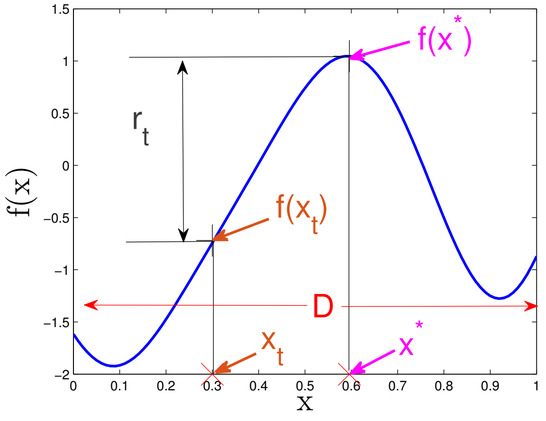
Bayesian Optimization meets Reinforcement Learning

Sayak Ray Chowdhury

Department of Electrical Communication Engineering, Indian Institute of Science

Black-Box Optimization with Gaussian Processes

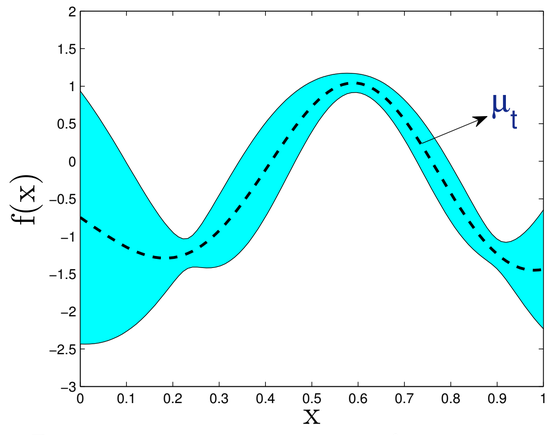
Sequentially maximize an *unknown* function $f : D \to \mathbb{R}$



At every round t, an agent:

1. Chooses $x_t \in D$ based on past observations
2. Observes noisy samples of $f(x_t)$
3. Suffers regret $f(x^\star) - f(x_t)$

Goal: Minimize cumulative regret

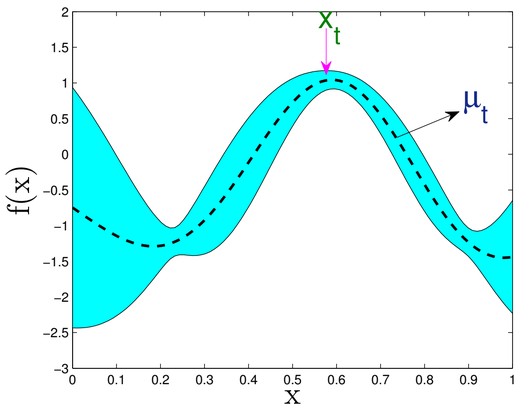


- Represent uncertainty over f using **G**aussian **p**rocess prior $GP(0, k(x, y))$
- Squared Exponential (SE) kernel: $k(x,y) = \exp\left(\frac{-\|x-y\|_2^2}{2l^2}\right)$
- Posterior of f: $GP(\mu_t(x), k_t(x, y))$

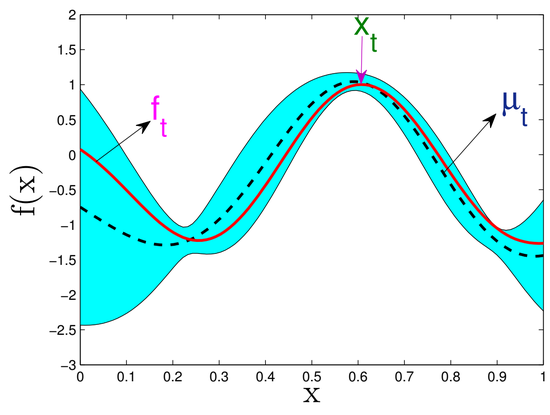
Bayesian optimization application and key idea. Hyperparameter tuning in DeepNN – huge set of parameters to tune – *number of layers, weight regularization, layer size, nonlinearity type, batch size, learning rate schedule, stopping conditions etc* – grid search is expensive – *optimize a cheap proxy function instead !*

