# Bayesian Optimization under Heavy-tailed Payoffs

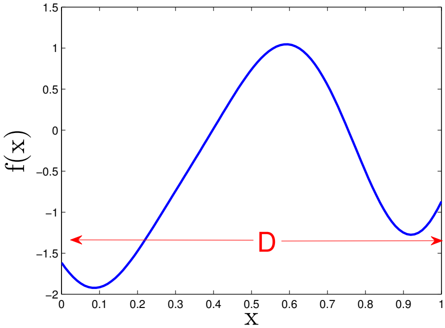
Sayak Ray Chowdhury
(Joint work with Aditya Gopalan)

Department of Electrical Communication Engineering
Indian Institute of Science

September 25, 2019

# Black-box optimization

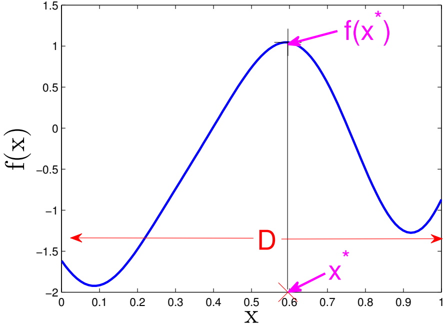
Sequentially Maximize $f : D \to \mathbb{R}$




- $f$ **unknown**, $D \subset \mathbb{R}^d$

# Black-box optimization

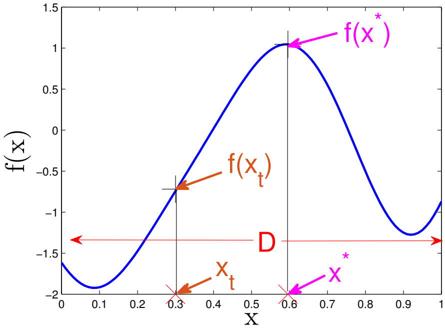
Sequentially Maximize $f : D \to \mathbb{R}$




- $f$ **unknown**, $D \subset \mathbb{R}^d$
- $x^\star \in \underset{x \in D}{\mathrm{argmax}}\, f(x)$

# Black-box optimization

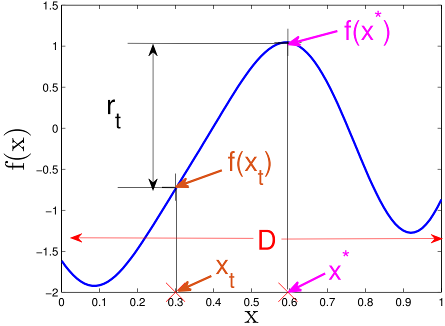
Sequentially Maximize $f : D \to \mathbb{R}$




- $f$ **unknown**, $D \subset \mathbb{R}^d$
- $x^\star \in \underset{x \in D}{\operatorname{argmax}} f(x)$
- At each round $t$:
  - Learner chooses $x_t \in D$ based on past
  - Observes noisy reward $y_t = f(x_t) + \varepsilon_t$

# Black-box optimization

Sequentially Maximize $f : D \to \mathbb{R}$




- $f$ **unknown**, $D \subset \mathbb{R}^d$
- $x^\star \in \operatorname{argmax}_{x \in D} f(x)$
- At each round $t$:
  - Learner chooses $x_t \in D$ based on past
  - Observes noisy reward $y_t = f(x_t) + \varepsilon_t$

## Goal

- Minimize Cumulative Regret: $\sum_{t=1}^{T} \Big( f(x^\star) - f(x_t) \Big)$

# Application: Hyperparameter tuning in ML models

- Hyperparameters in DeepNN training:
  - Learning rate
  - Regularizer
  - Number of hidden layers
  - Number of units in each layer
  - Optimizer (SGD, Adagrad, Adam, ...)
  - Nonlinearity (Relu, Softmax, ...)
  - ...

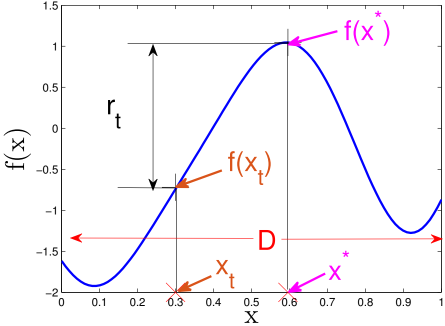
- Black-box optimization:
  - $D$ : all possible hyperparameter configurations
  - $f(x)$ : training error for configuration $x$
  - $x^\star$ : the best hyperparameter

# Traditional approaches

- Grid search, Random search
  - Doesnt use information from previous searches
  - Not good when training time is high
- Huge (possibly infinite) set of hyperparameters ($D \subset \mathbb{R}^d$)
- Need to make an educated decision about where to search
  - Bayesian optimization

# Bayesian optimization

Sequentially Maximize $f : D \to \mathbb{R}$




At each round $t$:

- Learner chooses $x_t \in D$ based on past knowledge
- Observes noisy reward $y_t = f(x_t) + \varepsilon_t$

Existing works: Rewards are light-tailed, e.g., Gaussian or sub-Gaussian (Srinivas et al., 2010, Chowdhury and Gopalan, 2017,...)

