Online Learning Applications

Part 4: Pricing

Introduction to pricing

Basic pricing setting

We will focus on this basic **pricing** problem:

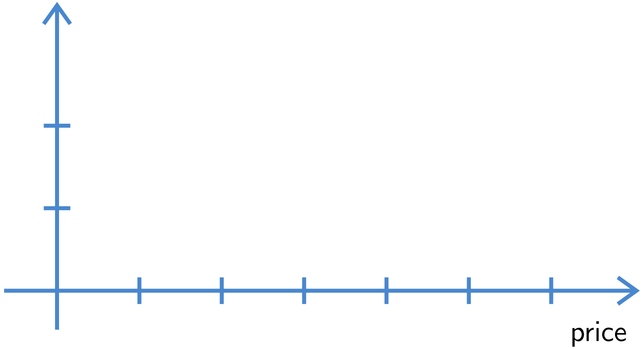
- Single **seller** (monopoly)
- Infinite units of a **single good**
- Multiple **costumers**

Seller model

- The seller has a cost c for each good (e.g., production cost)
- The seller can set a unique price p for all the units of the good

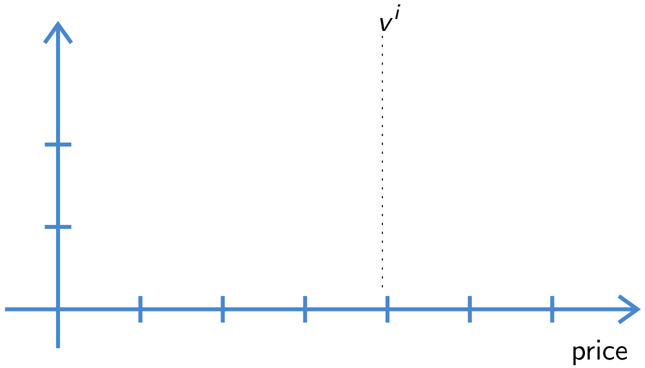
Buyer model

- Every customer i has a valuation v^i of the good



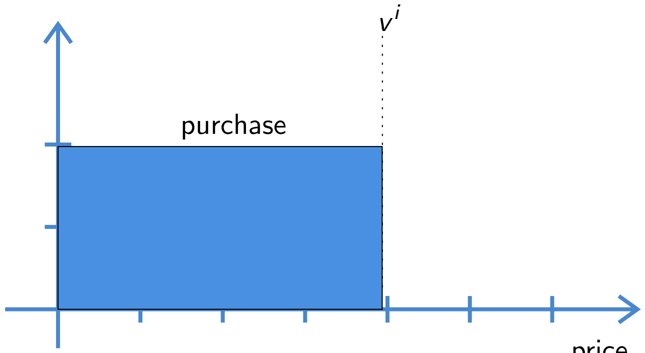
Buyer model

- Every customer i has a valuation v^i of the good



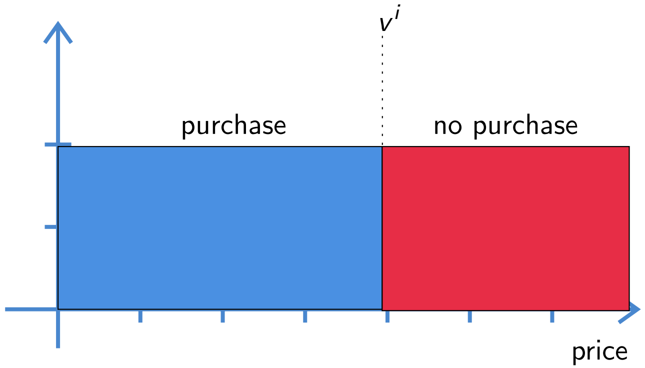
Buyer model

- Every customer i has a valuation v^i of the good



Buyer model

- Every customer i has a valuation v^i of the good



Aggregate Buyer model

- We can aggregate the behavior of all the buyer with the **demand curve**
- The demand curve $D(\cdot)$ specifies how many buyers will buy (in expectation) at each price
- The demand $D(p)$ specifies the expected number of purchases at price p.

