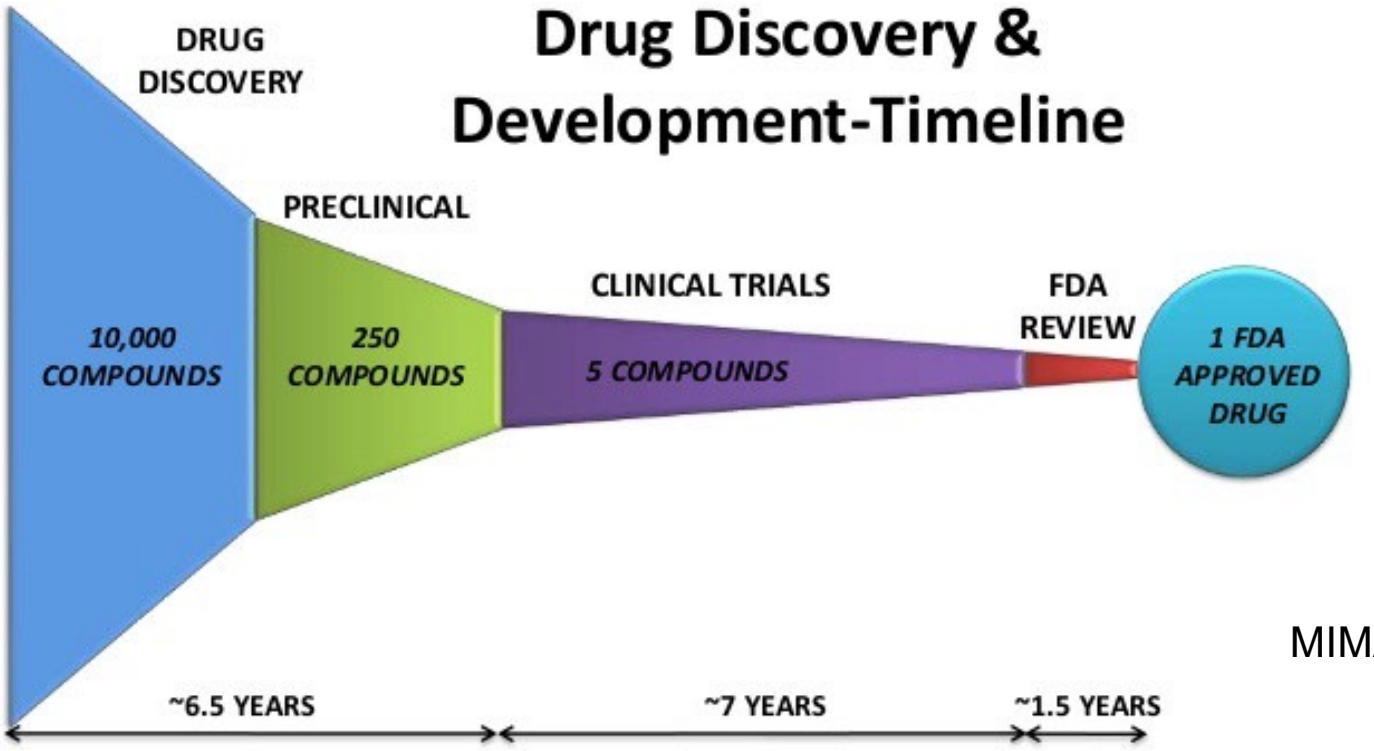
Advances in Bayesian Optimization

WASHINGTON STATE UNIVERSITY

Penn
UNIVERSITY of PENNSYLVANIA

DeepMind

Drug/Vaccine Design

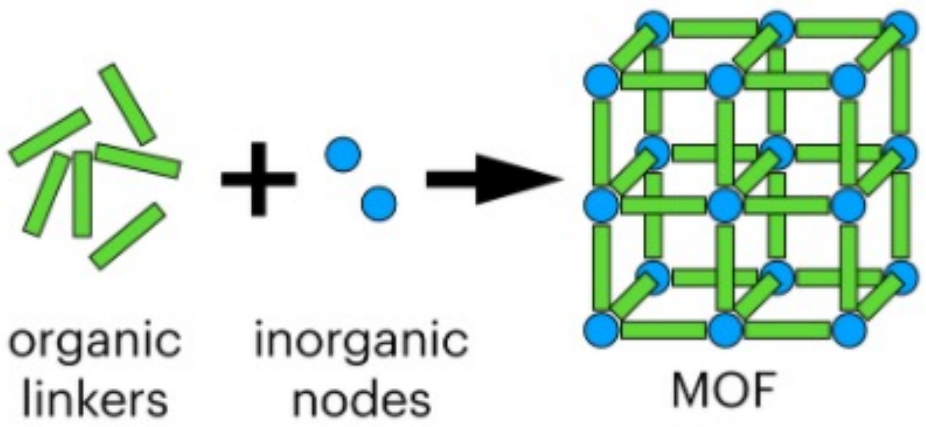


Credit:

MIMA healthcare

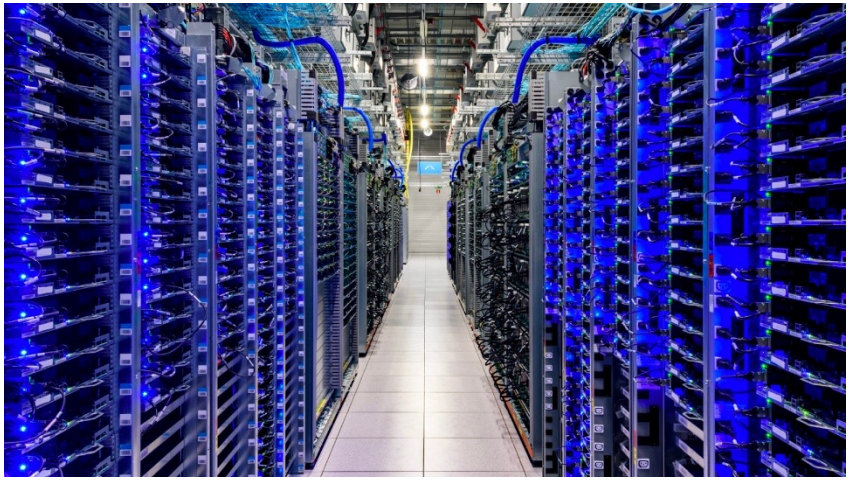
- Accelerate the discovery of promising designs

Nanoporous Materials Design



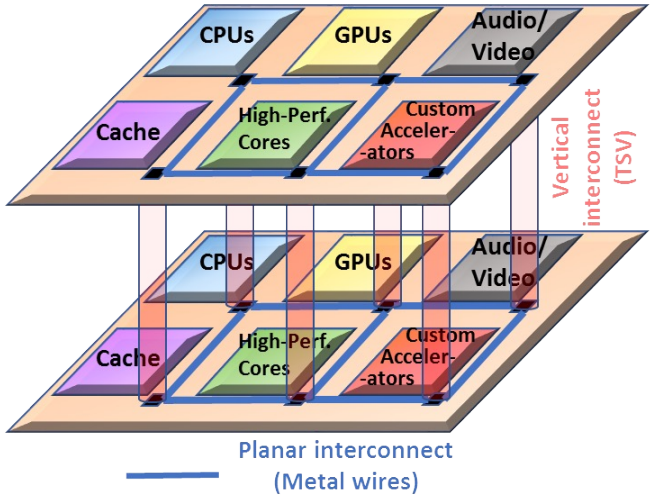
- **Sustainability applications**
  - Storing gases (e.g., hydrogen powered cars)
  - Separating gases (e.g., carbon dioxide from flue gas of coalfired power plants)
  - Detecting gases (e.g., detecting pollutants in outdoor air)

Sustainable Hardware Design for Data Centers



America's Data Centers Are Wasting Huge Amounts of Energy

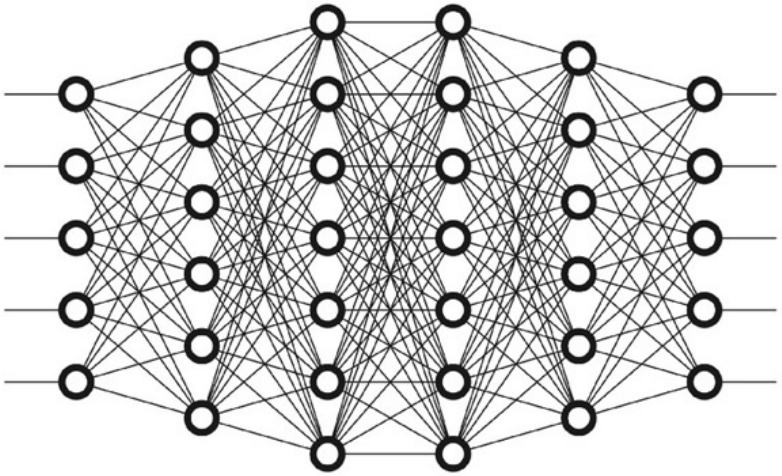
By 2020, data centers are projected to consume roughly 140 billion kilowatt-hours annually, costing American businesses $13 billion annually in electricity bills and emitting nearly 150 million metric tons of carbon pollution



High-performance and Energy-efficient manycore chips

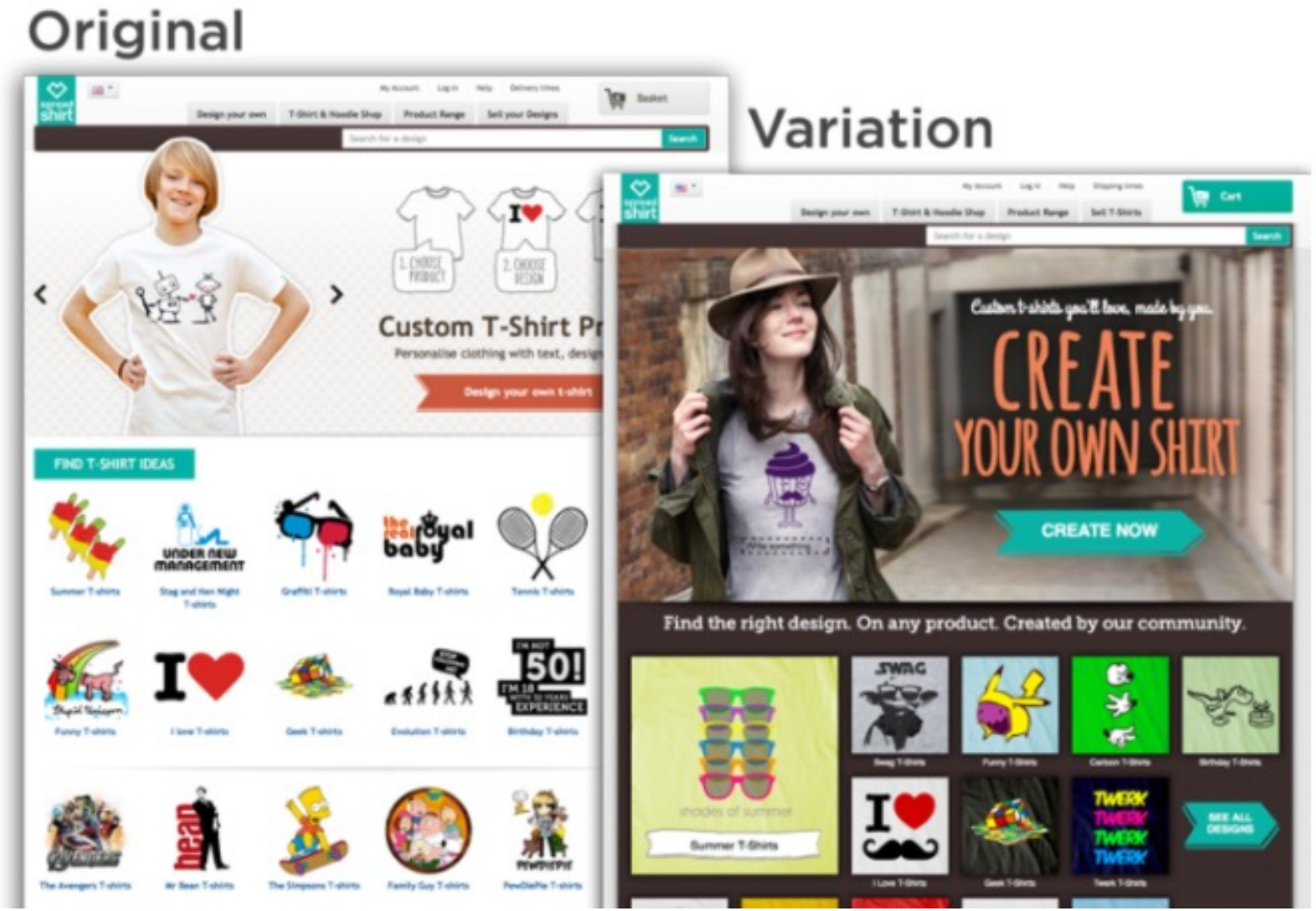
Report from Natural Resources Defense Council:.
https://www.nrdc.org/sites/default/files/data-center-efficiency-assessment-IB.pdf

Auto ML and Hyperparameter Tuning



- Accuracy of models critically depends on hyper-parameters
  - Optimization algorithm, learning rates, momentum, batch normalization, batch sizes, dropout rates, weight decay, data augmentation, …

A/B Testing to Configure Websites



Making Delicious Cookies

Bayesian Optimization for a Better Dessert

Greg Kochanski, Daniel Golovin, John Karro, Benjamin Solnik,
Subhodeep Moitra, and D. Sculley
{gpk, dgg, karro, bsolnik, smoitra, dsculley}@google.com; Google Brain Team

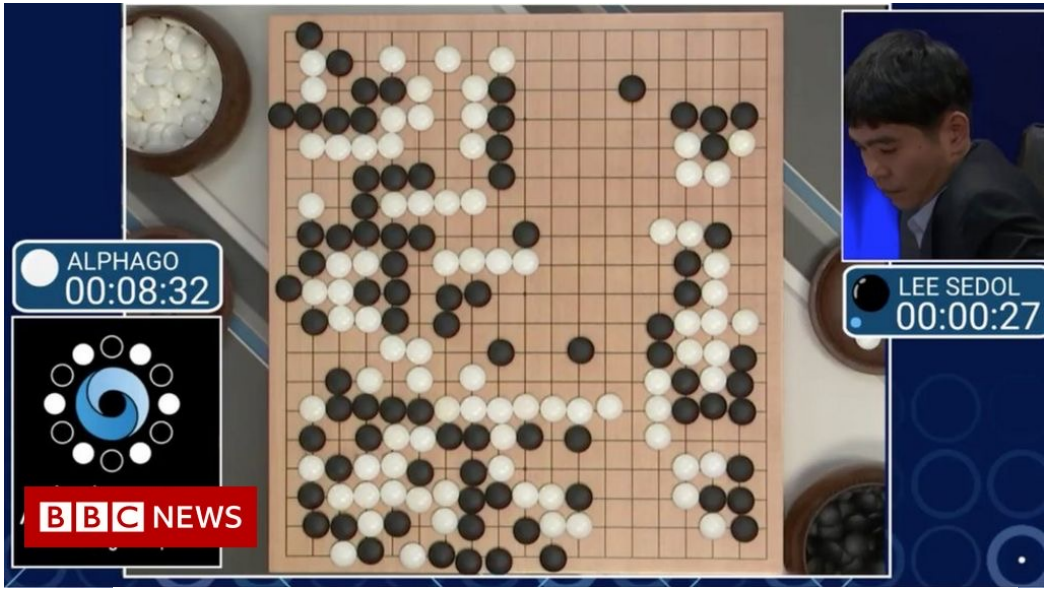
Abstract

We present a case study on applying Bayesian Optimization to a complex real-world system; our challenge was to optimize chocolate chip cookies. The process was a mixed-initiative system where both human chefs, human raters, and a machine optimizer participated in 144 experiments. This process resulted in highly rated cookies that deviated from expectations in some surprising ways – much less sugar in California, and cayenne in Pittsburgh. Our experience highlights the importance of incorporating domain expertise and the value of transfer learning approaches.

