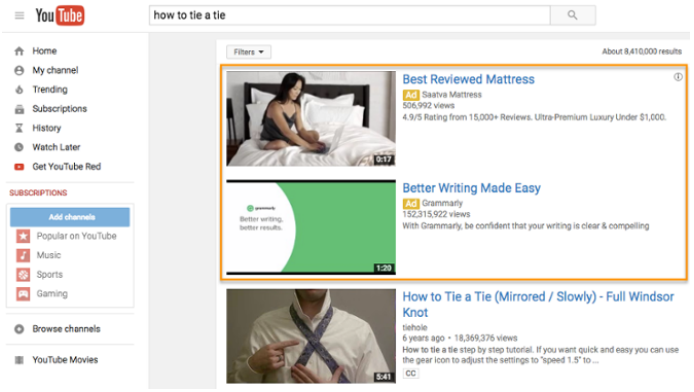
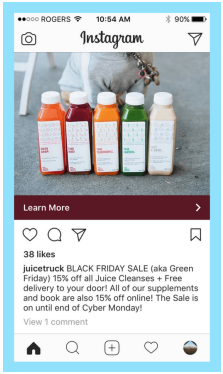
Introduction to Multi-Armed Bandits & Bayesian Optimization

Samarth Gupta & Ankur Mallick
18-847

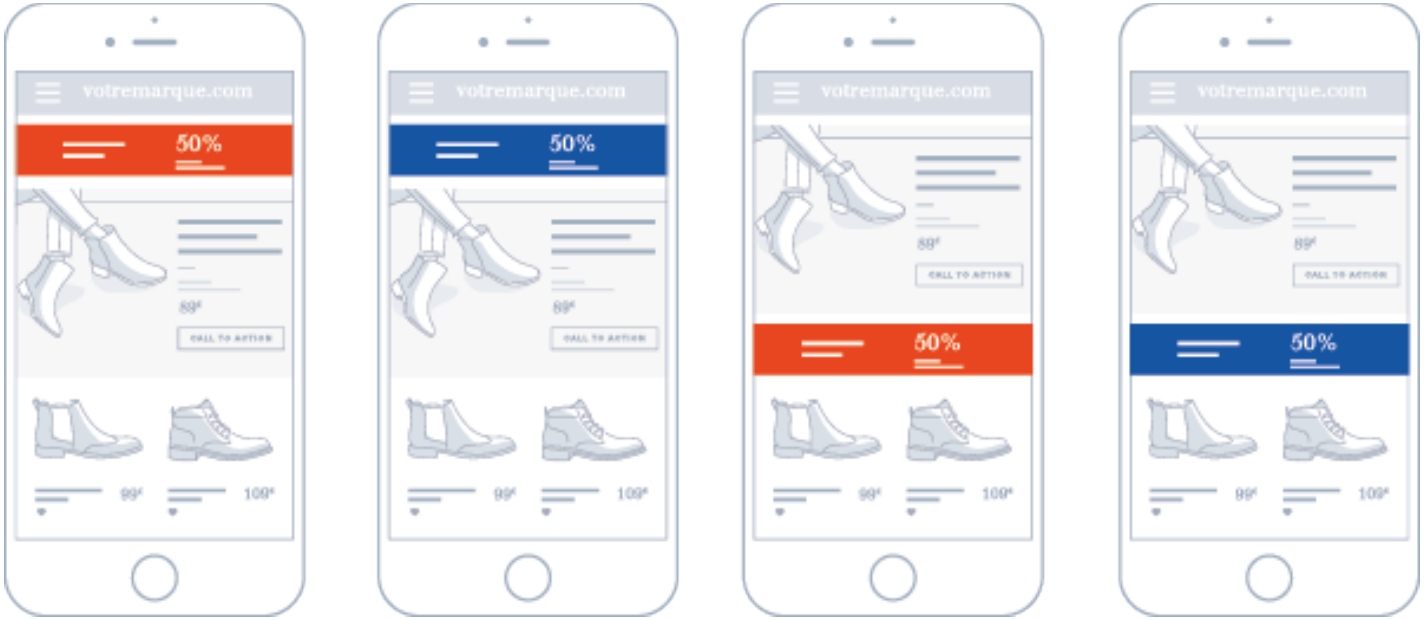
Ads everywhere!





Which version of ad to show?

What layout to use?



Political campaign

PLAY ANIMATION

PLAY ANIMATION

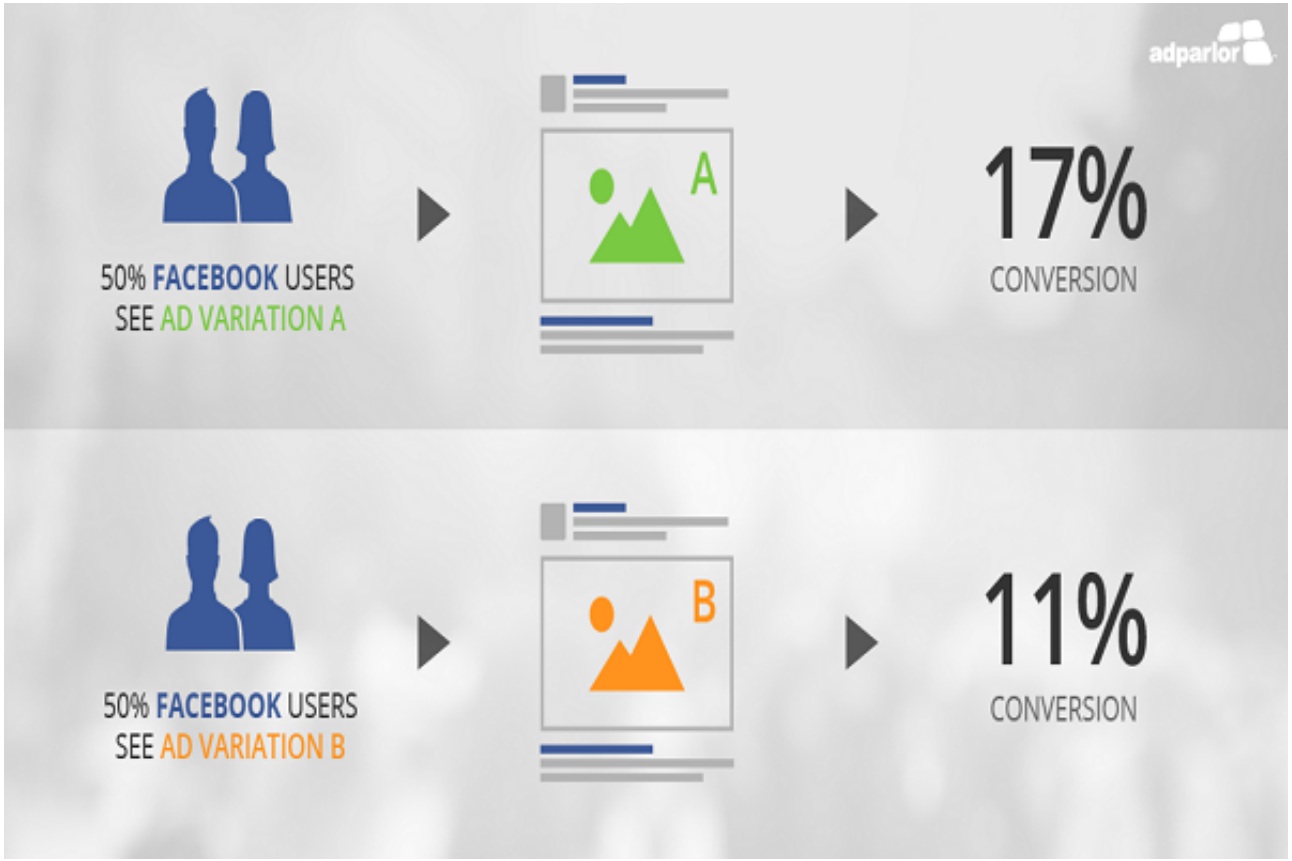
Different configuration lead to more donations!