Example: conversion rate

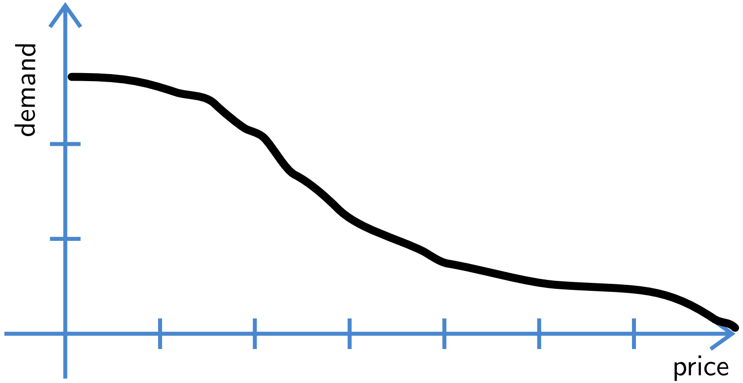
Conversion rate

What is the probability that a customer buys an item after she/he visited the webpage with a given price?

- We can aggregate the behavior of all the buyers with the demand curve
- The demand curve says how many buyers will buy at each price

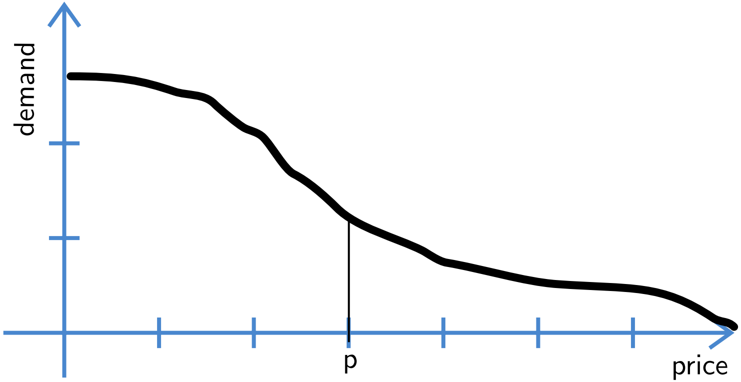
Profit maximization

The goal of the seller it to maximize profit = revenue - cost



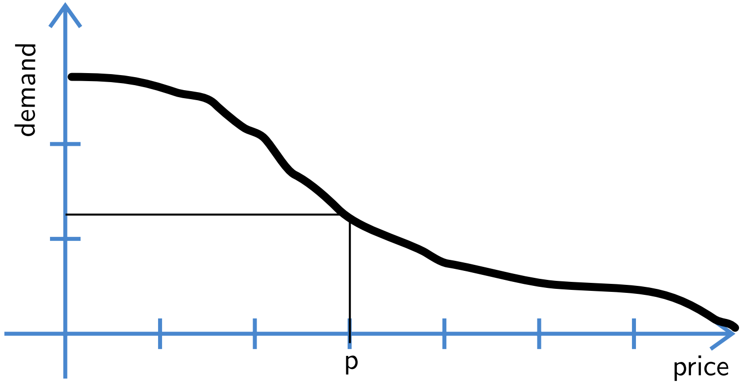
Profit maximization

The goal of the seller it to maximize profit = revenue - cost



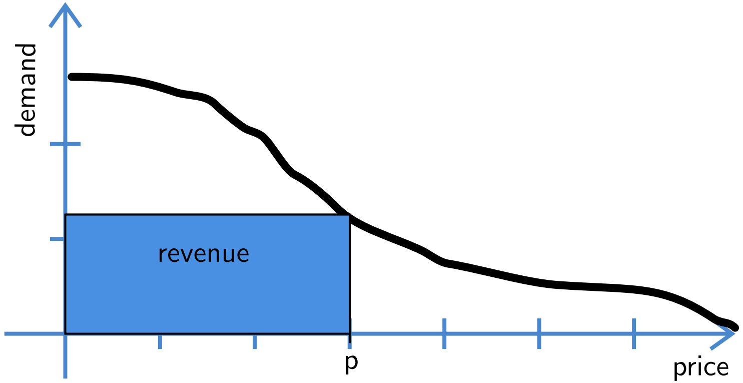
Profit maximization

The goal of the seller it to maximize profit = revenue - cost



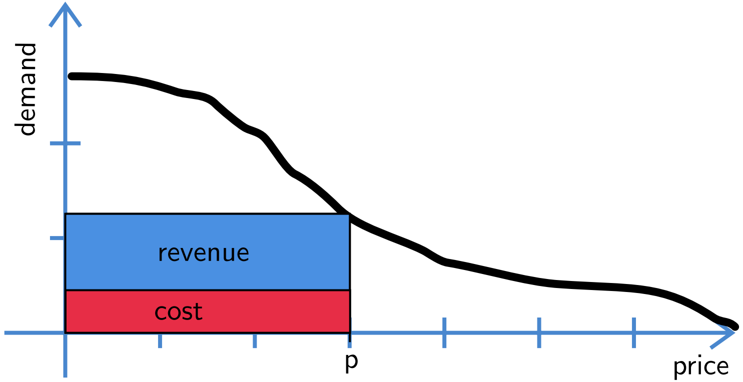
Profit maximization

The goal of the seller it to maximize profit $=$ revenue - cost



Profit maximization

The goal of the seller it to maximize profit = revenue - cost



Online pricing

Online pricing

We study a **online** pricing problem faced by a seller. At each round t:

- The seller sets a price p_t
- A number of buyers $n_t = D(p_t) + \eta_t$ buy the product, where η_t is a (subgaussian) random noise

goal

Maximize the seller's **profit**.

Online Pricing (formal)

This is an online learning problem:

- The seller sets a price $p_t \in [0,1]$ (chooses an arm **from a continuous set**)
- The seller receives a reward $r_t(p_t) = n_t(p_t - c)$, where $n_t = D(p_t) + \eta_t$
- The mean reward of a price p is $\mu(p) = D(p)(p - c)$

goal

Achieve sublinear pseudo-regret with respect to the optimal fixed price p^*, where the pseudo-regret is

$$\mathcal{R}_T = T \max_{p \in [0,1]} \mu(p) - \mathbb{E}\left[\sum_{t \in [T]} \mu(p_t)\right],$$

where the expectation is on the randomness of the algorithm.

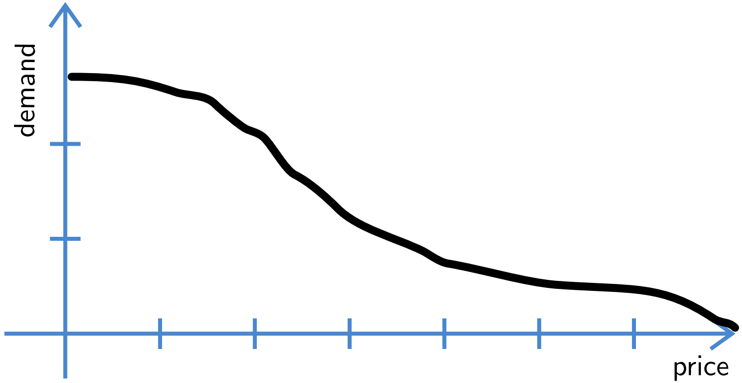
Online pricing with discretization

From continuous to discrete action set

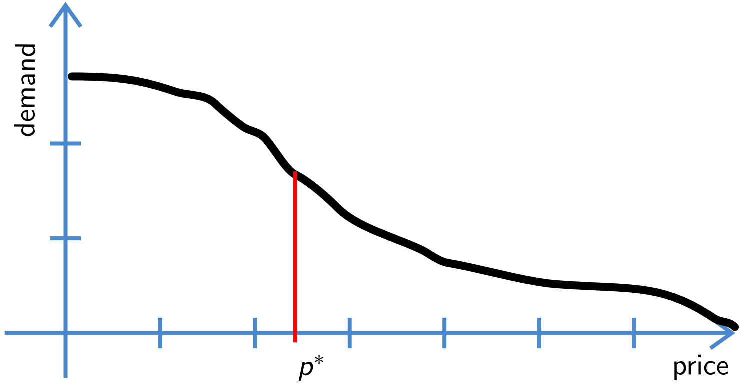
Idea: discretize the set of payments, i.e., consider only $p = 0, \epsilon, 2\epsilon, \dots, 1$.

