Bayesian Design of Experiments via Posterior Sampling

Applications in Hyper-parameter tuning, Astrophysics, and Materials Science

Kirthevasan Kandasamy
Carnegie Mellon University & ExperiML

June 1, 2018
Lawrence Berkeley National Lab, CA

slides: www.cs.cmu.edu/~kkandasa
Choose an experiment $x \in \mathcal{X}$.

Obtain the result (observation) $Y_x \sim P(y|x, \theta_\star)$. $\theta_\star$ (unknown) completely specifies the system.
Design of Experiments

Choose an experiment $x \in \mathcal{X}$.

Obtain the result (observation) $Y_x \sim \mathbb{P}(y|x, \theta_*)$.  
$\theta_*$ (unknown) completely specifies the system.

Repeat in an adaptive sequence to collect data 
$D_t = \{(x_t, Y_{x_t})\}_{t=1}^{n}$.

Typically some “goal/objective” in mind.
Black-box Optimisation: Model Selection

\[ \mathbf{x} \rightarrow \text{Neural Network} \rightarrow Y_{\mathbf{x}} \]

- Train NN using given hyper-params
- Compute accuracy on validation set

\( \mathbf{x} \): hyper-parameter values

\( \mathbf{Y} \): cross validation accuracy

\( \mathbf{Y} = [0, 1] \)

**Goal:** Find hyper-parameters with highest CV accuracy.
Black-box Optimisation: Architecture Search

$x \rightarrow$ Neural Network $\rightarrow Y_x$

$X$: Space of NN Architectures

- Train NN using given NN architecture
- Compute accuracy on validation set

$Y = [0, 1]$

**Goal:** Find NN architecture with highest CV accuracy.
Multi-objective Optimisation: Drug Discovery

\[ \chi \xrightarrow{\text{In Vitro/ In Vivo Test}} Y_x \]

\( \chi \): candidate drugs

\( Y_x \): specificity, solubility, potency, toxicity

\( Y \subset \mathbb{R}^K \)

**Goal:** Find drug with “good value” on all objectives.
**Goal:** Estimate relation between solution and interfacial tension.
Multiple Goals: Materials Science

**Goal:** Estimate relation between electrolyte solution and viscosity, while simultaneously optimising conductivity.
Goal: Estimate posterior for cosmological parameters given data.
**Goal:** Identify changes in crystal structure in an alloy.
Outline

- Part I: Preliminaries (Black-box Optimisation)
  1. Bayesian Models
  2. Black-box Optimisation via Thompson Sampling

- Part II: DOE via posterior sampling

- Part III: Scaling up DOE (back to Black-box Optimisation)
  1. Parallelising experiments
  2. Multi-fidelity experimentation
  3. High dimensional input spaces
  4. Beyond Euclidean/categorical domains

- Part IV: ExperiML & Collaborations with LBL
Outline

- Part I: Preliminaries (Black-box Optimisation)
  1. Bayesian Models
  2. Black-box Optimisation via Thompson Sampling

- Part II: DOE via posterior sampling

- Part III: Scaling up DOE (back to Black-box Optimisation)
  1. Parallelising experiments
  2. Multi-fidelity experimentation
  3. High dimensional input spaces
  4. Beyond Euclidean/categorical domains

- Part IV: ExperiML & Collaborations with LBL
Black-box Optimisation

\[ f : \mathcal{X} \rightarrow \mathbb{R} \text{ is an expensive, black-box function, accessible only via noisy evaluations.} \]
Black-box Optimisation

\[ f : \mathcal{X} \rightarrow \mathbb{R} \] is an expensive, black-box function, accessible only via noisy evaluations.
Black-box Optimisation

$f : \mathcal{X} \rightarrow \mathbb{R}$ is an expensive, black-box function, accessible only via noisy evaluations. Let $x^* = \arg\max_x f(x)$. 
Bayesian Models for $f$

Functions with no observations

$f(x)$

$x$
Bayesian Models for $f$

Prior $\mathbb{P}(\theta_*)$
Bayesian Models for $f$

Observations

$f(x)$

$x$
Bayesian Models for $f$

Posterior given observations $\mathbb{P}(\theta_\star|D_t)$
Bayesian Models for $f$

Posterior given observations $\mathbb{P}(\theta_\star|D_t)$
Thompson Sampling for Black-box Optimisation

(Thompson, 1933)

1) Construct posterior $P(\theta^\star | D_t)$.
2) Draw sample $g$ from posterior.
3) Choose $x_t = \arg\max_x g(x)$.
4) Evaluate $f$ at $x_t$. 

$f(x)$
Thompson Sampling for Black-box Optimisation

(Thompson, 1933)

\[
\begin{align*}
1) & \text{ Construct posterior } P(\theta^\ast|D_t). \\
\end{align*}
\]
Thompson Sampling for Black-box Optimisation

(Thompson, 1933)

1) Construct posterior $\mathbb{P}(\theta_\star | D_t)$. 2) Draw sample $g$ from posterior.
Thompson Sampling for Black-box Optimisation

(Thompson, 1933)

1) Construct posterior $\mathbb{P}(\theta_\star | D_t)$.  
2) Draw sample $g$ from posterior.  
3) Choose $x_t = \text{argmax}_x g(x)$. 

$f(x)$

$x_t$
Thompson Sampling for Black-box Optimisation

(Thompson, 1933)

1) Construct posterior $\mathbb{P}(\theta_*|D_t)$. 2) Draw sample $g$ from posterior. 3) Choose $x_t = \arg\max_x g(x)$. 4) Evaluate $f$ at $x_t$. 
Thompson Sampling for Black-box Optimisation

(Thompson, 1933)
Thompson Sampling for Black-box Optimisation

(Thompson, 1933)
Thompson Sampling for Black-box Optimisation

\( f(x) \)

\( t = 2 \)
Thompson Sampling for Black-box Optimisation

(Thompson, 1933)

\[ f(x) \]

\[ t = 3 \]
Thompson Sampling for Black-box Optimisation

(Thompson, 1933)
Thompson Sampling for Black-box Optimisation

(Thompson, 1933)
Thompson Sampling for Black-box Optimisation

\[ f(x) \]

\( t = 6 \)
Thompson Sampling for Black-box Optimisation

(Thompson, 1933)
Thompson Sampling for Black-box Optimisation

(Thompson, 1933)
Thompson Sampling for Black-box Optimisation

(Thompson, 1933)

\[ f(x) \]

\[ t = 25 \]
Black-box Optimisation in the Bayesian Paradigm

Other criteria for selecting $x_t$:

- Upper Confidence Bounds (Srinivas et al. 2010)
- Expected improvement (Jones et al. 1998)
- Probability of improvement (Kushner et al. 1964)
- ... and a few more.

