optimization, in which a learning algorithm's generalization performance is modeled as a sample from a Gaussian process (GP). We show that certain choices for the nature of the GP, such as the type of kernel and the treatment of its hyperparameters, can play a crucial role in obtaining a good optimizer that can achieve expert-level performance. We describe new algorithms that take into account the variable cost (duration) of learning algorithm experiments and that can leverage the presence of multiple cores for parallel experimentation. We show that these proposed algorithms improve on previous automatic procedures and can reach or *surpass* human expert-level optimization for many algorithms including latent Dirichlet allocation, structured SVMs and convolutional neural networks.

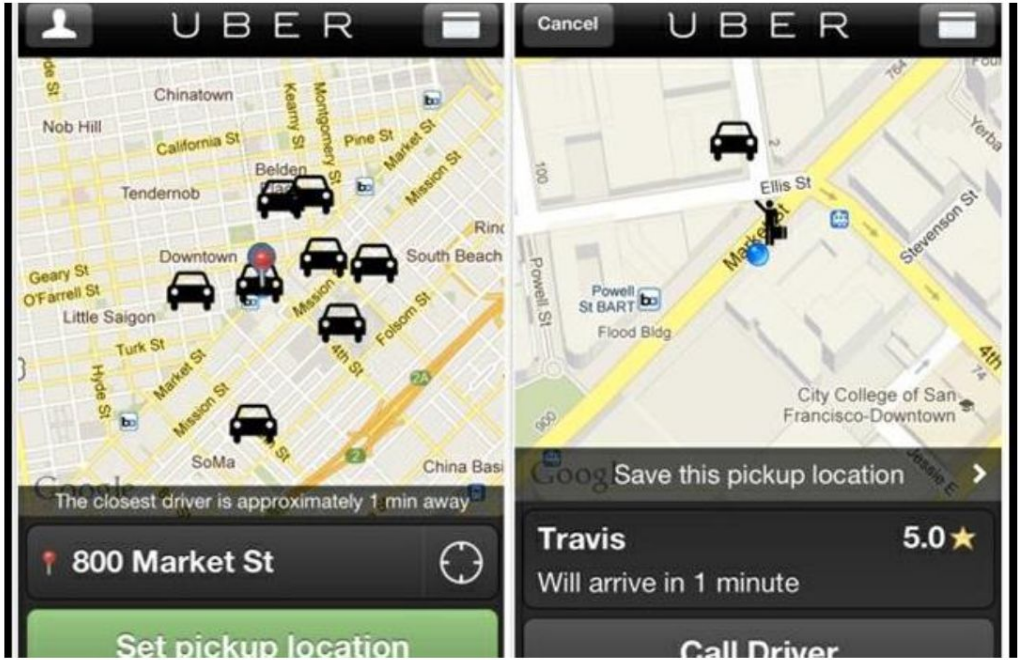
1 Introduction

Machine learning algorithms are rarely parameter-free: parameters controlling the rate of learning or the capacity of the underlying model must often be specified. These parameters are often considered nuisances, making it appealing to develop machine learning algorithms with fewer of them. Another, more flexible take on this issue is to view the optimization of such parameters as a procedure to be automated. Specifically, we could view such tuning as the optimization of an unknown black-box function and invoke algorithms developed for such problems. A good choice is Bayesian optimization [1], which has been shown to outperform other state of the art global optimization algorithms on a number of challenging optimization benchmark functions [2]. For continuous functions, Bayesian optimization typically works by assuming the unknown function was sampled from a Gaussian process and maintains a posterior distribution for this function as observations are made or, in our case, as the results of running learning algorithm experiments with different hyperparameters are observed. To pick the hyperparameters of the next experiment, one can optimize the expected improvement (EI) [1] over the current best result or the Gaussian process upper confidence bound (UCB)[3]. EI and UCB have been shown to be efficient in the number of function evaluations required to find the global optimum of many multimodal black-box functions [4, 3].

1

Snoek et al., NeurIPS 2012

Industry applications: Uber



Industry applications: Twitter

Applications: Biology, Chemistry

Frisby and Langmead *Algorithms Mol Biol* (2021) 16:13
https://doi.org/10.1186/s13015-021-00195-4

Algorithms for Molecular Biology

RESEARCH Open Access

Bayesian optimization with evolutionary and structure-based regularization for directed protein evolution

Trevor S. Frisby and Christopher James Langmead*

Abstract

Background: Directed evolution (DE) is a technique for protein engineering that involves iterative rounds of mutagenesis and screening to search for sequences that optimize a given property, such as binding affinity to a specified target. Unfortunately, the underlying optimization problem is under-determined, and so mutations introduced to improve the specified property may come at the expense of unmeasured, but nevertheless important properties (ex. solubility, thermostability, etc). We address this issue by formulating DE as a regularized Bayesian optimization problem where the regularization term reflects evolutionary or structure-based constraints.

Results: We applied our approach to DE to three representative proteins, GB1, BRCA1, and SARS-CoV-2 Spike, and evaluated both evolutionary and structure-based regularization terms. The results of these experiments demonstrate that: (i) structure-based regularization usually leads to better designs (and never hurts), compared to the unregularized setting; (ii) evolutionary-based regularization tends to be least effective; and (iii) regularization leads to better designs because it effectively focuses the search in certain areas of sequence space, making better use of the experimental budget. Additionally, like previous work in Machine learning assisted DE, we find that our approach significantly reduces the experimental burden of DE, relative to model-free methods.

Conclusion: Introducing regularization into a Bayesian ML-assisted DE framework alters the exploratory patterns of the underlying optimization routine, and can shift variant selections towards those with a range of targeted and desirable properties. In particular, we find that structure-based regularization often improves variant selection compared to unregularized approaches, and never hurts.

Keywords: Protein design, Bayesian optimization, Regularization, Directed evolution, Rational design, Gaussian process regression, Protein language model, Active learning

Introduction

The field of protein engineering seeks to design molecules with novel or improved properties [1]. The primary techniques used in protein engineering fall into two broad categories: *rational design* [2] and *directed evolution* (DE) [3]. Rational design uses model-driven in silico combinatorial searches to identify promising candidate designs, which are then synthesized and tested experimentally. Directed evolution, in contrast, involves iterative rounds of saturation mutagenesis at select residue positions, followed by in vitro or in vivo screening for desirable traits. The most promising sequences are then isolated and used to seed the next round of mutagenesis.

Traditionally, directed evolution is a model-free approach. That is, computational models are not used to guide or simulate mutagenesis. Recently, however, a technique for incorporating Machine learning (ML) into the DE workflow was introduced [4]. Briefly, this ML-assisted often utilizing deep learning, has now led to the prediction of synergistic drug combinations in a systematic manner. However, there is a dearth of drug combination datasets due to the large combinatorial space of possible experiments — ultimately limiting the quality of such predictions. A cost-efficient solution involves the use of sequential model optimization (SMO) where one performs both informative experiments ("*exploration*") and experiments that double-down on a promising hypothesis ("*exploitation*") [8]. Utilisation of such approaches can result in discovering most of the promising combinations within a given library while requiring substantially less experimentation than an exhaustive search. With an SMO platform available in conjunction with an appropriate *in vitro* assay, one has a powerful tool to rapidly respond to a future public health crisis.

There have now been a number of approaches for selection of drug combinations [9]. Classic bioinformatics approaches have focused on using machine learning and network statistics over specified features of drugs (e.g., molecular fingerprints [10]), cell lines (e.g., transcriptomics, copy number variations [11]) and interactome topology between biomolecules (e.g., protein–protein interactions, gene regulatory networks [12]). Initiatives such as the Dialogue on Reverse Engineering Assessment and Methods (DREAM) have led to a plethora of methods being benchmarked against one another in prospective challenges through the generation of novel datasets [13]. Relevant to searching through large

*Correspondence: cjl@cs.cmu.edu
Computational Biology Department, School of Computer Science, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA 15213, USA

Frisbry et al., AMB 2021

RECOVER: sequential model optimization platform for combination drug repurposing identifies novel synergistic compounds *in vitro*

Paul Bertin[1], Jarrid Rector-Brooks[1], Deepak Sharma[1], Thomas Gaudelet[2], Andrew Anighoro[2], Torsten Gross[2], Francisco Martínez-Peña[3], Eileen L. Tang[3], Suraj M S[2], Cristian Regep[2], Jeremy Hayter[2], Maksym Korablyov[1], Nicholas Valiante[4], Almer van der Sloot[5], Mike Tyers[5], Charles Roberts[2], Michael M. Bronstein[6,7], Luke L. Lairson[3], Jake P. Taylor-King[2], and Yoshua Bengio[1]

[1] Mila, the Quebec AI Institute, Canada [2] Relation Therapeutics, London, UK
[3] Department of Chemistry, The Scripps Research Institute, USA [4] Glyde Bio, Inc, USA.
[5] IRIC, Institute for Research in Immunology and Cancer, Université de Montréal, Canada
[6] Department of Computer Science, University of Oxford, UK [7] Twitter, UK
jake@relationrx.com, yoshua.bengio@mila.quebec

ABSTRACT

Selecting optimal drug repurposing combinations for further preclinical development is a challenging technical feat. Due to the toxicity of many therapeutic agents (e.g., chemotherapy), practitioners have favoured selection of synergistic compounds whereby lower doses can be used whilst maintaining high efficacy. For a fixed small molecule library, an exhaustive combinatorial chemical screen becomes infeasible to perform for academic and industry laboratories alike. Deep learning models have achieved state-of-the-art results *in silico* for the prediction of synergy scores. However, databases of drug combinations are highly biased towards synergistic agents and these results do not necessarily generalise out of distribution. We employ a sequential model optimization search applied to a deep learning model to quickly discover highly synergistic drug combinations active against a cancer cell line, while requiring substantially less screening than an exhaustive evaluation. Through iteratively adapting the model to newly acquired data, after only 3 rounds of ML-guided experimentation (including a calibration round), we find that the set of combinations queried by our model is enriched for highly synergistic combinations. Remarkably, we rediscovered a synergistic drug combination that was later confirmed to be under study within clinical trials.
Our method is available at: https://github.com/RECOVERcoalition/Recover.

1 INTRODUCTION

Drug combinations are an important therapeutic strategy for treating diseases that are subject to evolutionary dynamics, in particular cancers and infectious disease [1, 2]. Conceptually, as tumours or pathogens are subject to change over time, they may develop resistance to a single agent [3] — motivating one to target multiple biological targets simultaneously [4]. Discovering synergistic drug combinations is an important step towards developing robust therapies as they hold the potential for greater efficacy, reducing dose and thereby limiting the likelihood of adverse effects. There are approximately 20,000 proteins encoded by the human genome alone and many pathogenic targets beyond this. More concretely, in a drug repurposing scenario (i.e., uncovering new indications for known drugs), a small molecule library of >4,000 FDA approved drugs [5] leads to >8 million pairwise combinations; this does not appear tractable with standard high throughput screening (HTS) technology — even at a single dose [6].

With the recent COVID-19 global health crisis, there has been the need for rapid drug repurposing that would allow for expedited and de-risked clinical trials. Due to the complexity of selecting drug combinations and minimal training data publicly available, studies have typically been focused towards monotherapy repurposing from a variety of angles — often involving artificial intelligence (AI) techniques to provide recommendations [7]. Pioneering work,

Bertin et al., ArXiv 2022

Article

Bayesian reaction optimization as a tool for chemical synthesis

https://doi.org/10.1038/s41586-021-03213-y
Received: 24 June 2020
Accepted: 11 December 2020
Published online: 3 February 2021
Check for updates

Benjamin J. Shields[1], Jason Stevens[2], Jun Li[2], Marvin Parasram[1], Farhan Damani[2], Jesus I. Martinez Alvarado[1], Jacob M. Janey[2], Ryan P. Adams[3,4] & Abigail G. Doyle[1]

Reaction optimization is fundamental to synthetic chemistry, from optimizing the yield of industrial processes to selecting conditions for the preparation of medicinal candidates[1]. Likewise, parameter optimization is omnipresent in artificial intelligence, from tuning virtual personal assistants to training social media and product recommendation systems[2]. Owing to the high cost associated with carrying out experiments, scientists in both areas set numerous (hyper)parameter values by evaluating only a small subset of the possible configurations. Bayesian optimization, an iterative response surface-based global optimization algorithm, has demonstrated exceptional performance in the tuning of machine learning models[3]. Bayesian optimization has also been recently applied in chemistry[4–9]; however, its application and assessment for reaction optimization in synthetic chemistry has not been investigated. Here we report the development of a framework for Bayesian reaction optimization and an open-source software tool that allows chemists to easily integrate state-of-the-art optimization algorithms into their everyday laboratory practices. We collect a large benchmark dataset for a palladium-catalysed direct arylation reaction, perform a systematic study of Bayesian optimization compared to human decision-making in reaction optimization, and apply Bayesian optimization to two real-world optimization efforts (Mitsunobu and deoxyfluorination reactions). Benchmarking is accomplished via an online game that links the decisions made by expert chemists and engineers to real experiments run in the laboratory. Our findings demonstrate that Bayesian optimization outperforms human decisionmaking in both average optimization efficiency (number of experiments) and consistency (variance of outcome against initially available data). Overall, our studies suggest that adopting Bayesian optimization methods into everyday laboratory practices could facilitate more efficient synthesis of functional chemicals by enabling better-informed, data-driven decisions about which experiments to run.

Optimization of a chemical reaction is a complex, multidimensional challenge that requires experts to evaluate various reaction parameters, such as substrate, catalyst, reagent, additive, solvent, concentration, temperature and reactor type (Fig. 1a). Yet in a typical laboratory, bench chemists can evaluate only a small subset of these conditions during a standard optimization campaign owing to time and material limitations. Modern advances in high-throughput experimentation (HTE) have extended experimental capabilities to the collection of a few thousand data points under a limited set of conditions[10]. Thus, the chemist's art is to differentiate between millions of plausible configurations using a laboratory equipped to run only a tiny fraction of the possibilities. To do so, chemists typically carry out their experiments by scouring the chemical literature for similar reactions and intuiting the most influential dimensions (that is, reaction parameters) for reaction success on the basis of experience, mechanistic understanding, empirical data and simple heuristics (Fig. 1b).

Chemists also commonly utilize systematic, model-driven approaches to reaction optimization[11–15]. For example, design of experiments (DOE) seeks to sample experimental conditions that facilitate modelling of reaction parameters and deconvolution of interactions (Fig. 1b)[11,14,15]. In conjunction with a response surface model, DOE enables the exploitation of knowledge gained from previous evaluations to guide the selection of future experiments. However, the exploration of reaction space is typically left in the hands of pre-defined optimal designs, sensitivity analysis, literature precedence and the operator's intuition[1]. In addition, although a typical reaction requires the fine-tuning of numerous discrete parameters, screening requirements grow exponentially with the number of categorical components using

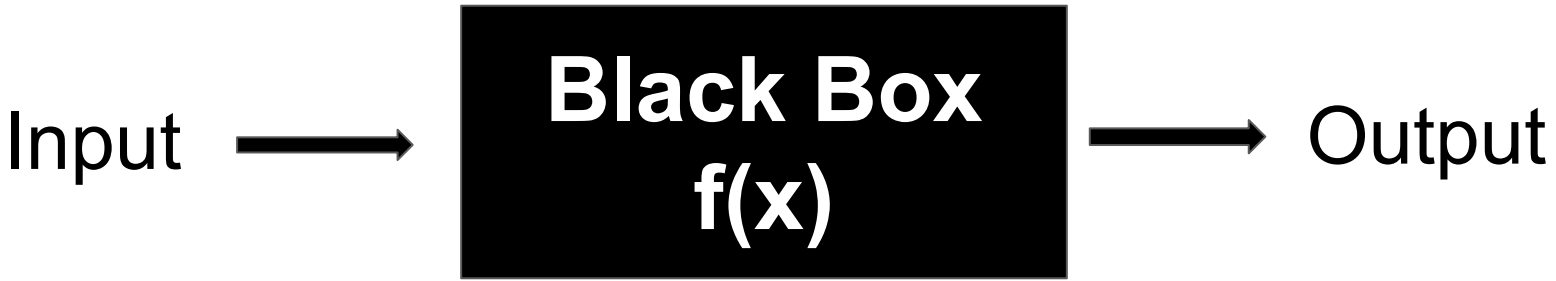
[1]Department of Chemistry, Princeton University, Princeton, NJ, USA. [2]Chemical Process Development, Bristol-Myers Squibb, New Brunswick, NJ, USA. [3]Department of Computer Science, Princeton University, Princeton, NJ, USA. ✉e-mail: rpa@princeton.edu; agdoyle@princeton.edu

Shields et al., Nature 2021

Motivation

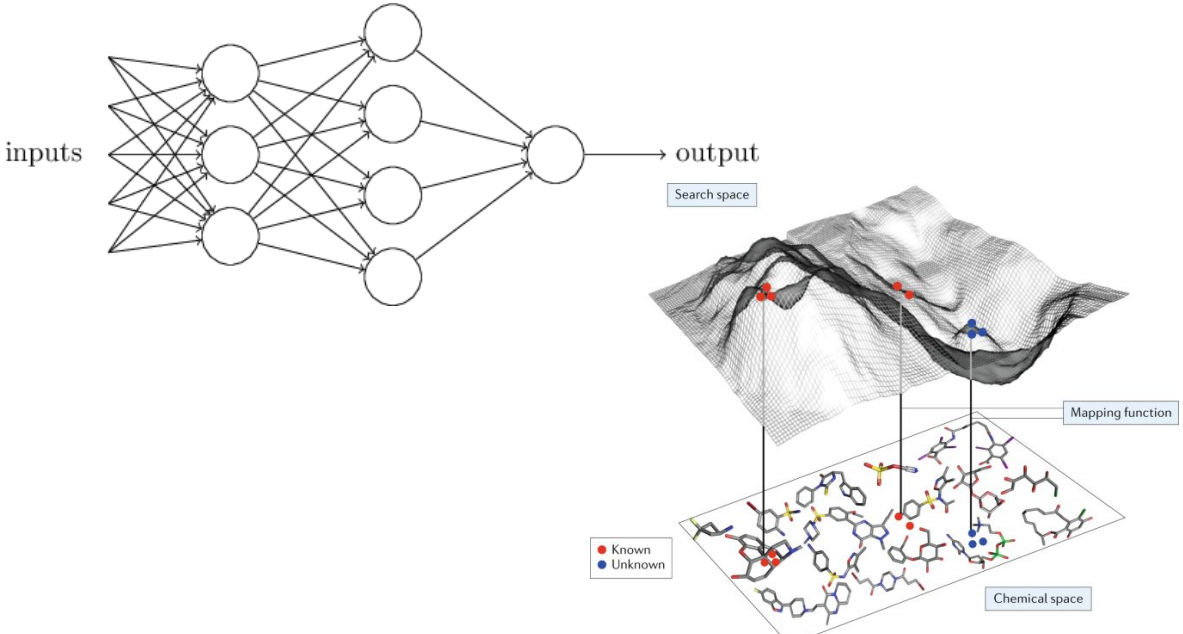
- Many optimization problems in machine learning are black box optimization problems where the function is a black box function

- Don’t have an analytical expression for the function



Motivation

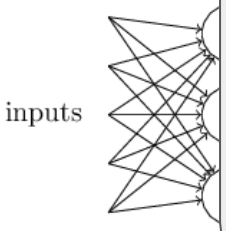
- Function evaluation is very expensive

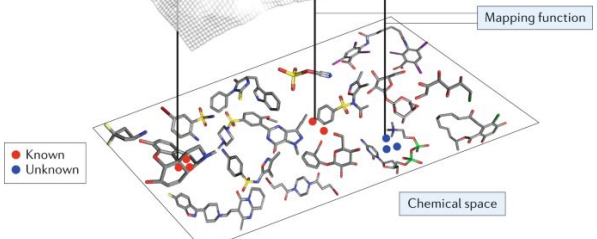


Motivation

- Function evaluation is very expensive

Optimize expensive-to-evaluate function





What do we do?

- Grid search

  If f(x) is L-Lipschitz continuous, and we are in a noise-free domain to guarantee that we propose some $x_{M,n}$ such that

$$f(x_M) - f(x_{M,n}) \leq \epsilon$$

  We need to evaluate f(x) on a D-dimensional unit hypercube:

$$(L/\epsilon)^D \text{ evaluations}$$

What do we do?