+5%

Can we design algorithms with the aim of maximizing reward?

Simple approach

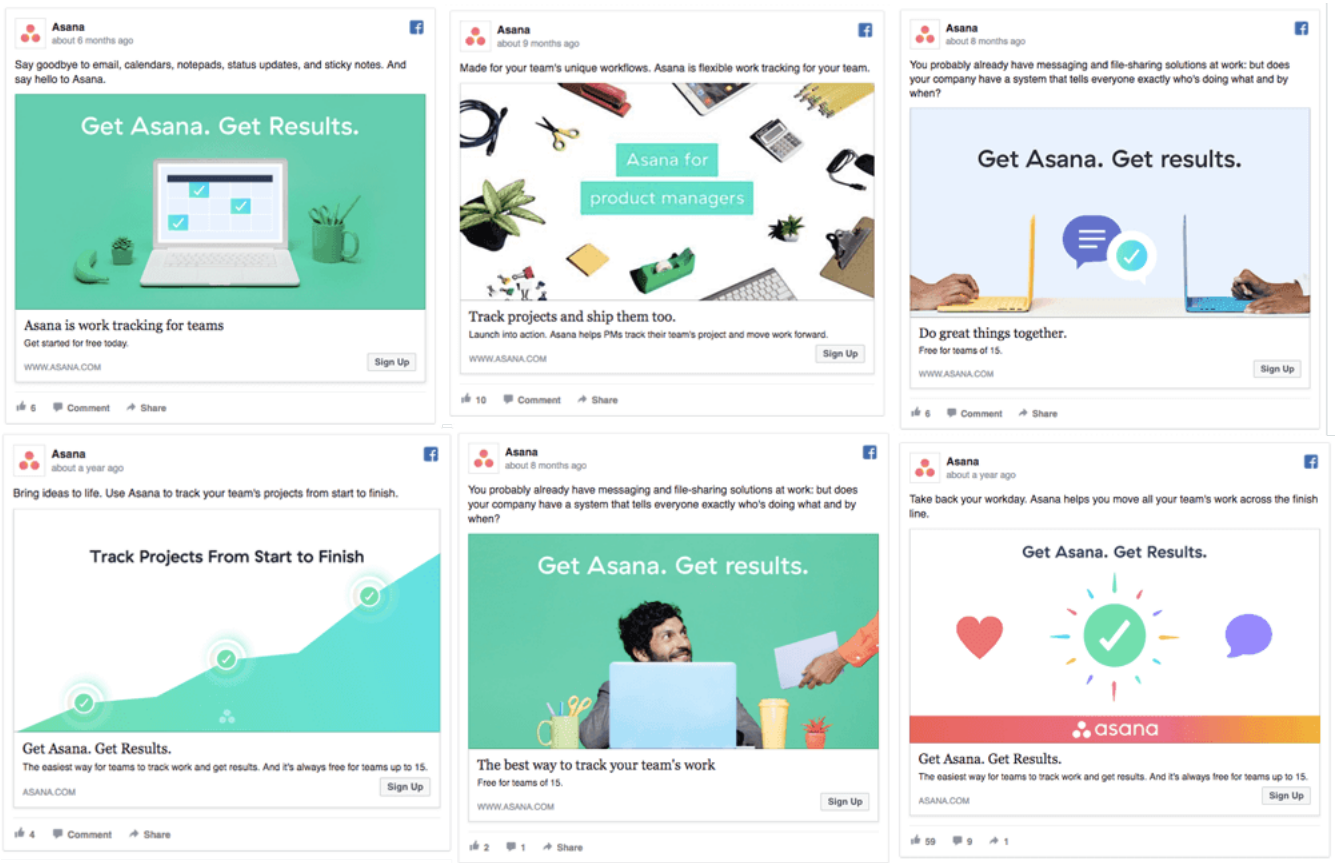
Simple approach



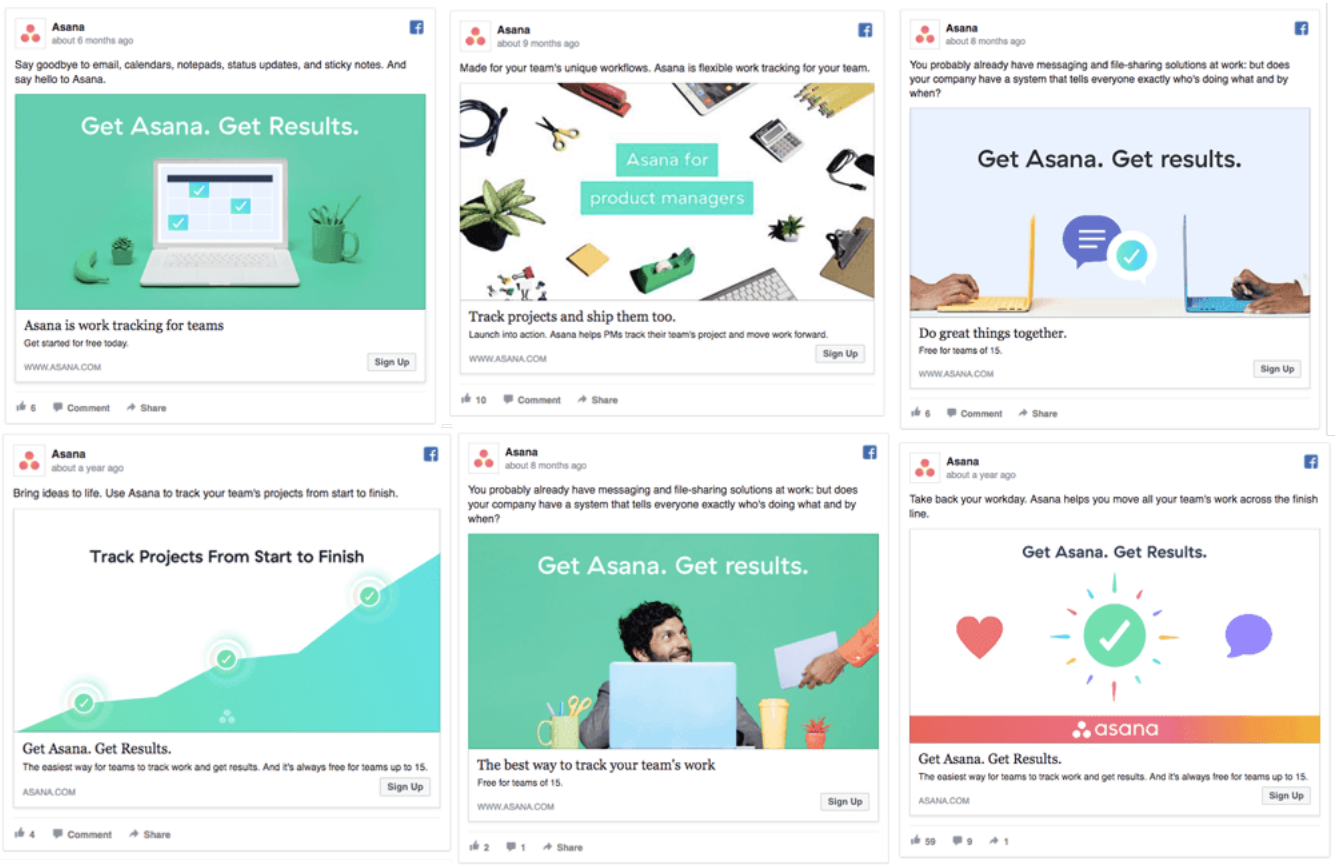
Problems with this?

- How long should be the test phase?
- In test phase half the users are seeing wrong ads?
- Can you be sure of outcome in the test phase?

Should we still split equally?



Should we still split equally?



A lot of users might see sub-optimal ads!

Can we do something smart?

- Make selection adaptive, instead of having a static scheme
- Observe the results at the end of each time step and make decisions based on that