Bayesian models for $f$:

- Gaussian Processes (most popular)
- Neural networks (Snoek et al. 2015)
- Random forests (Hutter 2009)

Off-the-shelf models: general, but can be inefficient.
Outline

- Part I: Preliminaries (Black-box Optimisation)
  1. Bayesian Models
  2. Black-box Optimisation via Thompson Sampling

- Part II: DOE via posterior sampling

- Part III: Scaling up DOE (back to Black-box Optimisation)
  1. Parallelising experiments
  2. Multi-fidelity experimentation
  3. High dimensional input spaces
  4. Beyond Euclidean/categorical domains

- Part IV: ExperiML & Collaborations with LBL
Design of Experiments

![Diagram of an experiment flowchart]

- Choose experiment $X \in \mathcal{X}$, obtain result $Y_X \sim \mathbb{P}(y|X, \theta_{\star})$.
- $\theta_{\star}$ represents everything that is unknown about the system.
- Repeat in a sequence to collect data $D_t = \{(X_j, Y_{X_j})\}_{j=1}^t$.
- Typically some “goal/objective” in mind.
Design of Experiments

Choose experiment $X \in \mathcal{X}$, obtain result $Y_{X} \sim \mathbb{P}(y|X, \theta_{\star})$.

- $\theta_{\star}$ represents everything that is unknown about the system.
- Repeat in a sequence to collect data $D_{t} = \{(X_{j}, Y_{X_{j}})\}_{j=1}^{t}$.
- Typically some “goal/objective” in mind.

Desiderata for a General Framework:

- Flexibility to capture custom/complex relations for $X \rightarrow Y_{X}$.
  - Incorporate domain expertise into models.
- Ability to achieve any desired goal.
Formalism for “goal-oriented” DOE

System:

- A true parameter $\theta_\star \in \Theta$ that completely specifies the system.
- $\Theta \leftarrow$ a parameter space.

Goal:

- Collect data $D_n = \{(x_t, y_{x_t})\}_{t=1}^n$ to achieve a goal specified by a penalty function $\lambda(\theta, D_n)$.

We wish to achieve small $\lambda(\theta_\star, D_n)$ after $n$ experiments.

Bayesian Models:

- A prior for $\theta_\star$: $P(\theta_\star)$.
- A discriminative model for observations $y|x, \theta$: $P(y|x, \theta)$. 
Formalism for “goal-oriented” DOE

**System:**
- A true parameter $\theta_* \in \Theta$ that completely specifies the system.
- $\Theta \leftarrow$ a parameter space.

**Goal:**
- Collect data $D_n = \{(x_t, y_{x_t})\}_{t=1}^n$ to achieve a goal specified by a penalty function $\lambda(\theta, D_n)$.
- We wish to achieve small $\lambda(\theta_*, D_n)$ after $n$ experiments.
Formalism for “goal-oriented” DOE

System:
- A true parameter $\theta^*_x \in \Theta$ that completely specifies the system.
- $\Theta \leftarrow$ a parameter space.

Goal:
- Collect data $D_n = \{(x_t, y_{x_t})\}_{t=1}^n$ to achieve a goal specified by a penalty function $\lambda(\theta, D_n)$.
- We wish to achieve small $\lambda(\theta^*_x, D_n)$ after $n$ experiments.

Bayesian Models:
- A prior for $\theta^*_x$: $\mathbb{P}(\theta^*_x)$.
- A discriminative model for observations $y|x, \theta$: $\mathbb{P}(y|x, \theta)$. 
Incorporating Domain Expertise via Bayesian Models
An example in Electrolyte Design

Three control variables:
\( Q \): EC-EMC fraction, \( S \): molarity of salt LiPF\(_6\), \( T \): temperature.
Incorporating Domain Expertise via Bayesian Models
An example in Electrolyte Design

Three control variables:
Q: EC-EMC fraction, S: molarity of salt LiPF₆, T: temperature.

\[ f_{\text{vis}}(Q, S, T) = \exp(-aT + bS) \cdot g_{\text{vis}}(Q). \]  
(Reynolds model)

\[ f_{\text{dissol}}(Q, S, T) = cT \cdot \frac{1}{1 + \exp(dS)} \cdot g_{\text{dissolv}}(Q). \]
Incorporating Domain Expertise via Bayesian Models
An example in Electrolyte Design

\[ f_{\text{vis}}(Q, S, T) = \exp(-aT + bS) \cdot g_{\text{vis}}(Q). \]  
\[ f_{\text{dissol}}(Q, S, T) = cT \cdot \frac{1}{1 + \exp(dS)} \cdot g_{\text{dissolv}}(Q). \]

Observations:  
\[ y_{\text{visc}} | x, \theta \sim \mathcal{N}(f_{\text{vis}}(Q, S, T), \eta^2). \]  
\[ y_{\text{dissol}} | x, \theta \sim \mathcal{N}(f_{\text{dissol}}(Q, S, T), \sigma^2). \]
Incorporating Domain Expertise via Bayesian Models
An example in Electrolyte Design

Three control variables:
Q: EC-EMC fraction, S: molarity of salt LiPF$_6$, T: temperature.

\[ f_{\text{vis}}(Q, S, T) = \exp(-aT + bS) \cdot g_{\text{vis}}(Q). \]  
(Reynolds model)

\[ f_{\text{dissol}}(Q, S, T) = cT \cdot \frac{1}{1+\exp(dS)} \cdot g_{\text{dissolv}}(Q). \]

Observations:  
\[ y_{\text{vis}}|\mathbf{x}, \theta \sim \mathcal{N}(f_{\text{vis}}(Q, S, T), \eta^2). \]
\[ y_{\text{dissol}}|\mathbf{x}, \theta \sim \mathcal{N}(f_{\text{dissol}}(Q, S, T), \sigma^2). \]

Unknown parameters:  
\[ \theta = (a, b, c, d, g_{\text{vis}}, g_{\text{dissolv}}) \in \Theta. \]
True parameter  
\[ \theta_* = (a_*, b_*, c_*, d_*, g_{\text{vis}*}, g_{\text{dissolv}*}). \]
Use prior \[ \mathbb{P}(\theta_*) \] to specify plausible values for \[ \theta_* \].
Specifying the goal via a Penalty Function

Practitioner specifies goal of the experiment via $\lambda(\theta, D_n)$.
Specifying the goal via a Penalty Function

Practitioner specifies goal of the experiment via $\lambda(\theta, D_n)$.