- Grid search

  If f(x) is L-Lipschitz continuous, and we are in a noise-free domain to guarantee that we propose some $x_{M,n}$ such that

$$f(x_M) - f(x_{M,n}) \leq \epsilon$$

  We need to evaluate f(x) on a D-dimensional unit hypercube:

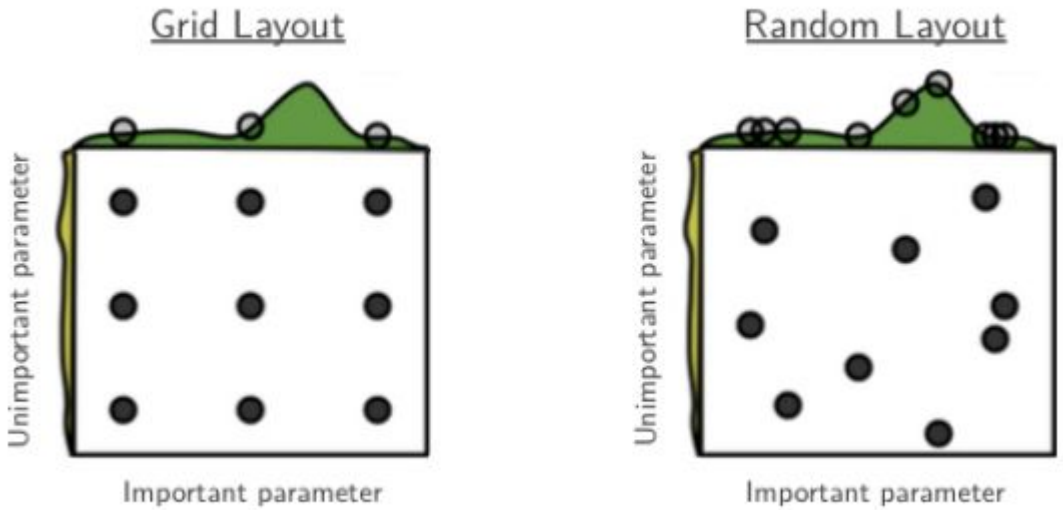
$$(L/\epsilon)^D \text{ evaluations}$$

$$(10/0.01)^5 = 10e14$$

What do we do?

- Random search

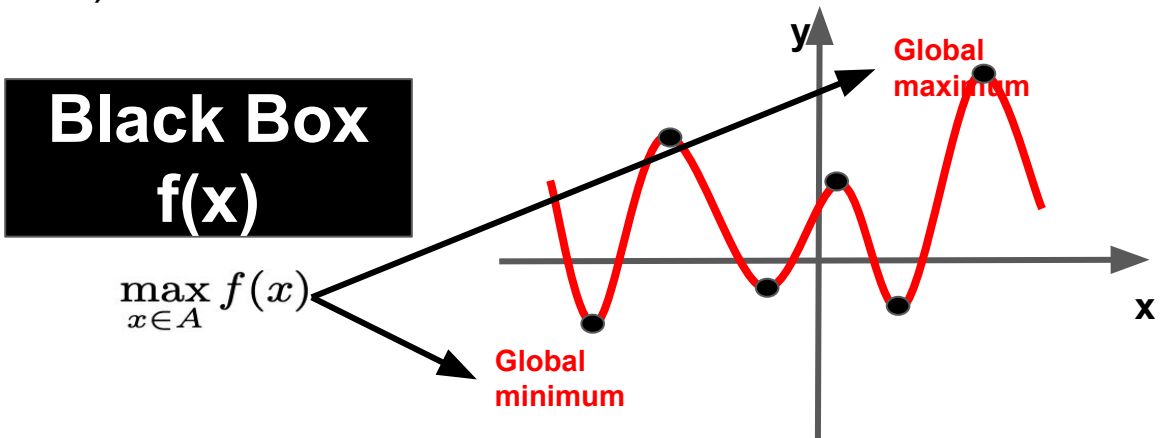
  Sample space uniformly [Bergstra and Bengio 2012]



Intuition

- BO uses all the information from previous evaluations and performs some computation to determine the next point to try

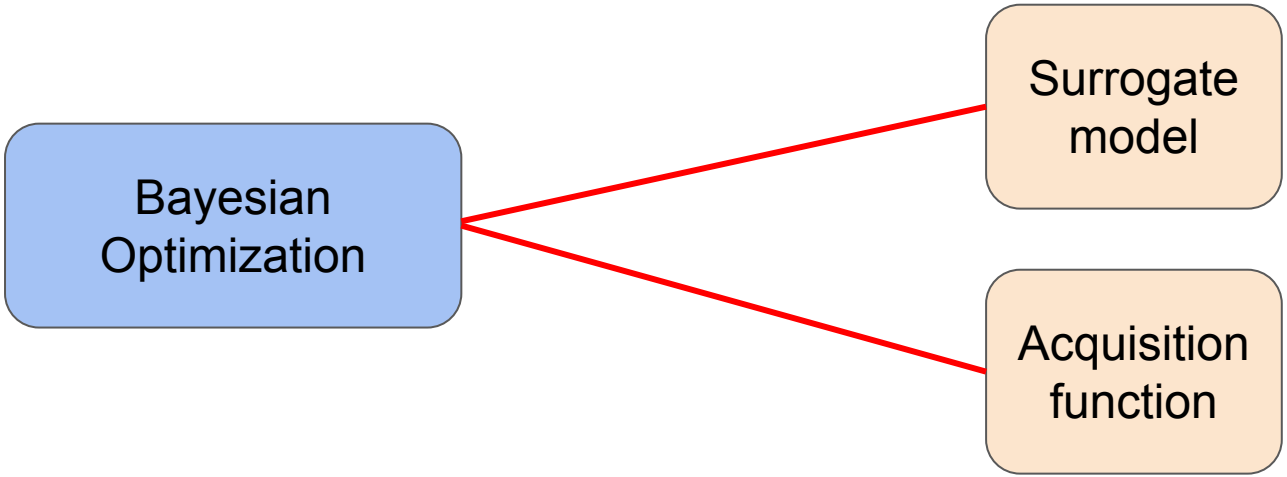
- We want to find the peak of our true function (accuracy of the function's hyperparameter)



Intuition

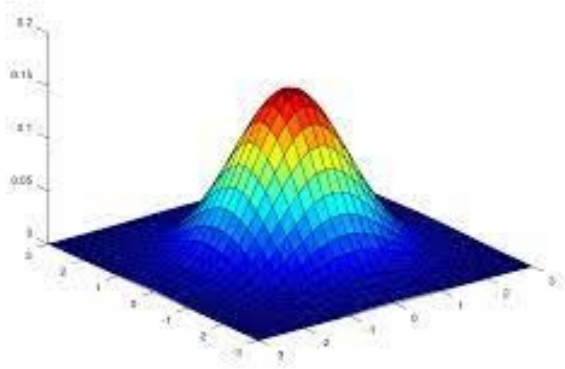
- **Surrogate model** for approximating the objective function

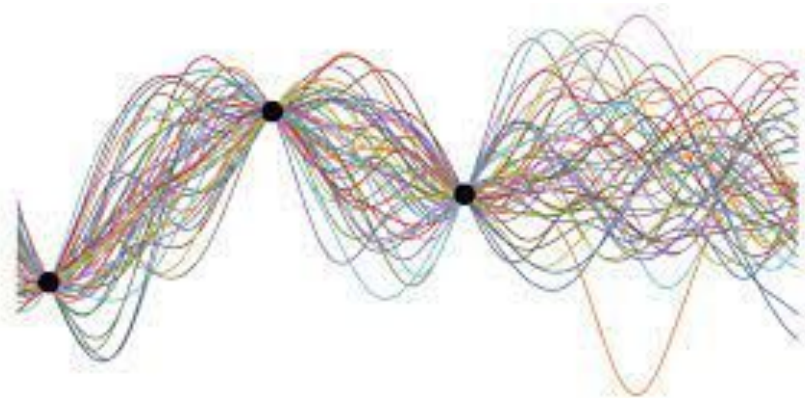
- **Acquisition function** for sampling to areas where there is an improvement over the current best observation



Gaussian Process (Surrogate Model)

- GP is a non-parametric model used to infer a distribution over functions





Multivariate normal distribution
Mean vector / covariance matrix

Gaussian Process
Prior mean function / Prior covariance function

Gaussian Process

- GP has the property that the function values at any finite set of inputs are jointly Gaussian distributed

$$\mathbf{f} = [f(\mathbf{x}_1), ..., f(\mathbf{x}_N)] \sim \mathcal{N}(\mu_X, K_{XX})$$

$$\mu_X = [\mu(\mathbf{x}_1), ..., \mu(\mathbf{x}_N)]^\top \qquad K_{XX} = [k(\mathbf{x}_i, \mathbf{x}_j)]_{i,j=1}^N$$

Gaussian Process

- Given a training dataset $\mathcal{D} = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\}$, the goal is to infer the posterior distribution $p(f \mid \mathcal{D})$

- Under the Gaussian observation model, the distribution is also a Gaussian

$$p(f(\mathbf{x}^*) \mid \mathcal{D}) \sim \mathcal{GP}\left(\mu_{f|\mathcal{D}}(\mathbf{x}^*), k_{f|\mathcal{D}}(\mathbf{x}^*, \mathbf{x}^{*'})\right)$$

Gaussian Process

- Given some set of test points X^*, the definition of a GP implies that $f(X^*) = [f(\mathbf{x}_1^*), ..., f(\mathbf{x}_t^*)]$ $[y_1, ..., y_n]$ ɔintly Gaussian distributed

$$\begin{bmatrix} \mathbf{y} \\ f(X^*) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu(\mathbf{y}) \\ \mu(f(X^*)) \end{bmatrix}, \begin{bmatrix} K_{XX} + \sigma^2 \mathbf{I} & K_{XX^*} \\ K_{XX^*}^\top & K_{X^*X^*} \end{bmatrix} \right)$$

$$p(\mathbf{y}_1, \mathbf{y}_2) = \mathcal{N}\left(\begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}, \begin{bmatrix} \mathrm{A} & \mathrm{B} \\ \mathrm{B}^\mathsf{T} & \mathrm{C} \end{bmatrix} \right)$$

Gaussian Process

$$p(\mathbf{y}_1, \mathbf{y}_2) = \mathcal{N}\left(\begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}, \begin{bmatrix} A & B \\ B^T & C \end{bmatrix}\right)$$

$$P(y_1|y_2) = N(a + BC^{-1}(y_2 - b), \quad A - BC^{-1}B^T)$$

$$\mu_{f|\mathcal{D}}(\mathbf{x}) = \mu(\mathbf{x}^*) + K_{\mathbf{x}^*X}\hat{K}_{XX}^{-1}\mathbf{y}$$

Predictive Mean

$$k_{f|\mathcal{D}}(\mathbf{x}^*, \mathbf{x}^*) = K_{\mathbf{x}^*\mathbf{x}^*} - K_{\mathbf{x}^*X}\hat{K}_{XX}^{-1}K_{\mathbf{x}^*X}^{\top}$$

Predictive Covariance (Kernel function)

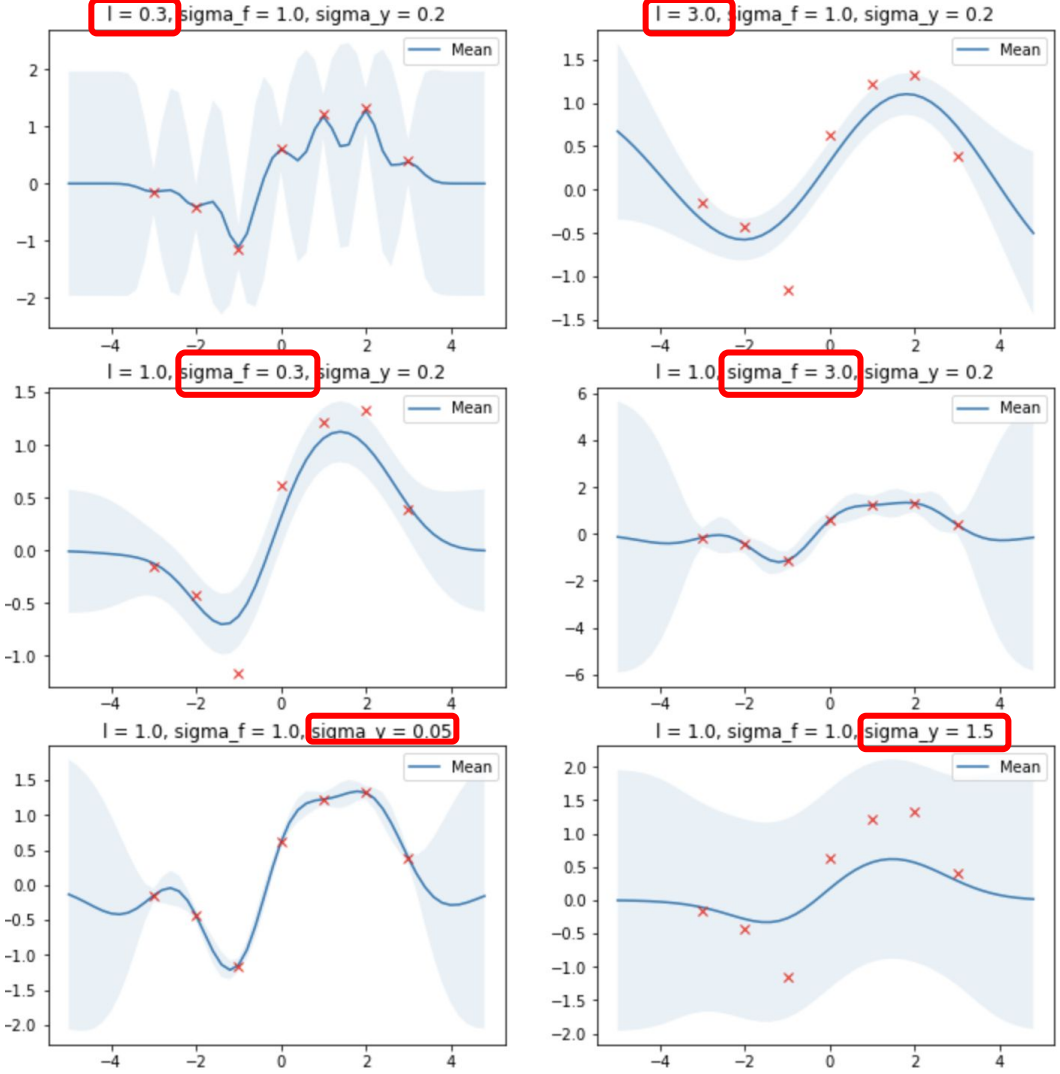
Gaussian Process

RBF Kernel (exponential kernel)

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \exp\left(-\frac{1}{2l^2}(\mathbf{x}_i - \mathbf{x}_j)^T(\mathbf{x}_i - \mathbf{x}_j)\right)$$

Vertical variation

Smoothness


l = 0.3, sigma_f = 1.0, sigma_y = 0.2
Mean
l = 3.0, sigma_f = 1.0, sigma_y = 0.2
Mean
l = 1.0, sigma_f = 0.3, sigma_y = 0.2
Mean
l = 1.0, sigma_f = 3.0, sigma_y = 0.2
Mean
l = 1.0, sigma_f = 1.0, sigma_y = 0.05
Mean
l = 1.0, sigma_f = 1.0, sigma_y = 1.5
Mean

Gaussian Process

- Kernel matrices depend on hyperparameters θ

- The Log marginal likelihood of the data is given by

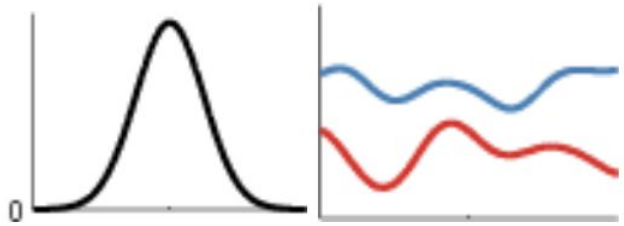
$$\log p(\mathbf{y} \mid \theta) = -\frac{1}{2}\mathbf{y}^\top \hat{K}_{XX}^{-1}\mathbf{y} - \frac{1}{2}\log|\hat{K}_{XX}| + \mathbf{c}$$

- All prior hyperparameters can be treated by maximizing or by choosing a prior p(θ) and sampling from $p(\theta|D) \propto p(y|\theta)p(\theta)$

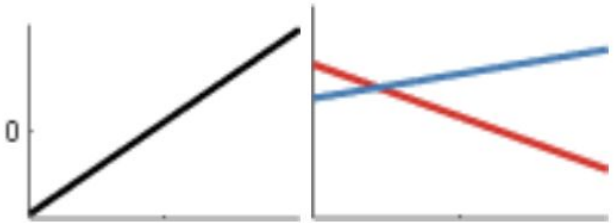
Gaussian Process

$$\begin{bmatrix} \mathbf{y} \\ f(X^*) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu(\mathbf{y}) \\ \mu(f(X^*)) \end{bmatrix}, \begin{bmatrix} K_{XX} + \sigma^2\mathbf{I} & K_{XX^*} \\ K_{XX^*}^\top & K_{X^*X^*} \end{bmatrix} \right)$$