NEURAL INFORMATION PROCESSING SYSTEMS

tion

Making AlphaGo Better



Bayesian Optimization in AlphaGo

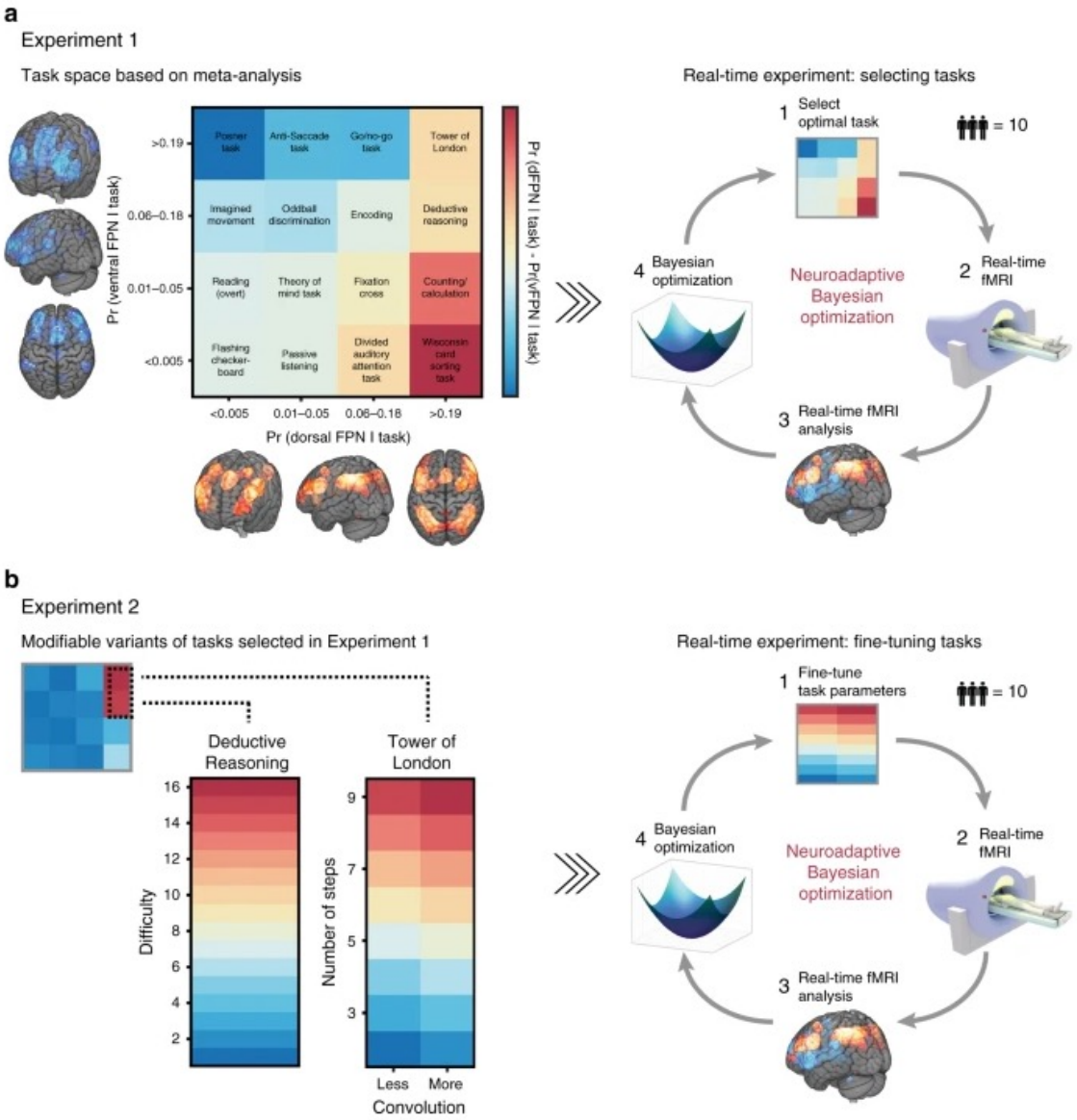
Yutian Chen, Aja Huang, Ziyu Wang, Ioannis Antonoglou, Julian Schrittwieser, David Silver & Nando de Freitas

DeepMind, London, UK
yutianc@google.com

Abstract

During the development of AlphaGo, its many hyper-parameters were tuned with Bayesian optimization multiple times. This automatic tuning process resulted in substantial improvements in playing strength. For example, prior to the match with Lee Sedol, we tuned the latest AlphaGo agent and this improved its win-rate from 50% to 66.5% in self-play games. This tuned version was deployed in the final match. Of course, since we tuned AlphaGo many times during its development cycle, the compounded contribution was even higher than this percentage. It is our hope that this brief case study will be of interest to Go fans, and also provide Bayesian optimization practitioners with some insights and inspiration.

Neuroscience and Brain Analytics



Credit: https://www.nature.com/articles/s41467-018-03657-3

Common Attributes of the Search Problem

- Search Space: Many candidate choices (inputs)
- Objective function: Need to perform an expensive experiment to evaluate the objective value of any input
- Optimization problem: find the candidate input with highest objective function value

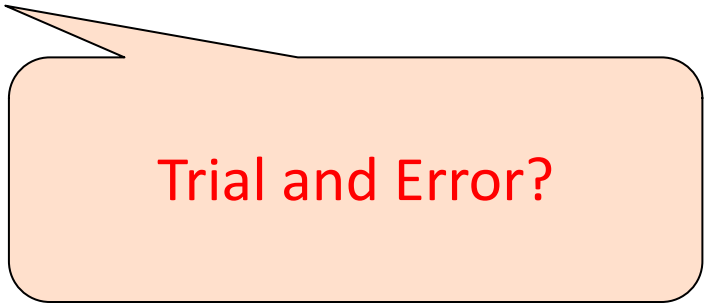
Common Attributes of the Search Problem

- Search Space: Many candidate choices (inputs)
- Objective function: Need to perform an expensive experiment to evaluate the objective value of any input
- Optimization problem: find the candidate input with highest objective function value

Cannot afford exhaustive search

Common Attributes of the Search Problem

- Search Space: Many candidate choices (inputs)
- Objective function: Need to perform an expensive experiment to evaluate the objective value of any input
- Optimization problem: find the candidate input with highest objective function value



Common Attributes of the Search Problem

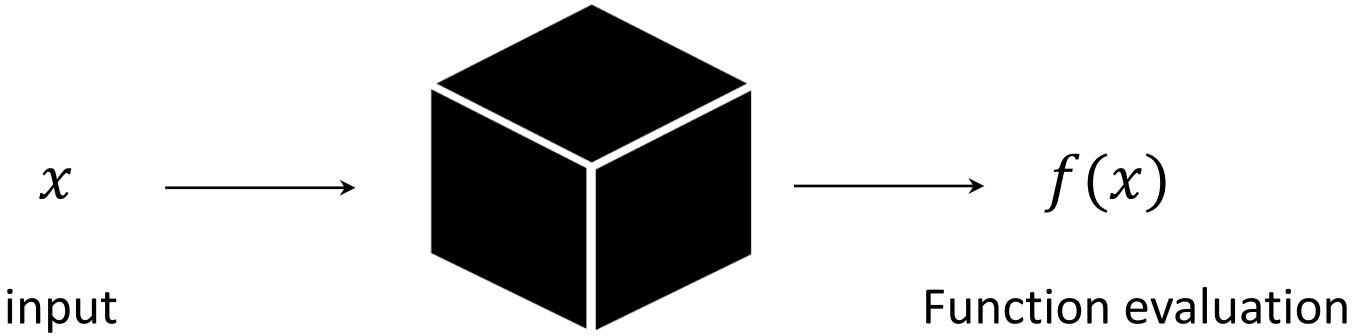
- Search Space: Many candidate choices (inputs)
- Objective function: Need to perform an expensive experiment to evaluate the objective value of any input
- Optimization problem: find the candidate input with highest objective function value

Can we do better than trial-and-error?

Accelerate Search via Bayesian Optimization

- Efficiently optimize expensive black-box functions

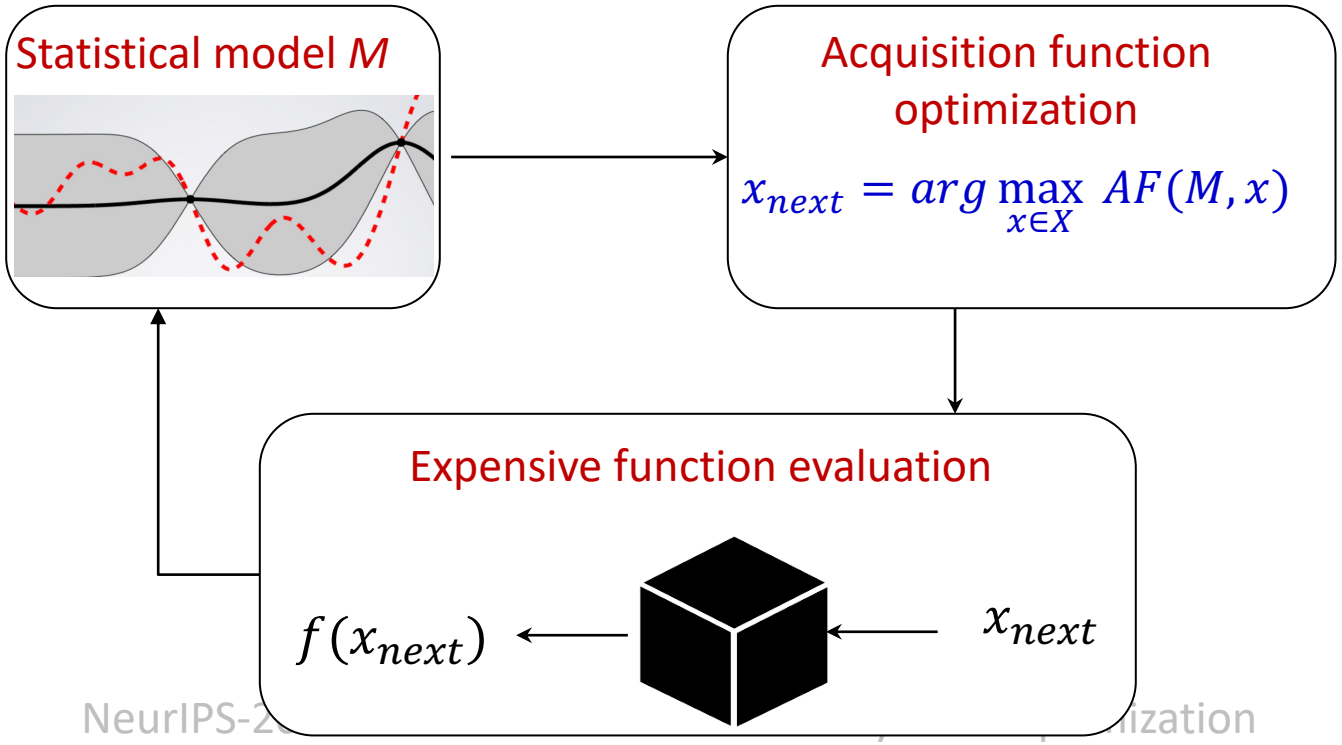
$$x^* = arg\max_{x \in X} f(x)$$



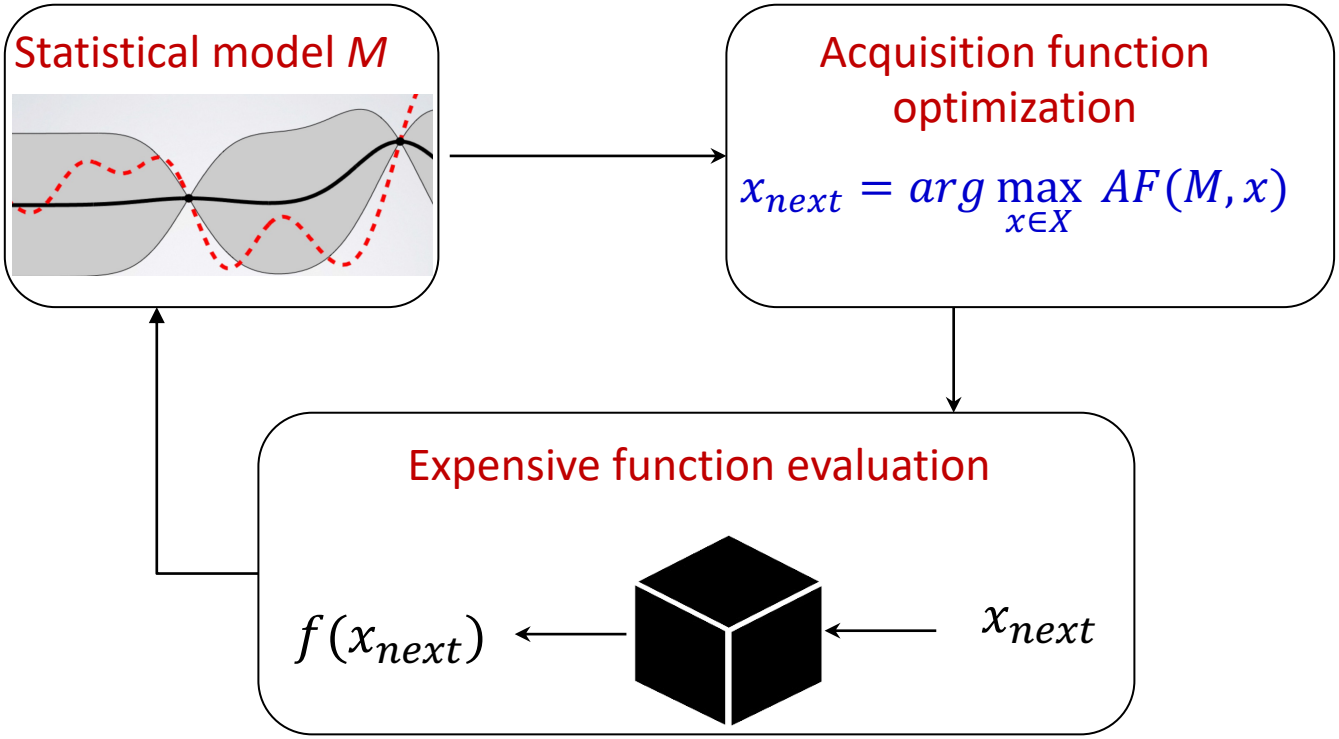
- Black-box queries (aka experiments) are expensive

Bayesian Optimization: Key Idea

- Build a surrogate statistical model and use it to intelligently search the space
  - Replace expensive queries with cheaper queries
  - Use uncertainty of the model to select expensive queries



Bayesian Optimization: Three Key Elements



- Statistical model (e.g., Gaussian process)
- Acquisition function (e.g., Expected improvement)
- Acquisition function optimizer (e.g., local search)

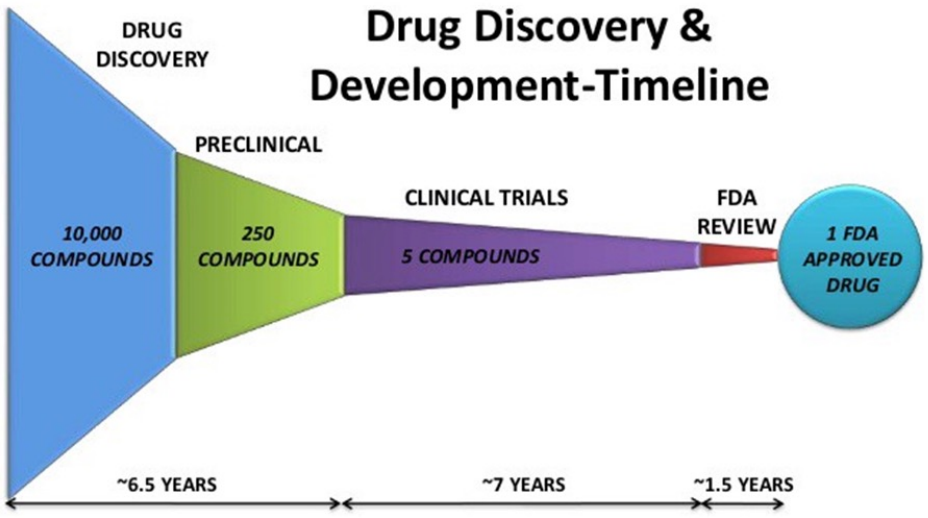
BO Dimensions: Input Space

- **Continuous space**
  - All variables of input x are continuous
- **Discrete / Combinatorial space**
  - Sequences, trees, graphs, sets, permutations etc.
- **Hybrid space**
  - x = mixture of x_d (discrete) and x_c (continuous) variables

BO Dimensions: No. of Objectives

- **Single objective**
  - For example, finding hyperparameters to optimize accuracy

- **Multiple objectives**

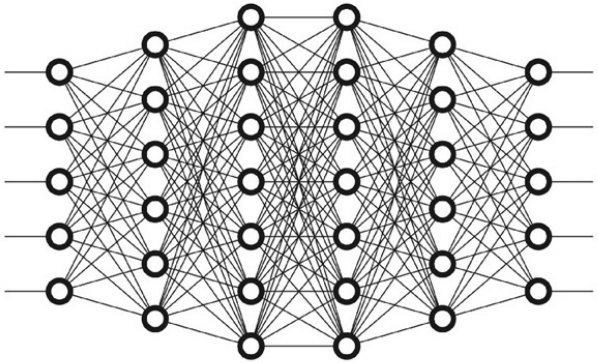


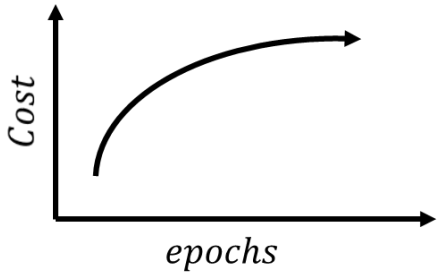
Credit: MIMA healthcare

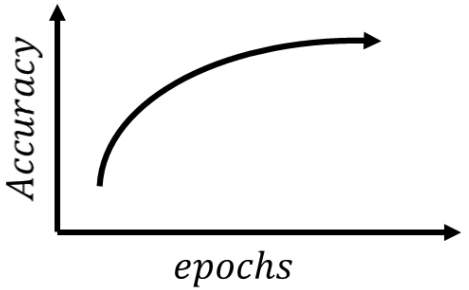
BO Dimensions: No. of Fidelities

- **Single-fidelity setting**
  - Most expensive and accurate function evaluation