# Motivation

Many real life environments exhibit heavy-tailed behavior

- Distribution of delays in data networks
- Bursty traffic flow distributions
- Price fluctuations in finance and insurance data

Can we develop efficient Bayesian optimization algorithms under heavy-tailed environments ?

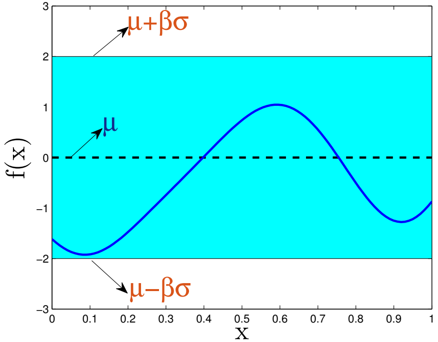
- Efficient algorithm: Sublinear growth of cumulative regret with time

# Assumptions

- Smoothness: $f$ lies in **R**eproducing **K**ernel **H**ilbert **S**pace (**RKHS**) of functions $D \to \mathbb{R}$
  - Reproducing property: $f(x) = \langle f, k(x, \cdot) \rangle_k$
  - Induces smoothness: $|f(x) - f(y)| \leq \|f\|_k \, \|k(x, \cdot) - k(y, \cdot)\|_k$
- Example kernel
  - RBF kernel: $k(x, y) = \exp\left(\frac{-\|x-y\|_2^2}{2l^2}\right)$
- Heavy-tailed noise $\varepsilon_t$ is zero mean and have bounded $(1+\alpha)$-th moment for $\alpha \in (0, 1]$.
  - Student's-$t$ distribution with 3 d.o.f. has bounded variance

# Algorithm design

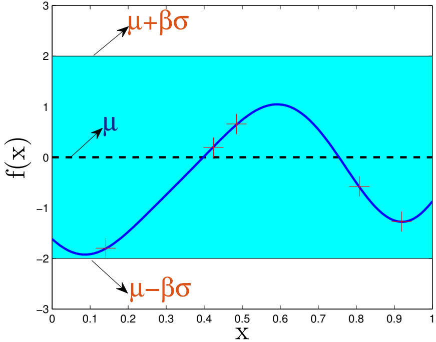
**Key idea:** Represent uncertainty over $f$ using **G**aussian **P**rocess (**GP**)




- Assume Gaussian Process Prior $\mathrm{GP}\Big(0, k(x, y)\Big)$

# Algorithm design

**Key idea:** Represent uncertainty over $f$ using **G**aussian **P**rocess (**GP**)



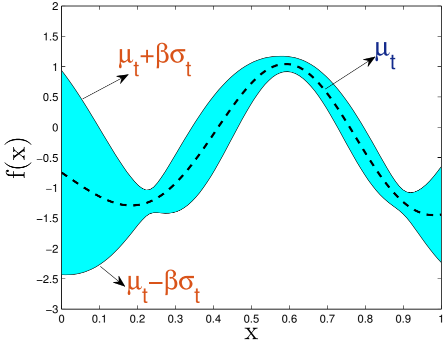

- Assume Gaussian Process Prior $\mathrm{GP}\Big(0, k(x, y)\Big)$
- Truncate high rewards:

$$\widehat{y}_t = \mathbf{1}_{|y_t| \leq b_t}$$

- $b_t$ governs the truncation

# Algorithm design

**Key idea:** Represent uncertainty over $f$ using **G**aussian **P**rocess (**GP**)




- Assume Gaussian Process Prior $\mathrm{GP}\Big(0, k(x,y)\Big)$
- Truncate high rewards:

$$\widehat{y}_t = \mathbf{1}_{|y_t| \le b_t}$$

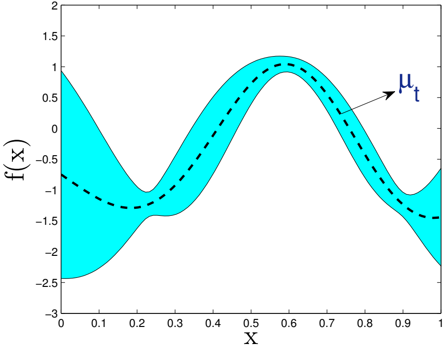
- $b_t$ governs the truncation

## (Approximate) GP posterior:

$$\begin{aligned}\mu_t(x) &= k_t(x)^T (K_t + \lambda I)^{-1} [\widehat{y}_1, \ldots, \widehat{y}_t]^T \\ \sigma_t^2(x) &= k(x,x) - k_t(x)^T (K_t + \lambda I)^{-1} k_t(x)\end{aligned}$$