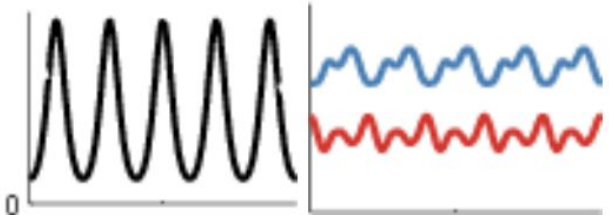
Squared Exponential Kernel



Linear Kernel

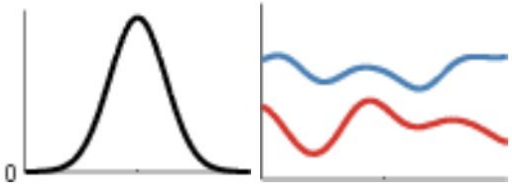


Periodic Kernel

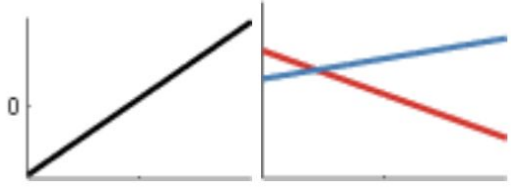


Gaussian Process

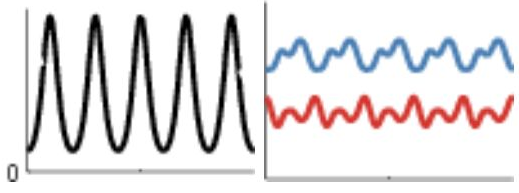
Squared Exponential Kernel

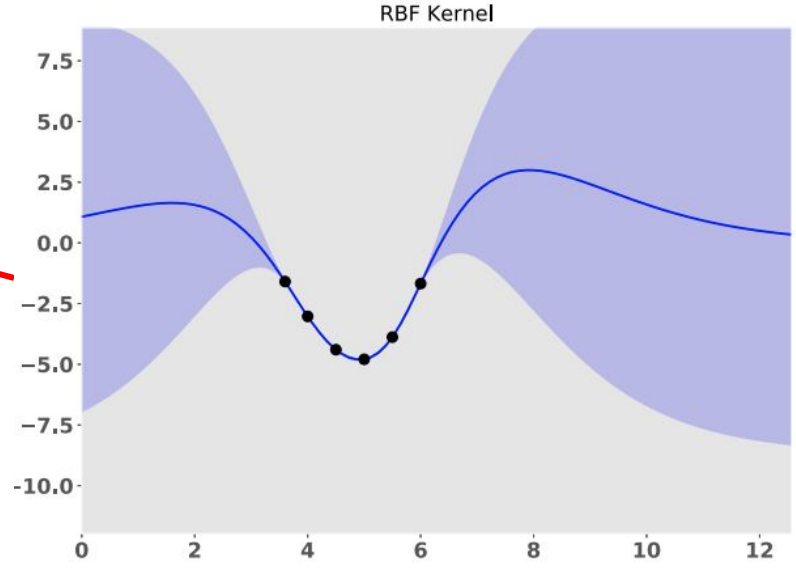


Linear Kernel



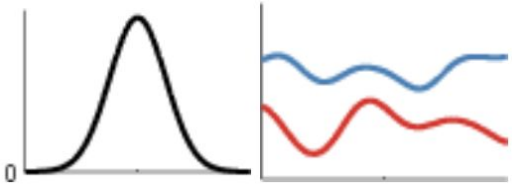
Periodic Kernel



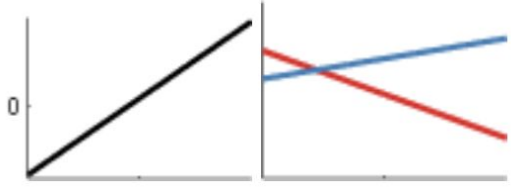


Gaussian Process

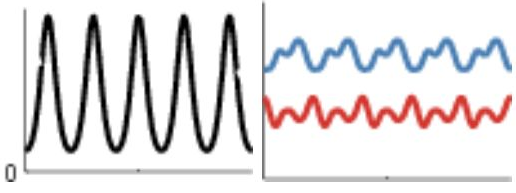
Squared Exponential Kernel

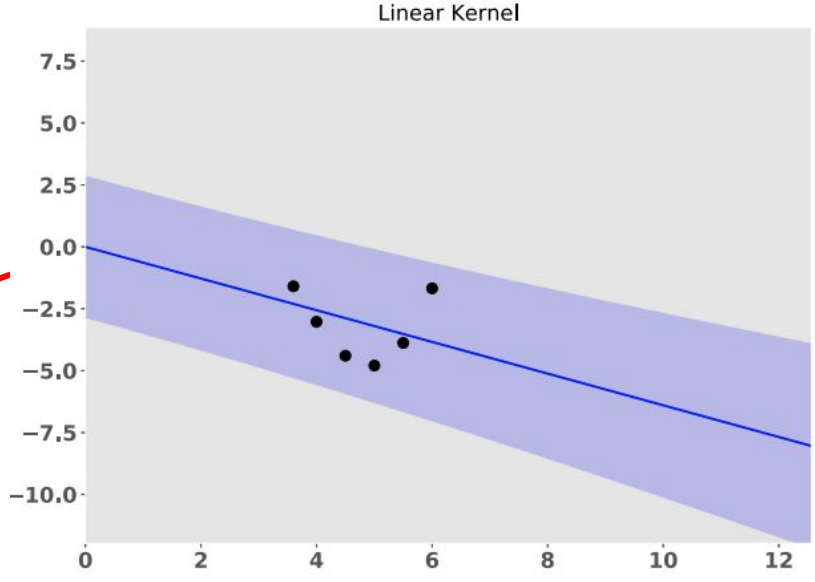


Linear Kernel



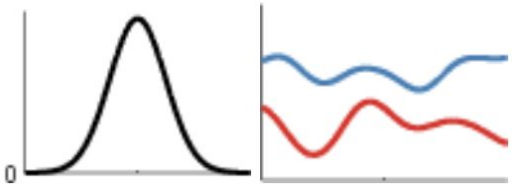
Periodic Kernel



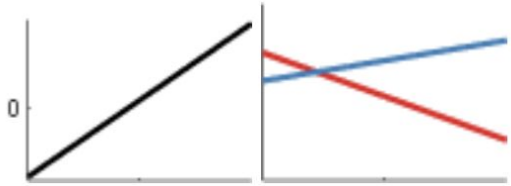


Gaussian Process

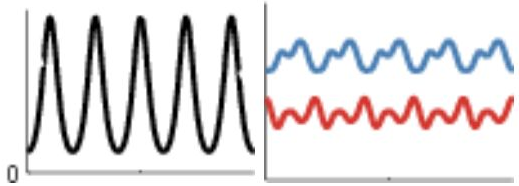
Squared Exponential Kernel



Linear Kernel



Periodic Kernel



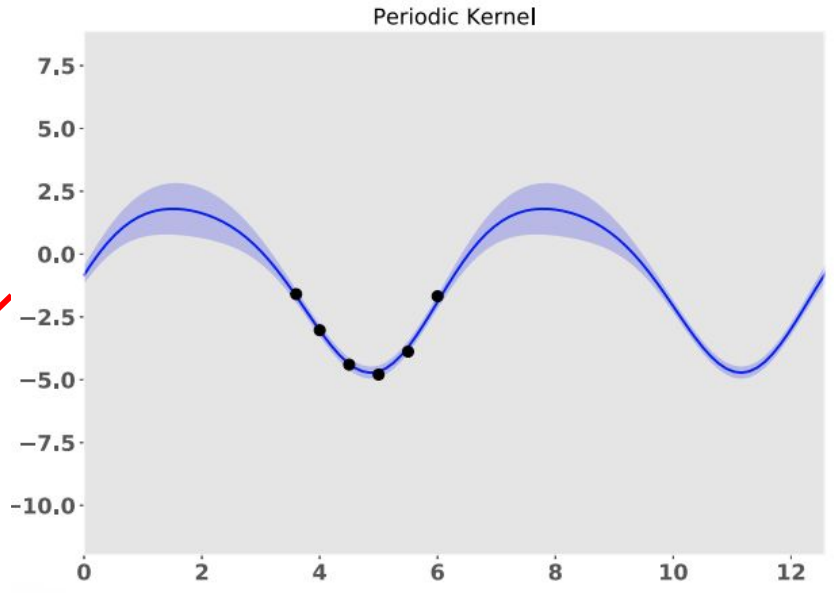


Illustration of Gaussian Process

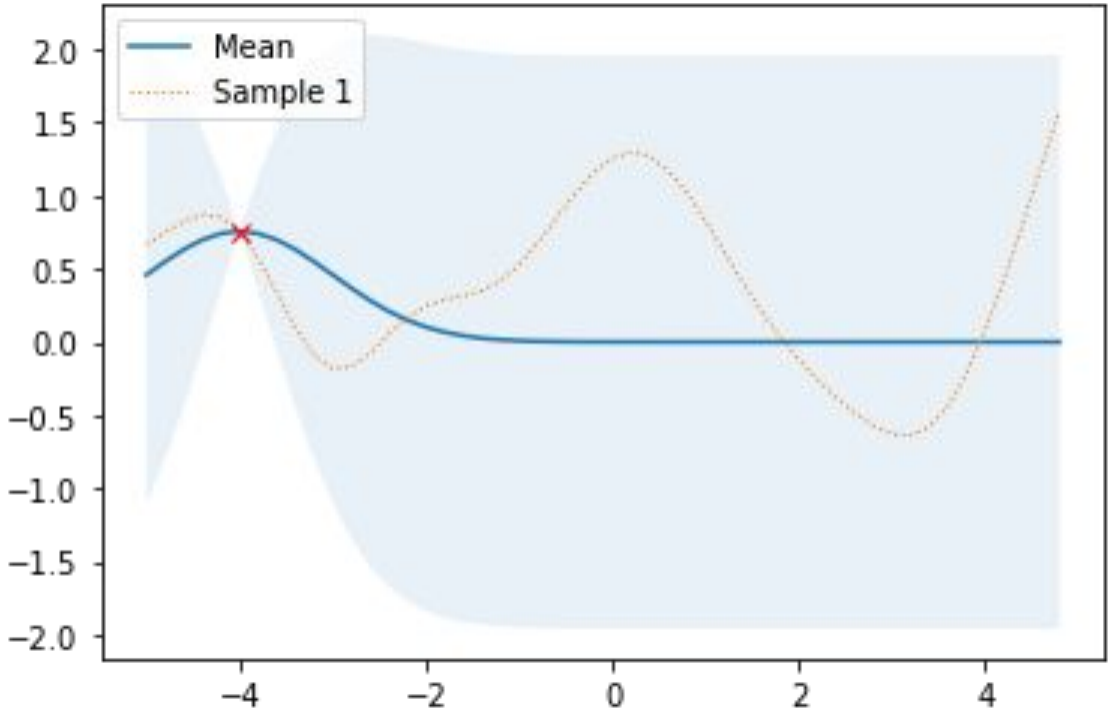


Illustration of Gaussian Process

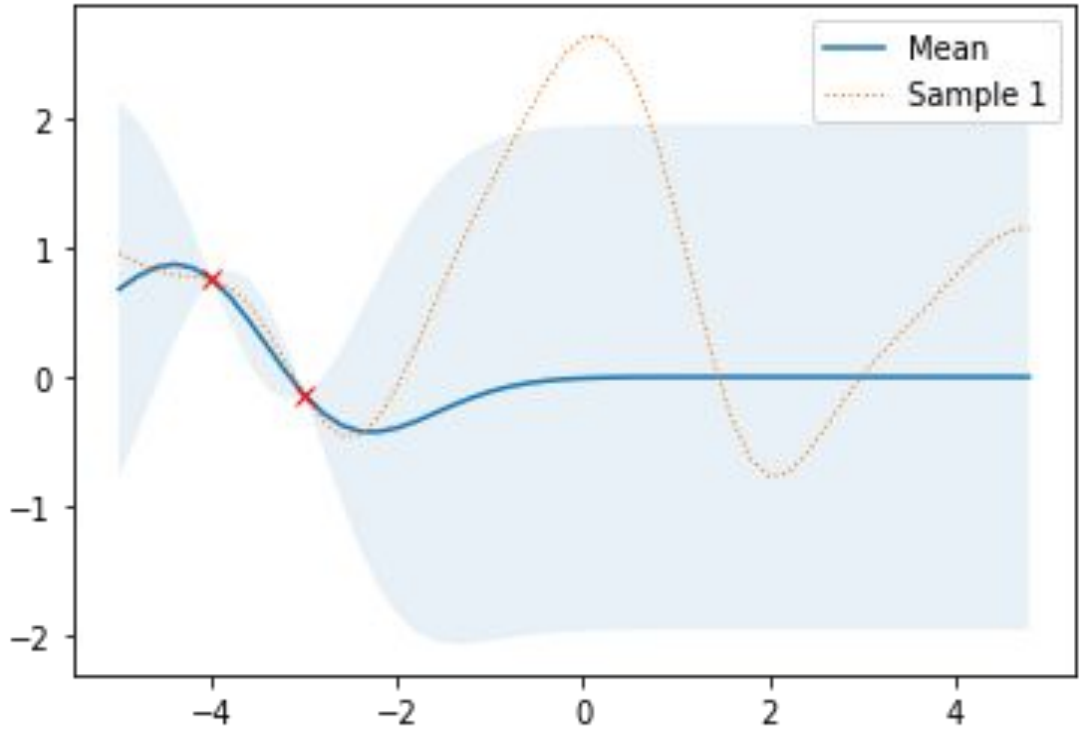


Illustration of Gaussian Process

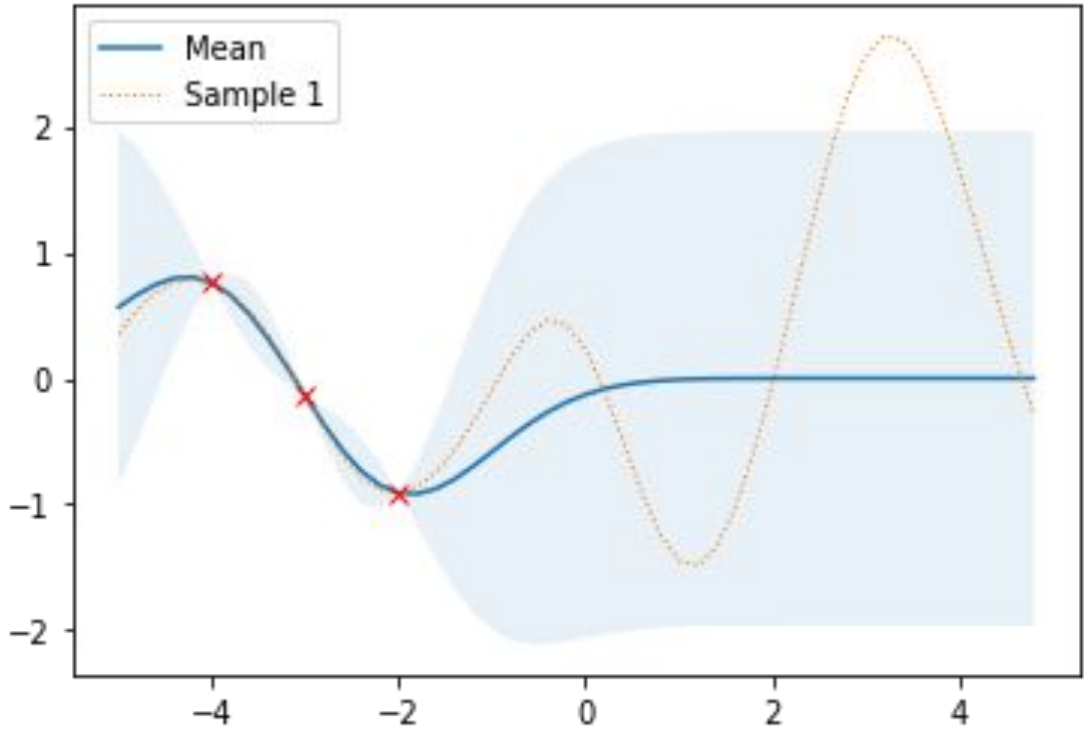


Illustration of Gaussian Process

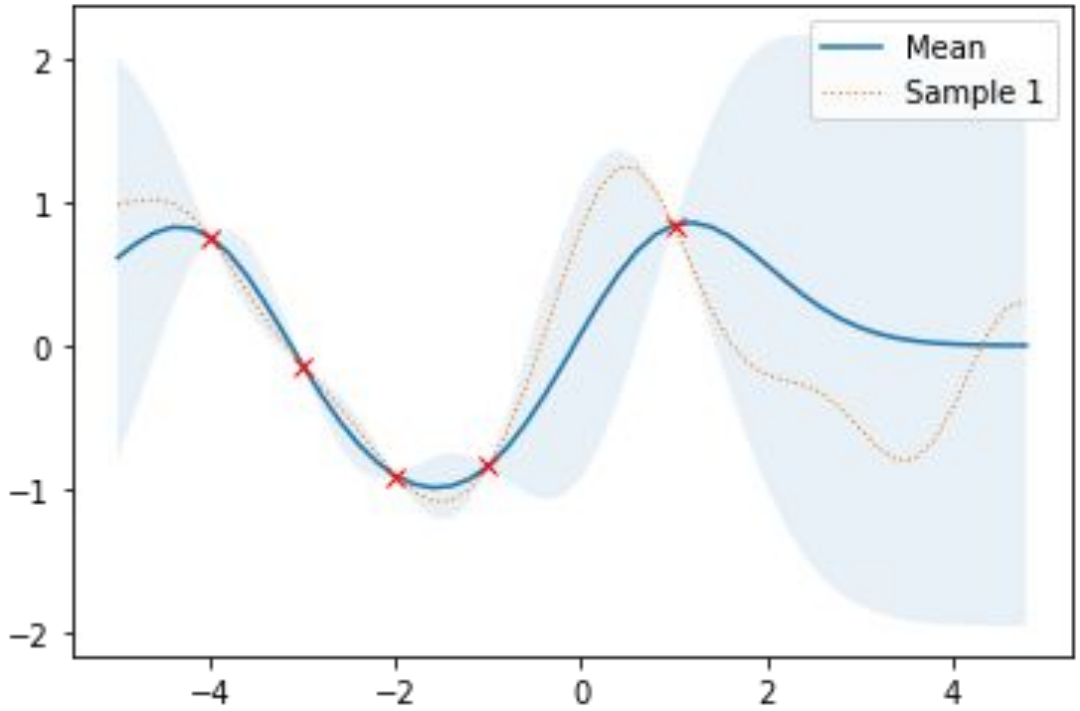
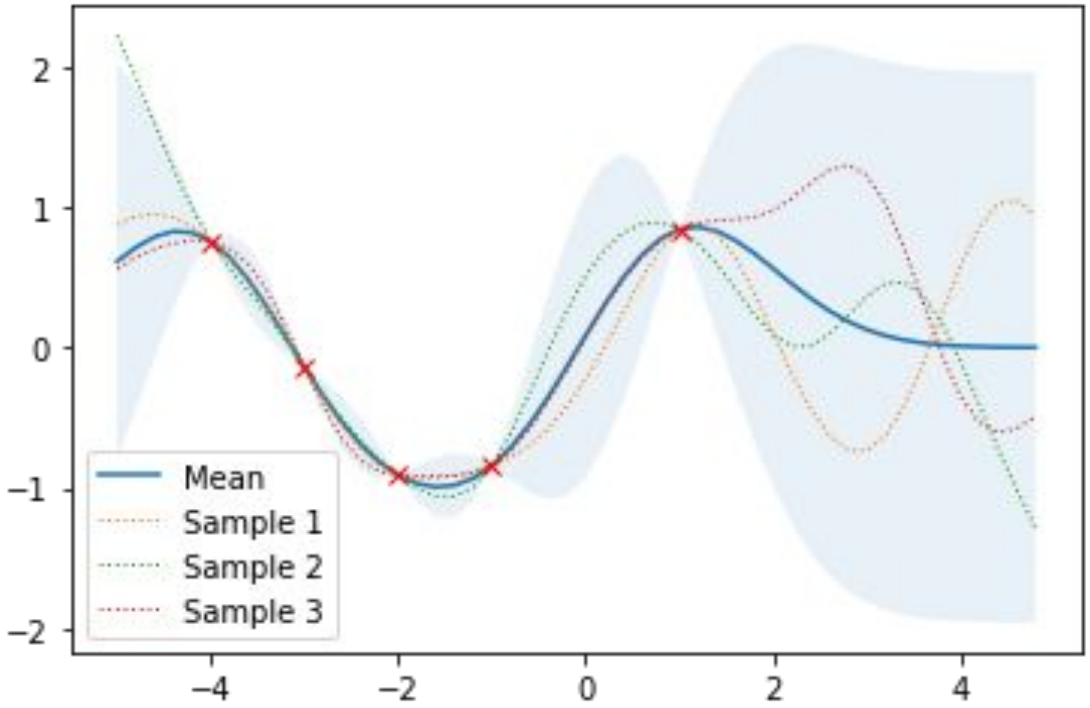


Illustration of Gaussian Process



Expected improvement (Acquisition Function)

- Used to evaluate how promising each candidate $\mathbf{x}^*$ is

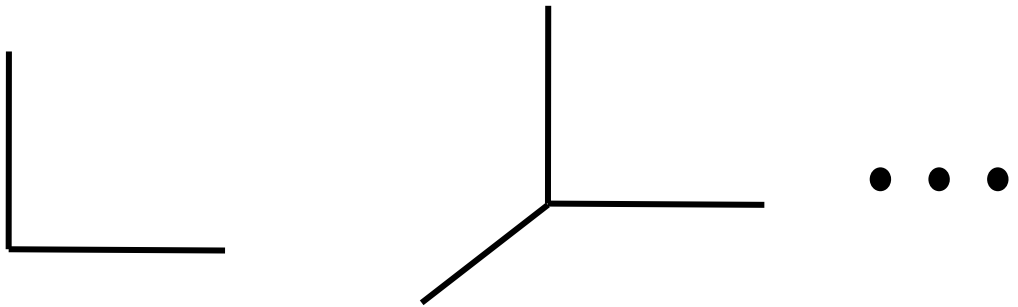
- Define $I(f(\mathbf{x}^*))$ as the improvement obtained by sampling at $\mathbf{x}^*$

$$I\big(f(x^*) - f(x_{best})\big) = \max(0, f_{best} - f(x^*))$$

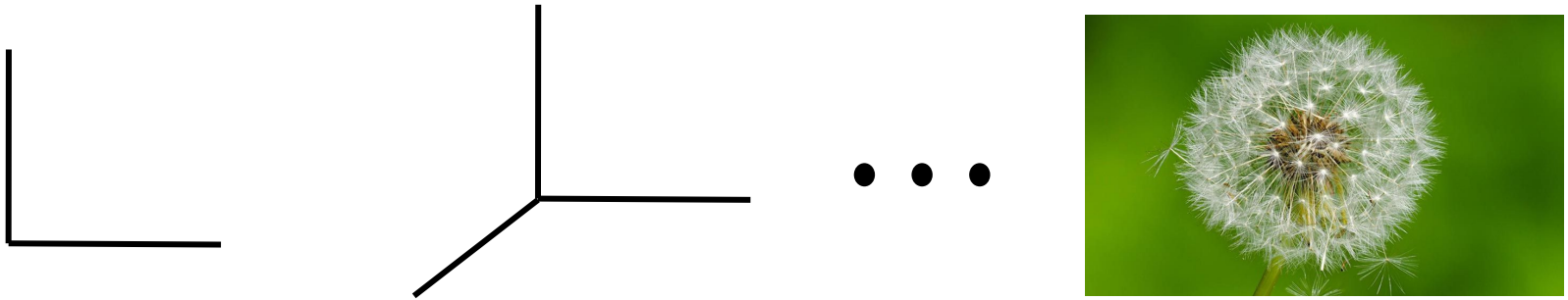
- The EI function evaluates the expectation of the above function over the predictive posterior

$$\begin{aligned}\mathrm{EI}(\mathbf{x}^*) &= \mathbb{E}\left[I\big(f(\mathbf{x}^*)\big)\right]_{p(f(\mathbf{x}^*)|\mathcal{D},\mathbf{x}^*)} \\ &= \int_{-\infty}^{f_{\text{best}}} \big(f_{\text{best}} - f(\mathbf{x}^*)\big) p(f(\mathbf{x}^*) \mid \mathcal{D}, \mathbf{x}^*)\, \mathrm{d}f(\mathbf{x}^*)\end{aligned}$$

Expected improvement (Acquisition Function)



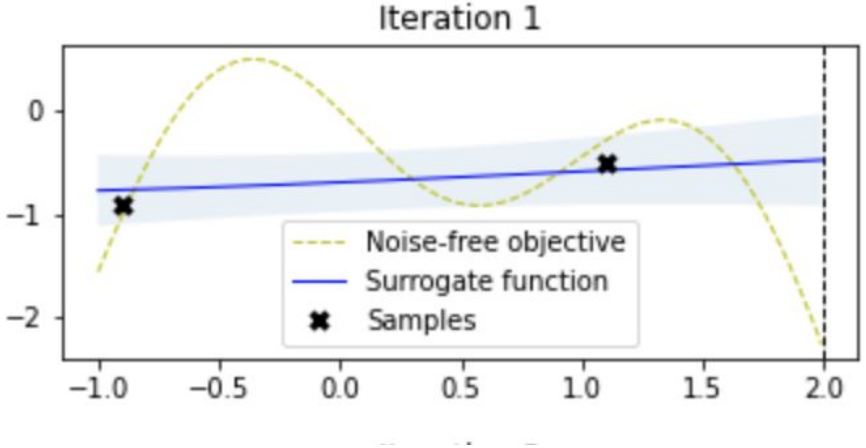
Expected improvement (Acquisition Function)