- **Multi-fidelity setting**
  - Function evaluations with varying trade-offs in cost and accuracy



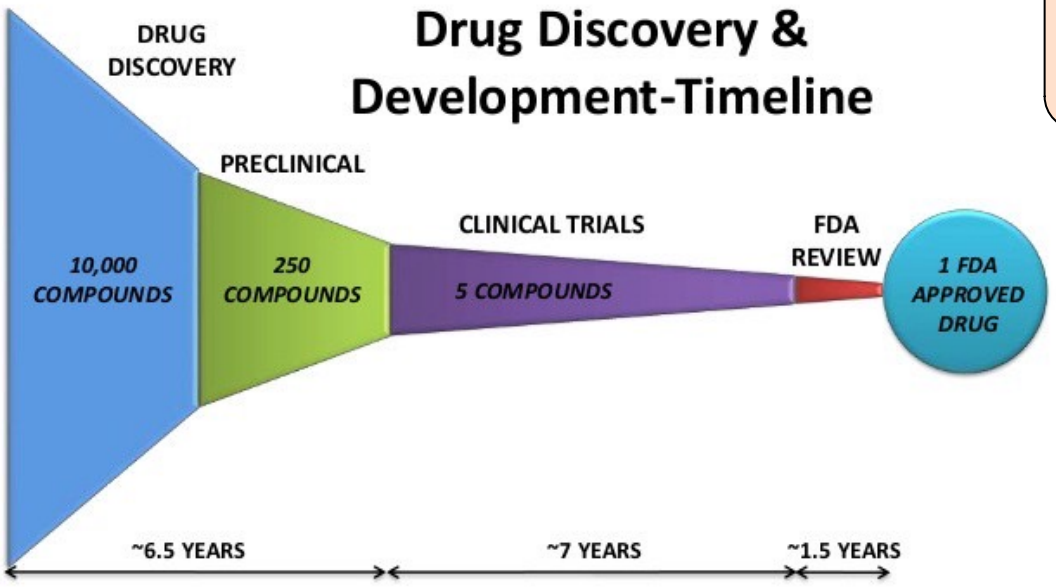




BO Dimensions: Constraints

- **Unconstrained setting**
  - all inputs are valid

- **Constrained setting**



Drugs/Vaccines that are safe

Outline of the Tutorial

- Quick Overview of the BO Framework and GPs
- Summary of advances in GPs and Acquisition Functions
- Bayesian Optimization over Discrete/Hybrid Spaces

- High-Dimensional Bayesian Optimization
- BoTorch Hands-on Demonstration

- Causal Bayesian Optimization
- Summary and Outstanding Challenges in BO

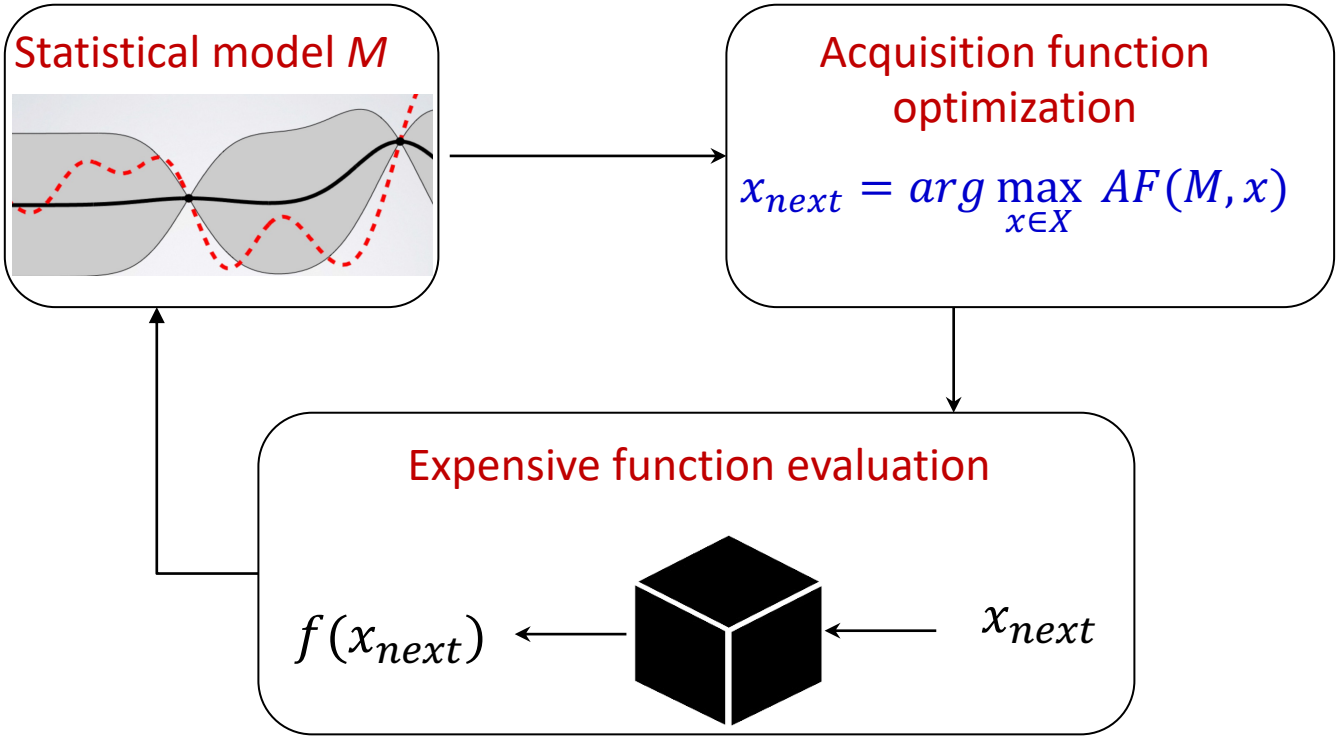
Outline of the Tutorial

- Quick Overview of the BO Framework and GPs
- Summary of advances in GPs and Acquisition Functions
- Bayesian Optimization over Discrete/Hybrid Spaces

- High-Dimensional Bayesian Optimization
- BoTorch Hands-on Demonstration

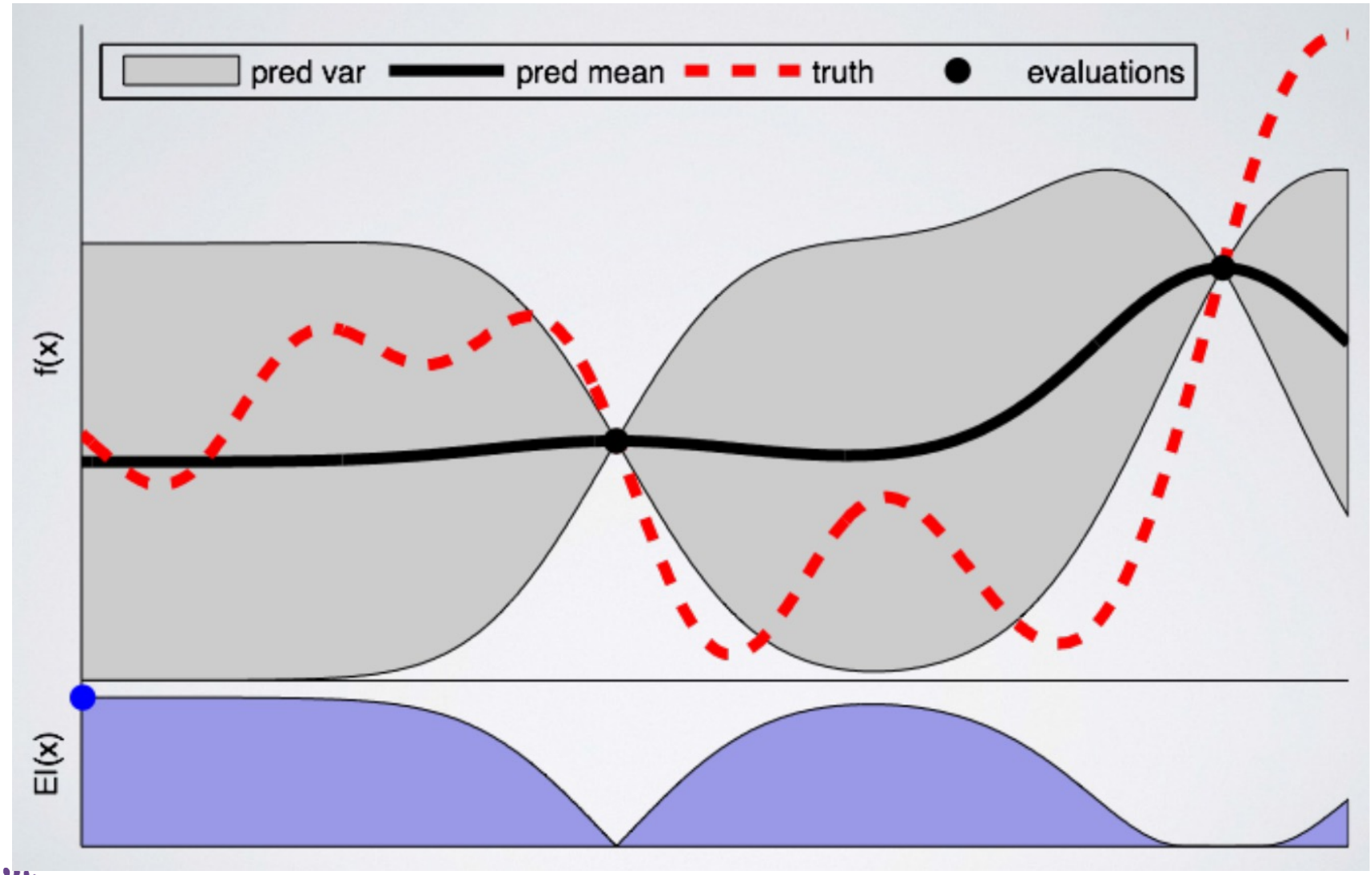
- Causal Bayesian Optimization
- Summary and Outstanding Challenges in BO

Bayesian Optimization Framework



- Statistical model (e.g., Gaussian process)
- Acquisition function (e.g., Expected improvement)
- Acquisition function optimizer (e.g., local search)

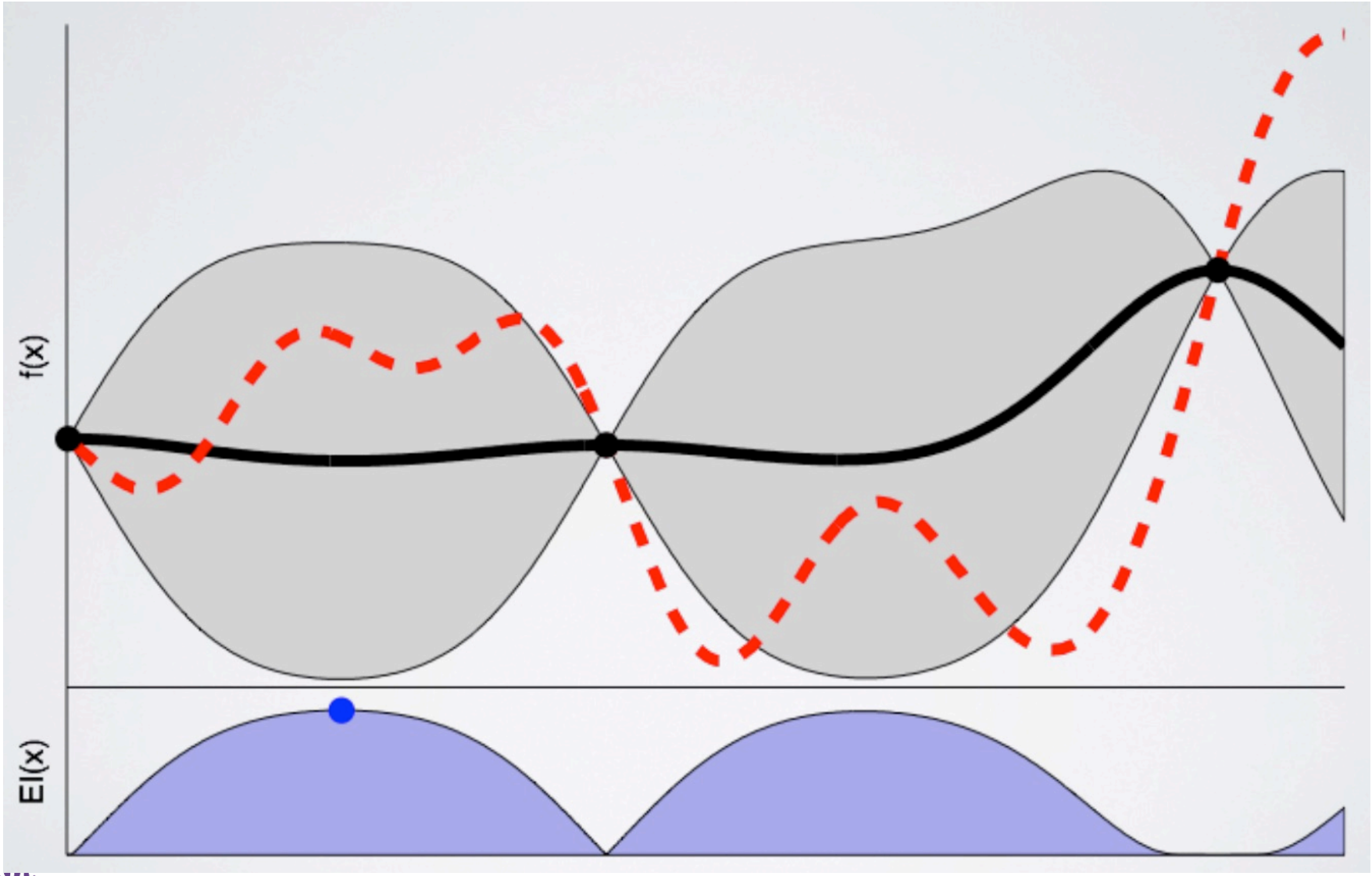
Bayesian Optimization: Illustration



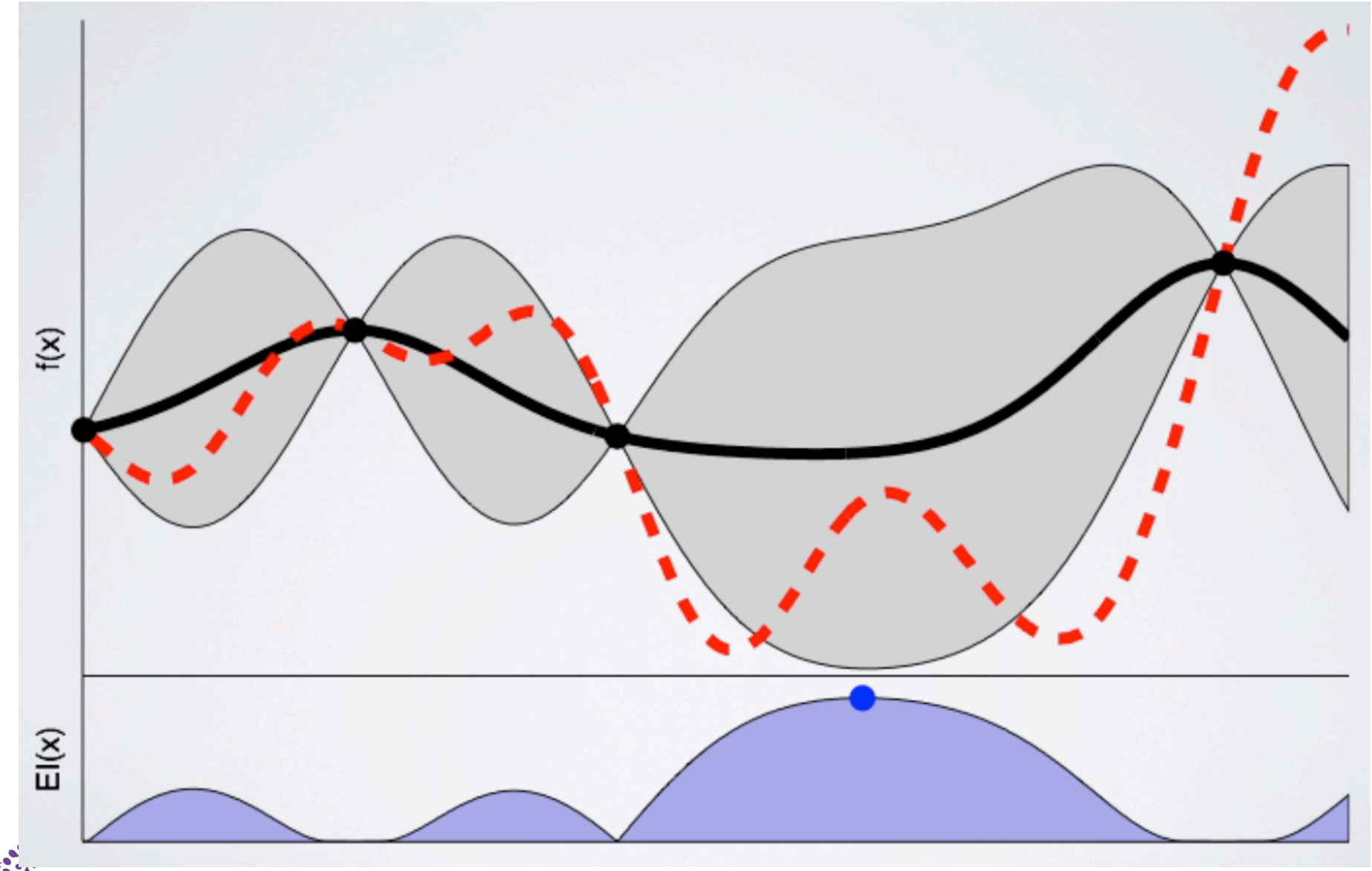
Credit: Ryan Adams

https://www.cs.toronto.edu/~rgrosse/courses/csc411_f18/tutorials/tut8_adams_slides.pdf