**Example 1: Optimisation**

$$\lambda(\theta, D_n) = \max_{x \in \mathcal{X}} f_\theta(x) - \max_{t \leq n} f_\theta(x_t)$$

**Example 2: Active Learning**

Estimate some parameter $\tau^\star = \tau(\theta^\star)$ of the system.

$$\lambda(\theta, D_n) = \|\tau(\theta) - \hat{\tau}(D_n)\|_2^2$$

$\hat{\tau} \leftarrow$ some prespecified (e.g. maximum likelihood) estimator for $\tau$ using data.
Specifying the goal via a Penalty Function

Practitioner specifies goal of the experiment via $\lambda(\theta, D_n)$.

Example 1: Optimisation

$$\lambda(\theta, D_n) = \max_{x \in \mathcal{X}} f_\theta(x) - \max_{t \leq n} f_\theta(x_t)$$

Example 2: Active Learning

Estimate some parameter $\tau_\star = \tau(\theta_\star)$ of the system.

$$\lambda(\theta, D_n) = \|\tau(\theta) - \hat{\tau}(D_n)\|_2^2.$$ 

$\hat{\tau} \leftarrow$ some prespecified (e.g. maximum likelihood) estimator for $\tau$ using data.
Specifying the goal via a Penalty Function

Practitioner specifies goal of the experiment via $\lambda(\theta, D_n)$.

**Example 1: Optimisation**

$$\lambda(\theta, D_n) = \max_{x \in \mathcal{X}} f_\theta(x) - \max_{t \leq n} f_\theta(x_t)$$

**Example 2: Active Learning**

Estimate some parameter $\tau_\star = \tau(\theta_\star)$ of the system.

$$\lambda(\theta, D_n) = \|\tau(\theta) - \hat{\tau}(D_n)\|_2^2.$$ 

$\hat{\tau} \leftarrow$ some prespecified (e.g. maximum likelihood) estimator for $\tau$ using data.

Will look at more examples shortly.
MPS: Myopic Posterior Sampling for DOE

Expected look-ahead penalty at $x$ if $\theta$ was the true parameter and we have already collected data $D$:

$$\lambda^+(\theta, D, x) = \mathbb{E}_{Y_x \sim \mathbb{P}(Y|x, \theta)} \left[ \lambda(\theta, D \cup \{(x, Y_x)\}) \right].$$
MPS: Myopic Posterior Sampling for DOE

Expected look-ahead penalty at $x$ if $\theta$ was the true parameter and we have already collected data $D$:

$$\lambda^+(\theta, D, x) = \mathbb{E}_{Y_x \sim \mathbb{P}(Y|x, \theta)} \left[ \lambda(\theta, D \cup \{(x, Y_x)\}) \right].$$

Algorithm: MPS ($\pi_{PS}^M$)

- Set $D_0 \leftarrow$ initial data.
- For $t = 1, 2, \ldots$, do
  1. Sample $\theta \sim \mathbb{P}(\theta_x | D_{t-1})$.
  2. Choose $x_t = \text{argmin}_{x \in \mathcal{X}} \lambda^+(\theta, D_{t-1}, x)$.
  3. $y_{x_t} \leftarrow$ conduct experiment at $x_t$.
  4. Set $D_t \leftarrow D_{t-1} \cup \{(x_t, y_{x_t})\}$. 
MPS: Myopic Posterior Sampling for DOE

Expected look-ahead penalty at \( x \) if \( \theta \) was the true parameter and we have already collected data \( D \):

\[
\lambda^+(\theta, D, x) = \mathbb{E}_{Y_x \sim \mathbb{P}(Y|\theta)} \left[ \lambda(\theta, D \cup \{(x, Y_x)\}) \right].
\]

**Algorithm: MPS (\( \pi_{\text{PS}}^M \))**

- Set \( D_0 \leftarrow \) initial data.
- For \( t = 1, 2, \ldots \), do
  1. Sample \( \theta \sim \mathbb{P}(\theta_x|D_{t-1}) \).
  2. Choose \( x_t = \arg\min_{x \in X} \lambda^+(\theta, D_{t-1}, x) \).
  3. \( y_{x_t} \leftarrow \) conduct experiment at \( x_t \).
  4. Set \( D_t \leftarrow D_{t-1} \cup \{(x_t, y_{x_t})\} \).

**N.B:** When the goal is optimisation, this reduces to exactly Thompson sampling.
Experiment: Active Learning

ActiveSel: (Chaudhuri et al. 2015)
Experiment: Posterior Estimation in Astrophysics

Astrophysicist defines prior on Hubble constant, and dark matter fraction and dark energy fraction. Computer posterior distribution given Type Ia supernova data \( Q \). Likelihood computed using the Robertson-Walker metric.

\[
\lambda(\theta_*, D_n) = \| p(\tau(\theta_*)|Q) - \hat{p}(\tau(\theta_*)|Q) \|_2
\]

**GP-EVR:** (Kandasamy et al. IJCAI 2015)
An experiment measures solubility, viscosity and conductivity of an electrolyte design.