Expected improvement (Acquisition Function)

- Propose sampling points in the search space
- The acquisition function trades-off exploitation and exploration
- Goal is to maximize the acquisition function to determine the next sampling point

Illustration of Bayesian Optimization



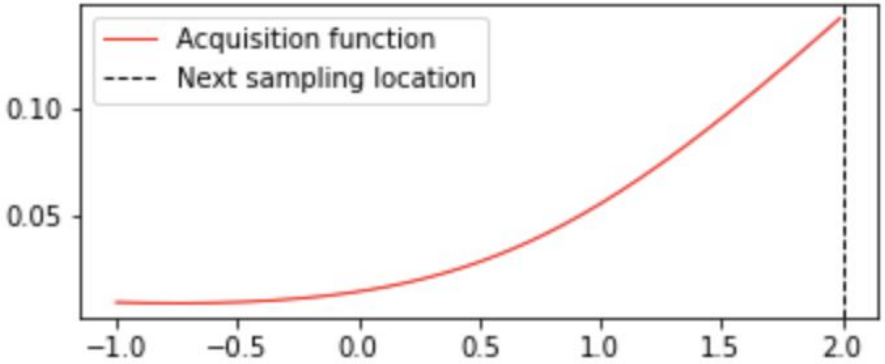
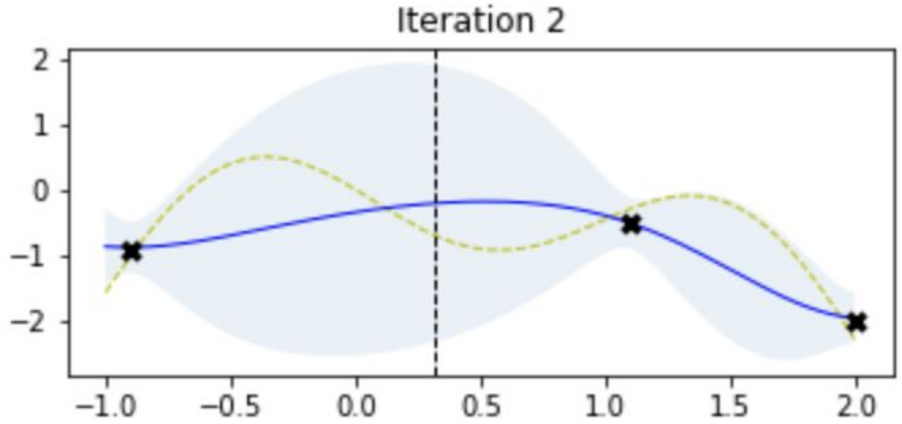


Illustration of Bayesian Optimization



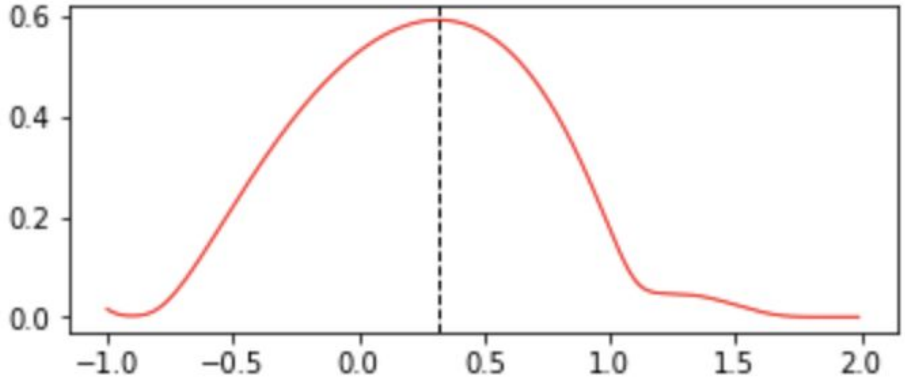


Illustration of Bayesian Optimization

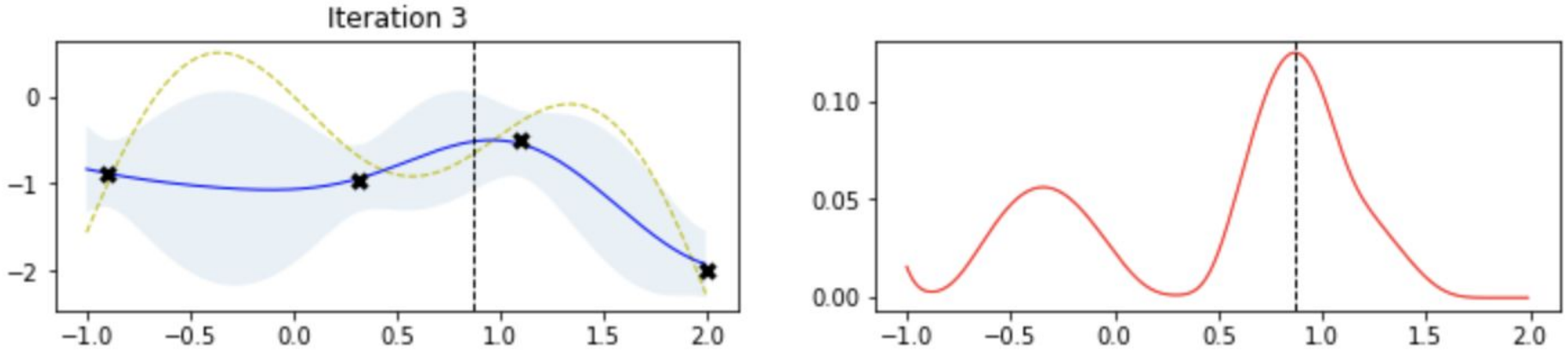
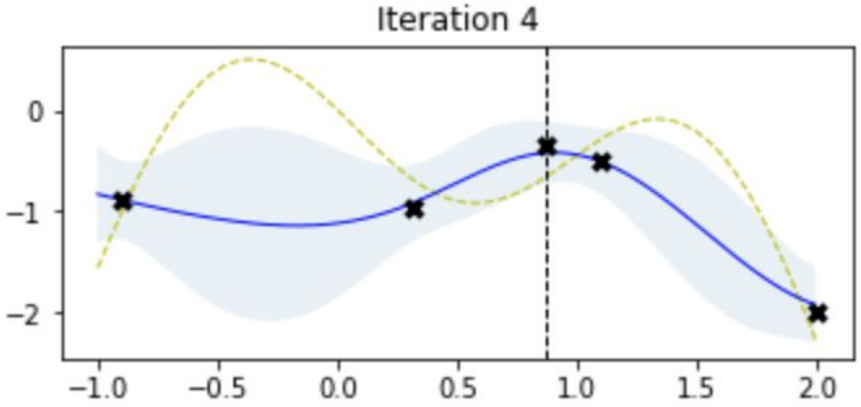


Illustration of Bayesian Optimization



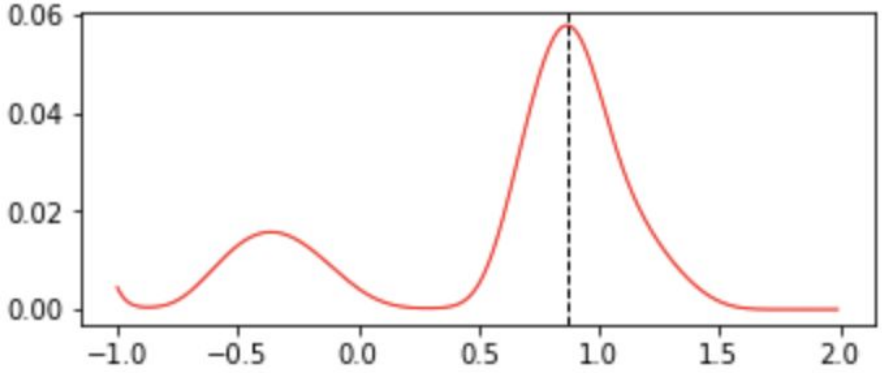
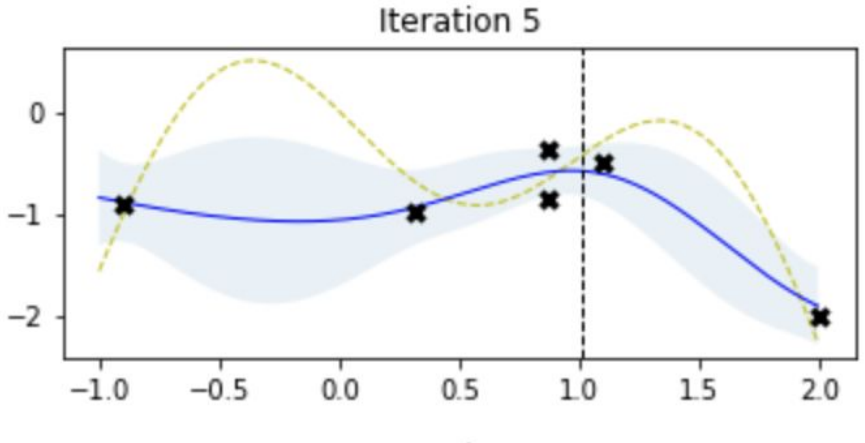


Illustration of Bayesian Optimization



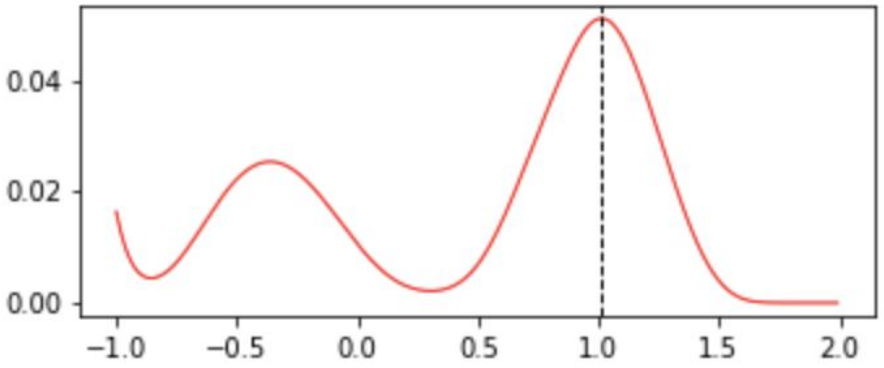


Illustration of Bayesian Optimization

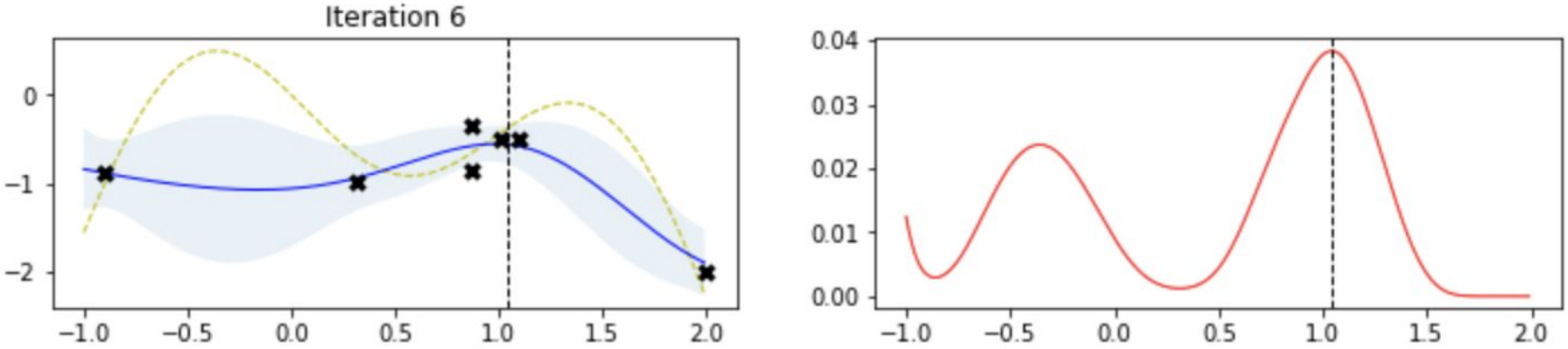


Illustration of Bayesian Optimization

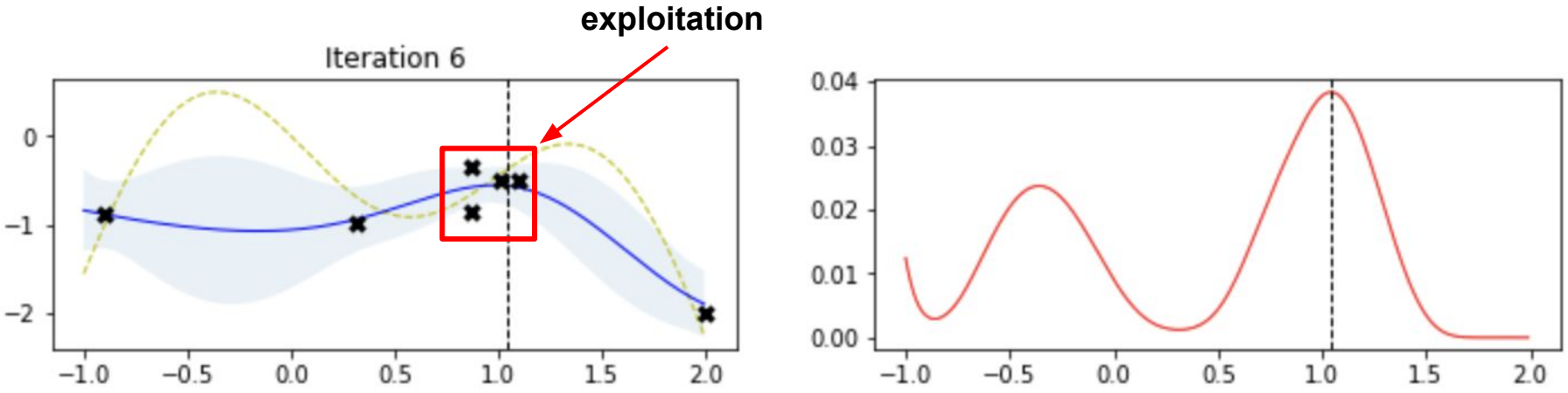
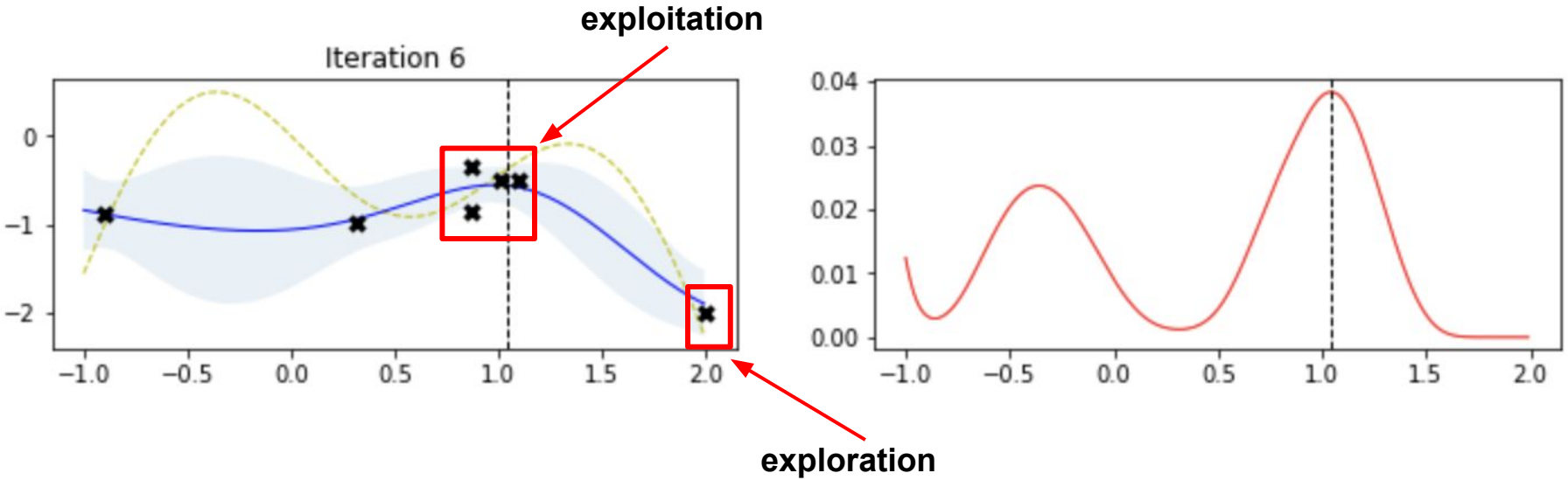
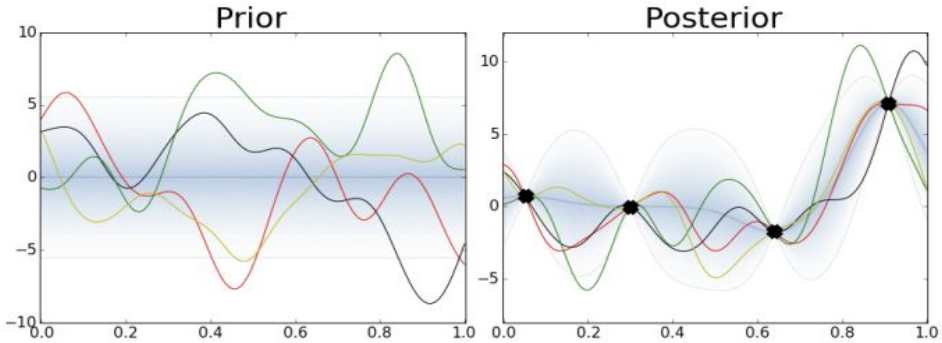


Illustration of Bayesian Optimization



Bayesian Optimization

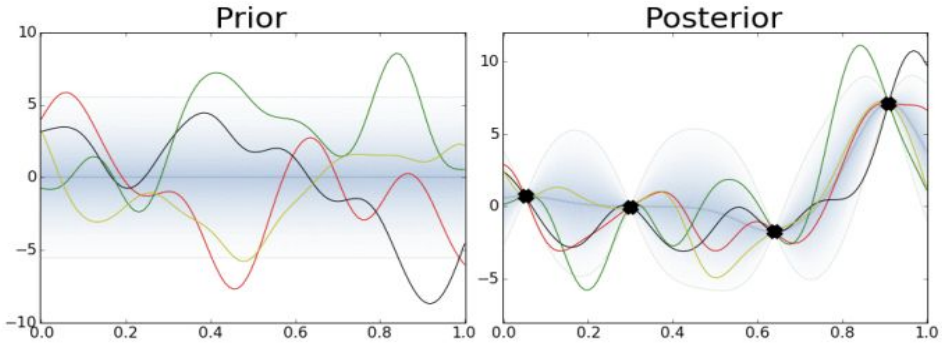
1. Choose some prior measure over the space of possible objectives f(x)



Bayesian Optimization

1. Choose some prior measure over the space of possible objectives f(x)

2. Combine prior and the likelihood to get a posterior measure over the objective given some observations

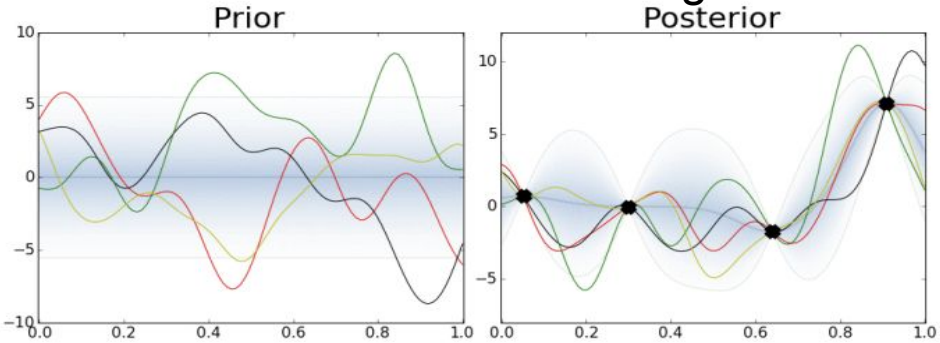


Bayesian Optimization

1. Choose some prior measure over the space of possible objectives f(x)

2. Combine prior and the likelihood to get a posterior measure over the objective given some observations

3. Use the posterior to decide where to take the next evaluation using acquisition function



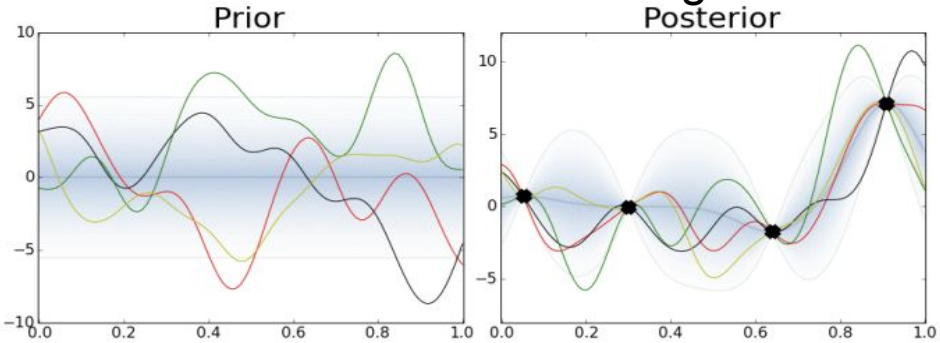
Bayesian Optimization

1. Choose some prior measure over the space of possible objectives f(x)

2. Combine prior and the likelihood to get a posterior measure over the objective given some observations

3. Use the posterior to decide where to take the next evaluation using acquisition function

4. Augment the data



Why Biology? Why Chemistry?