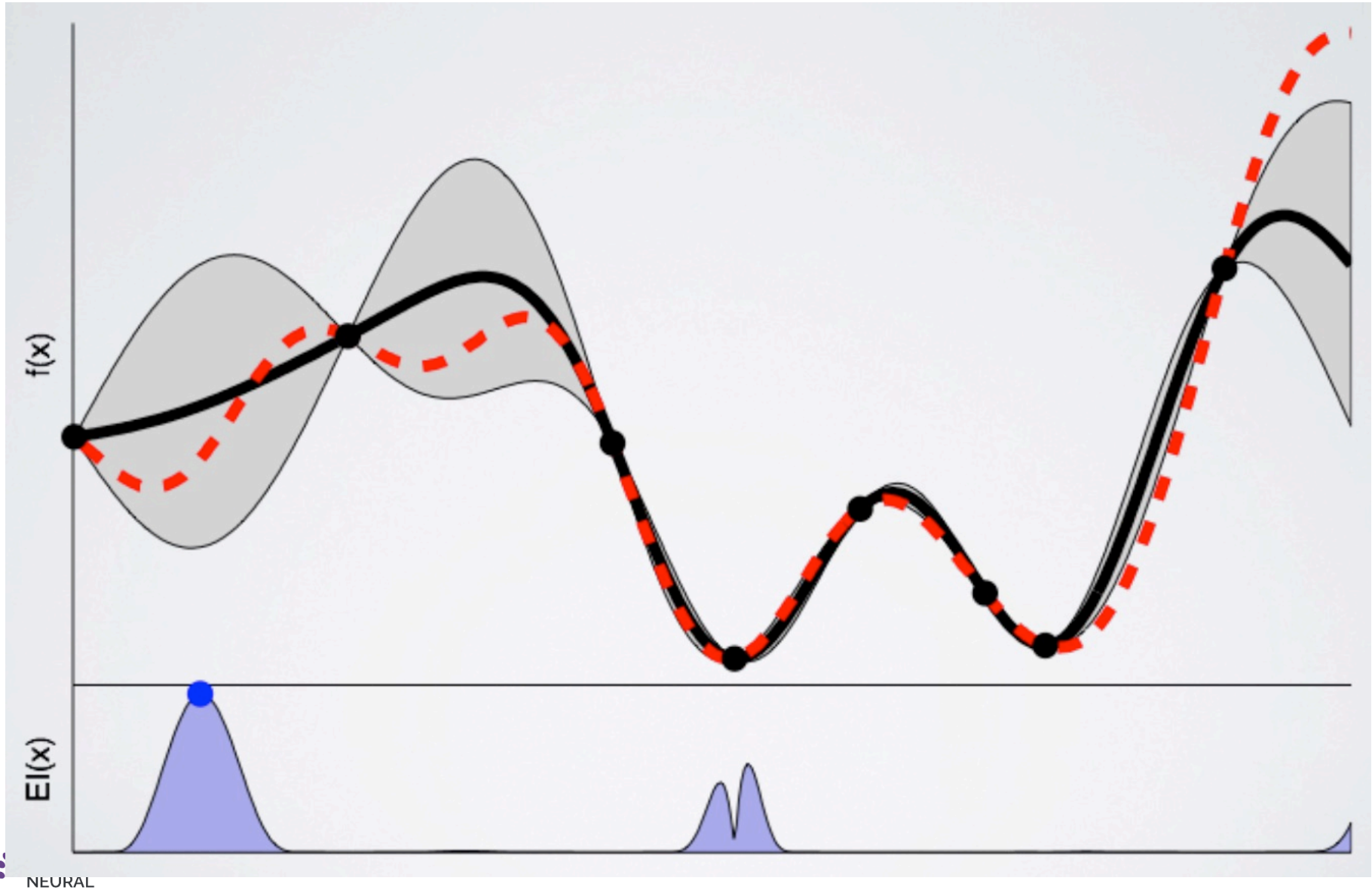
Bayesian Optimization: Illustration



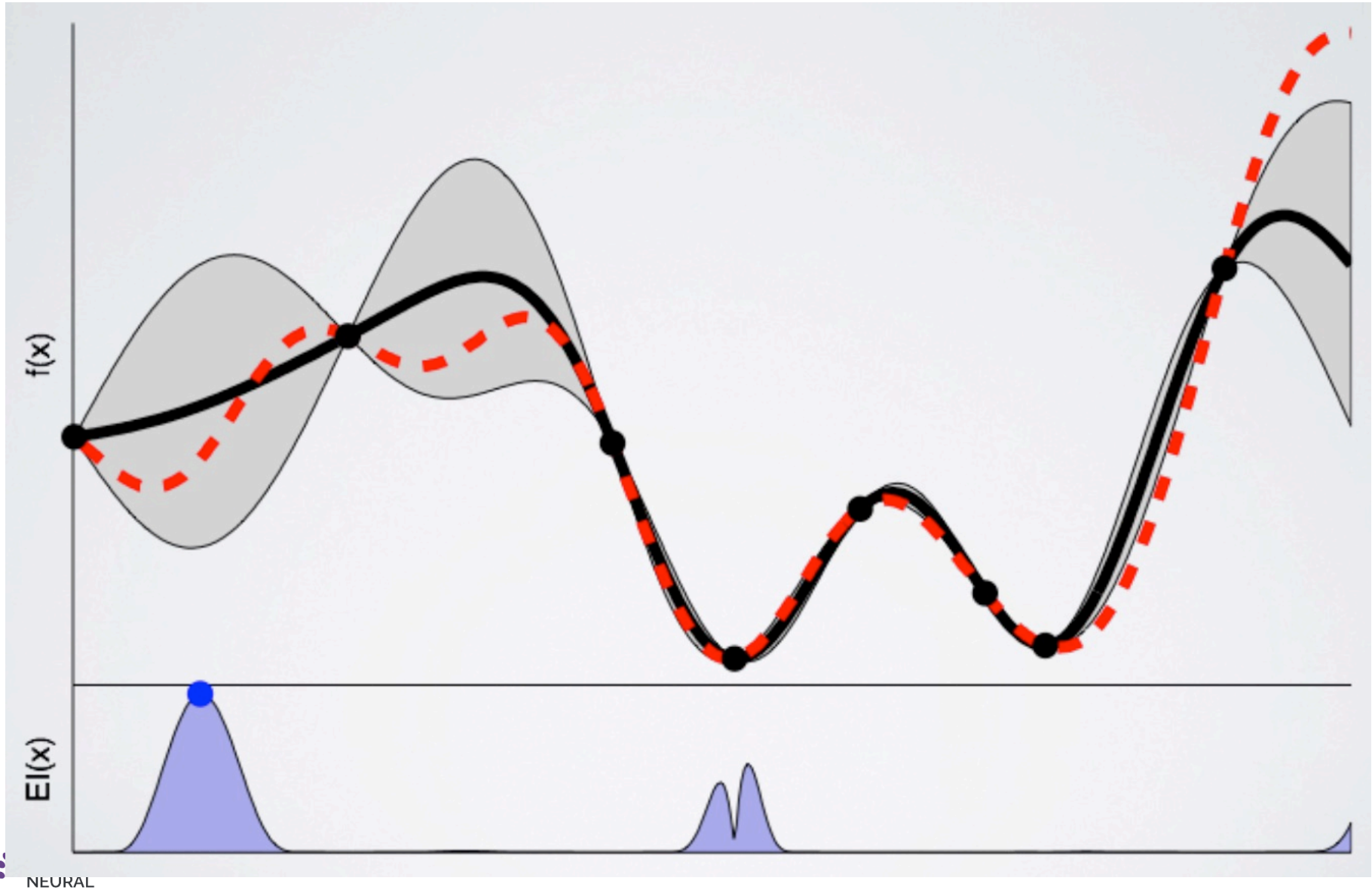
Bayesian Optimization: Illustration



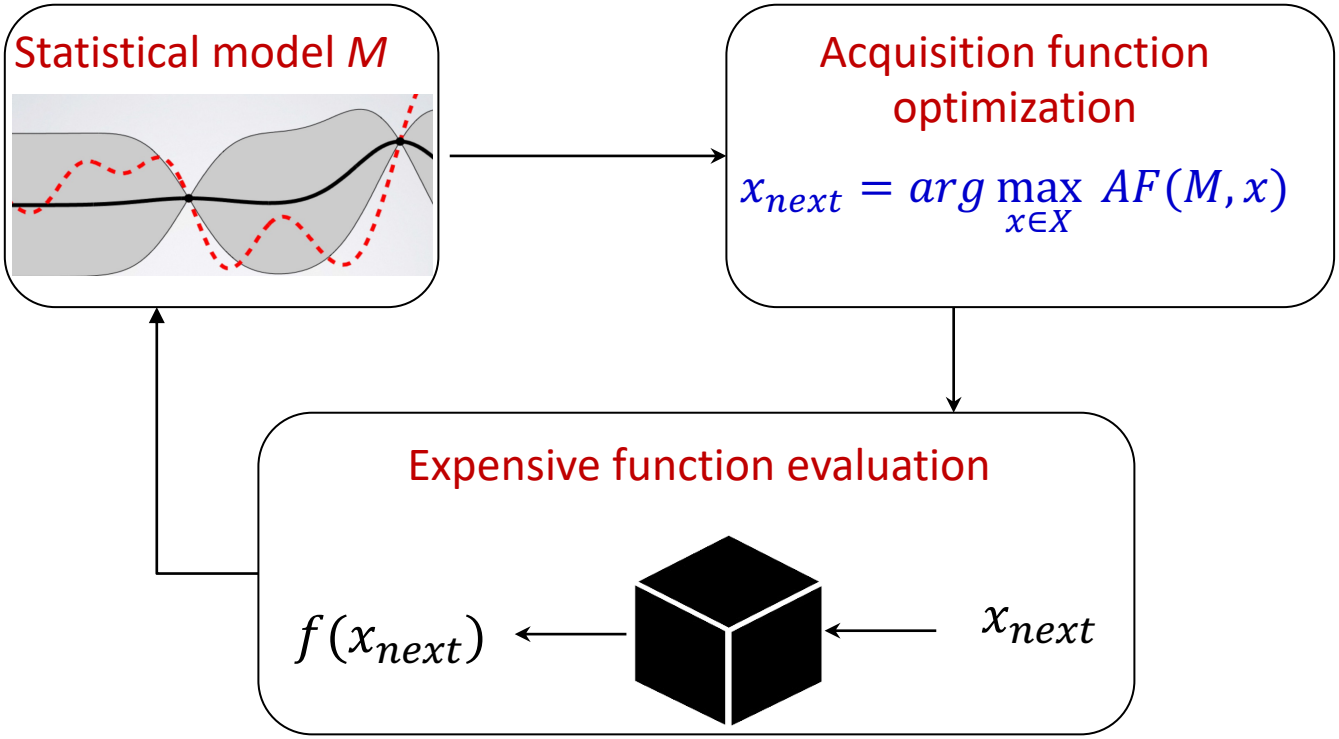
Bayesian Optimization: Illustration



Bayesian Optimization: Illustration



Bayesian Optimization: Three Key Elements



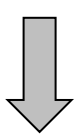
- Statistical model (e.g., Gaussian process)
- Acquisition function (e.g., Expected improvement)
- Acquisition function optimizer (e.g., local search)

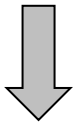
BO needs a Probabilistic Model

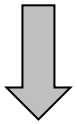
- To make predictions on unknown input
- To quantify the uncertainty in predictions

- One popular class of such models are Gaussian Processes (also called GPs)

Non-parametric, Bayesian and Kernel driven model







Flexibility

Principled uncertainty estimation

Specification of prior beliefs about rich function classes

Gaussian Process

- **Stochastic process definition**
  - Given any set of input points $\{x_1, x_2, \dots, x_m\}$, the output values follows a multi-variate Gaussian distribution

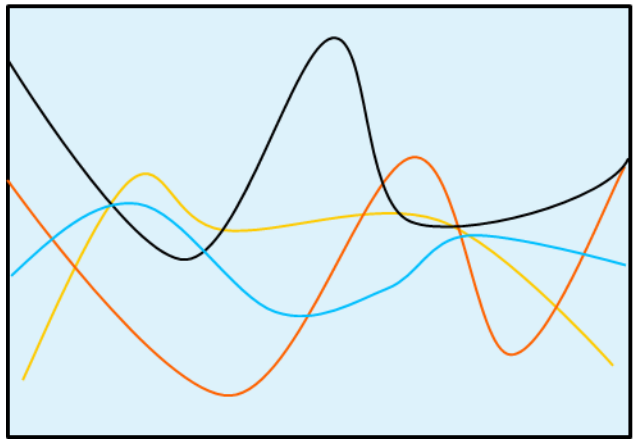
$$[f(x_1), f(x_2), f(x_3), \dots, f(x_m)] \sim \mathcal{N}(0, \Sigma)$$

- The covariance matrix Σ is given by a kernel function $k(x, x')$, i.e., $\Sigma_{ij} = k(x_i, x_j)$
  - Kernel captures the similarity between x and x'
- Choice of kernel $k(x, x')$ is critical for good performance
  - Allows to incorporate domain knowledge (e.g., Morgan fingerprints in chemistry)
  - Matern kernel is a popular choice for continuous spaces

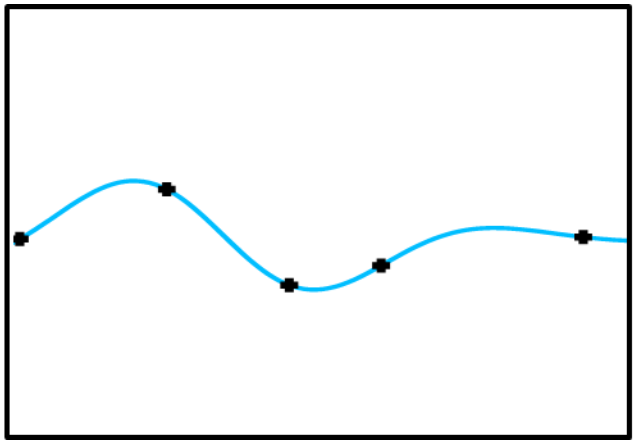
Gaussian Process: Two Views

- Function space view: distribution over functions
  - Function class is characterized by kernel

