An overview of multi armed bandits in the Bayesian context: Theory & applications

Rodrigo Rivera

Agenda

Abstract

Some decision problems have unknown probability distributions. To improve our understanding, we can collect new information. However, the cost of information can be significant or the number of choices overwhelming. In this talk, we will provide an overview of these problems and how they can be solved under the multi-armed bandit setting with a focus on one of its variants: the Bayesian optimization.

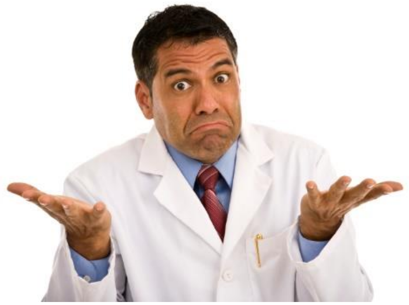
Motivation: Example from healthcare

We want to make (good) decisions under uncertainty

Setting

- Doctor wants to treat cancer patients
- All people are the same & only 2 drugs available.
- After diagnosis & without medicine, patient dies the same day
- With drug A they will be cured with probability p_1
- With drug B they will be cured with probability p_2

What should we do?



Motivation: Example from healthcare

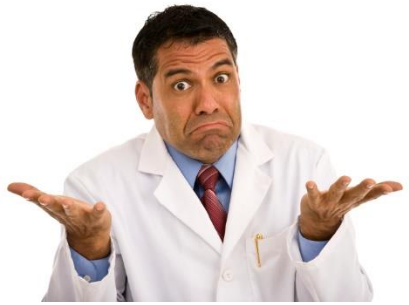
We want to make (good) decisions under uncertainty

Setting

- Doctor wants to treat cancer patients
- All people are the same & only 2 drugs available.
- After diagnosis & without medicine, patient dies the same day
- With drug A they will be cured with probability p_1
- With drug B they will be cured with probability p_2

What should we do?

- If we know the success rates, solution trivial: Pick the best

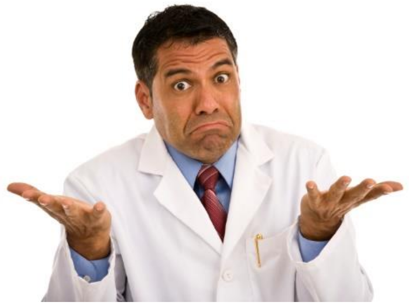


Motivation: Example from healthcare

We want to make (good) decisions under uncertainty

Problems

- In the real world, we never know the exact probabilities
- Our estimates might be wrong
- Even if we made the right conclusion, many had to die to test out a suboptimal drug
- What to do when a new drugs appears?



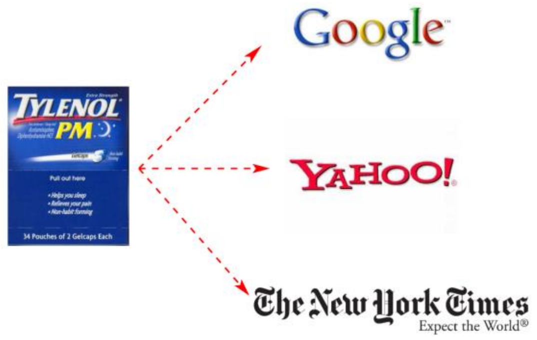
Motivation: Example from advertising

We want to make (good) decisions under uncertainty

Setting

- We want to start a marketing campaign
- We will book advertisement space across different websites
- We want to maximize our investment (increase profit, number of subscribers, etc)

What should we do?



Motivation: Example from advertising

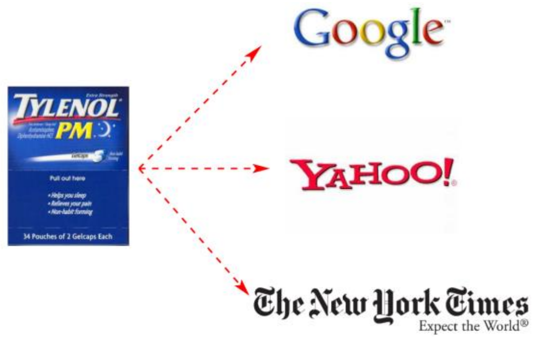
We want to make (good) decisions under uncertainty

Setting

- We want to start a marketing campaign
- We will book advertisement space across different websites
- Advertisers offer only a pay-per-impression model
- We want to maximize our investment (increase profit, number of subscribers, etc)

What should we do?