**Goal:** Optimise conductivity while learning solubility and viscosity.

\[
\lambda(\theta^*, D_n) = \|f_{\text{dissol}} - \hat{f}_{\text{dissol}}(D_n)\|^2 + \|f_{\text{vis}} - \hat{f}_{\text{vis}}(D_n)\|^2 + \\
\left(\max f_{\text{con}} - \max_{X_t, t \leq n} f_{\text{con}}(X_t)\right),
\]

Electrolyte Design

![Graph showing electrolyte design with axes labeled Number of Experiments (n) on the x-axis and \(\lambda(\theta^*, n)\) on the y-axis. The graph includes curves for RAND, GRID, \(\pi^M\), and \(\pi^*_M\).]
Theory

Theorem (Informal): Under certain assumptions on the problem, MPS does almost as well as the optimal algorithm that knows $\theta^\star$. We use ideas/conditions from

- Adaptive Submodularity
- Reinforcement Learning
- Bandits
Theory

It works!
Theory

It works!

**Theorem (Informal):** Under certain assumptions on the problem, MPS does almost as well as the optimal algorithm that knows $\theta_\star$. 
It works!

**Theorem (Informal):** Under certain assumptions on the problem, MPS does almost as well as the optimal algorithm that knows $\theta_\star$.

We use ideas/conditions from
- Adaptive Submodularity
- Re-inforcement Learning
- Bandits
Outline

- Part I: Preliminaries (Black-box Optimisation)
  1. Bayesian Models
  2. Black-box Optimisation via Thompson Sampling

- Part II: DOE via posterior sampling

- Part III: Scaling up DOE (back to Black-box Optimisation)
  1. Parallelising experiments
  2. Multi-fidelity experimentation
  3. High dimensional input spaces
  4. Beyond Euclidean/categorical domains

- Part IV: ExperiML & Collaborations with LBL
Part 3.1: Parallel Experiments

Sequential experiments with one worker

\[ \begin{array}{cccccc}
1 & 2 & 3 & 4 & \ldots \\
\end{array} \]

Time $\rightarrow$
Part 3.1: Parallel Experiments

Sequential experiments with one worker

Parallel experiments with $M$ workers (Asynchronous)
Part 3.1: Parallel Experiments

Sequential experiments with one worker

Parallel experiments with $M$ workers (Asynchronous)

Parallel experiments with $M$ workers (Synchronous)
Parallelised DOE via Posterior Sampling

Asynchronous:

At any given time,
1. \((x', y') \leftarrow \text{Wait for a worker to finish.}\)
2. Update posterior for \(\theta^*_\).
3. Draw a sample \(\theta \sim \mathbb{P}(\theta_*|D_t).\)
4. Re-deploy worker at \(\text{argmin } \lambda^+(\theta, D_t, x).\)
Parallelised DOE via Posterior Sampling

**Asynchronous:**

At any given time,

1. \((x', y') \leftarrow \text{Wait for a worker to finish.}\)
2. Update posterior for \(\theta_\star\).
3. Draw a sample \(\theta \sim \mathbb{P}(\theta_\star | D_t)\).
4. Re-deploy worker at \(\text{argmin} \lambda^+(\theta, D_t, x)\).

**Synchronous:**

At any given time,

1. \(\{(x'_m, y'_m)\}_{m=1}^M \leftarrow \text{Wait for all workers to finish.}\)
2. Update posterior for \(\theta_\star\).
3. Draw \(M\) samples \(\theta_m \sim \mathbb{P}(\theta_\star | D_t), \forall m\).
4. Re-deploy worker \(m\) at \(\text{argmin} \lambda^+(\theta_m, D_t, x)\).
Theory: parallel DOE via posterior sampling

**Conjecture:** For synchronous & asynchronous parallel DOE via posterior sampling

\[ \mathbb{E}[\lambda(\theta^*, D_n)] \lesssim \frac{M \log(M)}{n} + \text{sequential result} \]

**Theorem:** For parallelised Thompson sampling (Black-box Optimisation) (Kandasamy et al. AISTATS 2018)

\[ \mathbb{E}[f(x^*) - \max_{t \leq n} f(x_t)] \lesssim \frac{M \log(M)}{n} + \frac{C}{\sqrt{n}} \]

Can also quantify difference between synchronous and asynchronous settings. (Kandasamy et al. AISTATS 2018)

- If evaluation times are the same, synchronous is slightly better.
- When there is high variability in evaluation times, asynchronous is much better than synchronous.
Outline

- Part I: Preliminaries (Black-box Optimisation)
  1. Bayesian Models
  2. Black-box Optimisation via Thompson Sampling

- Part II: DOE via posterior sampling

- Part III: Scaling up DOE (back to Black-box Optimisation)
  1. Parallelising experiments
  2. Multi-fidelity experimentation
  3. High dimensional input spaces
  4. Beyond Euclidean/categorical domains

- Part IV: ExperiML & Collaborations with LBL
Part 2.2: Multi-fidelity Experiments

Motivating question:
What if we have cheap approximations to an experimentation?

1. Hyper-parameter tuning: Train & validate with a subset of the data, and/or early stopping before convergence.
   E.g. Bandwidth ($\ell$) selection in kernel density estimation.
Part 2.2: Multi-fidelity Experiments

**Motivating question:**
What if we have cheap approximations to an experimentation?

1. **Hyper-parameter tuning:** Train & validate with a subset of the data, and/or early stopping before convergence.
   E.g. Bandwidth \((\ell)\) selection in kernel density estimation.

2. **Computational astrophysics:** cosmological simulations and numerical computations with less granularity.

3. In many applications: real world experiment vs simulation.
Multi-fidelity Hyper-parameter tuning

E.g. Train an ML model with $N_*$ data and $T_*$ iterations.
- But use $N < N_*$ data and $T < T_*$ iterations to approximate cross validation performance at $(N_*, T_*)$. 

![Diagram showing CV Log Likelihood vs log(h) with curves for n = 3000 and n = 300](image)
Multi-fidelity Hyper-parameter tuning

E.g. Train an ML model with \( N_\bullet \) data and \( T_\bullet \) iterations.
- But use \( N < N_\bullet \) data and \( T < T_\bullet \) iterations to approximate cross validation performance at \((N_\bullet, T_\bullet)\).