Prior



Posterior



- Weight space view: Bayesian linear regression in kernel's feature space

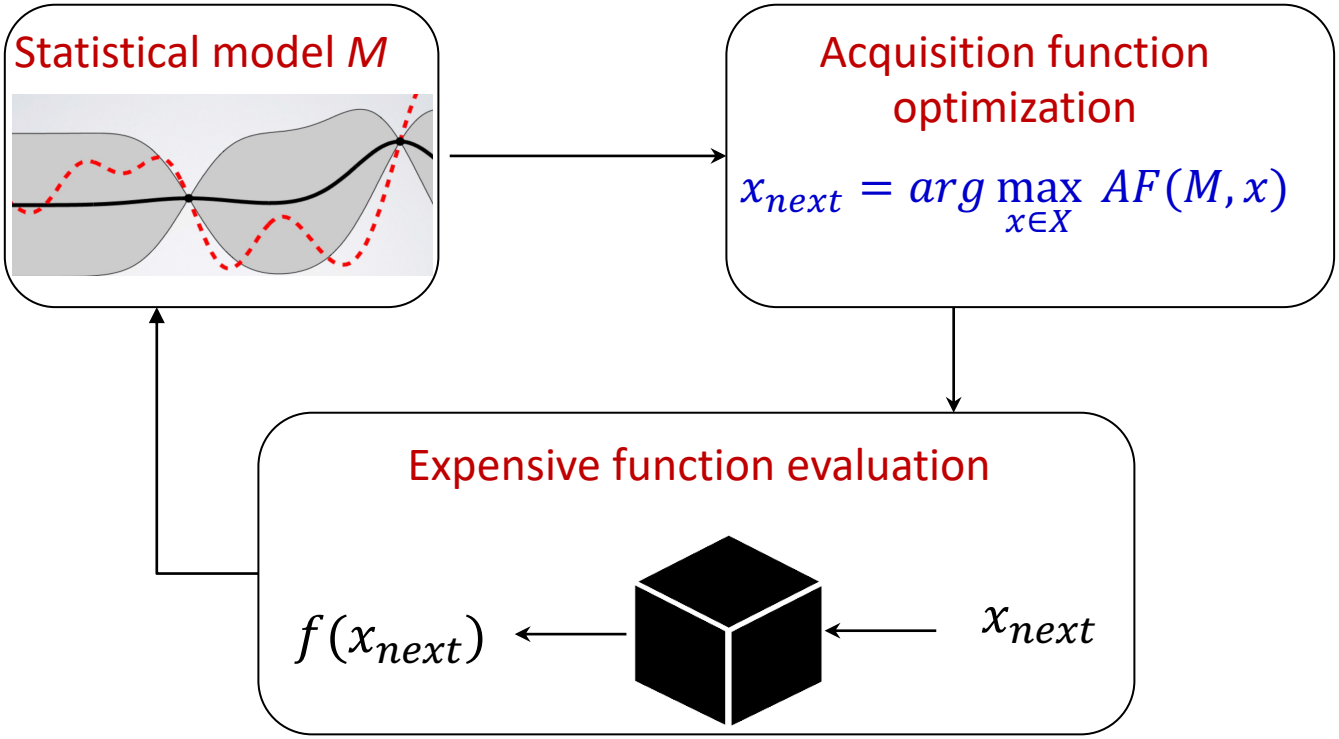
$$f(x) = w^T \tau(x)$$

$$k(x, x') = < \tau(x), \tau(x') >$$

Alternate Model Choices

- Random forest [Hutter et al., 2010, 2011]
- Bayesian neural networks
- Classification models
  - BORE [Tiao et al., 2021] and LFBO [Song et al., 2022]
  - Can work with any input space (continuous, discrete, hybrid)

Bayesian Optimization: Three Key Elements

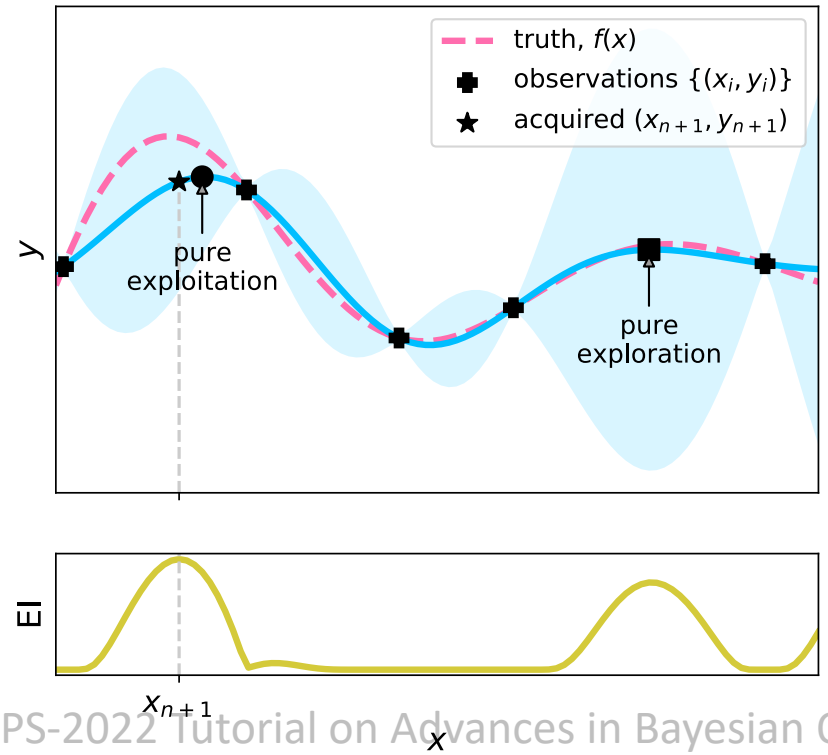


- Statistical model (e.g., Gaussian process)
- Acquisition function (e.g., Expected improvement)
- Acquisition function optimizer (e.g., local search)

Acquisition Function

- Intuition: captures utility of evaluating an input

- Challenge: trade-off exploration and exploitation
  - Exploration: seek inputs with high variance
  - Exploitation: seek inputs with high mean



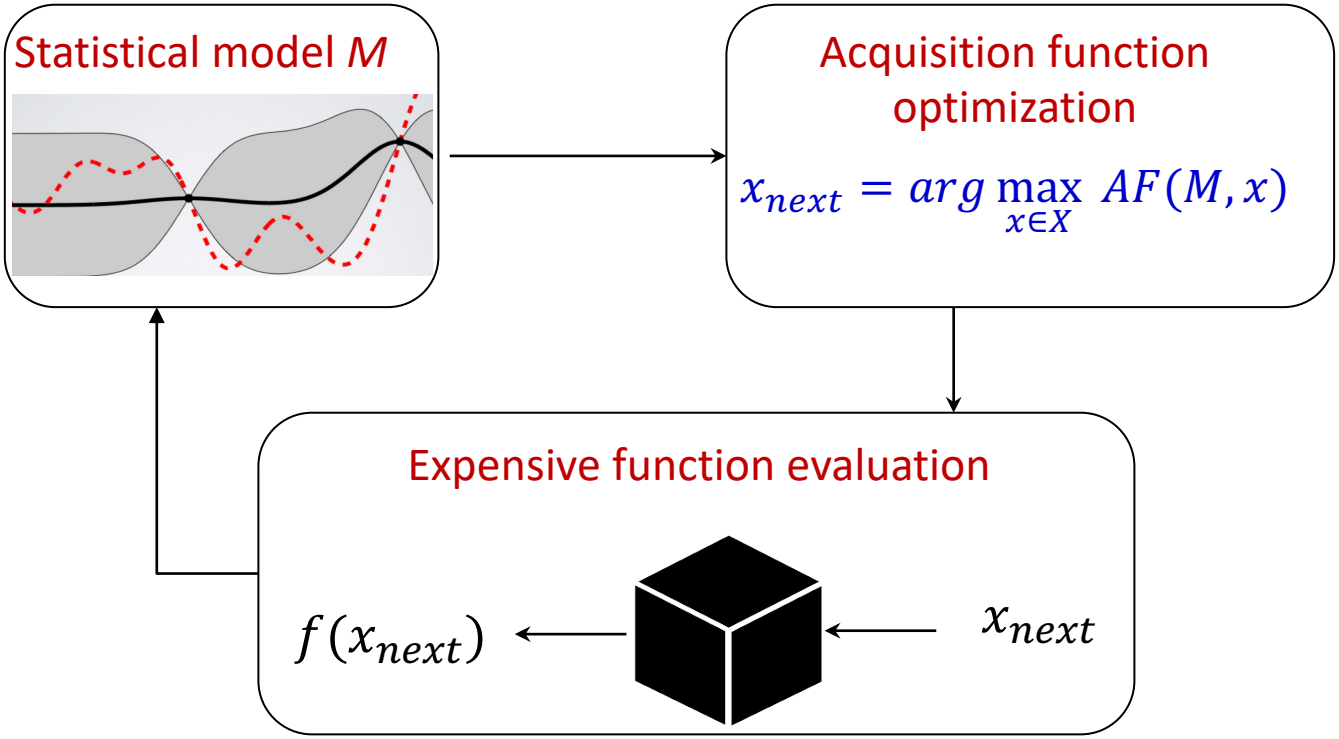
Acquisition Function: Examples

- Upper Confidence Bound (UCB)
  - Selects input that maximizes upper confidence bound

$$AF(x) = y^*(x) + \beta\, \sigma^*(x)$$

- Expected Improvement (EI)
  - Selects input with highest expected improvement over the incumbent
- Thompson Sampling (TS)
  - Selects optimizer of a function sampled from the surrogate model's posterior
- Knowledge Gradient

Bayesian Optimization: Three Key Elements



- Statistical model (e.g., Gaussian process)
- Acquisition function (e.g., Expected improvement)
- Acquisition function optimizer (e.g., local search)

Acquisition Function Optimizer

- Challenge: non-convex/multi-modal optimization problem

- Commonly used approaches for continuous spaces
  - Space partitioning methods (e.g., DIRECT, LOGO)
  - Gradient based methods (e.g., Gradient descent)
  - Evolutionary search (e.g., CMA-ES)

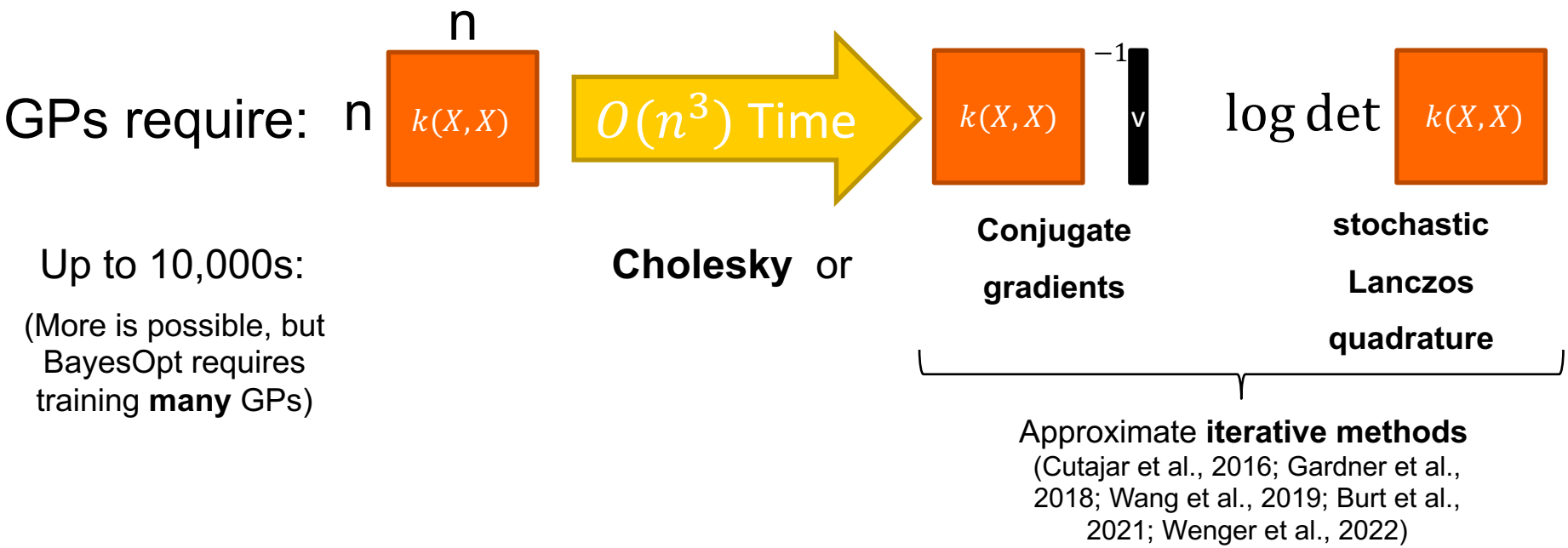
Outline of the Tutorial

- Quick Overview of the BO Framework and GPs
- Summary of advances in GPs and Acquisition Functions
- Bayesian Optimization over Discrete/Hybrid Spaces

- High-Dimensional Bayesian Optimization
- BoTorch Hands-on Demonstration

- Causal Bayesian Optimization
- Summary and Outstanding Challenges in BO

Scaling GPs for Bayesian Optimization



Scaling GPs for Bayesian Optimization

GPs require: n $k(X,X)$ n $O(n^3)$ Time $k(X,X)^{-1}$ v log det $k(X,X)$

Up to 10,000s:

(More is possible, but BayesOpt requires training **many** GPs)

Cholesky or

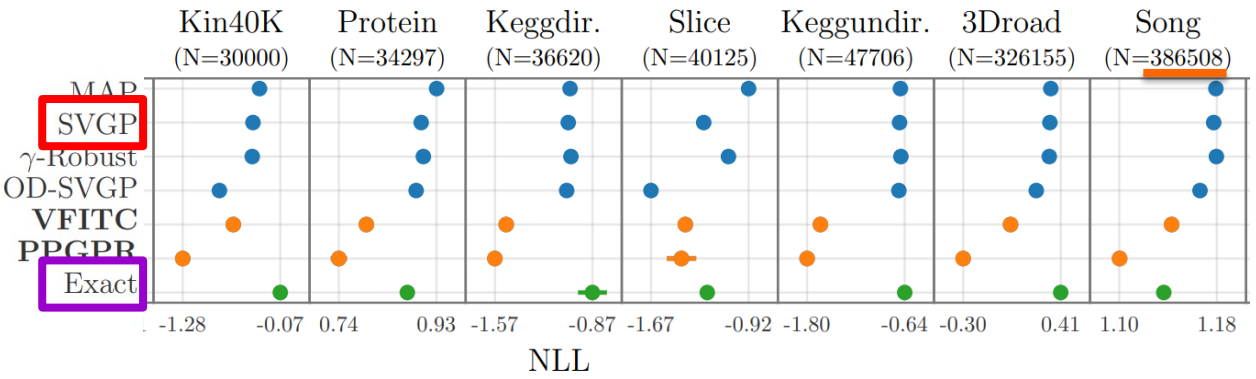
Conjugate gradients

stochastic Lanczos quadrature

Approximate **iterative methods**
(Cutajar et al., 2016; Gardner et al., 2018; Wang et al., 2019; Burt et al., 2021; Wenger et al., 2022)

More data: use **Stochastic Variational Inference**

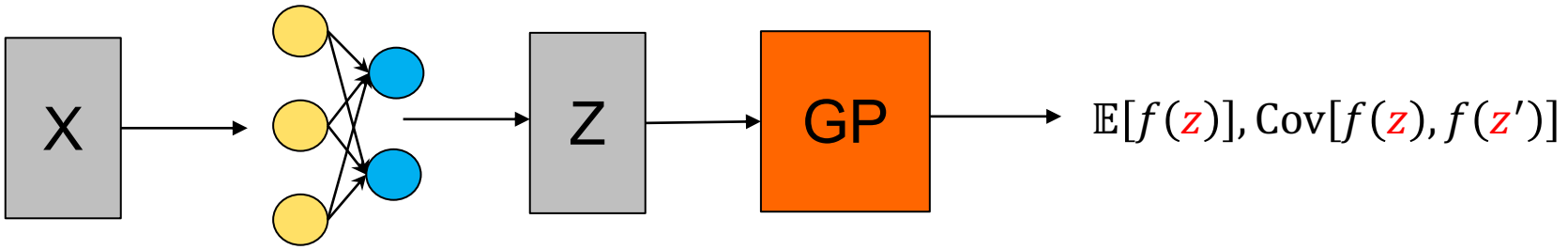
(Hensman et al., 2013; Salimbeni et al., 2018; Jankowiak et al., 2020)



(Lower is better)

More Expressive Models for More Data

GP Model with Deep Kernel:



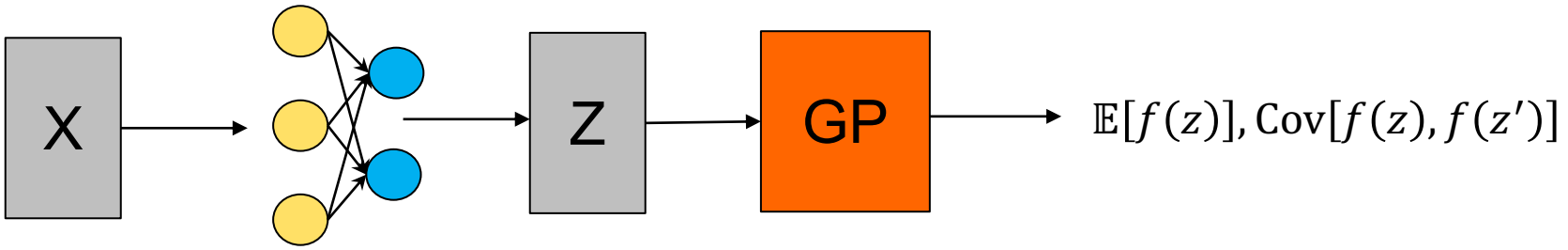
GPyTorch model implementation:

```
class GP(gpytorch.models.ExactGP):
    def __init__(self, train_x, train_y, likelihood):
        super().__init__(train_x, train_y, likelihood)
        self.covar_module = gpytorch.kernels.RBFKernel()
        self.mean_module = gpytorch.kernels.MeanModule()

    def forward(self, x):
        mean_x = self.mean_module(x)
        covar_x = self.covar_module(x)
        return gpytorch.distributions.MultivariateNormal(mean_x, covar_x)
```