- Buy ad space in all websites with high-traffic



Motivation: Example from advertising

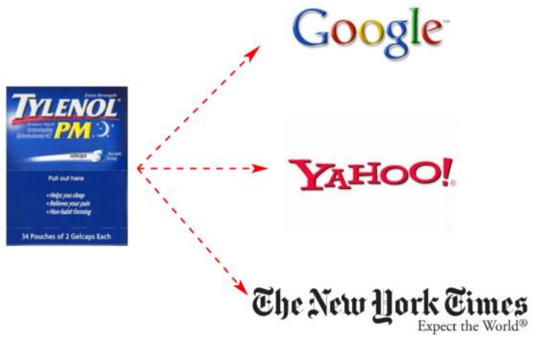
We want to make (good) decisions under uncertainty

Problems

- We have a limited budget
- The campaign will only run for a finite time
- Not all booked websites might have the same performance

What should we do?

- Buy ad space in all websites with high-traffic



Motivation: Example from advertising

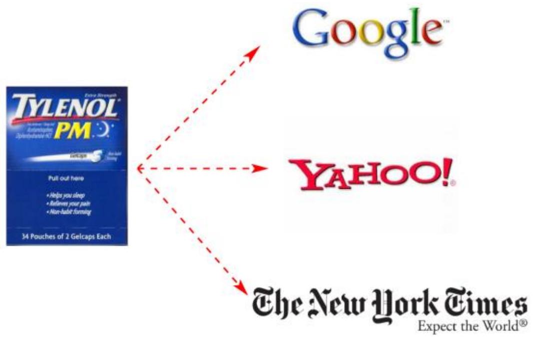
We want to make (good) decisions under uncertainty

Problems

- We have a limited budget
- The campaign will only run for a finite time
- Not all booked websites might have the same performance

What should we do?

- Buy ad space in all websites with high-traffic



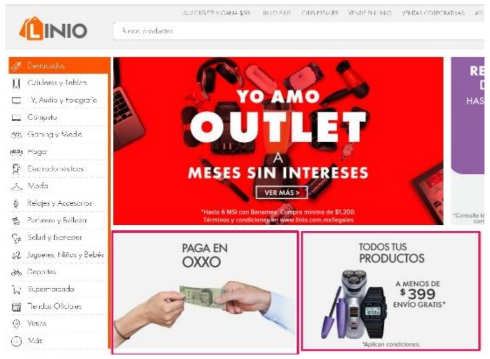
Motivation: Example from web optimization

We want to make (good) decisions under uncertainty

Setting

- We want to redesign a website
- We have multiple proposals
- The website has space for 2 large pictures and we have 51 of them.
- There are 51*50 = 2550 possible ways

What should we do?



Motivation: Example from web optimization

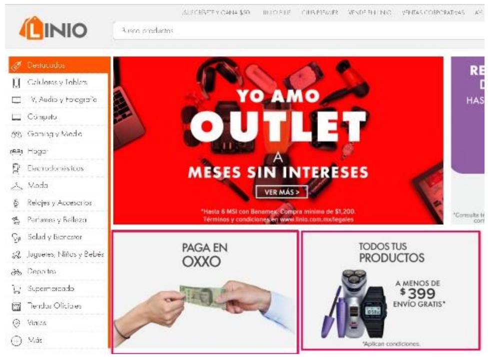
We want to make (good) decisions under uncertainty

Setting

- We want to redesign a website
- We have multiple proposals
- The website has space for 2 large pictures and we have 51 of them.
- There are 51*50 = 2550 possible ways

What should we do?

- The industry standard: A/B Test



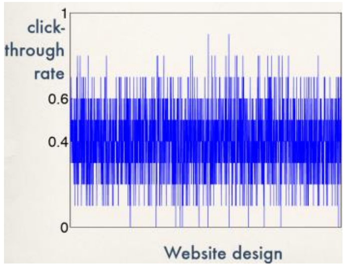
Motivation: Example from web optimization

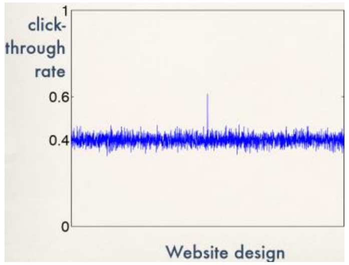
We want to make (good) decisions under uncertainty

Problems: Comparing many users requires time

- 2.55k designs
- CTR=click-through rate
- Best test has CTR=0.6
- Rest have CTR=0.4
- 10 users per design
- 25k users overall
- 2.5 days (10k visits / day)

- 2.55k designs
- CTR=click-through rate
- Best test has CTR=0.6
- Rest have CTR=0.4
- 500 users per design
- 1.25M users overall
- 4 months(10k visits / day)





Generalization

In all these problems, A/B testing does not give satisfactory answers

"Bandit problems embody in essential form a conflict evident in all human action:
Choosing actions which yield immediate reward vs. choosing actions(e.g., acquiring
Information or preparing the ground) whose benefit will come only later."