Approximations from a continuous 2D “fidelity space” \((N, T)\).
Multi-fidelity Hyper-parameter tuning

E.g. Train an ML model with $N^*$ data and $T^*$ iterations.
- But use $N < N^*$ data and $T < T^*$ iterations to approximate cross validation performance at $(N^*, T^*)$.

Approximations from a continuous 2D “fidelity space” $(N, T)$.

Multi-fidelity Black-box optimisation using GPs:
(Kandasamy et al. NIPS 2016a&b, Kandasamy et al. ICML 2017, Sen, Kandasamy et al. ICML 2018)
A fidelity space $\mathcal{Z}$ and domain $\mathcal{X}$

$\mathcal{Z} \leftarrow$ all $(N, T)$ values.

$\mathcal{X} \leftarrow$ all hyper-parameter values.
Multi-fidelity Optimisation

(Kandasamy et al. ICML 2017)

A fidelity space $\mathcal{Z}$ and domain $\mathcal{X}$

$\mathcal{Z} \leftarrow$ all $(N, T)$ values.

$\mathcal{X} \leftarrow$ all hyper-parameter values.

$g : \mathcal{Z} \times \mathcal{X} \rightarrow \mathbb{R}$.

$g([N, T], x) \leftarrow$ cv accuracy when training with $N$ data for $T$ iterations at hyper-parameter $x$. 
Multi-fidelity Optimisation

(Kandasamy et al. ICML 2017)

A fidelity space $\mathcal{Z}$ and domain $\mathcal{X}$

$\mathcal{Z} \leftarrow$ all $(N, T)$ values.

$\mathcal{X} \leftarrow$ all hyper-parameter values.

$g : \mathcal{Z} \times \mathcal{X} \rightarrow \mathbb{R}$.  

g([N, T], x) \leftarrow \text{cv accuracy when training with } N \text{ data for } T \text{ iterations at hyper-parameter } x.$

Denote $f(x) = g(z_\bullet, x)$ where $z_\bullet \in \mathcal{Z}$.  

$z_\bullet = [N_\bullet, T_\bullet]$.  

Multi-fidelity Optimisation (Kandasamy et al. ICML 2017)

A fidelity space $\mathcal{Z}$ and domain $\mathcal{X}$

$\mathcal{Z} \leftarrow$ all $(N, T)$ values.

$\mathcal{X} \leftarrow$ all hyper-parameter values.

$g : \mathcal{Z} \times \mathcal{X} \rightarrow \mathbb{R}$.

$g([N, T], x) \leftarrow$ cv accuracy when training with $N$ data for $T$ iterations at hyper-parameter $x$.

Denote $f(x) = g(z_\bullet, x)$ where $z_\bullet \in \mathcal{Z}$.

$z_\bullet = [N_\bullet, T_\bullet]$.

End Goal: Find $x_\star = \text{argmax}_x f(x)$.

Therefore, $\lambda(f, D_t) = f(x_\star) - \max_{t: z_t = z_\bullet} f(x_t)$. 

Multi-fidelity Optimisation

(Kandasamy et al. ICML 2017)

A fidelity space $\mathcal{Z}$ and domain $\mathcal{X}$

$\mathcal{Z} \leftarrow$ all $(N, T)$ values.

$\mathcal{X} \leftarrow$ all hyper-parameter values.

$g : \mathcal{Z} \times \mathcal{X} \rightarrow \mathbb{R}$.

$g([N, T], x) \leftarrow$ cv accuracy when training with $N$ data for $T$ iterations at hyper-parameter $x$.

Denote $f(x) = g(z_\bullet, x)$ where $z_\bullet \in \mathcal{Z}$.

$z_\bullet = [N_\bullet, T_\bullet]$.

**End Goal:** Find $x_* = \text{argmax}_x f(x)$.

Therefore,

$$\lambda(f, D_t) = f(x_*) - \max_{t: z_t = z_\bullet} f(x_t).$$

A cost function, $\gamma : \mathcal{Z} \rightarrow \mathbb{R}_+$. 

$\gamma(z) = \gamma(N, T) = \mathcal{O}(N^2 T)$ (say).
Algorithms

- Finite number of approximations: MF-GP-UCB
  (Kandasamy et al. NIPS 2016b)
- Continuous approximations: BOCA
  (Kandasamy et al. ICML 2017)

**Key intuition in both algorithms**

- By default, will evaluate at the low (cheap) fidelities.
- Proceed to higher (expensive) fidelities when there is a good information to cost trade-off.
**(Algorithm: BOCA)**

(Kandasamy et al. ICML 2017)

Model $g \sim \mathcal{GP}(0, \kappa)$ and compute posterior $\mathcal{GP}$:

- **mean** $\mu_{t-1} : \mathcal{Z} \times \mathcal{X} \to \mathbb{R}$
- **std-dev** $\sigma_{t-1} : \mathcal{Z} \times \mathcal{X} \to \mathbb{R}_+$

*(1) $x_t \leftarrow$ maximise upper confidence bound for $f(x) = g(z_\bullet, x)$. *

$x_t = \arg\max_{x \in \mathcal{X}} \mu_{t-1}(z_\bullet, x) + \beta_t^{1/2} \sigma_{t-1}(z_\bullet, x)$
Algorithm: BOCA

(Kandasamy et al. ICML 2017)

Model \( g \sim \mathcal{GP}(0, \kappa) \) and compute posterior \( \mathcal{GP} \):

- mean \( \mu_{t-1} : \mathcal{Z} \times \mathcal{X} \rightarrow \mathbb{R} \)
- std-dev \( \sigma_{t-1} : \mathcal{Z} \times \mathcal{X} \rightarrow \mathbb{R}_+ \)

1. \( x_t \leftarrow \text{maximise upper confidence bound for } f(x) = g(z_\bullet, x) \).

   \[
   x_t = \text{argmax}_{x \in \mathcal{X}} \mu_{t-1}(z_\bullet, x) + \beta_t^{1/2} \sigma_{t-1}(z_\bullet, x)
   \]

2. \( \mathcal{Z}_t \approx \{ z_\bullet \} \cup \left\{ z : \sigma_{t-1}(z, x_t) \geq \gamma(z) = \left( \frac{\gamma(z)}{\gamma(z_\bullet)} \right)^q \xi(z) \right\} \)