More Expressive Models for More Data

GP Model with Deep Kernel:



GPyTorch model implementation:

```
class DKLGP(gpytorch.models.ExactGP):
    def __init__(self, train_x, train_y, likelihood):
        super().__init__(train_x, train_y, likelihood)
        self.covar_module = gpytorch.kernels.RBFKernel()
        self.mean_module = gpytorch.kernels.MeanModule()
        self.feature_extractor = torch.nn.Sequential(
            torch.nn.Linear(train_x.size(-1), 32),
            torch.nn.BatchNorm1d(),
            torch.nn.ReLU(),
            torch.nn.Linear(32, 16),
            torch.nn.BatchNorm1d(),
        )

    def forward(self, x):
        z = self.feature_extractor(x)
        mean_z = self.mean_module(z)
        covar_z = self.covar_module(z)
        return gpytorch.distributions.MultivariateNormal(mean_z, covar_z)
```

Information-Theoretic Acquisition Functions

- Key principle: select inputs for evaluation which provide maximum information about the optimum

- Concretely, pick observations which quickly decrease the entropy of distribution over the optimum

$$AF(x) = \text{ Expected decrease in entropy}$$
$$AF(x) = H(\alpha \mid D) - E_y[H(\alpha | D \cup \{x, y\}$$
$$= \text{Information Gain}(\alpha; y)$$

- Design choices of α leads to different algorithms

Information-Theoretic Acquisition Functions

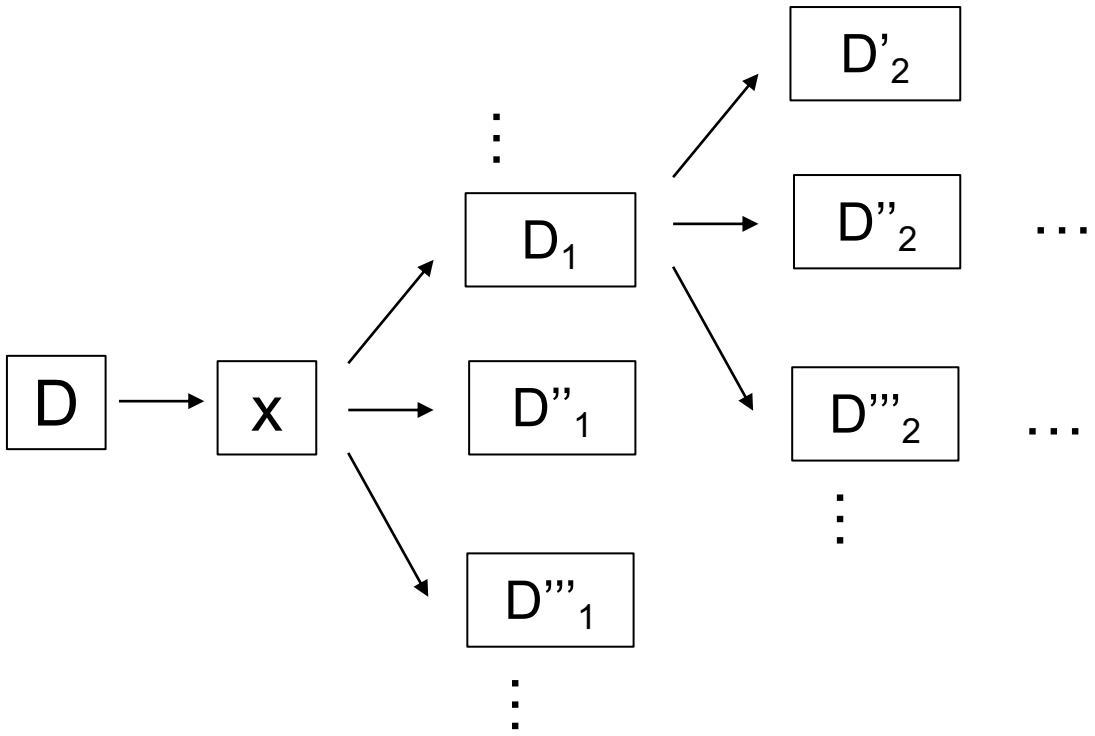
- Design choices of α leads to different algorithms

$$AF(x) = \text{Expected decrease in entropy}$$
$$AF(x) = H(\alpha \mid D) - E_y[H(\alpha | D \cup \{x, y\}$$
$$= \text{Information Gain}(\alpha; y)$$

- α as input location of optima x^*
  - Entropy Search (ES) / Predictive Entropy Search (PES)
  - Intuitive but requires expensive approximations
- α as output value of optima y^*
  - Max-value Entropy Search (MES) and it's variants
  - Computationally cheaper and more robust
- Generalization via decision-theoretic entropies [Neiswanger et al., 2022]

Non-Myopic / Lookahead Acquisition Functions

- Myopic acquisition functions (e.g., EI) reason about immediate utility
- Non-myopic variants consider BO as a MDP and reason about longer decision horizons



Non-Myopic / Lookahead Acquisition Functions

- Non-myopic variants consider BO as MDP and reason about longer decision horizons

$$u_k(x|D) = u_1(x|D) + E_y\left[\max_{x'} u_{t-1}(x'|D \cup \{x, y\})\right]$$

- Challenge: curse of dimensionality

$$u_k(x|D) = u_1(x|D) + E_y\left[\max_{x1}\{u(x_1|D_1) + E_{y1}[\max_{x2}\{u(x_2|D_2) \ldots.\}]\}\right]$$

- Some solutions [Lam et al., 2016, Lee et al., 2020, Gonzalez et al 2016, Jiang et al 2020]
  - Multi-step lookahead policies with approximations
  - Rollout based approximate dynamic programming

Outline of the Tutorial

- Quick Overview of the BO Framework and GPs
- Summary of advances in GPs and Acquisition Functions
- Bayesian Optimization over Discrete/Hybrid Spaces

- High-Dimensional Bayesian Optimization
- BoTorch Hands-on Demonstration

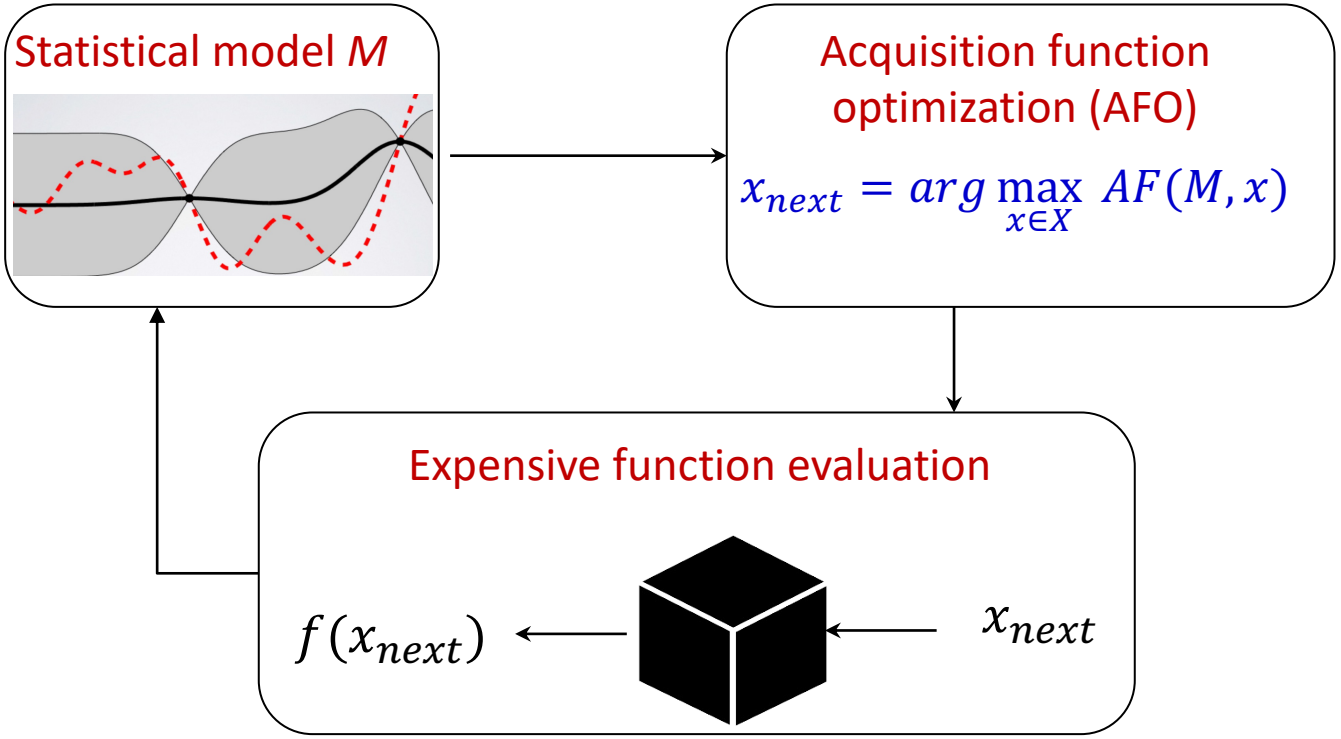
- Causal Bayesian Optimization
- Summary and Outstanding Challenges in BO

Combinatorial/Discrete BO

- **Given:** a combinatorial space of structures X (e.g., sequences, trees, graphs)
- **Find:** optimized combinatorial structure $x^* \in X$

- **Space of binary structures** $X = \{0,1\}^n$
  - Each structure $x \in X$ be represented using n binary variables $x_1, x_2, \ldots, x_n$
- **Categorical variables**
  - x_i can take more than two candidate values
- **How to deal with categorical variables?**
  - Option 1: Encode them as binary variables (a common practice)
  - Option 2: Modeling and reasoning over categorical variables

Discrete BO: Technical Challenges



- Effective modeling over combinatorial structures (e.g., sequences, graphs)
- Solving hard combinatorial optimization problem to select next structure
  - Not an issue if we are searching over a fixed database of structures

Discrete BO: Summary of Approaches

- Trade-off complexity of model and tractability of AFO

- Simple statistical models and tractable search for AFO
  - BOCS [Baptista et al., 2018]

- Complex statistical models and heuristic search for AFO
  - SMAC [Hutter et al., 2011] and COMBO [Oh et al., 2019]

- Complex statistical models and tractable/accurate AFO
  - L2S-DISCO [Deshwal et al., 2020] and MerCBO [Deshwal et al., 2021]
  - Reduction to continuous BO [Gómez-Bombarelli et al., 2018]…

Aside: Discrete BO vs. Structured Prediction

- **Structured prediction (SP)** [Lafferty et al., 2001] [Bakir et al., 2007]
  - Generalization of classification to structured outputs (e.g., sequences, trees, and graphs)
  - CRFs, Structured Perceptron, Structured SVM

- Complexity of cost function vs. tractability of inference

  - Simple cost functions (e.g., first-order) and tractable inference
  - Complex cost functions (e.g., higher-order) and heuristic inference
  - Learning to search for SP [Daume' et al., 2009] [Doppa et al., 2014]

Key Difference:

Small data vs. big data setting

Surrogate Models: BOCS [Baptista et al., 2018]

- Linear surrogate model over binary structures
  - $f(x \in X) = \theta^T . \phi(x)$
  - $\phi(x)$ consists of up to Quadratic (second-order) terms
  - $\phi(x) = [x_1, x_2, \dots, x_d, x_1 . x_2, x_1 . x_3, \dots, x_{d-1} . x_d]$

May not be sufficient to capture desired dependencies

Surrogate Models: SMAC [Hutter et al., 2010, 2011]

- Random forest as surrogate model
  - works naturally for categorical and mixed variables
  - Prediction/Uncertainty (= empirical mean/variance over trees)

Uncertainty estimates can be poor

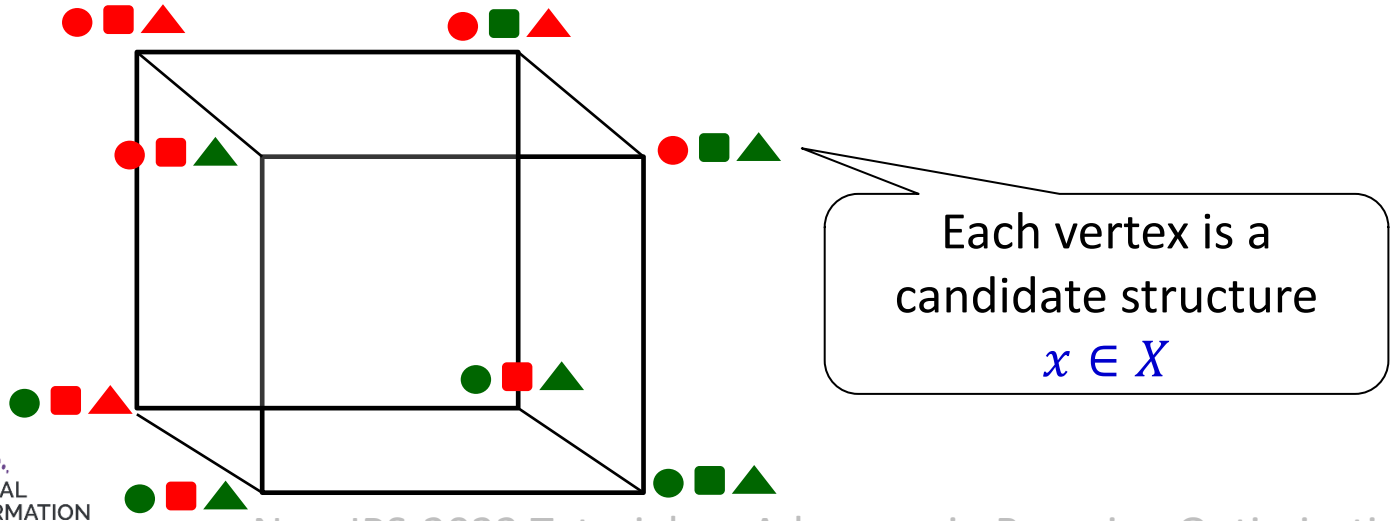
- Improvements for better uncertainty estimates
  - Bagging with oversampling [Kim and Choi, 2022]

Surrogate Models: COMBO [Oh et al., 2019]

- GP with diffusion kernel [Kondor and Lafferty 2002]
  - Requires a graph representation of the input space X

$$K(V,V) = exp(-\beta L(G))$$

- **Combinatorial graph representation** [Oh et al., 2019]

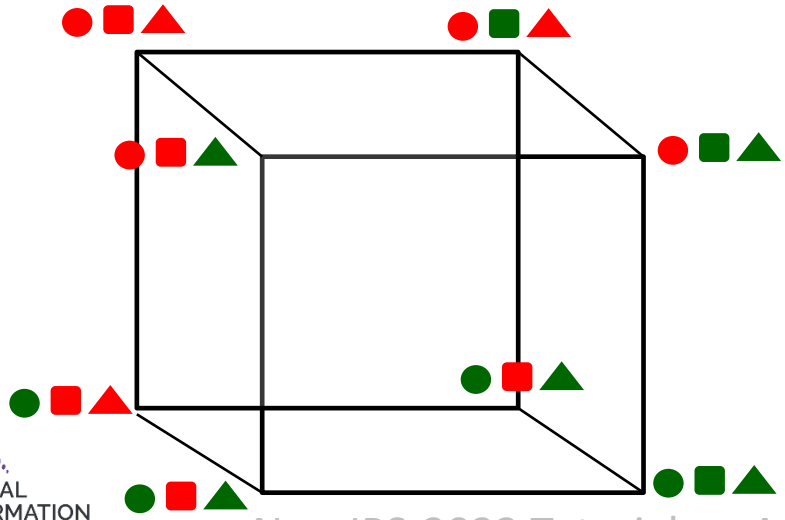


Surrogate Models: COMBO [Oh et al., 2019]

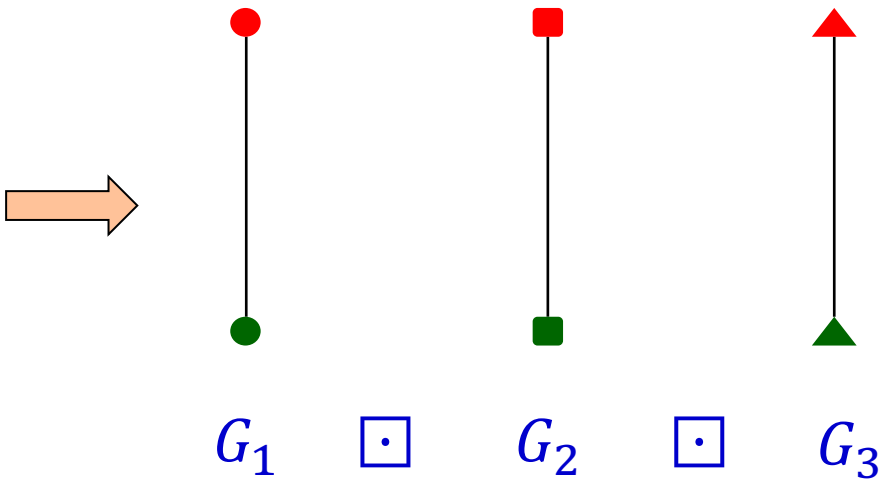
- GP with diffusion kernel [Kondor and Lafferty 2002]
  - Requires a graph representation of the input space X

$$K(V,V) = exp(-\beta L(G))$$

- **Combinatorial graph representation** [Oh et al., 2019]