- This motivates the study of Multi-Armed Bandit setup

Multi-Armed Bandits

1$

0$

0$

1$

4$

0$

2$

1$

3$

1$

2$

2$

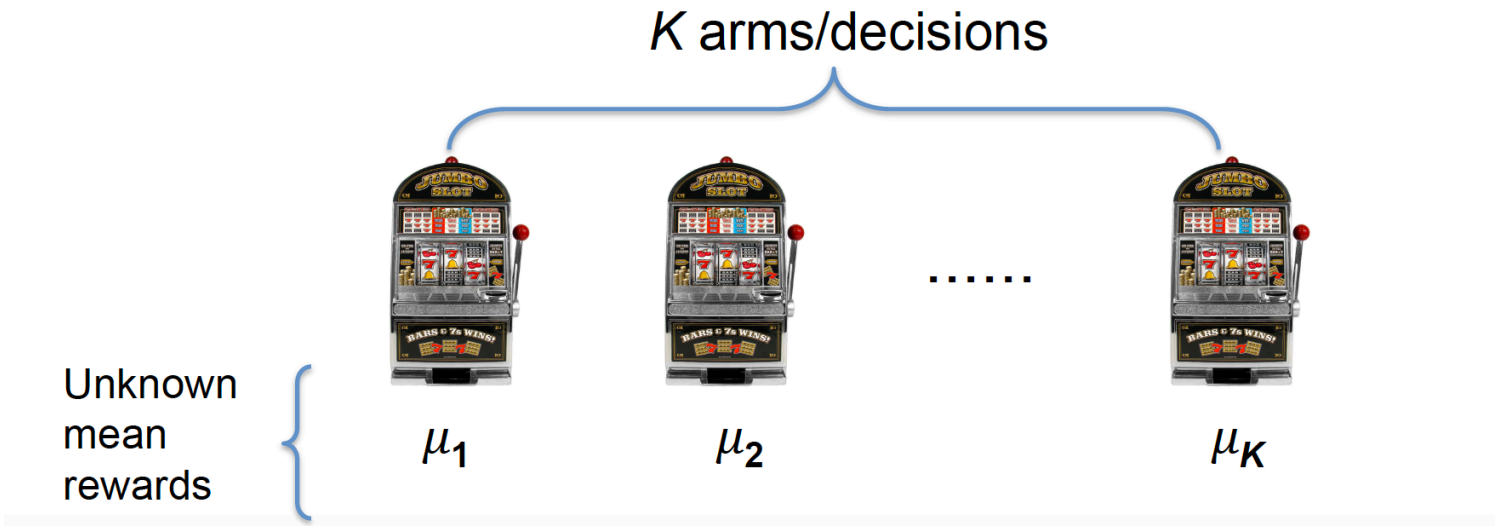
0$

2$

1$

0$

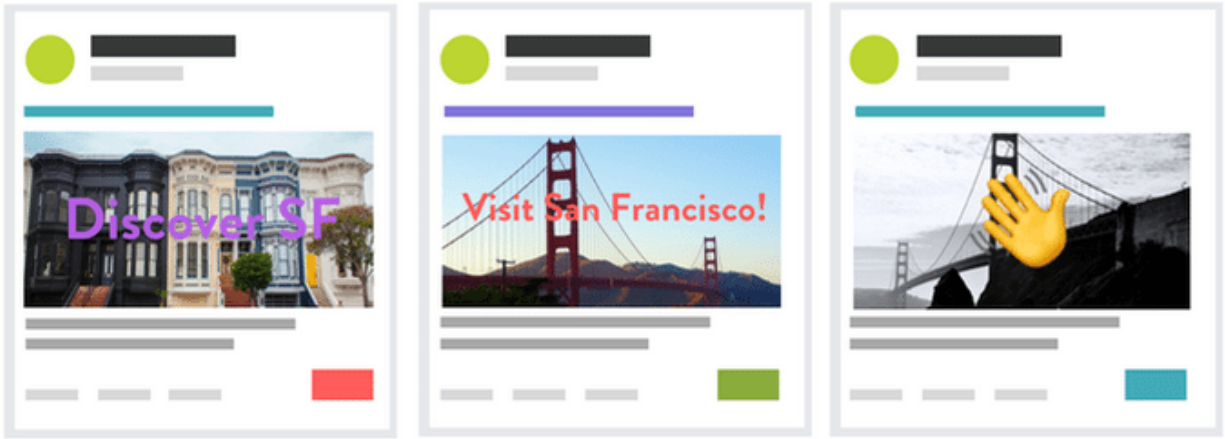
Multi-Armed Bandit Problem



- At every round: decide which arm to play (k_t)
- Receive a random reward corresponding to the arm played
- Mean rewards are unknown

Goal: Maximize cumulative reward in T rounds

Connection with Ad selection



- Arms: Different version of an ad
- Choice: Which version to display to the user
- Reward: Time spent / Clicked or not?