3. \( z_t = \text{argmin}_{z \in \mathcal{Z}_t} \gamma(z) \) (cheapest \( z \) in \( \mathcal{Z}_t \))
Theorem: (Informal) (Kandasamy et al. ICML 2017) BOCA does better, i.e. achieves better Simple regret, than GP-UCB. The improvements are better in the "good" setting when compared to the "bad" setting.
Theoretical Results for BOCA

Theorem: (Informal) (Kandasamy et al. ICML 2017)
BOCA does better, i.e. achieves better Simple regret, than GP-UCB. The improvements are better in the “good” setting when compared to the “bad” setting.
Experiment: SVM with 20 News Groups

Tune two hyper-parameters for the SVM. Dataset has $N_\bullet = 15K$ data and use $T_\bullet = 100$ iterations. But can choose $N \in [5K, 15K]$ or $T \in [20, 100]$ (2D fidelity space).
Experiment: SVM with 20 News Groups

Tune two hyper-parameters for the SVM. Dataset has $N = 15K$ data and use $T = 100$ iterations. But can choose $N \in [5K, 15K]$ or $T \in [20, 100]$ (2D fidelity space).
**Experiment:** Cosmological inference on Type-1a supernovae data

Estimate Hubble constant, dark matter fraction & dark energy fraction by maximising likelihood on $N_\bullet = 192$ data.

Requires numerical integration on a grid of size $G_\bullet = 10^6$.

Approximate with $N \in [50, 192]$ or $G \in [10^2, 10^6]$ (2D fidelity space).
Experiment: Cosmological inference on Type-1a supernovae data

Estimate Hubble constant, dark matter fraction & dark energy fraction by maximising likelihood on $N_\star = 192$ data.
Requires numerical integration on a grid of size $G_\star = 10^6$.
Approximate with $N \in [50, 192]$ or $G \in [10^2, 10^6]$ (2D fidelity space).
Outline

► Part I: Preliminaries (Black-box Optimisation)
  1. Bayesian Models
  2. Black-box Optimisation via Thompson Sampling

► Part II: DOE via posterior sampling

► Part III: Scaling up DOE (back to Black-box Optimisation)
  1. Parallelising experiments
  2. Multi-fidelity experimentation
  3. High dimensional input spaces
  4. Beyond Euclidean/categorical domains

► Part IV: ExperiML & Collaborations with LBL
Part 3.3: Optimisation in High Dimensional Input Spaces

E.g. Tuning a machine learning model with several hyper-parameters
Part 3.3: Optimisation in High Dimensional Input Spaces

E.g. Tuning a machine learning model with several hyper-parameters

At each time step
Part 3.3: Optimisation in High Dimensional Input Spaces

E.g. Tuning a machine learning model with several hyper-parameters

At each time step

1. **Statistical Difficulty:** estimating a high dimensional GP.

2. **Computational Difficulty:** maximising a high dimensional acquisition (e.g. sample or UCB) \( \varphi_t \).
Additive Models for High Dimensional BO
(Kandasamy et al. ICML 2015)

**Structural assumption:**

\[ f(x) = f^{(1)}(x^{(1)}) + f^{(2)}(x^{(2)}) + \ldots + f^{(M)}(x^{(M)}). \]

\( x^{(j)} \in \mathcal{X}^{(j)} = [0, 1]^p, \quad p \ll d, \quad x^{(i)} \cap x^{(j)} = \emptyset. \)
Structural assumption:

\[ f(x) = f^{(1)}(x^{(1)}) + f^{(2)}(x^{(2)}) + \ldots + f^{(M)}(x^{(M)}). \]

\( x^{(j)} \in \mathcal{X}^{(j)} = [0, 1]^p, \quad p \ll d, \quad x^{(i)} \cap x^{(j)} = \emptyset. \)

E.g. \( f(x_{\{1,\ldots,10\}}) = f^{(1)}(x_{\{1,3,9\}}) + f^{(2)}(x_{\{2,4,8\}}) + f^{(3)}(x_{\{5,6,10\}}). \)

Call \( \{\mathcal{X}^{(j)}\}_{j=1}^M = \{(1, 3, 9), (2, 4, 8), (5, 6, 10)\} \) the “decomposition”. 
Additive Models for High Dimensional BO

(Kandasamy et al. ICML 2015)

**Structural assumption:**

$$f(x) = f^{(1)}(x^{(1)}) + f^{(2)}(x^{(2)}) + \ldots + f^{(M)}(x^{(M)}).$$

$$x^{(j)} \in \mathcal{X}^{(j)} = [0, 1]^p, \quad p \ll d, \quad x^{(i)} \cap x^{(j)} = \emptyset.$$

E.g.  
$$f(x_{\{1,\ldots,10\}}) = f^{(1)}(x_{\{1,3,9\}}) + f^{(2)}(x_{\{2,4,8\}}) + f^{(3)}(x_{\{5,6,10\}}).$$

Call $\{\mathcal{X}^{(j)}_{j=1}^M\} = \{(1, 3, 9), (2, 4, 8), (5, 6, 10)\}$ the “decomposition”.

**Advantages:**

- Statistical: Better bias-variance trade-offs in high dimensions.
- Computational: Easy to maximise acquisition and choose $x_t$. 
Upper Confidence Bound:

\[ \varphi_t(x) = \sum_{j=1}^{M} \mu_{t-1}^{(j)}(x^{(j)}) + \beta_t^{1/2} \sigma_{t-1}^{(j)}(x^{(j)}). \]

Maximise each \( \tilde{\varphi}_t^{(j)}(x^{(j)}) \) separately.

Requires only \( \mathcal{O}(\text{poly}(d)\epsilon^{-p}) \) effort (vs \( \mathcal{O}(\epsilon^{-d}) \) for GP-UCB).
Additive models can still be useful in non-additive settings

► Additive models common in high dimensional regression. E.g.: Backfitting, MARS, COSSO, RODEO, SpAM etc. (Friedman '91, Lin et al. '06, Lafferty et al '05, Ravikumar et al. '09)

► Additive models are statistically simpler $\implies$ worse bias, but much better variance in low sample regime.