G



$G_1 \quad \boxdot \quad G_2 \quad \boxdot \quad G_3$

Surrogate Models: GP via Structured Kernels

- Leverage prior work on kernels over structured data [Gartner, 2003] to build GP surrogate models

- Some examples from BO literatures
  - String kernels [Moss et al., 2020]
  - Kernels for protein design [Gruver et al., 2021]
  - Permutation kernels [Deshwal et al., 2022; Oh et al., 2021]
  - Weisfeiler-Lehman kernels for NAS [Ru et al., 2021]

Acquisition Functions

- Can use acquisition functions that don't require sampling from the posterior model
  - EI and UCB

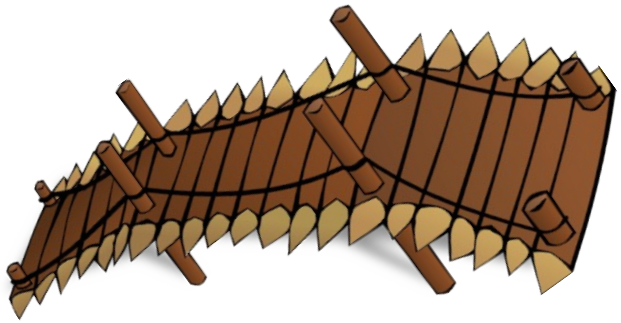
- Many advanced acquisition functions require sampling functions from posterior
  - Thompson sampling
  - Predictive entropy search
  - Max-value entropy search
  - …

Acquisition Function: Mercer Features [Deshwal et al., 2021]

- Mercer features allow sampling functions from GP posterior

- Missing puzzle to leverage prior acquisition functions
  - Thompson Sampling (TS)
  - Predictive Entropy Search (PES)
  - Max-value Entropy Search (MES)
  - …

BO for continuous spaces



BO for discrete spaces

Acquisition Function: Mercer Features [Deshwal et al., 2021]

- COMBO surrogate model: GP with discrete diffusion kernel and graph representation G

- **Key Idea:** exploit the structure of combinatorial graph G to compute its eigenspace in closed-form

- **Eigenvalue set:** $\{0, 2, \ldots, 2n\}$
  - j^{th} eigenvalue occurs with $\binom{n}{j}$ multiplicity

- **Eigenvector set:** Hadamard matrix (H) of order 2^n

$$H_{ij} = (-1)^{\langle r_i, r_j \rangle}$$

Acquisition Function: Mercer Features [Deshwal et al., 2021]

$$K(x_1, x_2) = \sum_{i=0}^{2^n - 1} e^{-\beta \lambda_i} \; -1^{<r_i, x_1>} \; -1^{<r_i, x_2>}$$

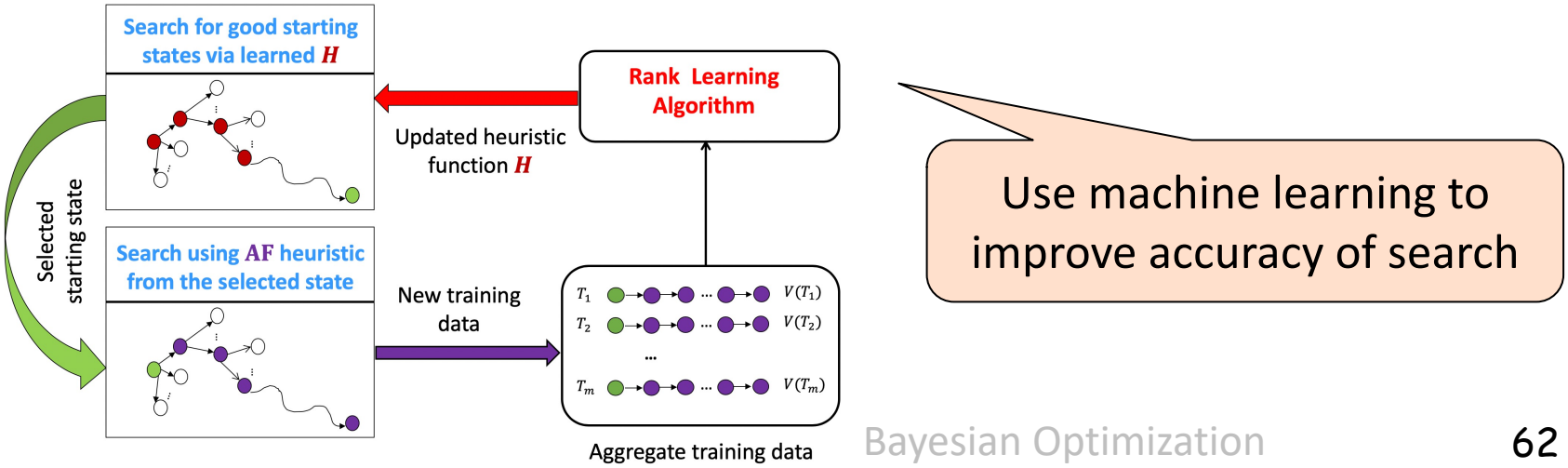
$$K(x_1, x_2) = \phi(x_1)^T \phi(x_2)$$

$$\phi(x)_i = \{\sqrt{e^{-\beta \lambda_i}} \; -1^{<r_i, x>}\}$$

j^{th} order Mercer features: first j distinct eigenvalues

Acquisition Function Optimization (AFO)

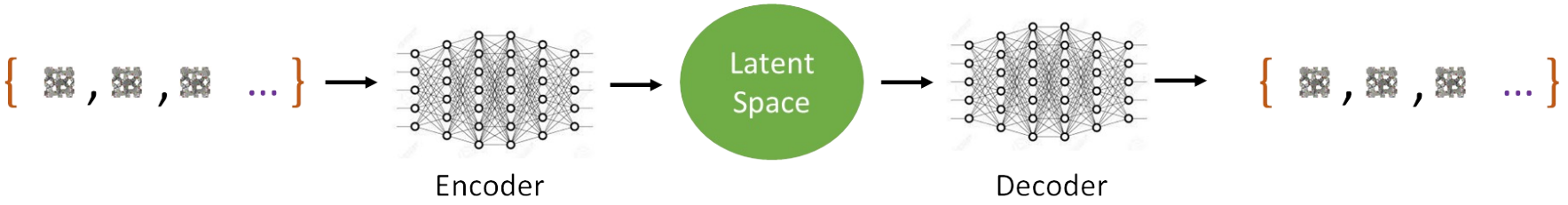
- Tractable search for AFO via Thompson sampling
  - BOCS [Baptista et al., 2018]
  - MerCBO [Deshwal et al., 2021]
  - Neural network + MILP [Papalexopoulos et al., 2022]
- Heuristic search (local search with restarts) for AFO
  - SMAC [Hutter et al., 2011] and COMBO [Oh et al., 2019]
- Learning-to-search framework for AFO [Deshwal et al., 2020]



Reduction to Continuous BO [Gómez-Bombarelli et al., 2018]…

- **Key Idea:** Convert discrete space into continuous space

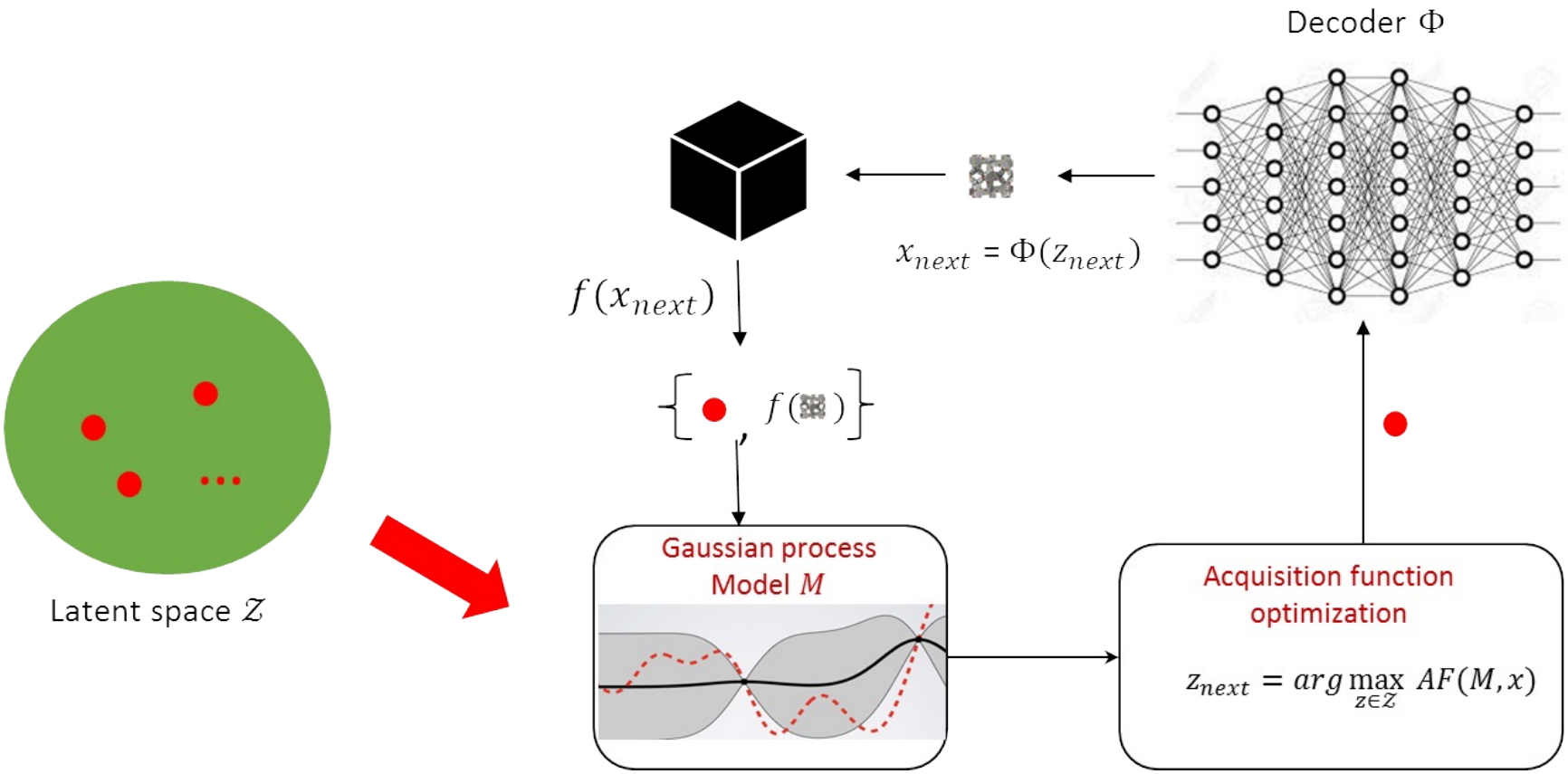
- Train a deep generative model (VAE) using unsupervised structures



- Perform BO in the learned continuous latent space
  - Surrogate modeling and acquisition function optimization in latent space (vs. combinatorial space)

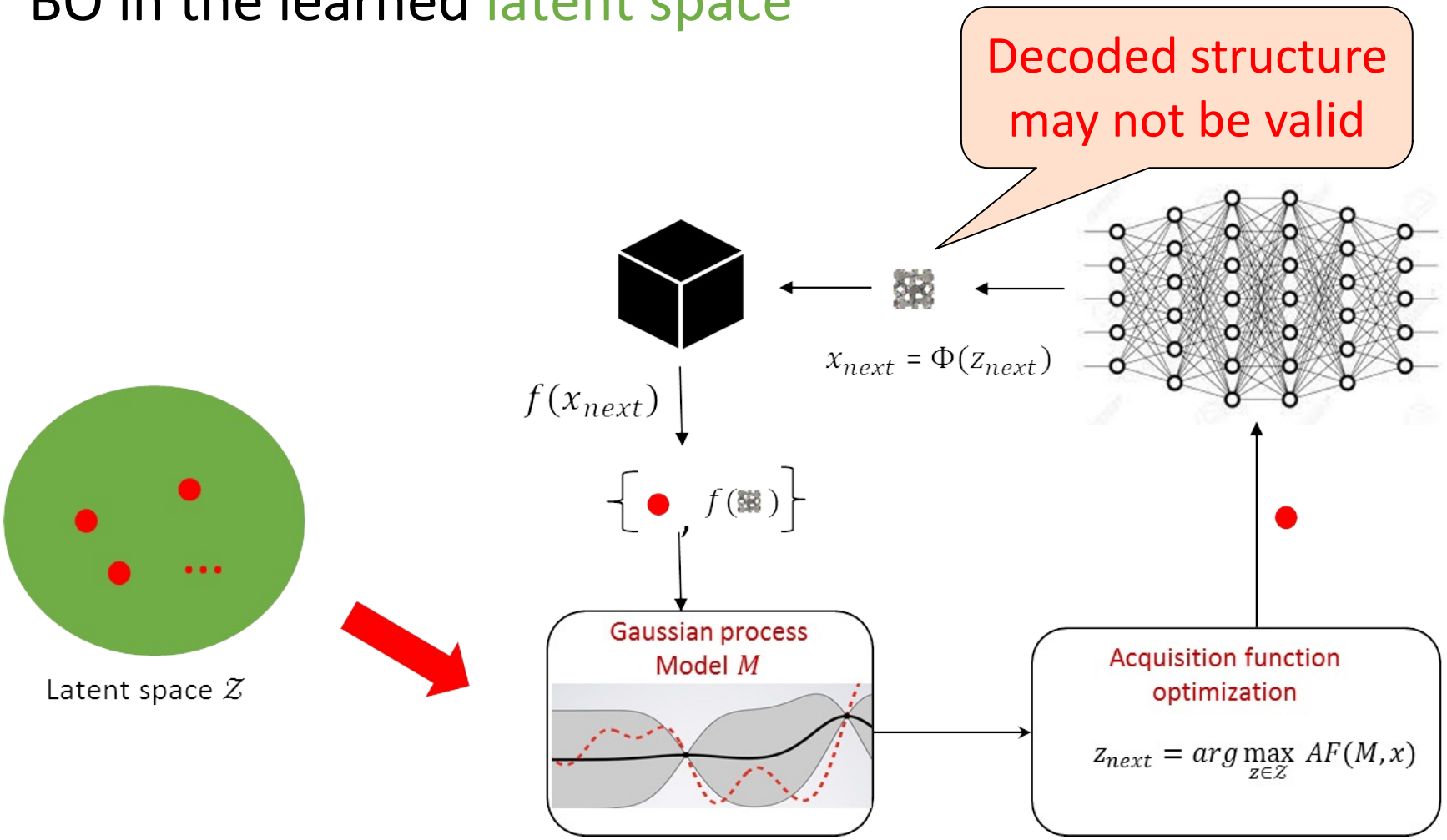
Reduction to Continuous BO [Gómez-Bombarelli et al., 2018]…

- BO in the learned latent space



Reduction to Continuous BO [Gómez-Bombarelli et al., 2018]…

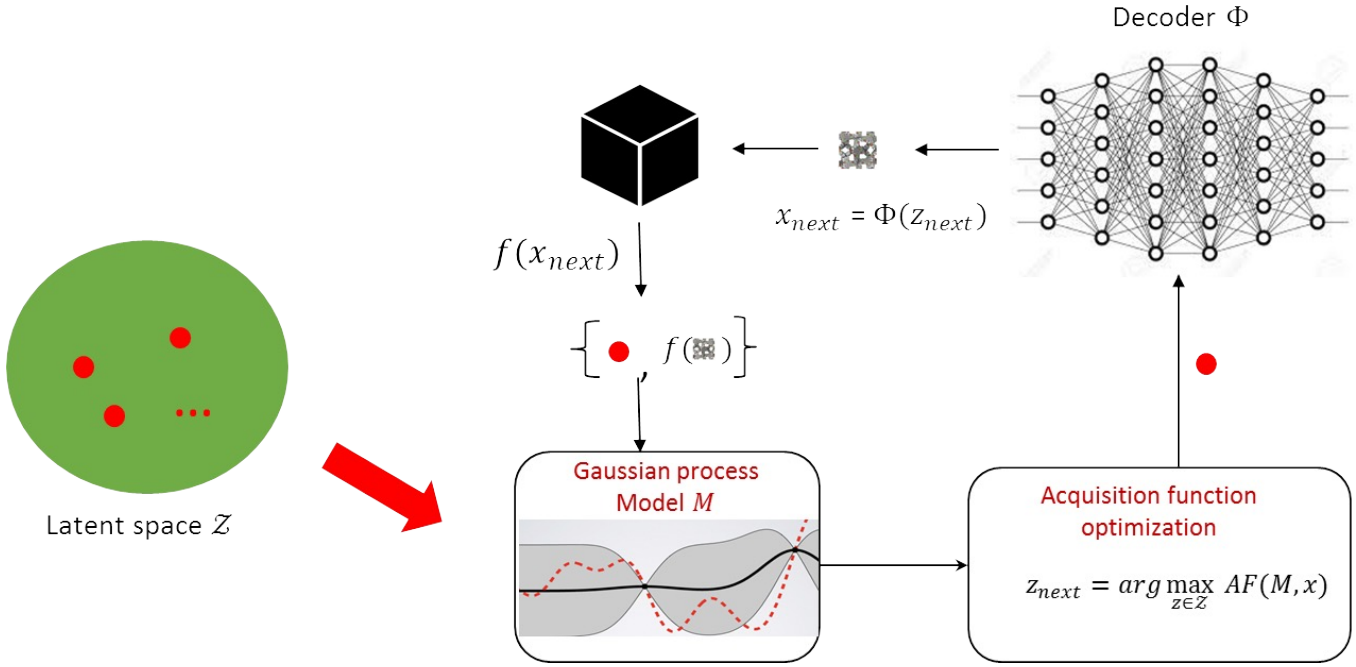
- BO in the learned latent space



Griffiths R.-R. and Hernández-Lobato J. M.: Constrained Bayesian optimization for Automatic Chemical Design Using Variational Autoencoders, Chemical Science, 2019