Which arm to pick at next round?

1$
0$
1$

$\hat{\mu}_1(t) = 0.67$

$n_1(t) = 3$

1$
4$
0$
2$
0$
3$
1$
1$

$\hat{\mu}_2(t) = 1.5$

$n_2(t) = 8$

2$
0$
2$
1$
1$

$\hat{\mu}_3(t) = 1.2$

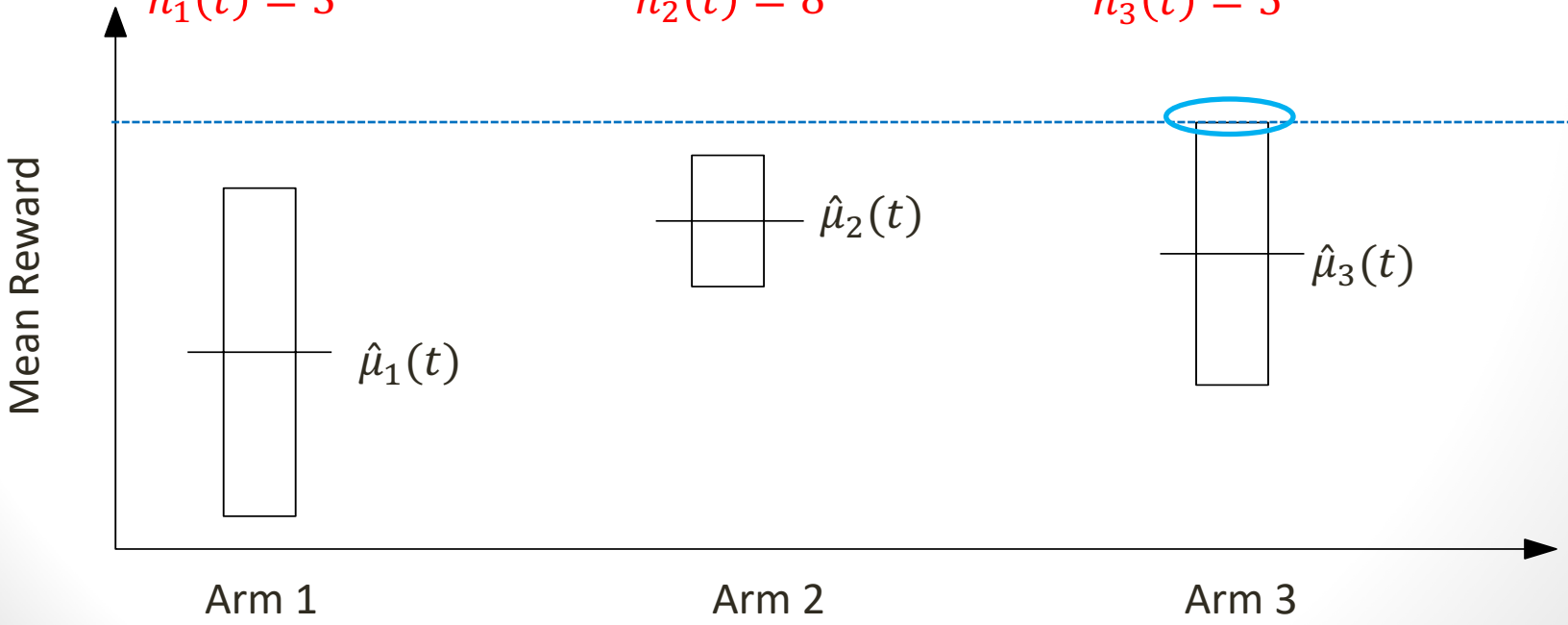
$n_3(t) = 5$

UCB (Upper Confidence Bound) Algorithm



$\hat{\mu}_1(t) = 0.67$ $\hat{\mu}_2(t) = 1.5$ $\hat{\mu}_3(t) = 1.2$

$n_1(t) = 3$ $n_2(t) = 8$ $n_3(t) = 5$

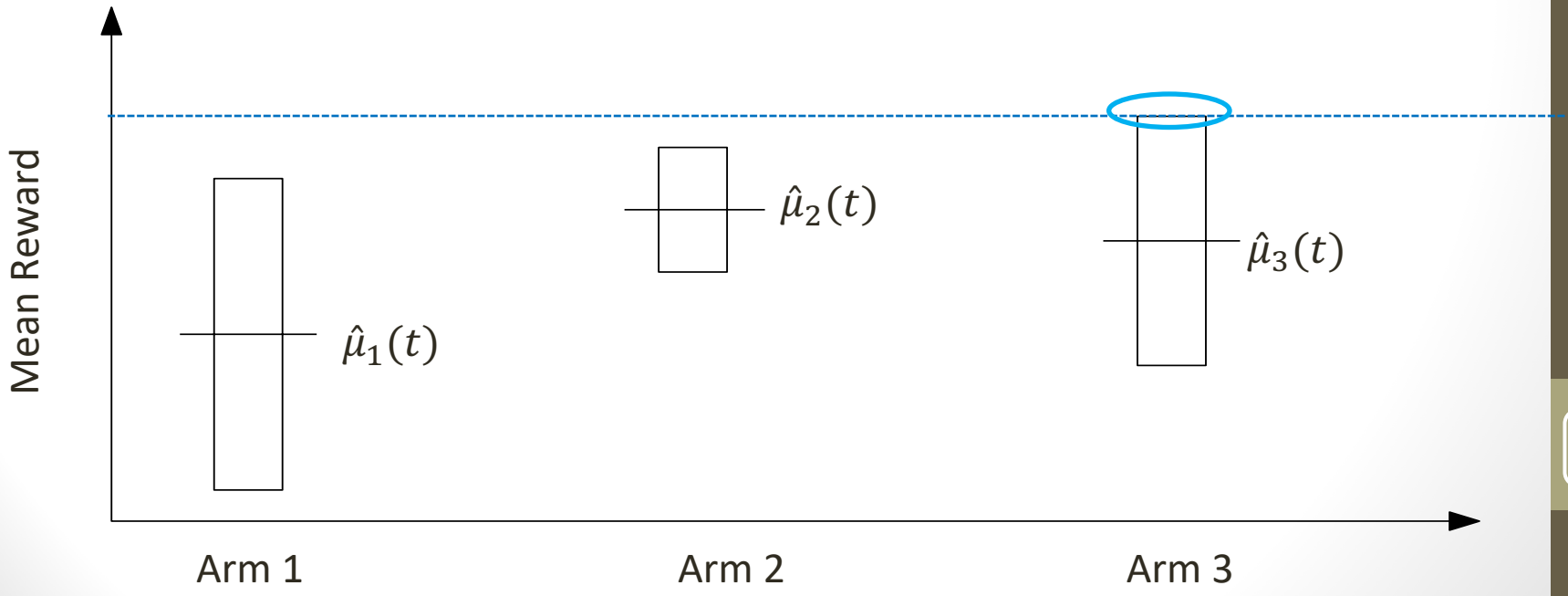


Lai, T. L., & Robbins, H. (1985). Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, *6*(1), 4-22.