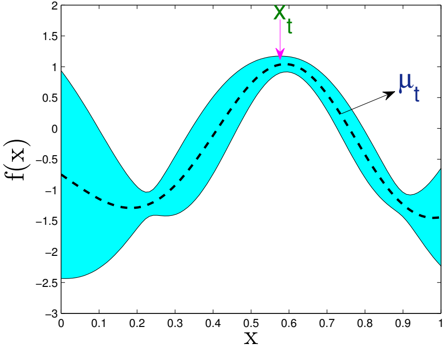
# Algorithm 1: Truncated GP-UCB (TGP-UCB)

**Key Idea:** Play the arm with highest Upper Confidence Bound (UCB)



# Algorithm 1: Truncated GP-UCB (TGP-UCB)

**Key Idea:** Play the arm with highest Upper Confidence Bound (UCB)

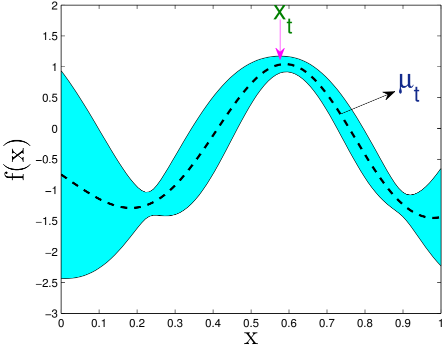



At each round $t$, play:

$$x_t = \underset{x \in D}{\operatorname{argmax}}\, \mu_t(x) + \beta_t \sigma_t(x)$$

# Algorithm 1: Truncated GP-UCB (TGP-UCB)

**Key Idea:** Play the arm with highest Upper Confidence Bound (UCB)




At each round $t$, play:

$$x_t = \operatorname*{argmax}_{x \in D} \mu_t(x) + \beta_t \sigma_t(x)$$

- $\beta_t$ trades off b/w **exploration** and **exploitation**
- Well known algorithm under Gaussian payoffs (but no reward truncation)

# Cumulative regret of TGP-UCB

## Upper bound of TGP-UCB (Informal)

Cumulative regret of **TGP-UCB** is $O\left(\gamma_T T^{\frac{2+\alpha}{2(1+\alpha)}}\right)$ *with high probability*

- $\gamma_T$ is a function of the kernels and quantifies Reduction in uncertainty about $f$ after observing rewards
  - RBF kernel: $\gamma_T = \mathrm{polylog}(T)$
- (Recall) assumption: rewards have bounded $(1+\alpha)$-th moment
  - Bounded variance ($\alpha = 1$): Cumulative regret is $\tilde{O}(T^{3/4})$
  - Sublinear growth with $T$: TGP-UCB is efficient

# A fundamental lower bound

## Lower bound (Informal)

**RBF kernel:** Expected cumulative regret of any algorithm is $\Omega(T^{\frac{1}{1+\alpha}})$

- Bounded variance ($\alpha = 1$): Lower bound is $\Omega(T^{1/2})$
- TGP-UCB is effecient, but may be suboptimal
- *Fails to completely nullify the effect of heavy-tail fluctuations*
- Similar result for Matérn kernel

# An (almost) optimal algorithm

## Adaptively Truncated Approximate GP-UCB algorithm

- Perform truncation in the feature space as opposed to truncating raw observations
- **Challenge:** Feature space is (possibly) infinite dimensional
- **Solution:** Find a "good" finite-dimensional approximation of the features
  - Keep error of approximation in control
- Perform feature adaptive truncation in this approximate feature space

Regret upper bound (Informal): $O(\gamma_T T^{\frac{1}{1+\alpha}})$

# Algorithm 2: ATA-GP-UCB

$$x_t = \mathrm{argmax}_{x \in D}\, \mu_t(x) + \beta_t \sigma_t(x)$$

- Feature approximation of kernel $\longrightarrow \varphi_t$ (of dimension $m_t$)
- $\Psi_t = [\varphi_t(x_1), \ldots, \varphi_t(x_t)]^T \qquad V_t = \Psi_t^T \Psi_t + \lambda I_{m_t}$,
- Rows of $V_t^{-1/2} \Psi_t^T \longrightarrow u_1, \ldots, u_{m_t}$
- Feature adaptive truncation of rewards in each dimension:

$$r_i = \sum_{\tau=1}^{t} u_{i,\tau} y_\tau \mathbf{1}_{|u_{i,\tau} y_\tau| \le b_t}$$