Reduction to Continuous BO [Gómez-Bombarelli et al., 2018]…

- BO in the learned latent space



- Challenges
  - Doesn't (explicitly) incorporate information about decoded structures
  - Surrogate model may not generalize well for small data setting

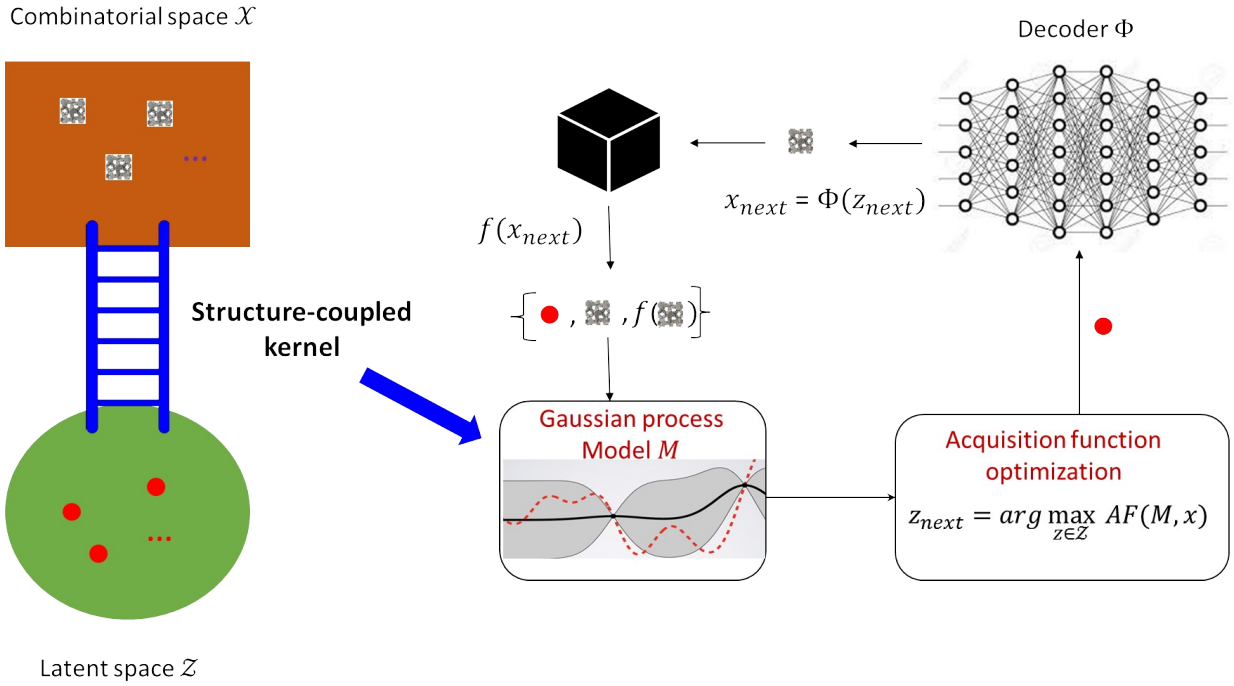
Improvements to Latent Space BO

- Weighted retraining [Tripp et al., 2020]
  - Periodically retrain the deep generative model
  - Assign importance weights to training data proportional to their objective function value

- Uncertainty-guided latent space BO [Notin et al., 2021]
  - Leverage the epistemic uncertainty of the decoder to guide the optimization process
  - No retraining of deep generative model is needed

Improvements to Latent Space BO

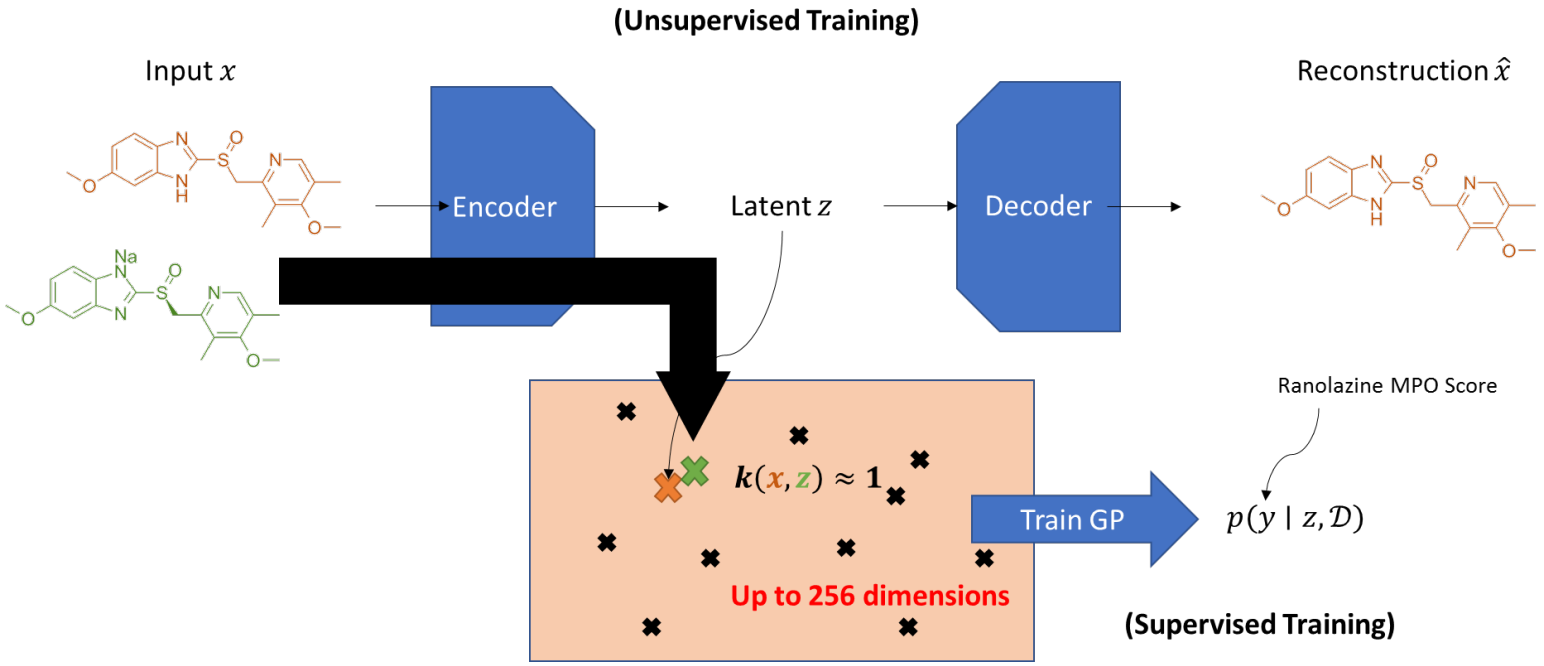
- LADDER algorithm [Deshwal and Doppa, 2021]
  - Incorporate domain knowledge via (hand-designed) structured kernels



- Key Idea
  - Extrapolate <u>eigenfunctions</u> of the latent space kernel matrix **L** with basis functions from the structured kernel k

Improvements to Latent Space BO

- LOL-BO [Maus et al., 2022]
  - Idea 1: train GP and VAE jointly
  - Idea 2: adapt high-dimensional BO methods in continuous spaces



Outline of the Tutorial

- Quick Overview of the BO Framework and GPs
- Summary of advances in GPs and Acquisition Functions
- Bayesian Optimization over Discrete/Hybrid Spaces

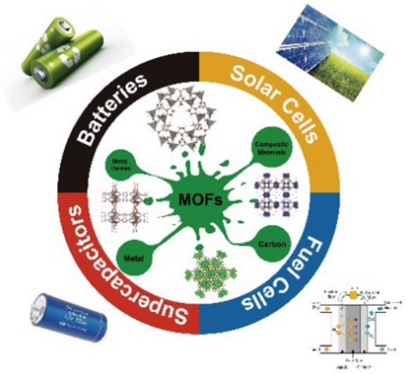
- High-Dimensional Bayesian Optimization
- BoTorch Hands-on Demonstration

- Causal Bayesian Optimization
- Summary and Outstanding Challenges in BO

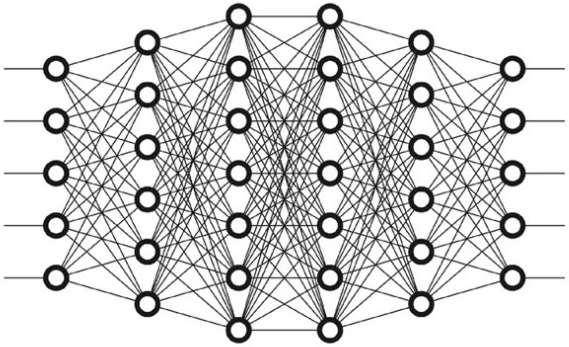
BO Over Hybrid Spaces: The Problem

- **Goal:** find optimized hybrid structures via expensive experiments
  - x = mixture of x_d (discrete) and x_c (continuous) variables

Microbiome design



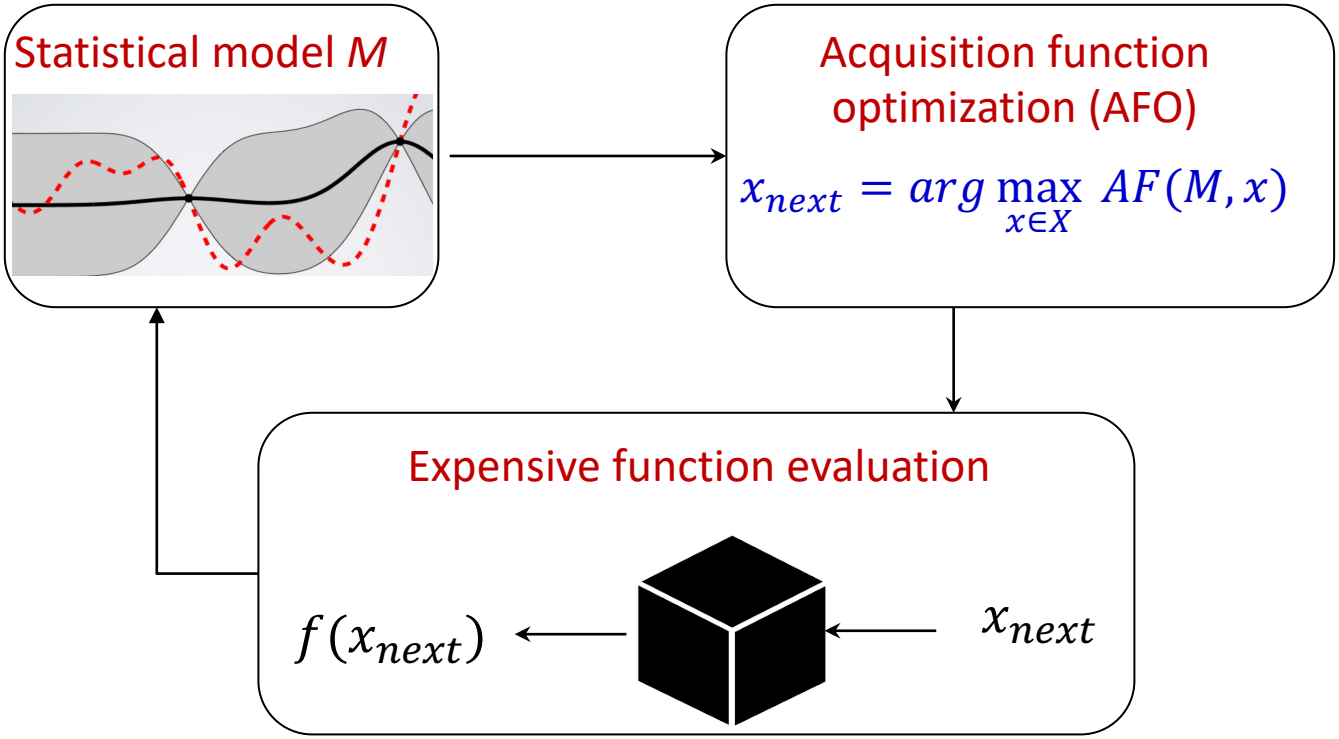
Material design



Hyper-parameter tuning / Auto ML

- Many other science, engineering, industrial applications

Hybrid BO: Technical Challenges



- Effective modeling over hybrid structures (capture complex interactions among discrete and continuous variables)
- Solving hard optimization problem over hybrid spaces for AFO

Surrogate Models: MiVaBO [Daxberger et al., 2019]

- Linear surrogate model over binary structures
  - $f(x \in X) = \theta^T . \phi(x)$
  - $\phi(x)$ consists of continuous (random Fourier features), discrete (BOCS representation for binary variables), and mixed (products of all pairwise combinations) features

May not be sufficient to capture desired dependencies

Surrogate Models: SMAC [Hutter et al., 2010, 2011]

- Random forest as surrogate model
  - works naturally for categorical and mixed variables
  - Prediction/Uncertainty (= empirical mean/variance over trees)

Uncertainty estimates can be poor

- Improvements for better uncertainty estimates
  - Bagging with oversampling [Kim and Choi, 2022]

Surrogate Models: GP with Sum-and-Product Kernels

[Ru et al., 2021; Wan et al., 2021]

$$k(z, z') = (1-\lambda) * (k_d(x_d, x_d') + k_c(x_c, x_c')) + \lambda * k_d(x_d, x_d') k_c(x_c, x_c')$$

Sum

Product

- CoCaBO [Ru et al., 2021]
  - $k_d(x_d, x_d') = \sum(x_d^i == x_d^i)$ // Hamming similarity
  - k_c is Matern Kernel
- Casmopolitan [Wan et al., 2021]
  - $k_d(x_d, x_d') = \exp(\sum l_i(x_d^i == x_d^i))$ // exponentiated Hamming
  - k_c is Matern Kernel

Surrogate Models: GP w/ Diffusion Kernels [Deshwal et al., 2021]

- GP surrogate model with additive diffusion kernels
  - Exploits the general recipe of additive kernels [Duvenaud et al., 2011]
  - Instantiation w/ discrete & continuous diffusion kernels
  - Bayesian treatment of the hyper-parameters

$$\mathcal{K}_{HYB} = \sum_{p=1}^{m+n} \left(\theta_p^2 \sum_{i_1, \cdots, i_p} \prod_{d=1}^{p} k_{i_d}(x_{i_d}, x'_{i_d}) \right)$$

Surrogate Models: GP w/ Frequency Modulation Kernels [Oh et al., 2021]

- Key idea: Generalize the COMBO kernel [Oh et al., 2019] by parametrizing via a function of continuous variables

$$\mathrm{K} = exp(-\beta L(G))$$

$$\mathrm{K} = U^T exp(-\beta\Sigma)U$$

$$\mathrm{K} = U^T f(\Sigma, X_c, X_{c\prime})U$$

Remember the COMBO kernel

- Requirement on f for K to be a positive definite kernel
  - f should be positive definite w.r.t $X_c, X_{c\prime}$

Acquisition Functions

- Thompson Sampling
  - MiVaBO: linear surrogate model $f(x \in X) = \theta^T . \phi(x)$
- Expected Improvement
  - SMAC: random forest model
  - HyBO: GP with hybrid diffusion kernel
  - BO-FM: GP with frequency modulated kernel
  - CoCaBO: GP with sum and product kernel over subspaces

Acquisition Function Optimization

- MiVaBO, HyBO, BO-FM: Alternating search
  - Step 1: Search over continuous sub-space
  - Step 2: Search over discrete sub-space using output of Step #1
  - Repeat if needed
- SMAC
  - Hand-designed local search with restarts
- CoCaBO
  - Continuous variables via gradient-based search
  - Categorical variables via multi-armed bandit algorithms
- Casmopolitan
  - Trust region-based AFO to scale to high-dimensional spaces