UCB (Upper Confidence Bound) Algorithm

$$k_{t+1} = \arg\max \hat{\mu}_k(t) + \sqrt{\frac{a \log t}{n_k(t)}}$$

- Selects arms with large $\hat{\mu}_k(t)$ or small $n_k(t)$
- Confidence set shrinks as $n_k(t)$ increases



Performance Measure

μ_1

μ_2

μ_3

- Goal: Maximize cumulative reward
- Equivalent to minimizing “regret”
- Regret: How much loss do you incur on pulling the suboptimal arms?

Performance Measure: Regret

μ_1

$n_1(t)$

Best Arm

$\mu_2 = \mu^*$

$n_2(t)$

μ_3

$n_3(t)$

- Ideally: Always pull arm 2
- Algorithm does not know mean reward
- Due to adaptive nature it pulls each arm different number of times

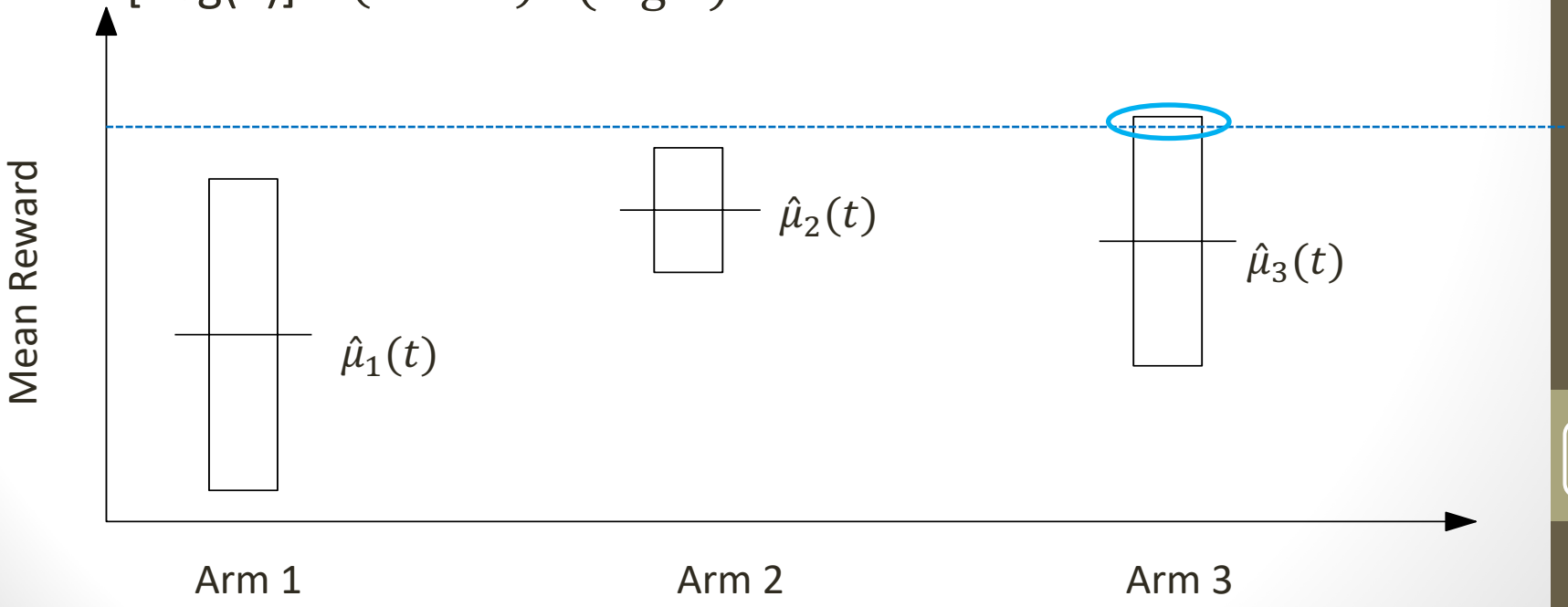
- Loss incurred / Regret = $n_1(t)\times(\mu_2 - \mu_1) + n_3(t)\times(\mu_2 - \mu_3)$

$= \sum_k n_k(t)\times(\mu^* - \mu_k)$

Regret for UCB

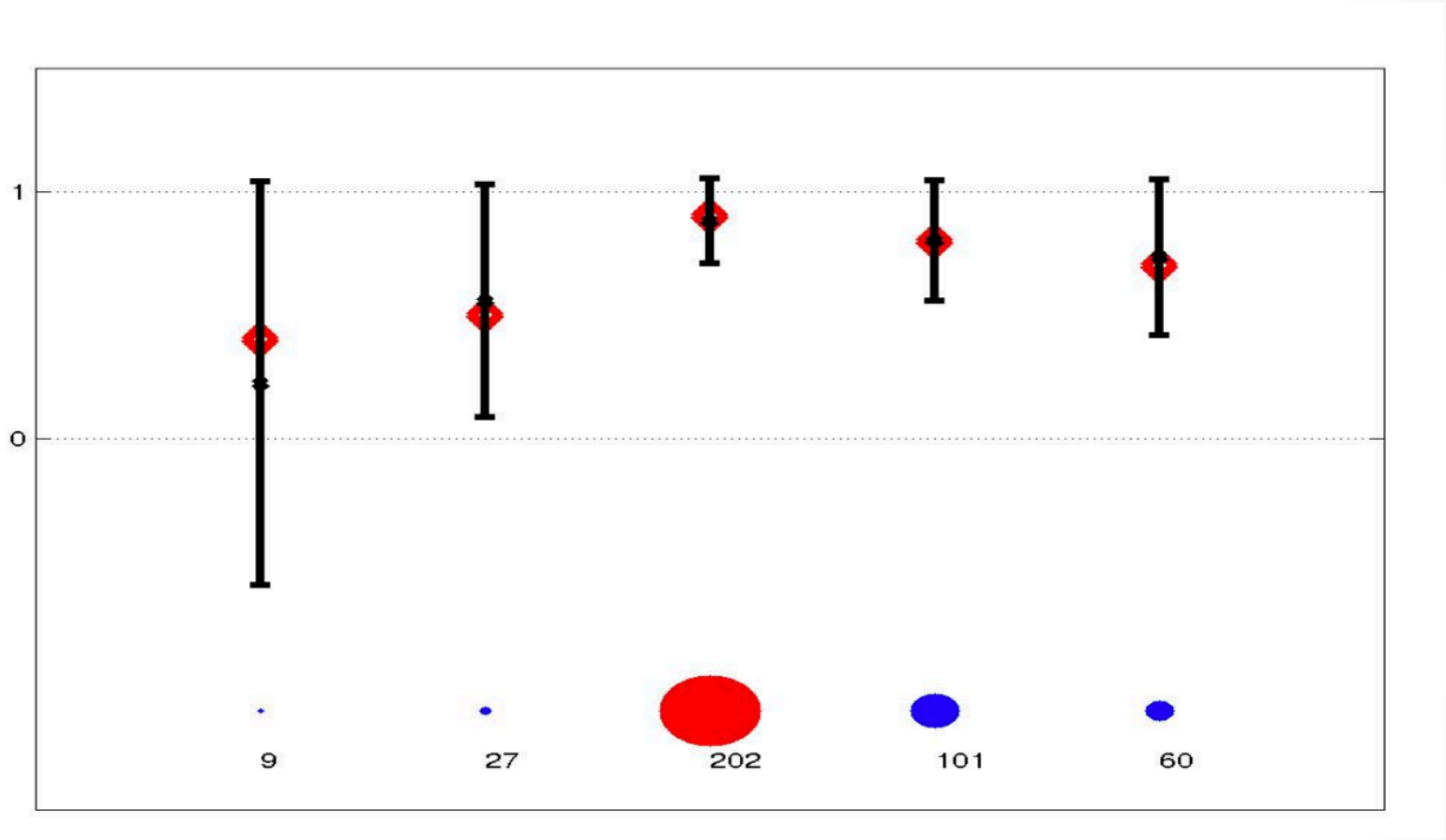
$$k_{t+1} = \arg\max \hat{\mu}_k(t) + \sqrt{\frac{a \log t}{n_k(t)}}$$