- The problem reduces to a (stochastic) MAB with a finite number of actions $O(1/\epsilon)$.
- There is a tradeoff between
  - The **regret** of the MAB $\rightarrow$ **increases** with the number of arms
  - How **close** we are to the optimal price $\rightarrow$ **decreases** with the number of arms

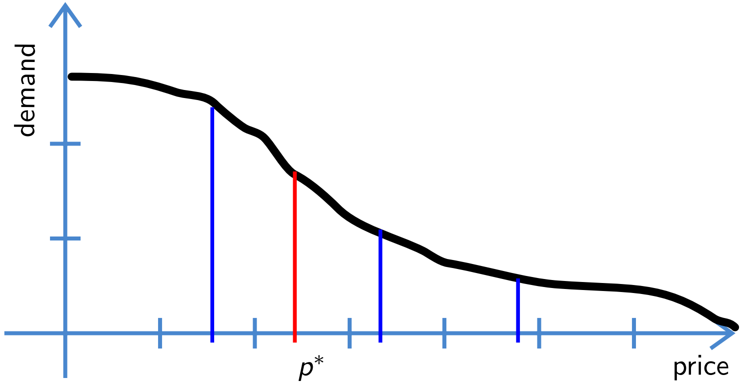
How many prices?



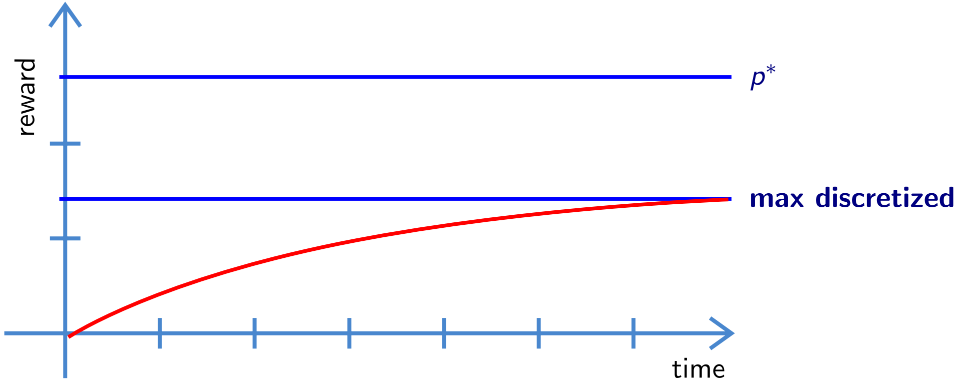
How many prices?



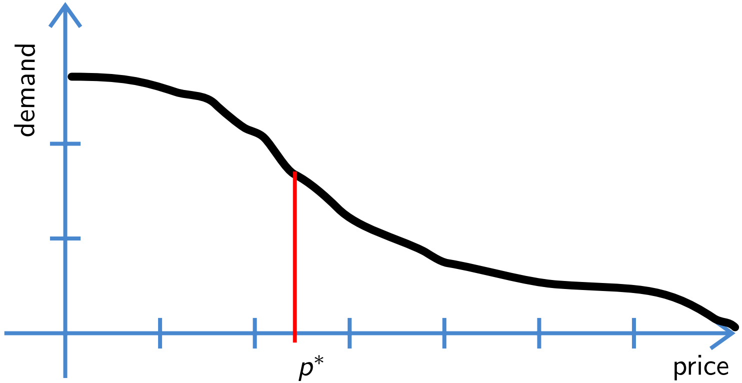
How many prices?



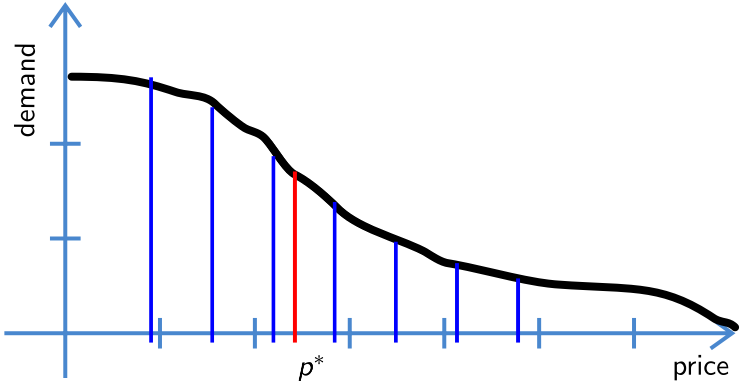
How many prices?