Why Biology? Why Chemistry?

DRUG
DISCOVERY

Why Biology? Why Chemistry?

DRUG DISCOVERY

Massive amount of data

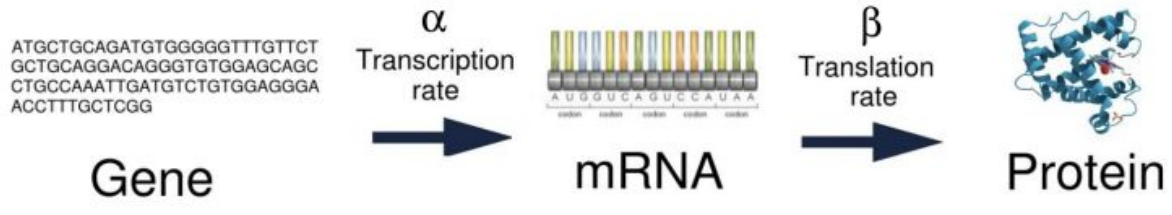
Why Biology? Why Chemistry?

A Mammalian cell in numbers

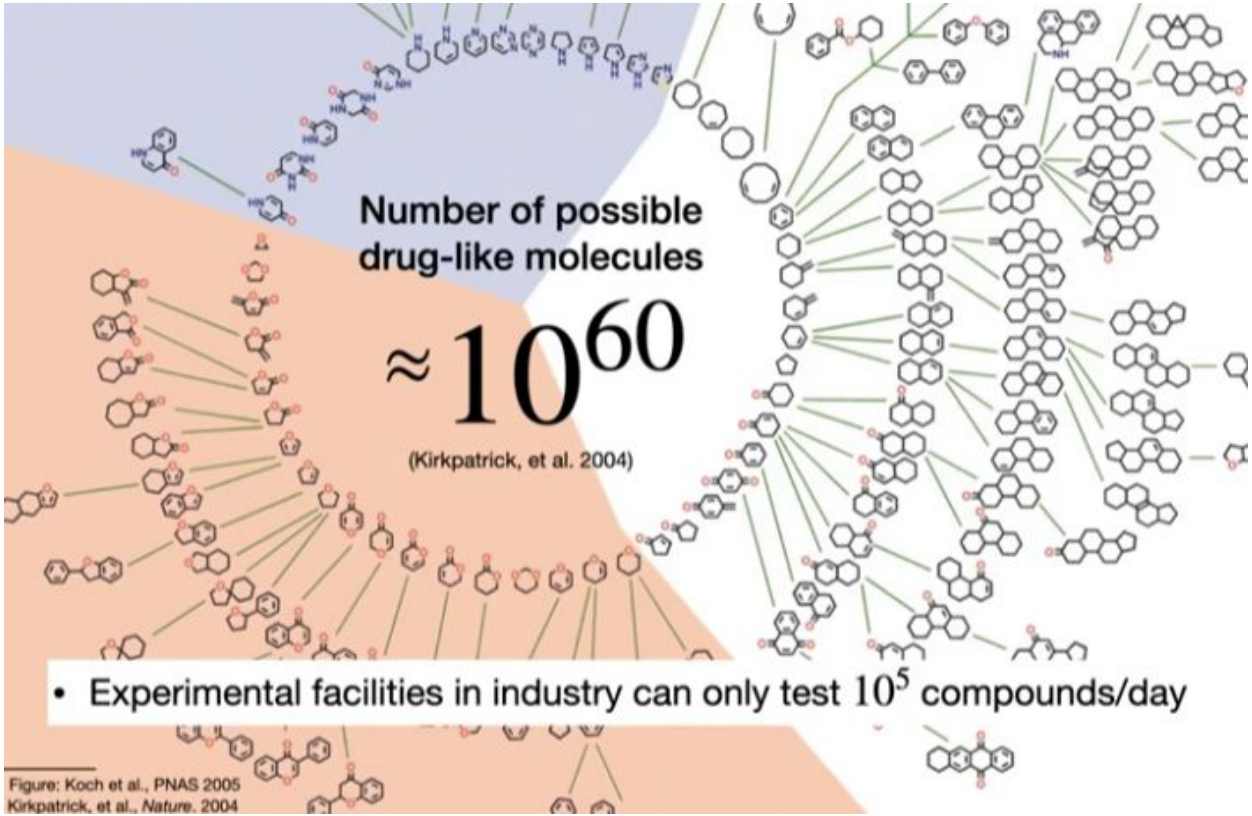
- Approximately 20,000 genes able to produce 20,000 proteins

- A few of them are of therapeutical interest

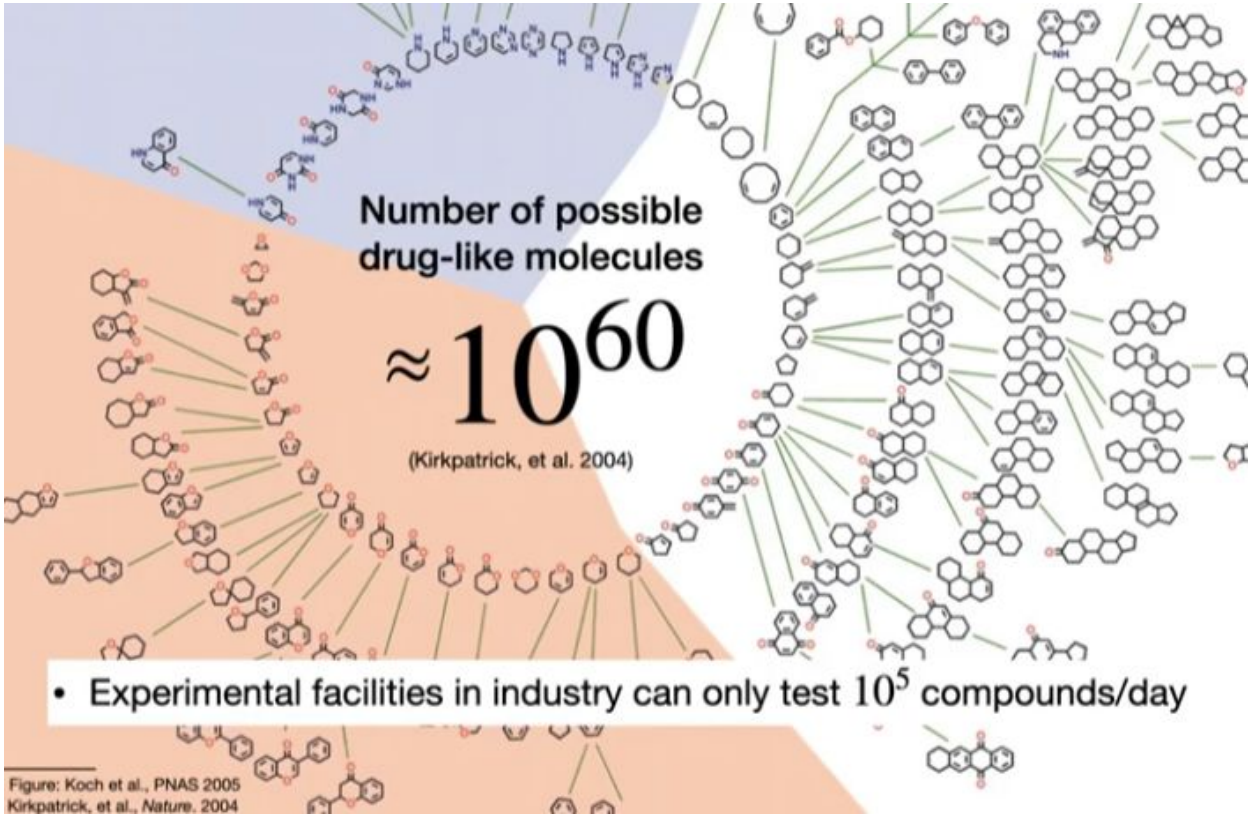
- The average gene length is 7092 base pairs (A, T, C, G)



Why Biology? Why Chemistry?



Why Biology? Why Chemistry?

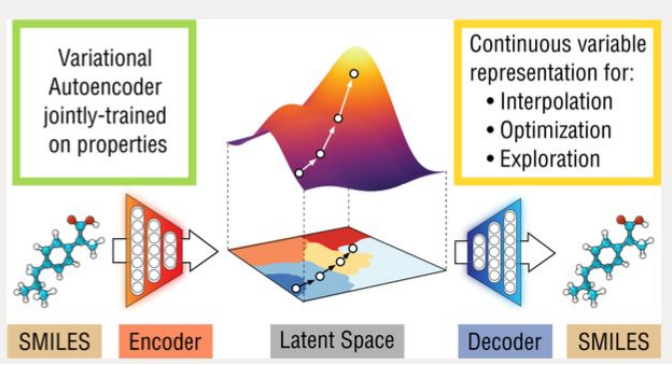


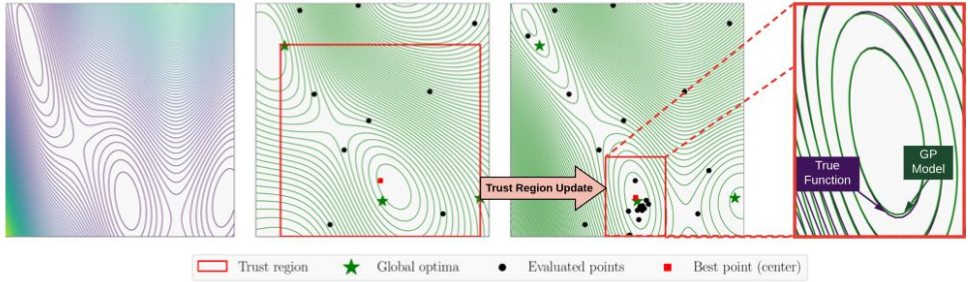
Faster

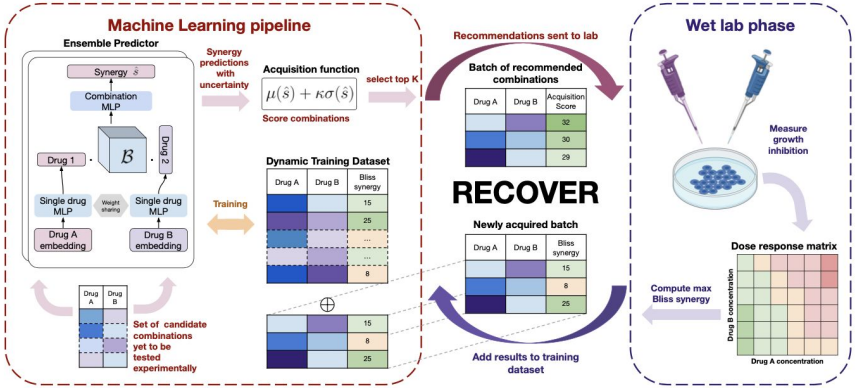
+

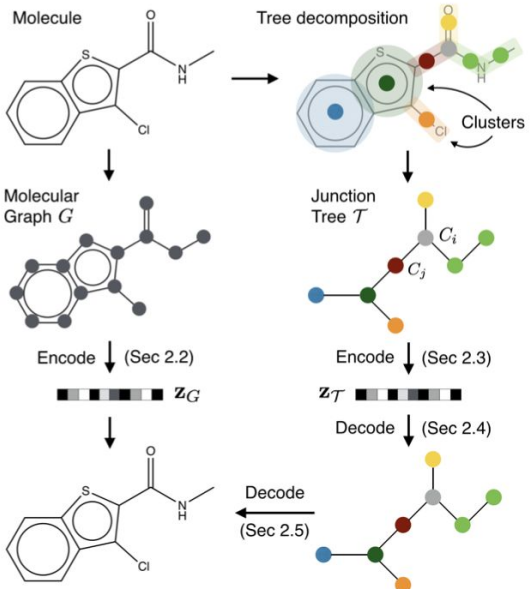
Accurate

Applications?



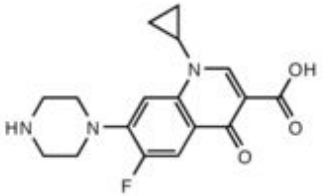




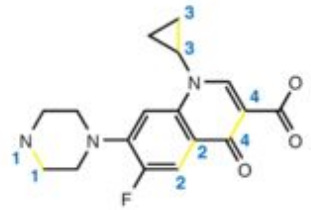


Why molecules?

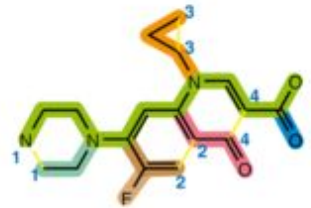
A



B



C

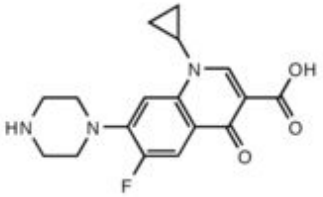


D

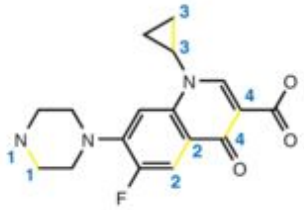
N1CCN(CC1)C(C(F)=C2)=CC(=C2C4=O)N(C3CC3)C=C4C(=O)O

Why molecules?

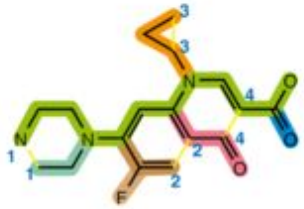
A



B



C



D

N1CCN(CC1)C(C(F)=C2)=CC(=C2C4=O)N(C3CC3)C=C4C(=O)O

SMILES strings

(Simplified molecular-input line-entry system)

Applications in Bayesian Optimization

- Variational AutoEncoder + BO
- Graph Neural Network + BO
- Local latent space + BO

Variational AutoEncoder + Bayesian Optimization

ACS central science

Research Article

Cite This: ACS Cent. Sci. 2018, 4, 268−276

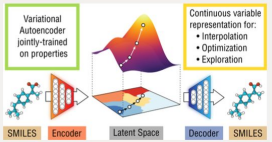
Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules

Rafael Gómez-Bombarelli,[†,#] Jennifer N. Wei,[‡,#] David Duvenaud,[¶,#] José Miguel Hernández-Lobato,[§,#] Benjamín Sánchez-Lengeling,[‡] Dennis Sheberla,[‡] Jorge Aguilera-Iparraguirre,[†] Timothy D. Hirzel,[†] Ryan P. Adams,[∇,∥] and Alán Aspuru-Guzik[*,‡,⊥]

[†]Kyulux North America Inc., 10 Post Office Square, Suite 800, Boston, Massachusetts 02109, United States
[‡]Department of Chemistry and Chemical Biology, Harvard University, Cambridge, Massachusetts 02138, United States
[¶]Department of Computer Science, University of Toronto, 6 King's College Road, Toronto, Ontario M5S 3H5, Canada
[§]Department of Engineering, University of Cambridge, Trumpington Street, Cambridge CB2 1PZ, U.K.
[∇]Google Brain, Mountain View, California, United States
[∥]Princeton University, Princeton, New Jersey, United States
[⊥]Biologically-Inspired Solar Energy Program, Canadian Institute for Advanced Research (CIFAR), Toronto, Ontario M5S 1M1, Canada

Supporting Information

ABSTRACT: We report a method to convert discrete representations of molecules to and from a multidimensional continuous representation. This model allows us to generate new molecules for efficient exploration and optimization through open-ended spaces of chemical compounds. A deep neural network was trained on hundreds of thousands of existing chemical structures to construct three coupled functions: an encoder, a decoder, and a predictor. The encoder converts the discrete representation of a molecule into a real-valued continuous vector, and the decoder converts these continuous vectors back to discrete molecular representations. The predictor estimates chemical properties from the latent continuous vector representation of the molecule. Continuous representations of molecules allow us to automatically generate novel chemical structures by performing simple operations in the latent space, such as decoding random vectors, perturbing known chemical structures, or interpolating between molecules. Continuous representations also allow the use of powerful gradient-based optimization to efficiently guide the search for optimized functional compounds. We demonstrate our method in the domain of drug-like molecules and also in a set of molecules with fewer that nine heavy atoms.



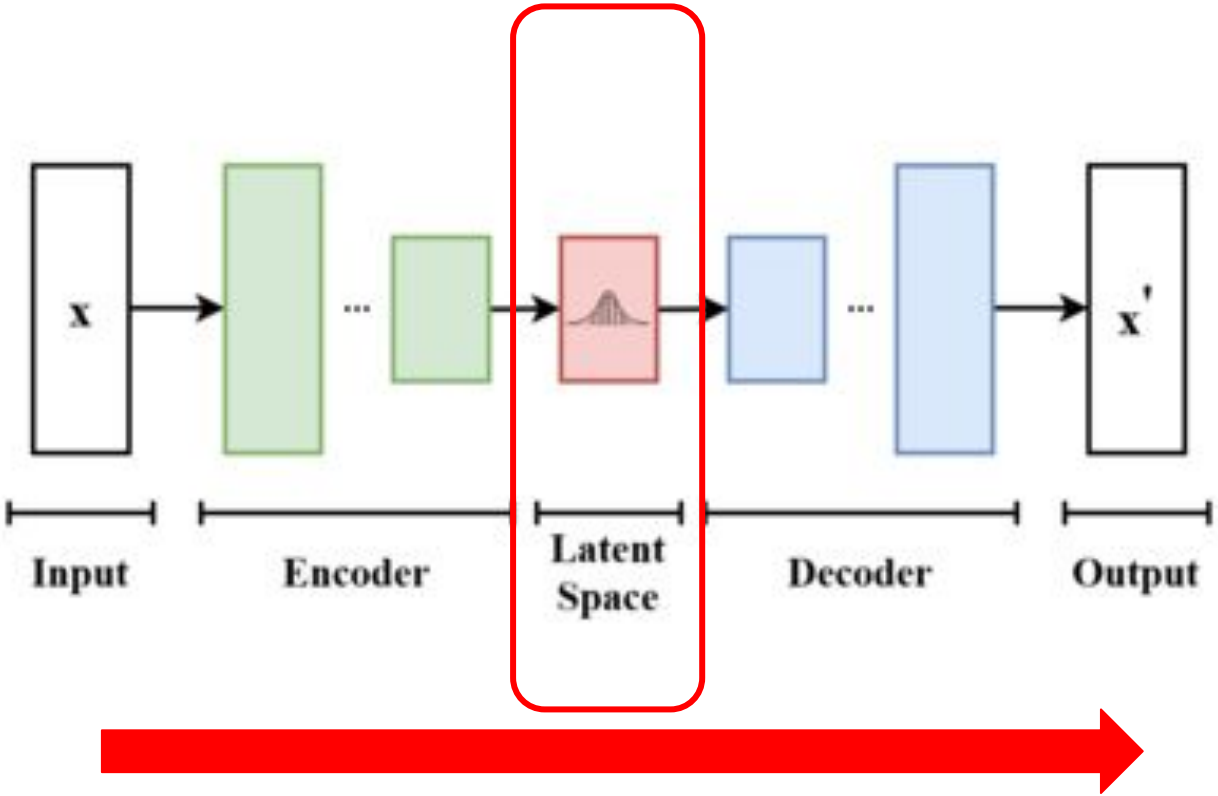
INTRODUCTION

The goal of drug and material design is to identify novel molecules that have certain desirable properties. We view this as an optimization problem, in which we are searching for the molecules that maximize our quantitative desiderata. However, optimization in molecular space is extremely challenging, because the search space is large, discrete, and unstructured. Making and testing new compounds are costly and time-consuming, and the number of potential candidates is overwhelming. Only about 10^8 substances have ever been synthesized,[1] whereas the range of potential drug-like molecules is estimated to be between 10^{23} and 10^{60}.[2] the most promising leads are selected and tested experimentally.