- $\mathrm{E}[n_k(t)] = O(\log T)$: Each sub-optimal arm pulled O(log T) times
- E[Reg(T)] = $(K - 1)O(\log T)$



Auer, P., Cesa-Bianchi, N., & Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Machine learning*, *47*(2-3), 235-256.

UCB (Upper Confidence Bound) Algorithm



How does this compare with the naïve approach?

- Let's say, you split equally for first 20% data and then play the best arm from there on

- Regret in this case is linear!

Alternatives to UCB

- Thompson Sampling

Summary: UCB overview

- UCB algorithm pulls sub-optimal arms much fewer times than the naïve approach

- The sequential selection allows UCB to choose arms smartly and reduce regret significantly!

Thompson Sampling: Which arm to pick at next round?

$\hat{\mu}_1(t) = 0.67$ $\quad$ $\hat{\mu}_2(t) = 1.5$ $\quad$ $\hat{\mu}_3(t) = 1.2$

$n_1(t) = 3$ $\quad$ $n_2(t) = 8$ $\quad$ $n_3(t) = 5$

- Assume rewards are coming from a gaussian distribution
- Construct posterior distribution on mean as

$$\mu_k \sim N\left(\hat{\mu}_k, \frac{1}{n_k(t)+1}\right)$$

- Generate fake samples from the posterior distribution of each arm
- Select arm with the highest value of the sample

Alternate Objectives

- So far, we have looked at the objective of maximizing the cumulative reward

- A common objective is Best-Arm identification
  - Identify the best arm with probability $1 - \delta$ in as few samples as possible

  - Given a budget of T pulls, identify the best arm with lowest error probability

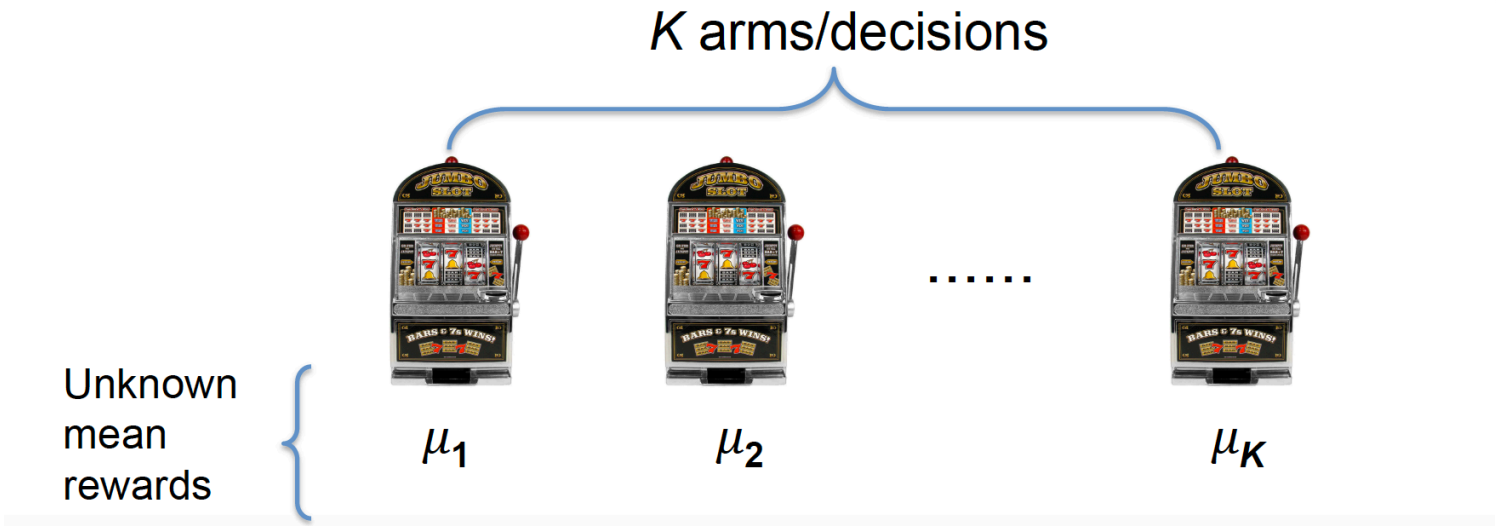
Hyperparameter Optimization

- Arms: Different hyperparameters
- Objective: Find the best hyperparameter setting which leads to lowest validation error
- Constraints: Fixed Budget

- Can use standard best-arm identification algorithms such as Sequential Elimination, LIL-UCB etc. to solve this problem
- For instance: see HyperBand paper (presented next week)