BO Algorithms: IGP-UCB and GP-TS [1]

Improved GP-UCB. Choose the maximizer x_t of the Upper Confidence Bound (UCB) envelope of posterior Gaussian process



GP-Thompson sampling. Sample a random function f_t from posterior Gaussian process and choose its maximizer x_t

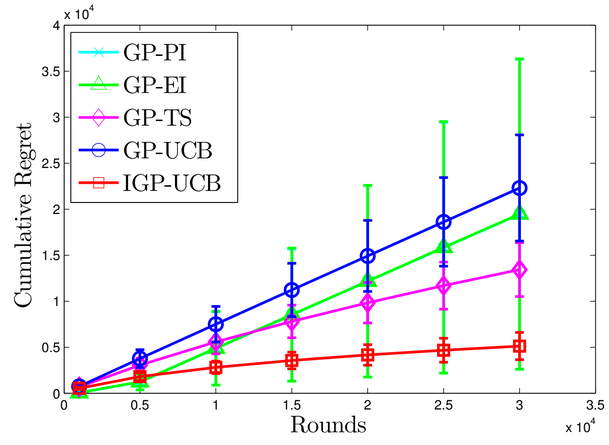


Acknowledgment. Google India PhD fellowship grant in Machine Learning, 2017

Performance of BO Algorithms

*For sufficiently smooth f, the **cumulative regret** of Bayesian optimization algorithms in T rounds are upper bounded by $\tilde{O}(\gamma_T\sqrt{T})$*

- γ_T: *Maximum Information Gain* about f after T rounds – quantifies reduction in uncertainty after observing T samples
- **SE** kernel: $\gamma_T \asymp O(\ln T)$
- Cumulative regret grows **sublinearly** with T – *average per-round regret vanishes*



Reinforcement Learning with Gaussian Processes

Episodically maximize reward in an *unknown* Markov Decision Process $M = \{\mathcal{S}, \mathcal{A}, R, P\}$

- State space $\mathcal{S}$, Action space $\mathcal{A}$, known
- Reward distribution $R(s,a)$, unknown
- Transition distribution $P(s,a)$, unknown

Goal: Minimize the loss incurred in the *Value function* due to not knowing the optimal policy of the unknown **MDP** M

At every round within an episode, an agent:

1. Takes an action $a \in \mathcal{A}$ based on the current state $s \in \mathcal{S}$ and past observations
2. Receives reward $r \sim R(s,a)$
3. Observes next state $s' \sim P(s,a)$

RL Algorithms: GP-UCRL and GP-TSRL [2]

Bayesian inference philosophy. Put *separate* Gaussian process priors over mean reward and mean transition function, and update posteriors at the end of every episode

1. Construct two confidence sets, one each for mean reward and mean transition function, using parameters of posterior distributions
2. Find the set of all MDPs, for which mean reward and mean transition function lie within respective confidence sets and choose the optimal policy for that *set of MDPs*

1. Sample two random functions, one each from the posterior distributions of mean reward and mean transition function
2. Build an MDP using the random samples of mean reward and mean transition function and choose the optimal policy for that *sampled MDP*

Theoretical guarantees order-wise similar to Bayesian optimization. *At every episode/round, the unknown mean reward and mean transition function lie within properly constructed confidence sets of shrinking width*

References

[1] *On Kernelized Multi-armed Bandits, S. R. Chowdhury and A. Gopalan, ICML 2017.*

[2] *Online Learning in Kernelized Markov Decision Processes, S. R. Chowdhury and A. Gopalan, ArXiv e-prints, May 2018.*