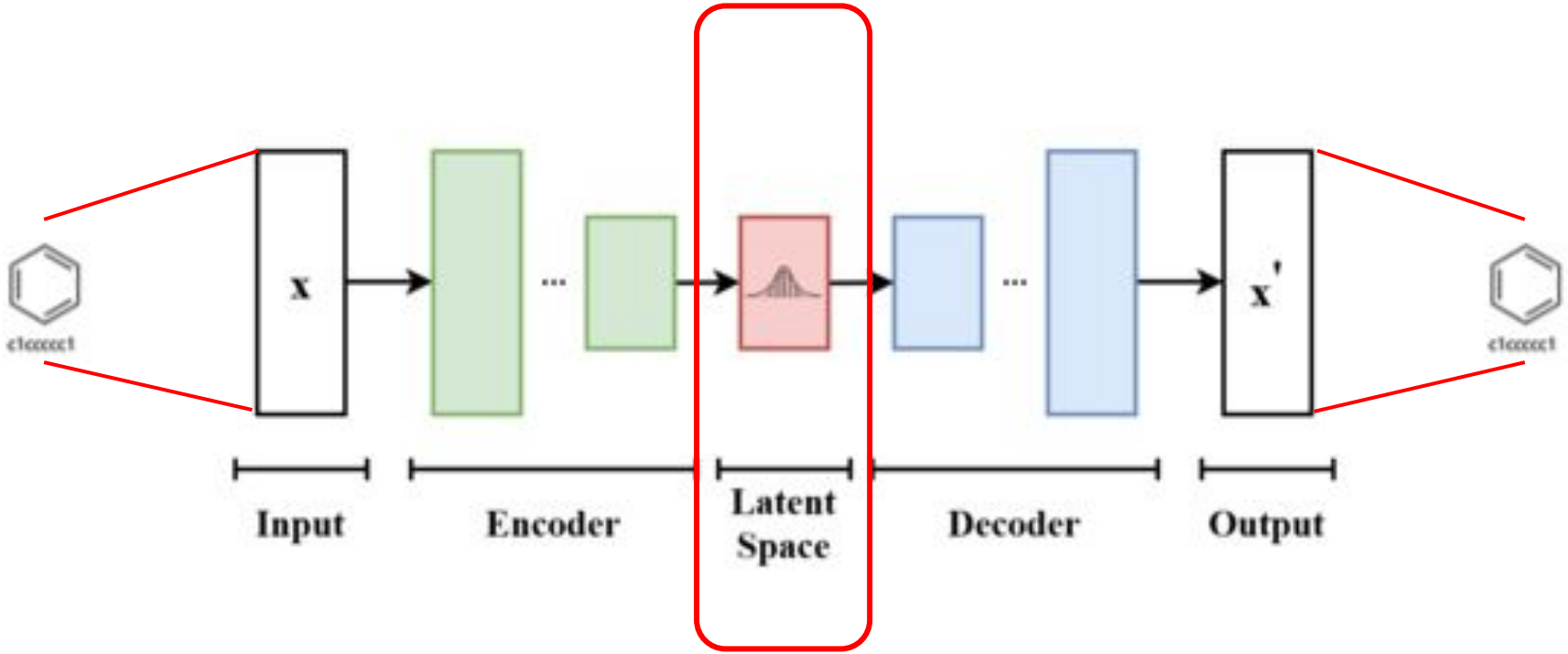
However, even when accurate simulations are available,[7] computational molecular design is limited by the search strategy used to explore chemical space. Current methods either exhaustively search through a fixed library,[8,9] or use discrete local search methods such as genetic algorithms[10−15] or similar discrete interpolation techniques.[16−18] Although these techniques have led to useful new molecules, these approaches still face large challenges. Fixed libraries are monolithic, costly to fully explore, and require hand-crafted rules to avoid impractical chemistries. The genetic generation of compounds requires manual specification of heuristics for mutation and

Gómez-Bombarelli et al., ACS central science 2018

Variational AutoEncoder + Bayesian Optimization

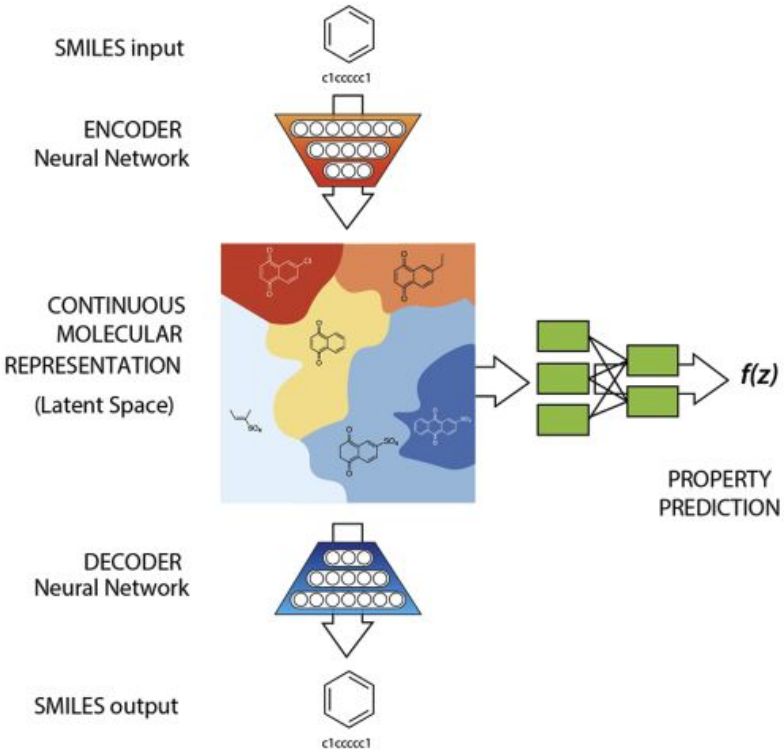


Variational AutoEncoder + Bayesian Optimization



Low dimension (relatively) + continuous representation

Variational AutoEncoder + Bayesian Optimization

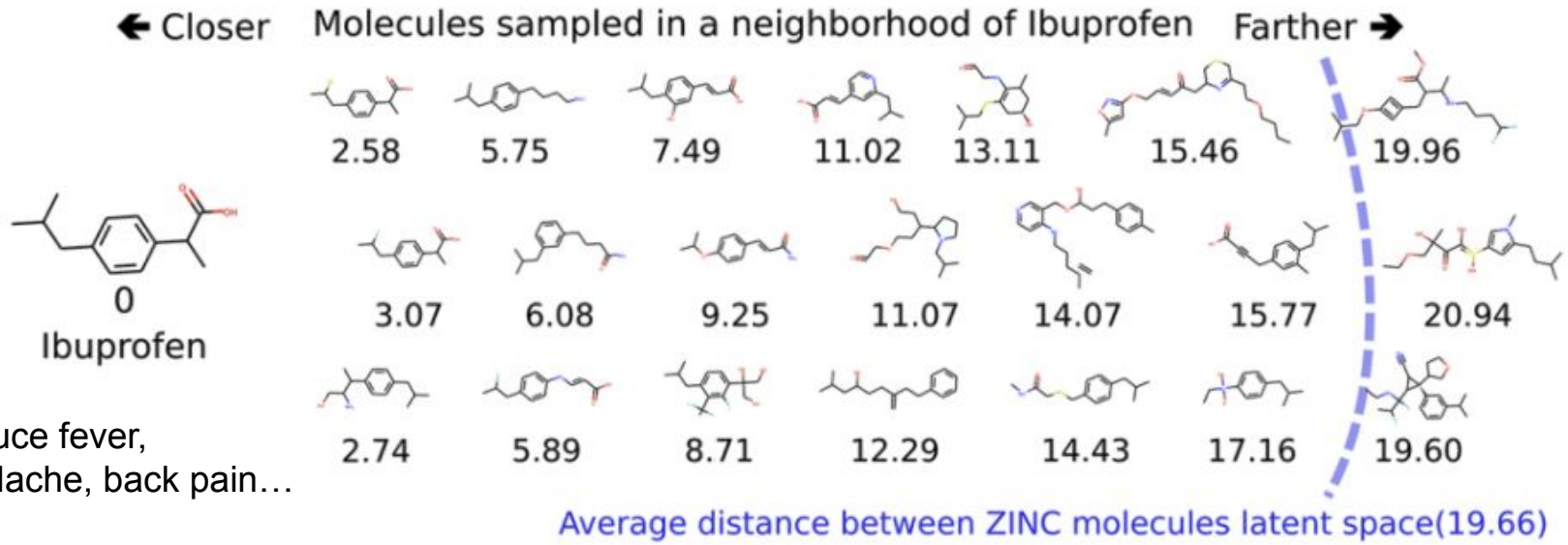


Compress molecules to a continuous latent representation

- Points that are close to each other in the latent space represent molecules that are similar chemically

Gómez-Bombarelli et al., ACS central science 2018

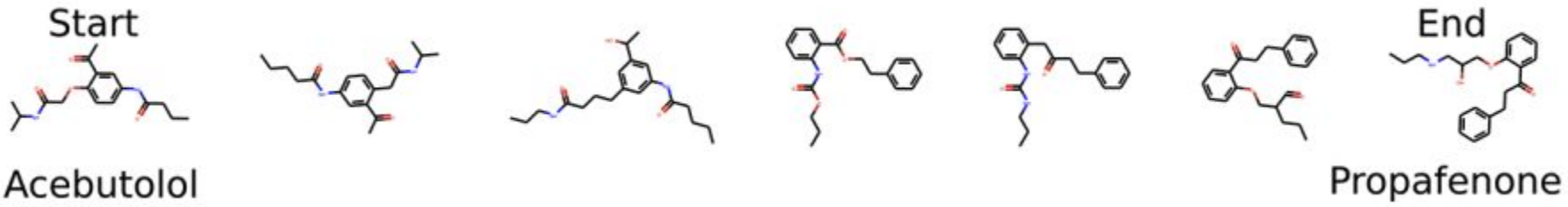
Variational AutoEncoder + Bayesian Optimization



Reduce fever, headache, back pain…

Gómez-Bombarelli et al., ACS central science 2018

Variational AutoEncoder + Bayesian Optimization

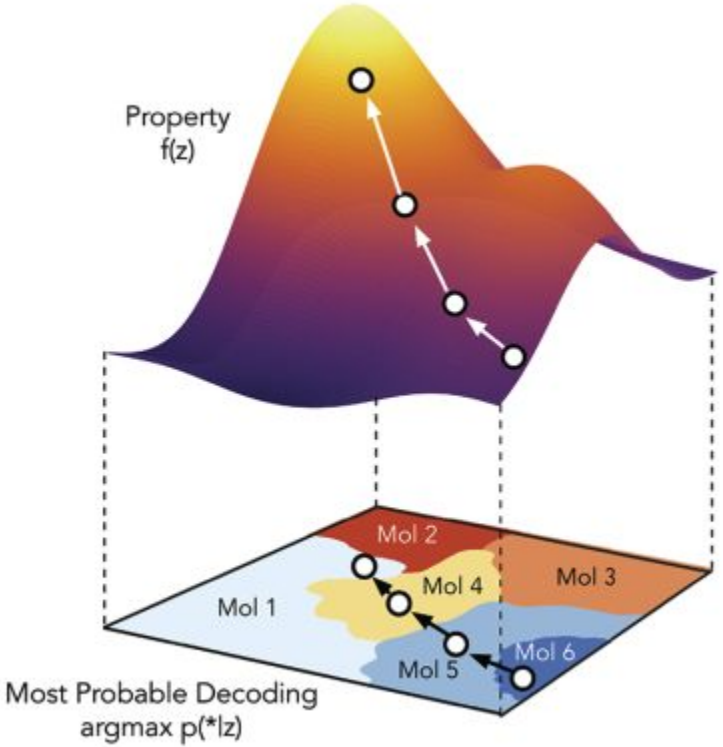


Used to treat high blood pressure

Used to treat irregular heartbeat

Gómez-Bombarelli et al., ACS central science 2018

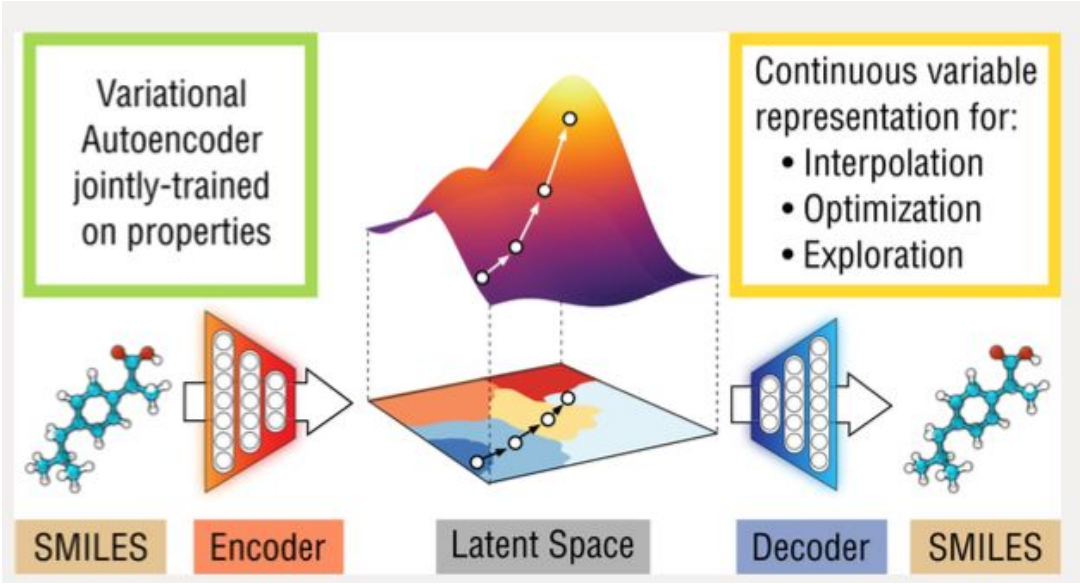
Variational AutoEncoder + Bayesian Optimization



Optimize the structure of molecules by using gradient-based optimization

Gómez-Bombarelli et al., ACS central science 2018

Variational AutoEncoder + Bayesian Optimization



- Easy to navigate in the latent space
- Design molecules with certain molecular properties

Gómez-Bombarelli et al., ACS central science 2018

Graph Neural Network + Bayesian Optimization

Junction Tree Variational Autoencoder for Molecular Graph Generation

Wengong Jin [1] **Regina Barzilay** [1] **Tommi Jaakkola** [1]

Abstract

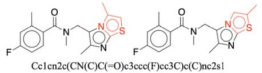
We seek to automate the design of molecules based on specific chemical properties. In computational terms, this task involves continuous embedding and generation of molecular graphs. Our primary contribution is the direct realization of molecular graphs, a task previously approached by generating linear SMILES strings instead of graphs. Our *junction tree variational autoencoder* generates molecular graphs in two phases, by first generating a tree-structured scaffold over chemical substructures, and then combining them into a molecule with a graph message passing network. This approach allows us to incrementally expand molecules while maintaining chemical validity at every step. We evaluate our model on multiple tasks ranging from molecular generation to optimization. Across these tasks, our model outperforms previous state-of-the-art baselines by a significant margin.

1. Introduction

The key challenge of drug discovery is to find target molecules with desired chemical properties. Currently, this task takes years of development and exploration by expert chemists and pharmacologists. Our ultimate goal is to automate this process. From a computational perspective, we decompose the challenge into two complementary subtasks: learning to represent molecules in a continuous manner that facilitates the prediction and optimization of their properties (encoding); and learning to map an optimized continuous representation back into a molecular graph with improved properties (decoding). While deep learning has been extensively investigated for molecular graph encoding (Duvenaud et al., 2015; Kearnes et al., 2016; Gilmer et al., 2017), the harder combinatorial task of molecular graph generation from latent representation remains under-explored.

[1]MIT Computer Science & Artificial Intelligence Lab. Correspondence to: Wengong Jin <wengong@csail.mit.edu>.

Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.



Cc1cn2c(CN(C)C(=O)c3ccc(F)cc3C)c(C)nc2s1
Cc1cc(F)ccc1C(=O)N(C)Cc1c(C)nc2scc(C)n12

Figure 1. Two almost identical molecules with markedly different canonical SMILES in RDKit. The edit distance between two strings is 22 (50.5% of the whole sequence).

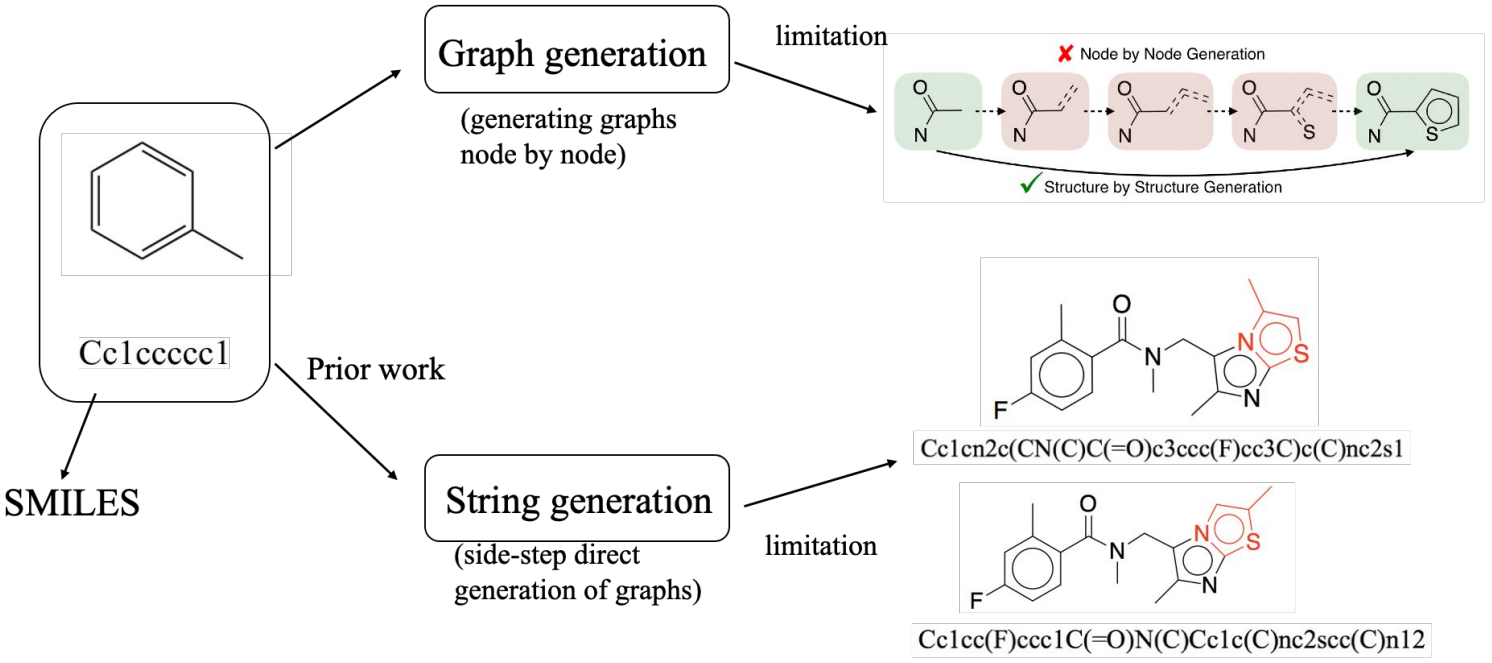
Prior work on drug design formulated the graph generation task as a string generation problem (Gómez-Bombarelli et al., 2016; Kusner et al., 2017) in an attempt to side-step direct generation of graphs. Specifically, these models start by generating SMILES (Weininger, 1988), a linear string notation used in chemistry to describe molecular structures. SMILES strings can be translated into graphs via deterministic mappings (e.g., using RDKit (Landrum, 2006)). However, this design has two critical limitations. First, the SMILES representation is not designed to capture molecular similarity. For instance, two molecules with similar chemical structures may be encoded into markedly different SMILES strings (e.g., Figure 1). This prevents generative models like variational autoencoders from learning smooth molecular embeddings. Second, essential chemical properties such as molecule validity are easier to express on graphs rather than linear SMILES representations. We hypothesize that operating directly on graphs improves generative modeling of valid chemical structures.