► In bandit applications queries are expensive. So we usually cannot afford many queries.

► Observation: Add-GP-UCB does well even when $f$ is not additive.
  ▶ Better bias/ variance trade-off in estimating the GP.
  ▶ Easy to maximise upper confidence bound.
Experiment: Viola & Jones Face Detection

A cascade of 22 weak classifiers. Image classified negative if the score $<\text{threshold}$ at any stage.
Experiment: Viola & Jones Face Detection

A cascade of 22 weak classifiers.
Image classified negative if the score $< \text{threshold}$ at any stage.

In the paper we go up to $> 100$ dimensions.
Outline

- Part I: Preliminaries (Black-box Optimisation)
  1. Bayesian Models
  2. Black-box Optimisation via Thompson Sampling

- Part II: DOE via posterior sampling

- Part III: Scaling up DOE (back to Black-box Optimisation)
  1. Parallelising experiments
  2. Multi-fidelity experimentation
  3. High dimensional input spaces
  4. Beyond Euclidean/categorical domains

- Part IV: ExperiML & Collaborations with LBL
Part 3.4: Tuning Neural Network Architectures

Feedforward network
Part 3.4: Tuning Neural Network Architectures

Feedforward network

GoogLeNet
(Szegedy et al. 2015)
Part 3.4: Tuning Neural Network Architectures

Feedforward network

GoogLeNet
(Szegedy et al. 2015)

ResNet
(He et al. 2016)
Part 3.4: Tuning Neural Network Architectures

Feedforward network

GoogLeNet (Szegedy et al. 2015)

ResNet (He et al. 2016)

DenseNet (Huang et al. 2017)
Part 3.4: Tuning Neural Network Architectures

At each time step

![Diagram 1]

![Diagram 2]
Main challenges
▶ Optimise
Define a distance between neural network architectures.
Part 3.4: Tuning Neural Network Architectures

At each time step

Main challenges

- Define a distance between neural network architectures.
- Optimise $\varphi_t$ on the space of neural networks.
**Key idea:** To compute distance between architectures $G_1$, $G_2$, match computation in layers in $G_1$ to $G_2$. 

$$Z \in \mathbb{R}^{n_1 \times n_2}.$$ 

$Z_{ij} \leftarrow$ amount matched between layer $i \in G_1$ and $j \in G_2$.

Minimise $\phi_{lmm}(Z) + \phi_{str}(Z) + \phi_{nas}(Z)$

- $\phi_{lmm}(Z)$: label mismatch penalty
- $\phi_{str}(Z)$: structural penalty
- $\phi_{nas}(Z)$: non-assignment penalty

Can prove that the solution is a distance.
**Key idea:** To compute distance between architectures $G_1$, $G_2$, match computation in layers in $G_1$ to $G_2$.

$Z \in \mathbb{R}^{n_1 \times n_2}$.

$Z_{ij} \leftarrow \text{amount matched between layer } i \in G_1 \text{ and } j \in G_2.$
Key idea: To compute distance between architectures $G_1$, $G_2$, match computation in layers in $G_1$ to $G_2$.

$Z \in \mathbb{R}^{n_1 \times n_2}$.

$Z_{ij} \leftarrow$ amount matched between layer $i \in G_1$ and $j \in G_2$.

Minimise $\phi_{lmm}(Z) + \phi_{str}(Z) + \phi_{nas}(Z)$

$\phi_{lmm}(Z)$: label mismatch penalty

$\phi_{str}(Z)$: structural penalty

$\phi_{nas}(Z)$: non-assignment penalty
**Key idea:** To compute distance between architectures $G_1$, $G_2$, match computation in layers in $G_1$ to $G_2$.

$Z \in \mathbb{R}^{n_1 \times n_2}$.

$Z_{ij} \leftarrow$ amount matched between layer $i \in G_1$ and $j \in G_2$.

Minimise $\phi_{\text{lmm}}(Z) + \phi_{\text{str}}(Z) + \phi_{\text{nas}}(Z)$

$\phi_{\text{lmm}}(Z)$: label mismatch penalty

$\phi_{\text{str}}(Z)$: structural penalty

$\phi_{\text{nas}}(Z)$: non-assignment penalty

Can prove that the solution is a distance.
Optimising the sample from the posterior

Via an evolutionary algorithm.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dec_single</td>
<td>Pick a layer at random and decrease the number of units by 1/8.</td>
</tr>
<tr>
<td>dec_en_masse</td>
<td>Pick several layers at random in topological order and decrease the number of units by 1/8 for all of them.</td>
</tr>
<tr>
<td>inc_single</td>
<td>Pick a layer at random and increase the number of units by 1/8.</td>
</tr>
<tr>
<td>inc_en_masse</td>
<td>Pick several layers at random in topological order and increase the number of units by 1/8 for all of them.</td>
</tr>
<tr>
<td>dup_path</td>
<td>Pick a random path $u_1, u_2, \ldots, u_{k-1}, u_k$, duplicate layers $u_2, \ldots, u_{k-1}$ and connect them to $u_1$ and $u_k$.</td>
</tr>
<tr>
<td>remove_layer</td>
<td>Pick a layer at random and remove it. Connect the layer’s parents to its children if necessary.</td>
</tr>
<tr>
<td>skip</td>
<td>Randomly pick layers $u, v$ where $u$ is topologically before $v$. Add $(u, v)$ to $\mathcal{E}$.</td>
</tr>
<tr>
<td>swap_label</td>
<td>Randomly pick a layer and change its label.</td>
</tr>
<tr>
<td>wedge_layer</td>
<td>Randomly remove an edge $(u, v)$ from $\mathcal{E}$. Create a new layer $w$ and add $(u, w), (w, v)$ to $\mathcal{E}$.</td>
</tr>
</tbody>
</table>
Optimising the sample from the posterior