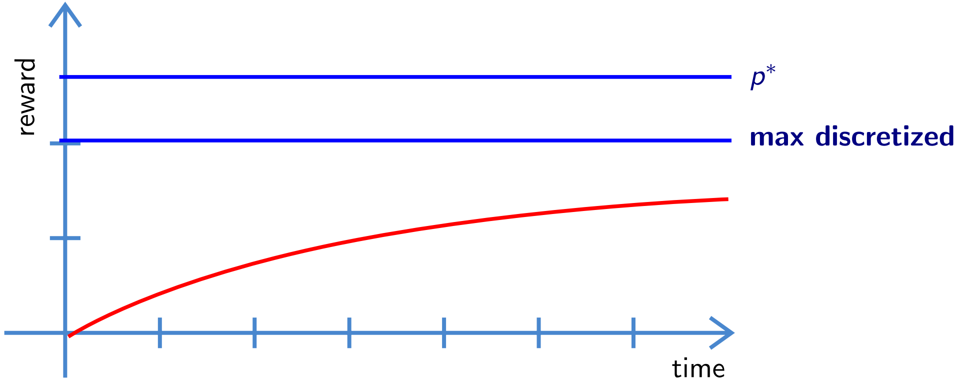
How many prices?



How many prices?



How many prices?



How many prices?

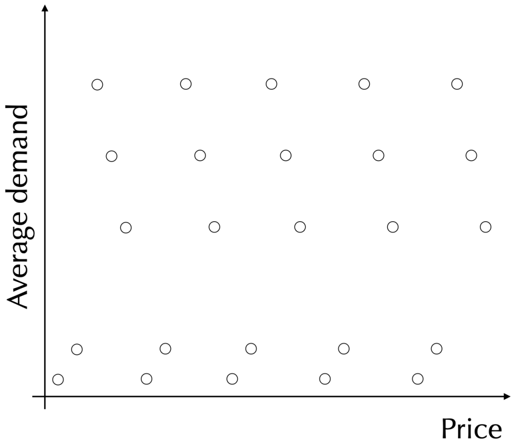
The optimal number of prices depends on the length of the horizon T.

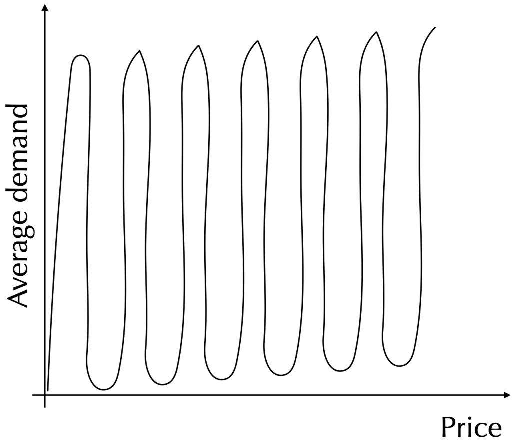
- Too **small** number of prices: too "far" from the optimal price
- Too **large** number of prices: no "enough time" to learn

We need to guarantee that the optimal price of the discretization performs similar to the optimal one.

- We will introduce some assumptions on the demand curve

Assumptions on the demand curve





No discontinuous function or with high derivative

Assumptions on the reward function

The expected reward of a price is $\mu(p) = D(p_t)(p - c)$.

Definition

A reward function is α-Lipschitz continuous if

$$|\mu(p) - \mu(p')| \leq \alpha|p - p'| \quad \forall p, p'$$

The reward in our pricing problem is usually Lipschitz continuous since the number of costumers that buy do not change too much if we slightly change the price. For simplicity, we will assume $\alpha = 1$.