Our primary contribution is a new generative model of molecular graphs. While one could imagine solving the problem in a standard manner – generating graphs node by node (Li et al., 2018) – the approach is not ideal for molecules. This is because creating molecules atom by atom would force the model to generate chemically invalid intermediaries (see, e.g., Figure 2), delaying validation until a complete graph is generated. Instead, we propose to generate molecular graphs in two phases by exploiting valid subgraphs as components. The overall generative approach, cast as a *junction tree variational autoencoder*[1], first generates a tree structured object (a junction tree) whose role is to represent the scaffold of subgraph components and their coarse relative arrangements. The components are

[1]https://github.com/wengong-jin/icml18-jtnn

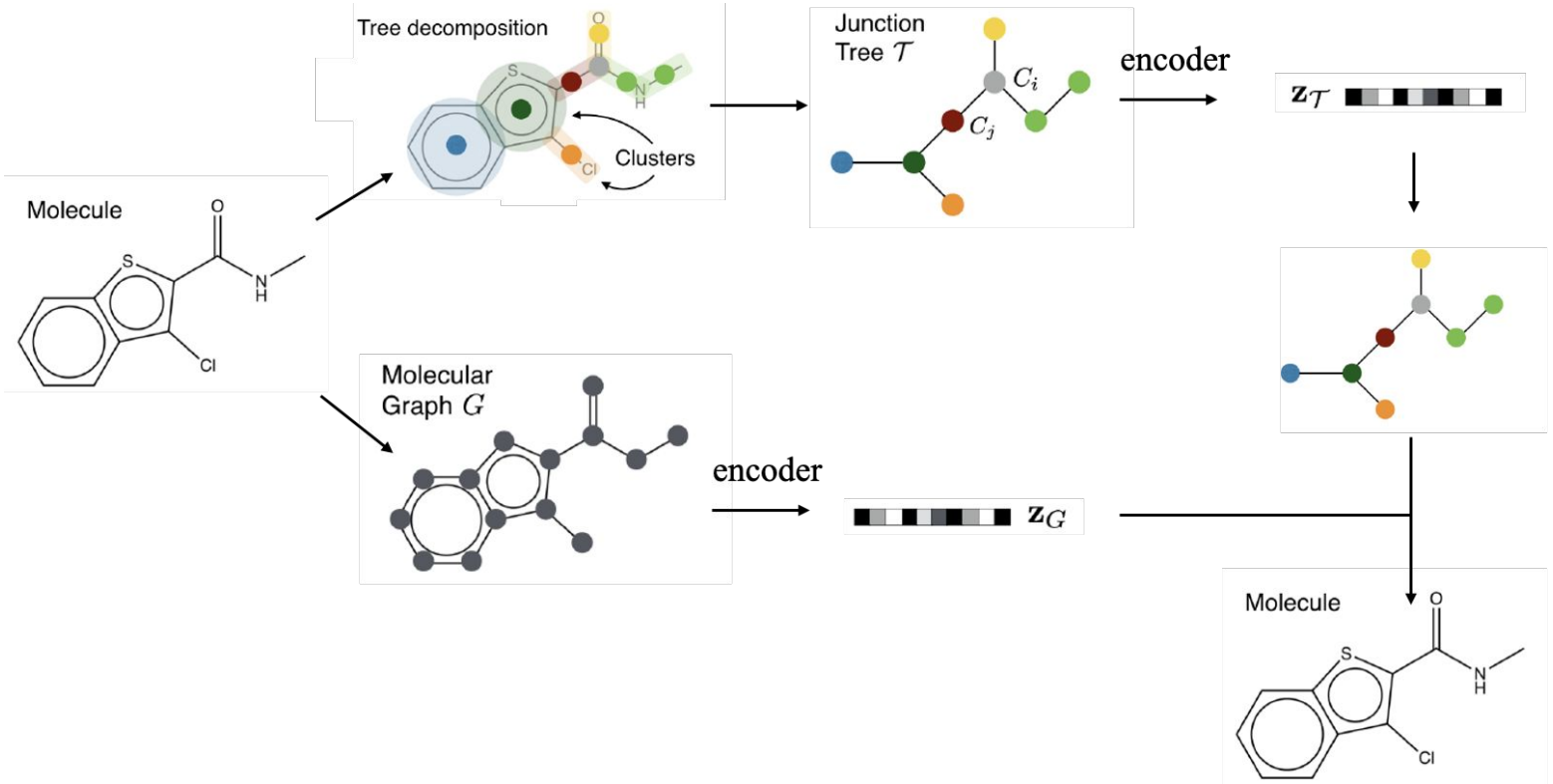
Jin et al., ICML 2018

Graph Neural Network + Bayesian Optimization



Jin et al., ICML 2018

Graph Neural Network + Bayesian Optimization



Jin et al., ICML 2018

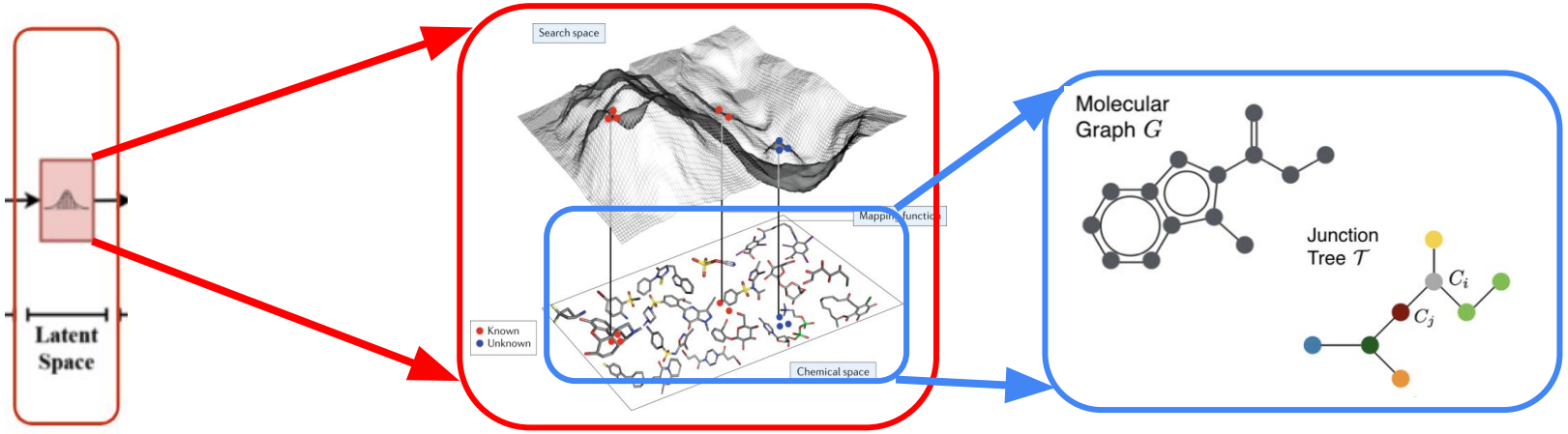
Graph Neural Network + Bayesian Optimization

Method	Reconstruction	Validity
CVAE	44.6%	0.7%
GVAE	53.7%	7.2%
SD-VAE	76.2%	43.5%
GraphVAE	-	13.5%
Atom-by-Atom LSTM	-	89.2%
JT-VAE	**76.7%**	**100.0%**

RDKit

Jin et al., ICML 2018

Graph Neural Network + Bayesian Optimization



Variational Autoencoder

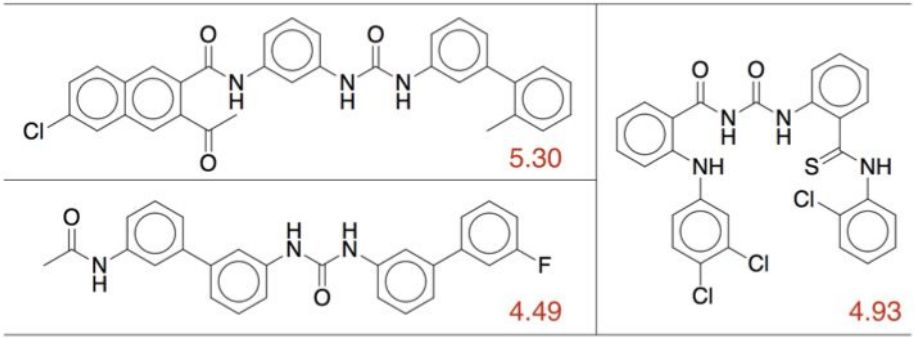
Bayesian Optimization

Molecular Graph (G)
Junction Tree (T)

Jin et al., ICML 2018

Graph Neural Network + Bayesian Optimization

Method	1st	2nd	3rd
CVAE	1.98	1.42	1.19
GVAE	2.94	2.89	2.80
SD-VAE	4.04	3.50	2.96
JT-VAE	**5.30**	**4.93**	**4.49**



Penalized logP

(Penalized water-octanol partition coefficient)

Jin et al., ICML 2018

Local latent space + Bayesian Optimization

Local Latent Space Bayesian Optimization over Structured Inputs

Natalie Maus [1] **Haydn T. Jones** [2] **Juston S. Moore** [2] **Matt J. Kusner** [3] **John Bradshaw** [4] **Jacob R. Gardner** [1]

Abstract

Bayesian optimization over the latent spaces of deep autoencoder models (DAEs) has recently emerged as a promising new approach for optimizing challenging black-box functions over structured, discrete, hard-to-enumerate search spaces (e.g., molecules). Here the DAE dramatically simplifies the search space by mapping inputs into a continuous latent space where familiar Bayesian optimization tools can be more readily applied. Despite this simplification, the latent space typically remains high-dimensional. Thus, even with a well-suited latent space, these approaches do not necessarily provide a complete solution, but may rather shift the structured optimization problem to a high-dimensional one. In this paper, we propose LOL-BO, which adapts the notion of trust regions explored in recent work on high-dimensional Bayesian optimization to the structured setting. By reformulating the encoder to function as both an encoder for the DAE globally and as a deep kernel for the surrogate model within a trust region, we better align the notion of local optimization in the latent space with local optimization in the input space. LOL-BO achieves as much as 20 times improvement over state-of-the-art latent space Bayesian optimization methods across six real-world benchmarks, demonstrating that improvement in optimization strategies is as important as developing better DAE models.

1. Introduction

Many challenges across the physical sciences and engineering require the optimization of *structured objects*. For example, in the drug design setting, it is common to seek a molecule that maximizes some property, e.g., binding affinity for a target protein (Huang et al., 2021), or similarity to a known drug molecule under some metric (Brown et al., 2019). These structured optimization problems are particularly challenging, as the objective function is now typically over a large combinatorial space of discrete objects.

These challenging optimization problems have motivated a large amount of very recent interest in methods that perform *Bayesian optimization (BO) over latent spaces*. Using drug design as a running example, a deep auto-encoder (DAE) is first trained on a large set of unsupervised molecules. Bayesian optimization is then run as normal in the continuous latent space of the autoencoder to produce candidate latent codes. These candidate latent codes are then decoded back to molecules, and the objective function is evaluated. Bayesian optimization then proceeds as normal, with the surrogate model trained on a dataset of latent codes and their resulting objective values after decoding.

Rapid progress has been made in this emerging topic. In large part, this has been to improve the DAE model. The early work of Kusner et al. (2017) and Jin et al. (2018) investigated domain specific autoencoder architectures that much more reliably produced valid molecules. The latest work of Tripp et al. (2020) and Grosnit et al. (2021) investigate methods to inject supervised signal from objective function evaluations into the latent space, enabling better surrogate modeling.[1] However, the *optimization* component of latent space Bayesian optimization remains under-explored: indeed, it is common practice to directly apply standard Bayesian optimization with expected improvement once a latent space has been constructed (all of the above cited methods do this). We conjecture that, despite the ability of a deep encoder to dramatically simplify a structured input space, optimization remains challenging due to the high-dimensionality of the extracted latent space.

In this paper, we propose to jointly co-design the latent space model and a local Bayesian optimization strategy to produce a method for structured optimization. Concretely, we make the following contributions:

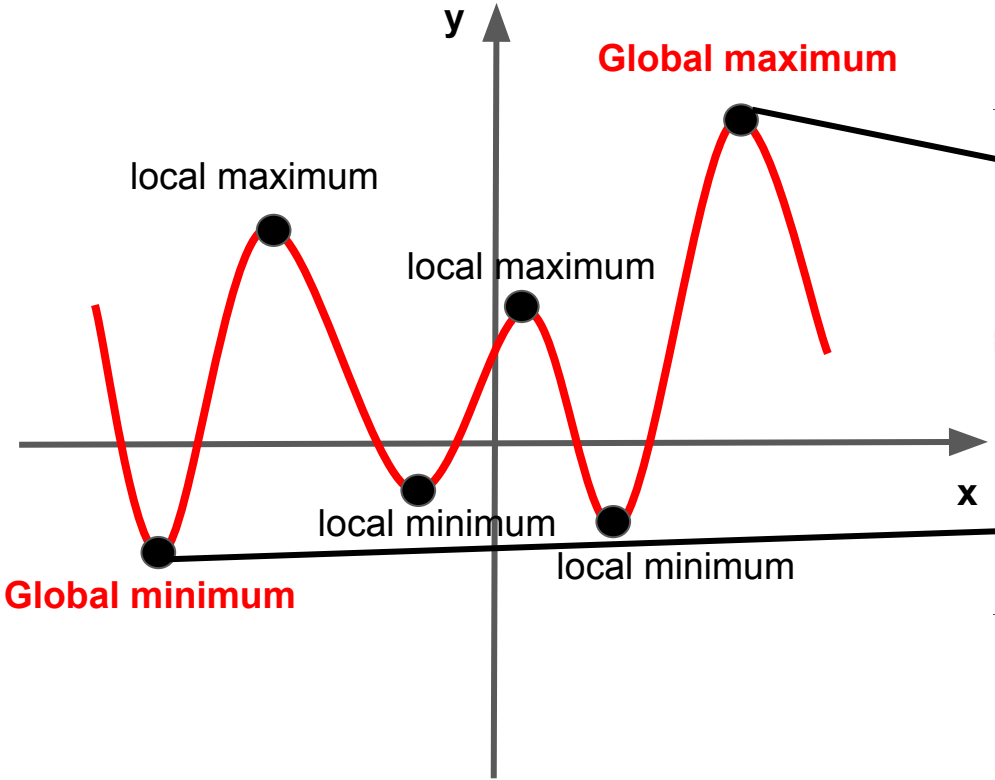
1. We develop local latent Bayesian optimization, LOL-BO, for the structured optimization setting, addressing the inherent mismatch between the notion of

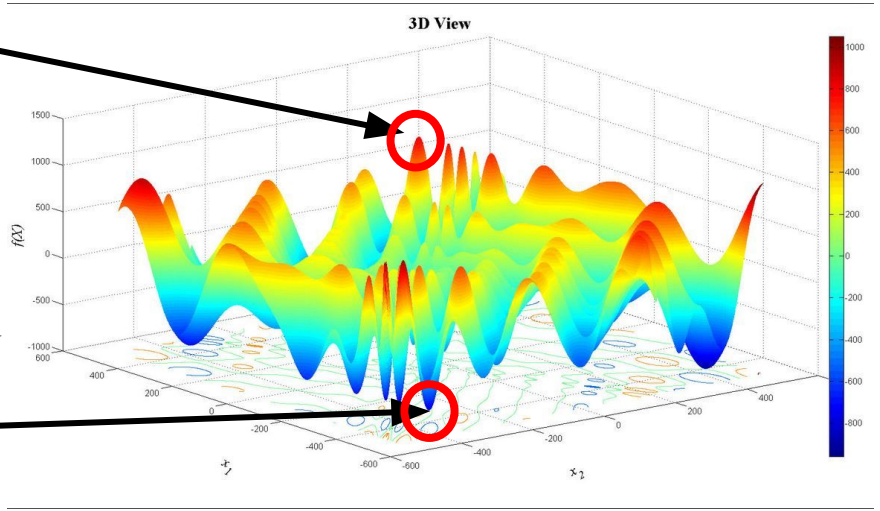
[1]University of Pennsylvania [2]Los Alamos National Laboratory [3]University College London [4]Massachusetts Institute of Technology. Correspondence to: Natalie Maus <nmaus@seas.upenn.edu>.

[1]For the interested reader see these recent surveys for more details (Öztürk et al., 2020; Coley et al., 2020).