Via an evolutionary algorithm.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dec_single</td>
<td>Pick a layer at random and decrease the number of units by 1/8.</td>
</tr>
<tr>
<td>dec_en_masse</td>
<td>Pick several layers at random in topological order and decrease the number of units by 1/8 for all of them.</td>
</tr>
<tr>
<td>inc_single</td>
<td>Pick a layer at random and increase the number of units by 1/8.</td>
</tr>
<tr>
<td>inc_en_masse</td>
<td>Pick several layers at random in topological order and increase the number of units by 1/8 for all of them.</td>
</tr>
<tr>
<td>dup_path</td>
<td>Pick a random path $u_1, u_2, \ldots, u_{k-1}, u_k$, duplicate layers $u_2, \ldots, u_{k-1}$ and connect them to $u_1$ and $u_k$.</td>
</tr>
<tr>
<td>remove_layer</td>
<td>Pick a layer at random and remove it. Connect the layer’s parents to its children if necessary.</td>
</tr>
<tr>
<td>skip</td>
<td>Randomly pick layers $u, v$ where $u$ is topologically before $v$. Add $(u, v)$ to $\mathcal{E}$.</td>
</tr>
<tr>
<td>swap_label</td>
<td>Randomly pick a layer and change its label.</td>
</tr>
<tr>
<td>wedge_layer</td>
<td>Randomly remove an edge $(u, v)$ from $\mathcal{E}$. Create a new layer $w$ and add $(u, w), (w, v)$ to $\mathcal{E}$.</td>
</tr>
</tbody>
</table>

Resulting procedure: NASBOT
Neural Architecture Search with Bayesian Optimisation and Optimal Transport (Kandasamy et al. Arxiv 2018)
Architectures found on Cifar10
Architectures found on Indoor Location
Architectures found on Slice Localisation
DOE on other (graphical) structures

Drug Discovery with Small molecules

Crystal Structures

Social networks & viral marketing
Summary

- A framework for “goal oriented” DOE.
  - General: can achieve any desired goal.
  - Flexible: can incorporate domain expertise.


- Can be trivially parallelised.
Summary

- A framework for “goal oriented” DOE.
  - General: can achieve any desired goal.
  - Flexible: can incorporate domain expertise.


- Can be trivially parallelised.

Scaling up DOE

- Multi-fidelity experimentation: Use cheap approximations to an expensive experiment to speed things up.

- High dimensional DOE: Additive models have favourable statistical and computational properties.

- DOE in “complex” domains.
Outline

➤ Part I: Preliminaries (Black-box Optimisation)
   1. Bayesian Models
   2. Black-box Optimisation via Thompson Sampling

➤ Part II: DOE via posterior sampling

➤ Part III: Scaling up DOE (back to Black-box Optimisation)
   1. Parallelising experiments
   2. Multi-fidelity experimentation
   3. High dimensional input spaces
   4. Beyond Euclidean/categorical domains

➤ Part IV: ExperiML & Collaborations with LBL
What is an experiment?

An experiment is any action that has an opportunity cost attached to it.
What is an experiment?

An experiment is any action that has an opportunity cost attached to it.

Examples:

1. Experiments to design/discover new materials/drugs.
2. Experiments to optimise and industrial process.
3. Personalised (contextual) experiments: online advertising, search etc.
Currently companies have to choose between two bad alternatives

<table>
<thead>
<tr>
<th>Exhaustive search over restricted design space</th>
<th>Manual tuning by experts</th>
</tr>
</thead>
</table>

Value proposition: Faster & Better.
Our technology enables searching over large design spaces and identifies better designs in 10-100 times fewer trials than exhaustive search and expert tuning with significantly less effort from experts.
Currently companies have to choose between two bad alternatives

<table>
<thead>
<tr>
<th>Exhaustive search over restricted design space</th>
<th>Manual tuning by experts</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ Automated.</td>
<td></td>
</tr>
<tr>
<td>− Expensive testing on many poor designs.</td>
<td></td>
</tr>
<tr>
<td>− Misses many revolutionary designs since design space is restricted.</td>
<td></td>
</tr>
</tbody>
</table>
Currently companies have to choose between two bad alternatives

**Exhaustive search over restricted design space**
- Automated.
- Expensive testing on many poor designs.
- Misses many revolutionary designs since design space is restricted.

**Manual tuning by experts**
- Domain experts manually design each test, directly using human expertise.
- Requires significant time and effort from experts.
- Humans are bad at making sense of complex high dimensional data.

Value proposition: Faster & Better.
Our technology enables searching over large design spaces and identifies better designs in 10-100 times fewer trials than exhaustive search and expert tuning with significantly less effort from experts.
Currently companies have to choose between two bad alternatives

**Exhaustive search over restricted design space**
- Automated.
- Expensive testing on many poor designs.
- Misses many revolutionary designs since design space is restricted.

**Manual tuning by experts**
- Domain experts manually design each test, directly using human expertise.
- Requires significant time and effort from experts.
- Humans are bad at making sense of complex high dimensional data.

**Value proposition:** Faster & Better.
Our technology enables searching over large design spaces and identifies better designs in 10-100 times fewer trials than exhaustive search and expert tuning with significantly less effort from experts.
What we need for a collaboration

A well defined DOE problem. This includes,

▶ **Design variables** that can be tuned in the given problem and the constraints on each variable.

▶ **Experimental results.** E.g: how well did a design do on the criteria you care.

▶ **Goal:** What is the goal of conducting these experiments?

▶ **Means to experiment:** For each design, a means to conduct the experiment and obtain feedback, either in simulation or in a real system.
What we need for a collaboration

A well defined DOE problem. This includes,

▷ **Design variables** that can be tuned in the given problem and the constraints on each variable.

▷ **Experimental results.** E.g: how well did a design do on the criteria you care.

▷ **Goal:** What is the goal of conducting these experiments?

▷ **Means to experiment:** For each design, a means to conduct the experiment and obtain feedback, either in simulation or in a real system.

In addition ...

▷ **Domain expertise:** Explicit models, tacit knowledge etc.

▷ **Past data:** Data from past experiments on this or relevant tasks.
In return, we will ...

- Execute our methods on the data/problem given.
- ... and recommend new experiments(s) to conduct.
In return, we will ...

- Execute our methods on the data/problem given.
- ... and recommend new experiments(s) to conduct.
- If there is a simulation, we can run it ourselves. Otherwise, we will need your help to run experiments.
CMU, University of Massachusetts Amherst, Rice University, University of Texas Austin, Microsoft Research, ExperiML

Thank You