Assumptions on the reward function

Proposition

Assume the reward is α-Lipschitz continuous. Then,

$$\max_{p\in\{0,\epsilon,2\epsilon,\dots,1\}} \mu(p) \geq \max_{p\in[0,1]} \mu(p) - \frac{\epsilon}{2}$$

Proof.

- There is always a price $\hat{p}$ at distance at most $\epsilon/2$ from the optimal price p^*.
- $\mu(\hat{p}) \geq \mu(p^*) - |p^* - \hat{p}| \geq \mu(p^*) - \frac{\epsilon}{2}$

$\square$

How many prices?

The optimal choice to optimize the **trade-off** between the **regret of the learning algorithm** and the **suboptimality of the discretization** is to take $\epsilon = T^{-1/3}$.

Theorem

Assume that the reward is Lipschitz continuous. Take $\epsilon = T^{-1/3}$, and apply UCB1 to the set of discretized prices $p = 0, \epsilon, 2\epsilon, \ldots, 1$. Then, the regret incurred by the algorithm is at most:

$$R_T = \widetilde{O}(T^{2/3}).$$

How many prices?

Proof.

- Let $p^* \in \arg\max_{p \in [0,1]} \mu(p)$
- Let $\hat{p} \in \arg\max_{p \in \{0, \epsilon, 2\epsilon, \dots, 1\}} \mu(p)$
- We proved that $\mu(\hat{p}) \geq \mu(p^*) - \frac{\epsilon}{2}$
- We proved that the pseudo-regret of UCB (for the discretized prices) is at most

$$O\left(\sqrt{KT\log(T)}\right) = O\left(\sqrt{\frac{1}{\epsilon}T\log(T)}\right) = O\left(T^{2/3}\log(T)\right)$$

$\square$

How many prices?

Proof.

- The pseudo-regret is at most

$$
\begin{aligned}
\mathcal{R}_T &= T(\mu(p^*)) - \mathbb{E}\left[\sum_{t\in[T]} \mu(p_t)\right] \\
&\leq T(\mu(\hat{p})) - \mathbb{E}\left[\sum_{t\in[T]} \mu(p_t)\right] + T\frac{\epsilon}{2} \\
&\leq O\left(T^{2/3}\log(T)\right) + T\frac{\epsilon}{2} \\
&= O\left(T^{2/3}\log(T)\right) + O\left(T^{2/3}\right) \\
&= O\left(T^{2/3}\log(T)\right)
\end{aligned}
$$

□

Online pricing with Gaussian processes

Can we do better?

We showed how discretization can provide sublinear regret. **Can we do better?**

Drawbacks of discretization

- Learning the reward for every possible price is impractical, as it requires to learn **many** random variables
- We are not exploiting that the reward function is **smooth**
- The reward of a price gives us **additional information** on the reward of close prices.

Solution: update all the reward estimators at the same time!

Assumptions on the reward function

We need to make some assumptions about the structure of the function:

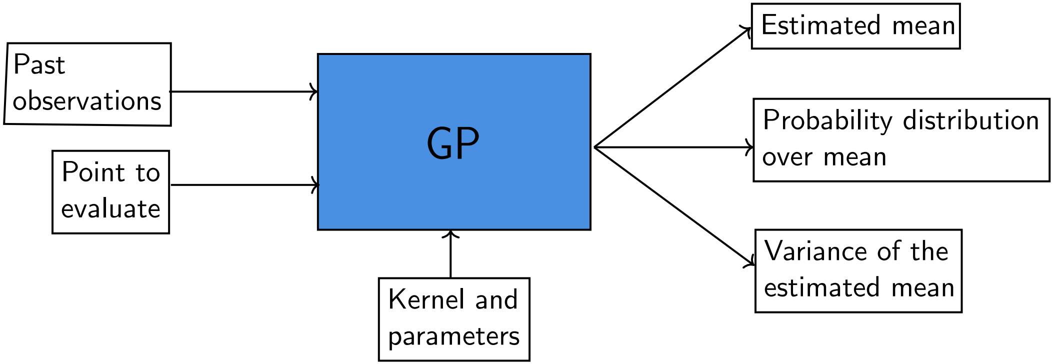
- Assumptions on the shape of the function (e.g., linear, polynomial) are quite restrictive
- We assume that the function can be modeled as a Gaussian Process (GP)
- We give as input only a kernel function that specifies the variance between two arms (i.e., prices)