Bayesian Optimization

Discrete Multi-Armed Bandit

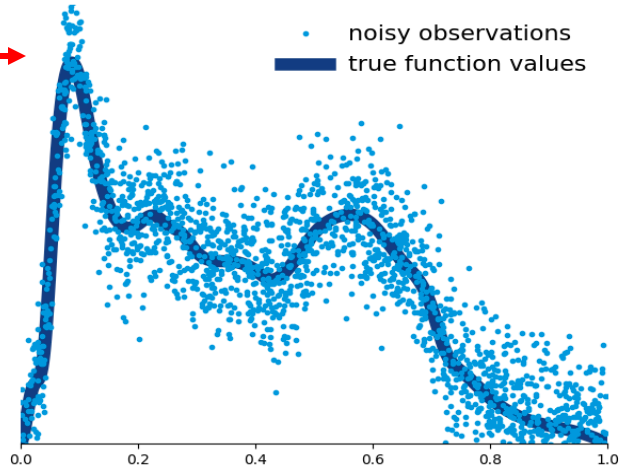


- At every round: decide which arm to play (k_t)
- Receive a random reward corresponding to the arm played
- Mean rewards are unknown

Goal: Maximize cumulative reward in T rounds

Continuous Multi-armed Bandit

Global Max



https://sigopt.com/blog/uncertainty-2-bayesian-optimization-with-uncertainty/

- At every round: decide which point to query (x_t)
- Receive a random reward corresponding to the point queried

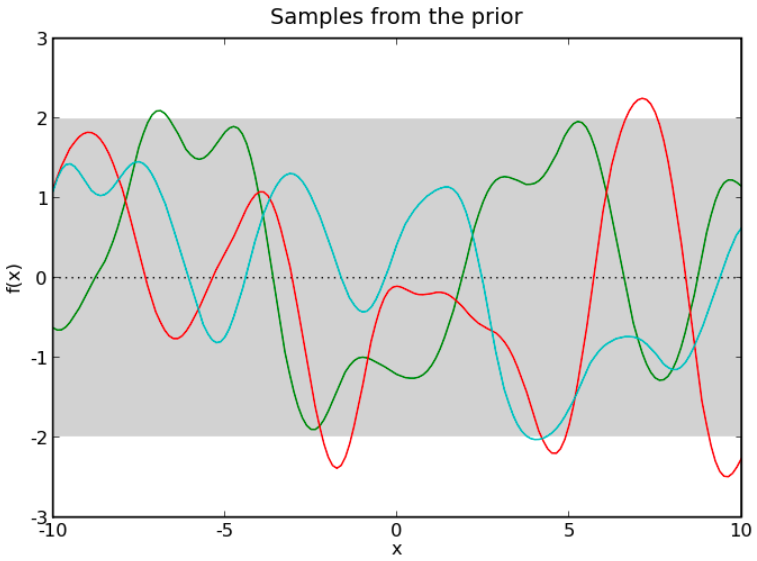
$$y_t = f(x_t) + \eta$$

- True function is unknown

Goal: Find the global maximum of f(x) with minimum queries

Gaussian Process Prior

- To design an algorithm we need to make some assumption about the structure of the function
- Parametric assumptions (For eg. Linear, quadratic etc.) are too restrictive
- Instead we assume that the function is drawn from a *Gaussian Distribution* over functions
- Essentially $[f(x_1),\ldots,f(x_N)] \sim N(0,\mathbf{K})$
- The choice of **K** captures our belief about the smoothness of the function
- For eg. $K_{ij} = \exp(-\frac{1}{2}||x_i - x_j||^2$

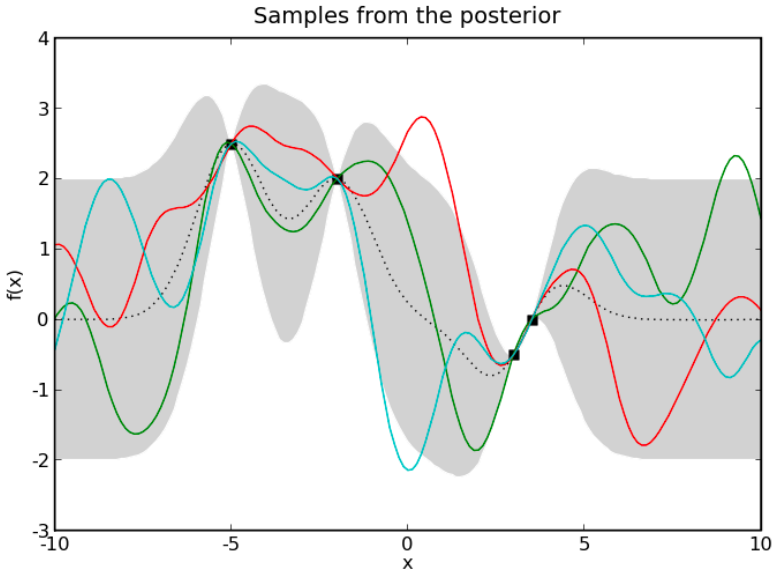


https://pythonhosted.org/infpy/gps.html

Gaussian Process Posterior

- A key advantage of using Gaussian process is that the posterior distribution is also Gaussian

- Given t samples $[f(x_1),\ldots f(x_t)]$, for any x_*, we have:



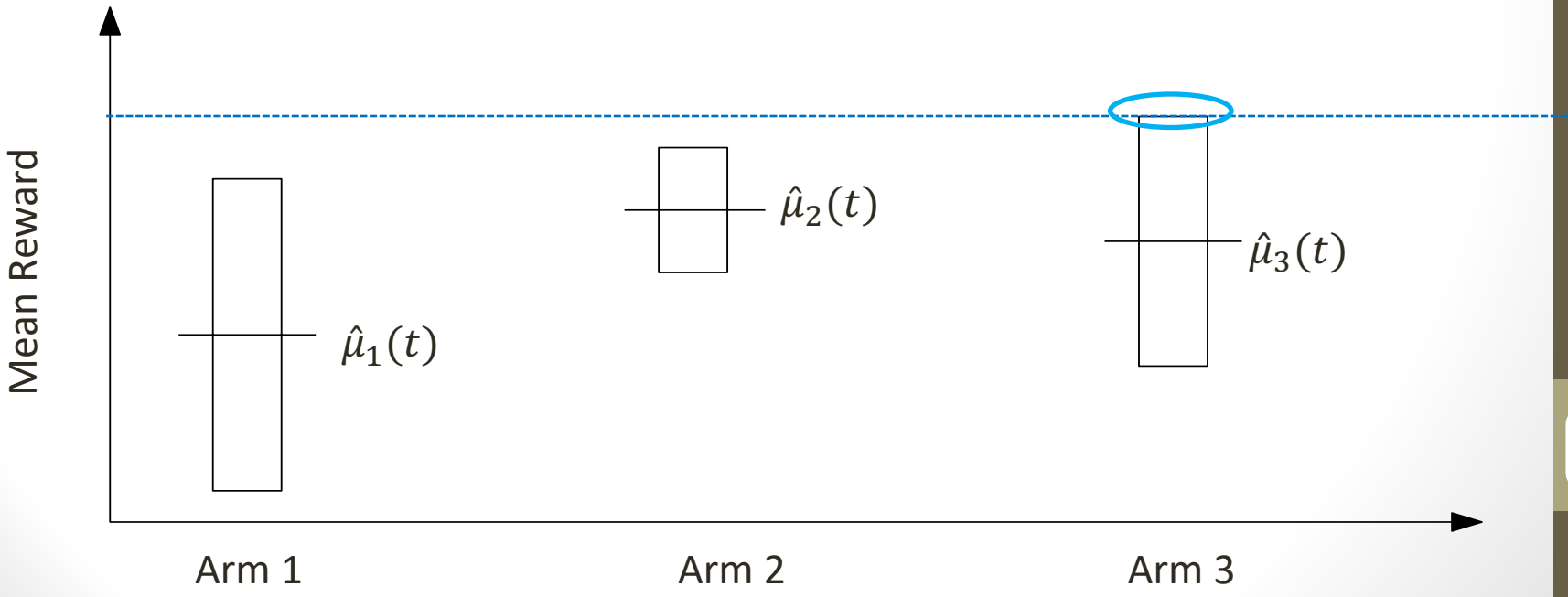
https://pythonhosted.org/infpy/gps.html

$$p\left(F(\boldsymbol{x}_*)|\mathbb{D},\boldsymbol{x}_*\right)=\mathcal{N}\left(\mu(\boldsymbol{x}_*),\sigma^2(\boldsymbol{x}_*)\right),$$
$$\mu(\boldsymbol{x}_*)=\boldsymbol{k}_*^T(\boldsymbol{K}+\sigma_w^2\boldsymbol{I})^{-1}\boldsymbol{Y},$$
$$\sigma^2(\boldsymbol{x}_*)=k_{**}-\boldsymbol{k}_*^T(\boldsymbol{K}+\sigma_w^2\boldsymbol{I})^{-1}\boldsymbol{k}_*$$

Recall: UCB

$$k_{t+1} = \arg\max \hat{\mu}_k(t) + \sqrt{\frac{a \log t}{n_k(t)}}$$

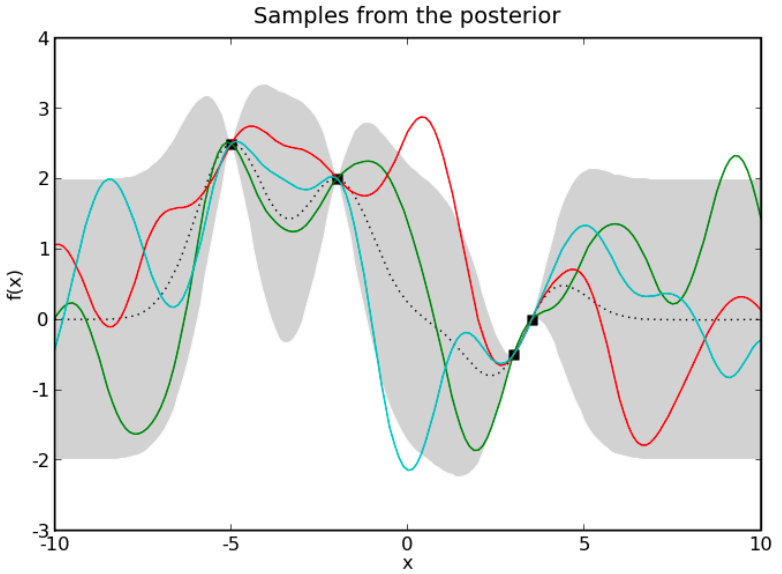
- Selects arms with large $\hat{\mu}_k(t)$ or small $n_k(t)$
- Confidence set shrinks as $n_k(t)$ increases



Bayesian Optimization: GP-UCB

- Assumption: f(x) is drawn from a Gaussian Process with kernel *k*
- Given: Samples $[f(x_1),\ldots f(x_{t-1})]$
- Goal: Find global maximum x*
- Algorithm:

$$x_t = \underset{x \in D}{\operatorname{argmax}}\, \mu_{t-1}(x) + \beta_t^{1/2} \sigma_{t-1}(x).$$



https://pythonhosted.org/infpy/gps.html

Reference: Srinivas et. al (https://arxiv.org/abs/0912.3995)

Recall: Thompson Sampling (TS)

$\hat{\mu}_1(t) = 0.67$

$n_1(t) = 3$

$\hat{\mu}_2(t) = 1.5$

$n_2(t) = 8$

$\hat{\mu}_3(t) = 1.2$

$n_3(t) = 5$

- Assume rewards are coming from a gaussian distribution
- Construct posterior distribution on mean as

$$\mu_k \sim N\left(\hat{\mu}_k, \frac{1}{n_k(t)+1}\right)$$

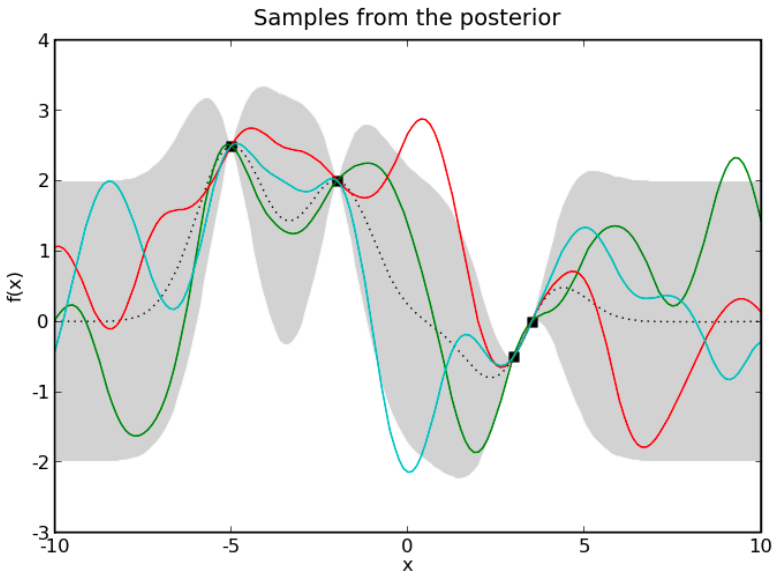
- Generate fake samples from the posterior distribution of each arm
- Select arm with the highest value of the sample

Bayesian Optimization: GP-TS

- Assumption: f(x) is drawn from a Gaussian Process with kernel *k*
- Given: Samples [$f(x_1)$,…$f(x_{t-1})$]
- Goal: Find global maximum x*
- Algorithm:

$$x_t = \text{argmax}_{x \in D} \hat{f}(x)$$

where $\hat{f}(x)$ is drawn according to the GP posterior



https://pythonhosted.org/infpy/gps.html

Reference: Ghosh et. al (https://arxiv.org/pdf/1705.06808.pdf)