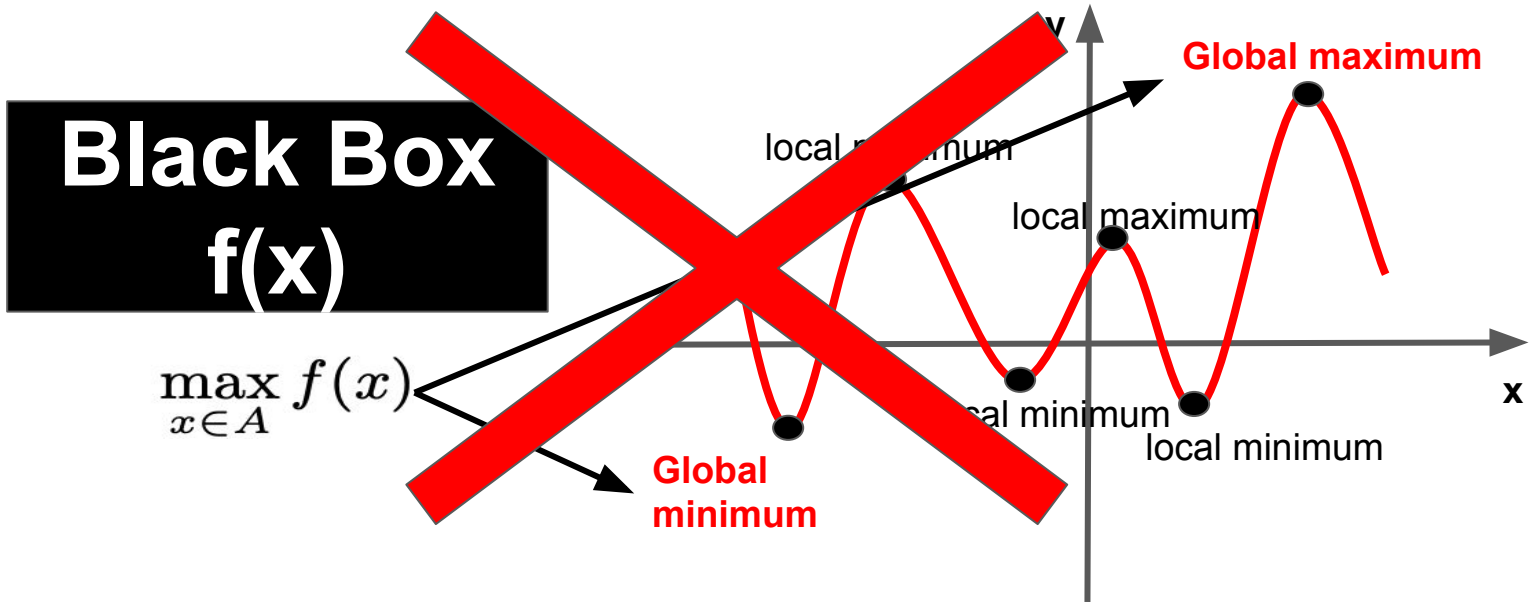
Maus et al., ArXiv 2022

Local latent space + Bayesian Optimization



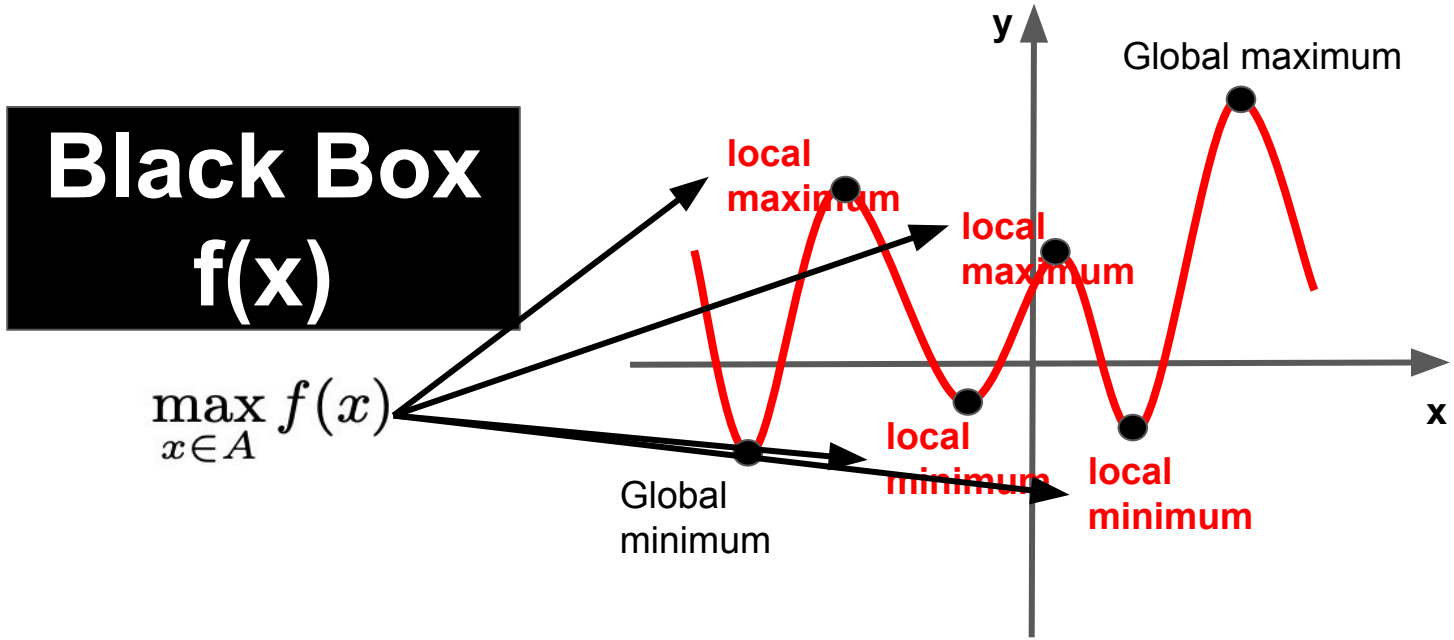


Local latent space + Bayesian Optimization



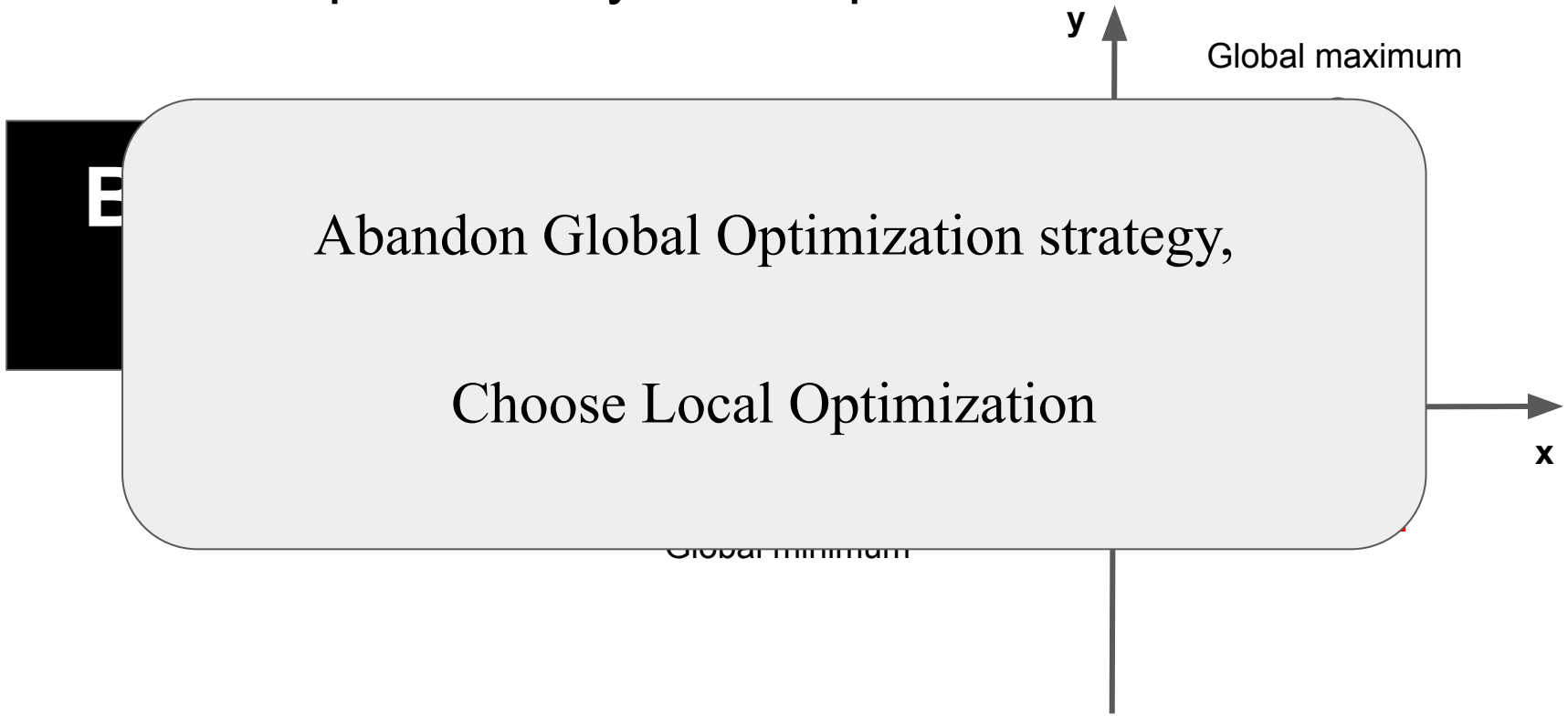
Maus et al., ArXiv 2022

Local latent space + Bayesian Optimization

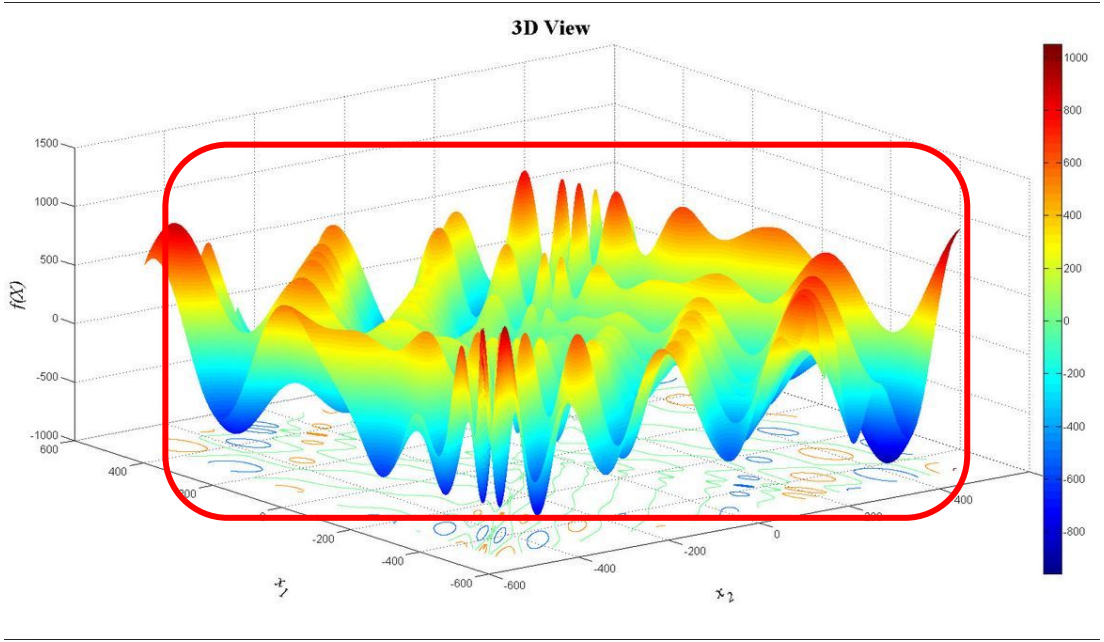


Maus et al., ArXiv 2022

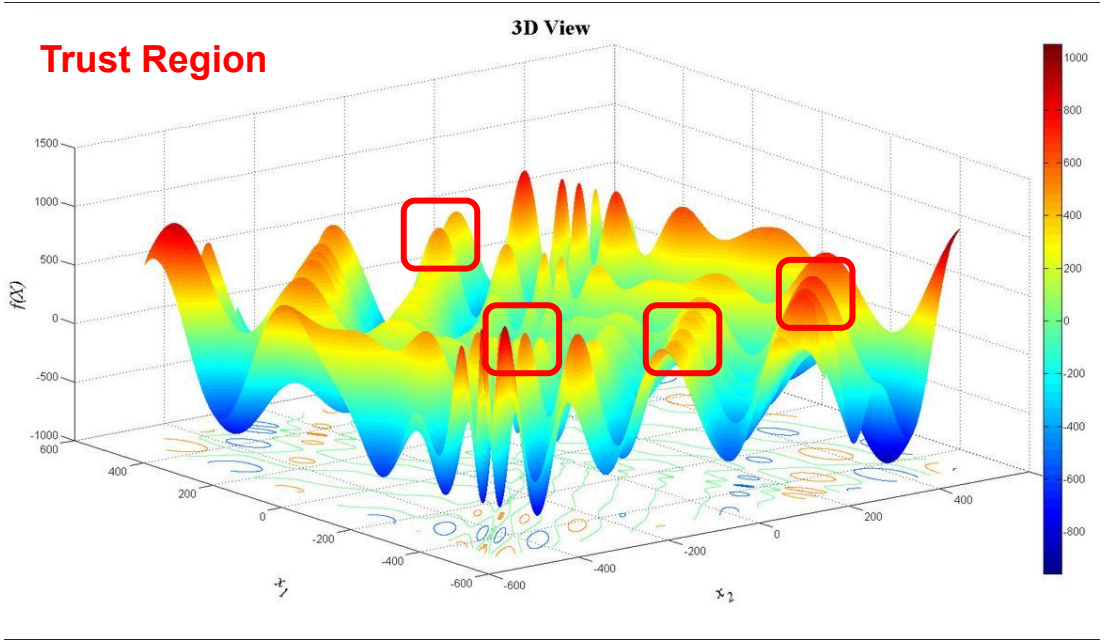
Local latent space + Bayesian Optimization



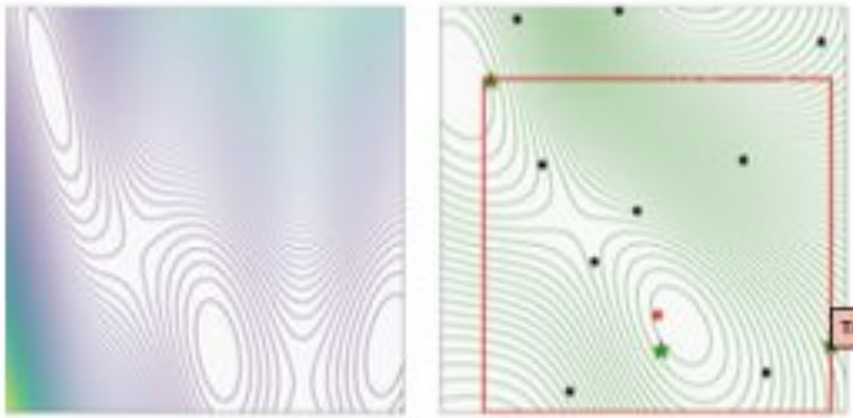
Local latent space + Bayesian Optimization



Local latent space + Bayesian Optimization



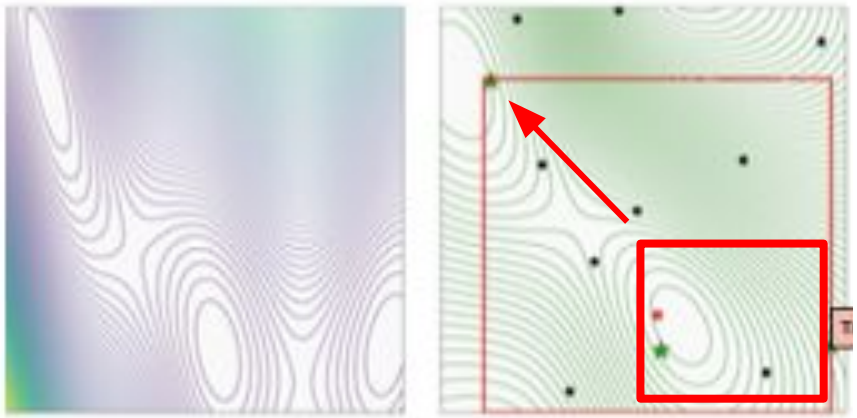
Local latent space + Bayesian Optimization



Choose the TR to be a rectangle centered at the best solution

Eriksson et al., NeurIPS 2019

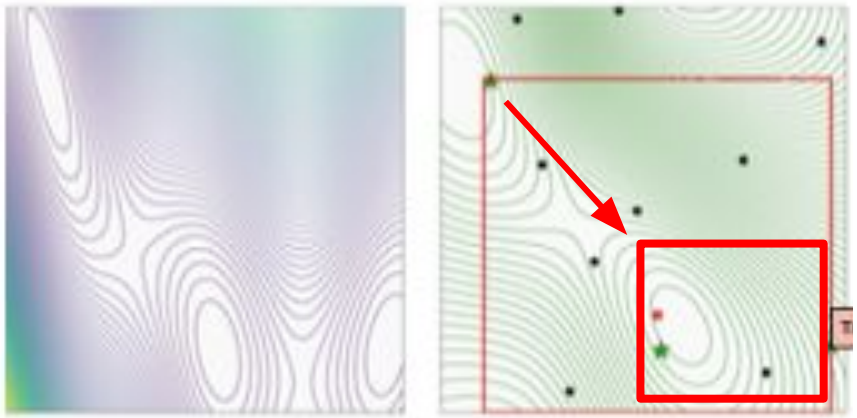
Local latent space + Bayesian Optimization



Explore by doubling the size of TR

Eriksson et al., NeurIPS 2019

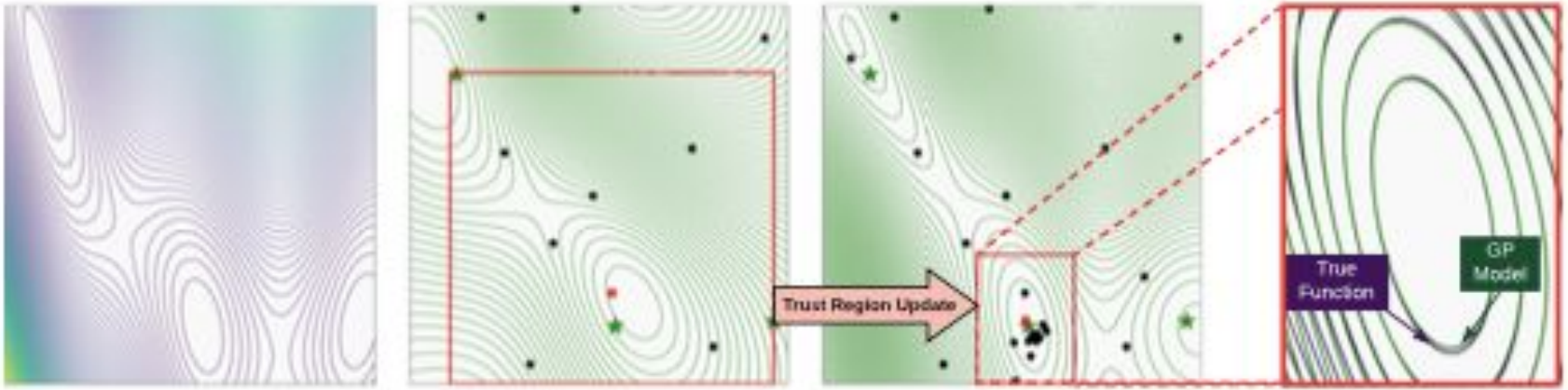
Local latent space + Bayesian Optimization



Exploit by halving the size of TR

Eriksson et al., NeurIPS 2019

Local latent space + Bayesian Optimization

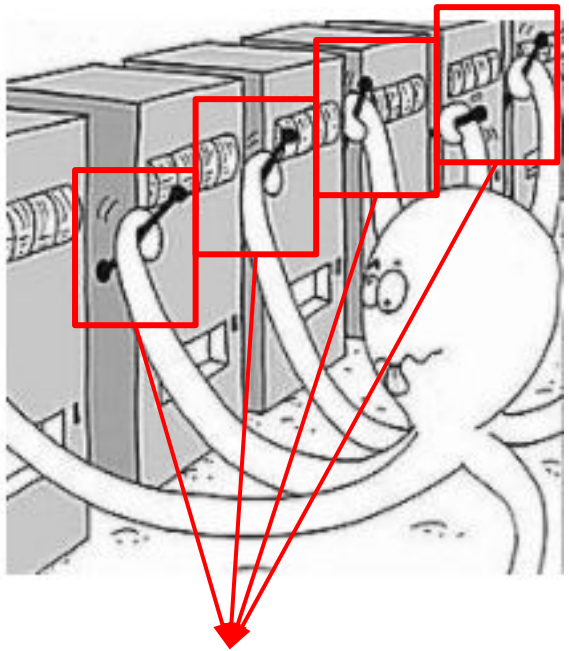


TuRBO (Trust region BO)

Eriksson et al., NeurIPS 2019

Local latent space + Bayesian Optimization

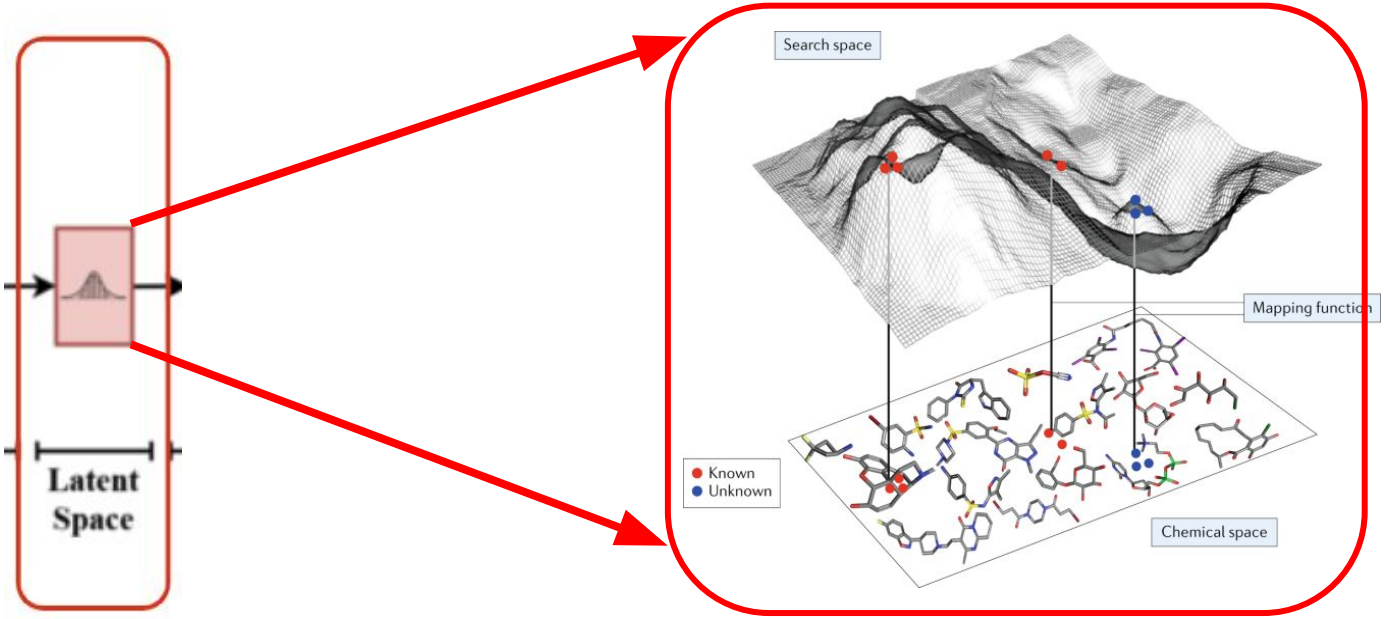
1. TuRBO maintains *m* trust regions simultaneously
2. Select a batch of *q* candidates drawn from the union of all trust regions
3. Update all local optimization problems
4. Allocate samples via a multi-armed bandit approach (Thompson sampling)



Trust Region (TR)

Eriksson et al., NeurIPS 2019

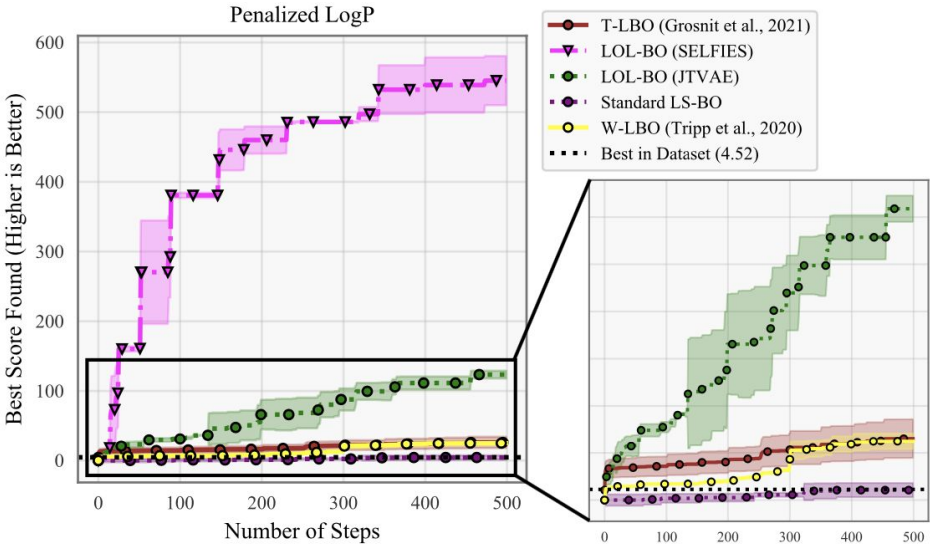
Local latent space + Bayesian Optimization



Local Latent Bayesian Optimization

Maus et al., ArXiv 2022

Local latent space + Bayesian Optimization



	SELFIES VAE	JTVAE
RECONSTRUCTION ACC.	**0.913**	0.767
AVG. DECODE TIME (S)	**0.02**	3.14
DECODE STDDEV	**0.04**	0.1
VALIDITY	1.0	1.0
GP TEST NLL (PERINDOPRIL)	**-2.156**	-1.481
GP TEST NLL(ZALEPLON)	**-2.705**	-1.781
GP TEST NLL (RANOLAZINE)	**-2.127**	-1.649

Maus et al., ArXiv 2022

Why doesn’t everyone use this then?

- Search space exponentially increases as the dimension increases
  - Surrogate model (GP…)
  - Acquisition function (Thompson sampling)

Takeaways

- Optimize expensive-to-evaluate functions.
- Human out of loop
- Reduce drastically the number of needed experiments (e.g. drug discovery)
- Save money