Assumptions on the reward function

We need to make some assumptions about the structure of the function:

- Assumptions on the shape of the function (e.g., linear, polynomial) are quite restrictive
- We assume that the function can be modeled as a Gaussian Process (GP)
- We give as input only a kernel function that specifies the variance between two arms (i.e., prices) $\rightarrow$ **more during lab**
- The kernel provides a measure of similarity between two arms
- The GP fits the reward function $\mu(\cdot)$

GP as a Black-box algorithm



Advantages of GP-regression

Advantages of GP-regression:

- Provide a probability distribution instead of only the mean (**useful for TS-likes techniques**)
- Provide the standard deviation σ (**useful for UCB-like techniques**)

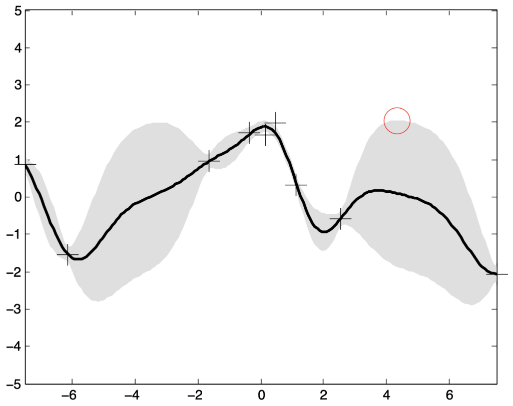
GP-UCB

GP-UCB

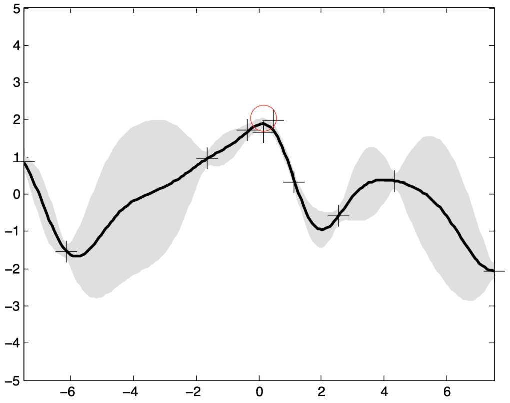
Algorithm: GP-UCB

```
1 input: continuous set of arms $A$, number of rounds $T$, kernel function $k(\cdot,\cdot)$,
    parameters $\beta_t$;
2 for $t = 1, \dots, T$ do
3     get $\mu_t(\cdot)$ and $\sigma_t(\cdot)$ from the GP;
4     $a_t = \arg\max_{a \in A} \mu_t(a) + \sqrt{\beta_t}\sigma_t(a)$;
5     play arm $a_t$;
6     observe reward $r_t(a_t)$;
7     update the GP with the new observation $r_t(a_t)$
```

Example: GP-UCB



(b) *Iteration t*



(c) *Iteration* $t + 1$

Picture from SRINIVAS, N. "Gaussian process optimization in the bandit setting: No regret and experimental design."

GP-Thompson Sampling

Algorithm: GP-THOMPSON SAMPLING

```
1 input: continuous set of arms $A$, number of rounds $T$, kernel function $k(\cdot,\cdot)$,;
2 for $t = 1, \dots, T$ do
3     use GP to sample a function $\theta(\cdot)$;
4     play arm $a_t \in \arg\max_{a \in A} \theta(a)$;
5     update the GP with the new observation $r_t(a_t)$;
```

Theoretical guarantees

Does GP-UCB and GP-TS provide theoretical guarantees?

- **Yes** (under some assumptions $\widetilde{O}(\sqrt{T})$)
- The theoretical analysis is complex and out of the scope of this course
- You will see the effectiveness of this methods during the lab

Pricing multiple items

Can we price multiple (different) items with correlated demands?

- **Yes**, GP-UCB and GP-TS work even in multidimensional action sets
- When a seller sells n different items, the action space is $p = [0, 1]^n$