## Approximate GP posterior

$$\begin{aligned} \mu_t(x) &= \varphi_t(x)^T V_t^{-1/2} [r_1, \ldots, r_{m_t}]^T \\ \sigma_t^2(x) &= k(x,x) - \varphi_t(x)^T \varphi_t(x) + \lambda \varphi_t(x)^T V_t^{-1} \varphi_t(x) \end{aligned}$$

# Fourier features approximation

- RBF kernel: $k(x,y) = e^{-\frac{(x-y)^2}{2l^2}}$

- Random Fourier features (Rahimi and Recht, 2008):
  - $\varphi(x) = \frac{1}{\sqrt{m}}[\cos(\omega_1 x), \ldots, \cos(\omega_m x), \sin(\omega_1 x), \ldots, \sin(\omega_m x)]^T$
  - $\omega_i \overset{\text{i.i.d.}}{\sim} \frac{l}{\sqrt{2\pi}} e^{-\frac{l^2\omega^2}{2}}$ [Classical method; not useful here]

- Quadrature Fourier features (Mutny and Krause, 2018):
  - $\varphi(x)_i = \begin{cases} \sqrt{\nu(\omega_i)} \cos\left(\frac{\sqrt{2}}{l}\omega_i x\right) & \text{if } 1 \leq i \leq m, \\ \sqrt{\nu(\omega_{i-m})} \sin\left(\frac{\sqrt{2}}{l}\omega_{i-m} x\right) & \text{if } m+1 \leq i \leq 2m \end{cases}$
  - $\omega_1, \ldots, \omega_m$ : roots of the $m$-th Hermite polynomial $H_m$
  - $\nu(z) = \frac{2^{m-1} m!}{m^2 H_{m-1}(z)^2}$ [Used in this work]

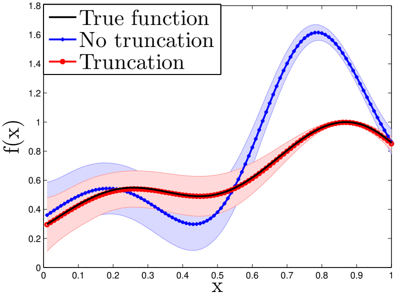
# Nyström approximation

- Data dependent: Approximate the gram matrix $K_t$
- Sample $m_t$ points from $\{x_1, \dots, x_t\}$ to construct a dictionary $\mathcal{D}_t$
- Include $x$ in $\mathcal{D}_t$ independently with probability $\sigma_t^2(x)$
- Feature approximation: $\varphi_t(x) = \left(K_{\mathcal{D}_t}^{1/2}\right)^{\dagger} k_{\mathcal{D}_t}(x)$
  - $K_{\mathcal{D}_t} = [k(x, y)]_{x,y \in \mathcal{D}_t}$
  - $k_{\mathcal{D}_t}(x) = [k(x_1, x), \dots, k(x_m, x)]^T_{x_i \in \mathcal{D}_t}$

(Alaoui and Mahony, 2015, Calandriello et. al, 2019)

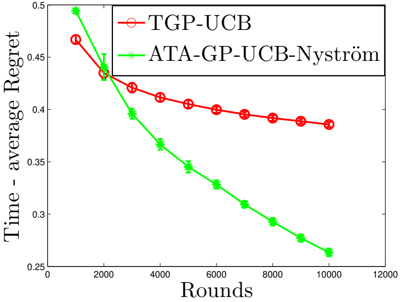
# Numerical Results

$f$ sampled from RKHS
(RBF kernel + Impulse noise)




- Reward truncation is necessary to find good estimates

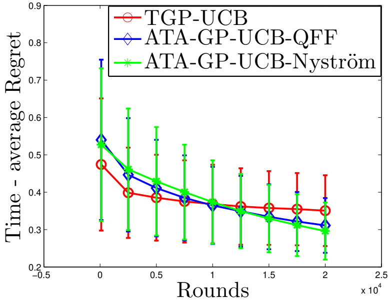
Financial data
(End of day stock prices)



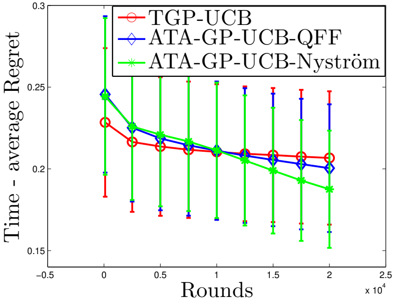

- Feature adaptive truncation nullify heavy tail fluctuations

# Numerical Results

$f$ sampled from RKHS
(RBF kernel + Student's-$t$ noise)




$f$ sampled from RKHS
(RBF kernel + Pareto noise)




- (Optimal) ATA-GP-UCB is better than (suboptimal) TGP-UCB
- Nyström embedding is better than Quadrature Fourier features

Thank You