—P.Whittle(1980)

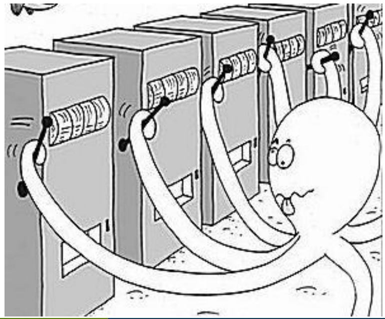
Multi-Armed Bandits: Problem Setting

MABs highlight the simple trade-off between exploration & exploitation

We have a casino with K one-armed bandit slot machines:

- Each arm *i* pays out 1$ with probability *pi,* if played; otherwise nothing
- $p_1, \ldots, p_k$ are fixed and we do not know any of their values
- Each time step t we pick a single arm a_t to play
- Based on our choice, we receive a return of $r_t \sim Ber(p_{a_t})$

How should we choose arms to maximize our expected return?



Multi-Armed Bandits: Problem Setting

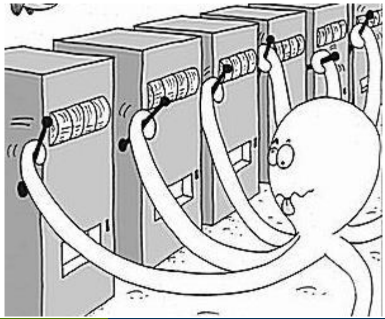
MABs highlight the simple trade-off between exploration & exploitation

More formally we have:

- $\exists n$ machines, each with a reward $y \sim P(y;\theta_i)$ and unknown θ_i
- $a_t \epsilon \{1, \dots, n\}$ is the chosen machine at time t
- $y_t \, \epsilon \, \mathbb{R}$ is the outcome with mean $< y_{a_t} >$
- f is the unknown function we want to optimize with x^* a member of $argmax_x f(x^*)$
- A strategy maps the history H_t to a new choice:
  - $\pi: [(a_1, y_1), \dots, (a_{t-1}, y_{t-1})] \longmapsto a_t$

Find a strategy π that

$max < \sum_{t=1}^{T} y_t >$ or $max < y_T >$



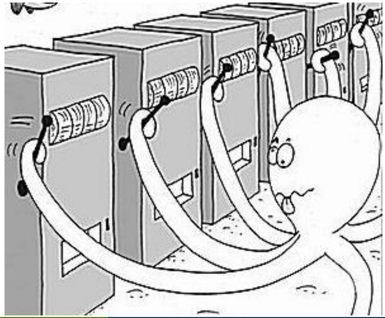
Excursus: Exploration vs Exploitation

Collect more data or increase reward

Choosing a machine has one of two consequences:

1. Exploration: Obtain more data about the machine
   Choose the next action a_t to $\min < H(b_t) >$, the entropy of the belief with $b_t(\theta) = P(\theta|\pi)$

2. Exploitation: Collect a reward
   Choose the next action a_t to $\max < y_t >$ immediate return



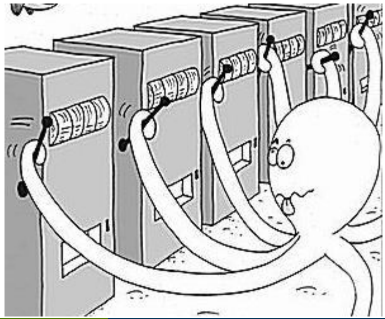
Multi Armed Bandits: Regret

We want to assess how much better things could have been

A learning algorithm might not attain good rewards on any bandit problem, but we want to get close to the best possible performance for that specific problem.

Regret: How much better we could have done, had we known the true function f from the start.

$$Regret(T, \pi, f) = \mathbb{E}\left[\sum_{t=1}^{T} f(x^*) - f(x_t)\right]$$



Overview of Bandits

In all these problems, A/B testing does not give satisfactory answers

Some of the most popular bandit algorithms are:

- Greedy
- Epsilon-Greedy
- Bayes optimal
- Upper Confidence Bound Bandit
- Thomson Sampling
- Epsilon-First

Overview of Bandits: Greedy Bandit

Greedily choose the best option without thinking on the future

Intuitively: Pick the arm with the higher success rate given the data Seen so far.

For each time step t:

$$Estimate\ \widehat{f_t} = \mathbb{E}[f|H_t]$$

$$Choose\ x_t \in argmax_x\ \widehat{f_t}$$

Problems:

- You may be choosing actions that do not increase your future understanding.
- By replacing the unknown f with the point estimate $\widehat{f_t}$ we overstate our knowledge.

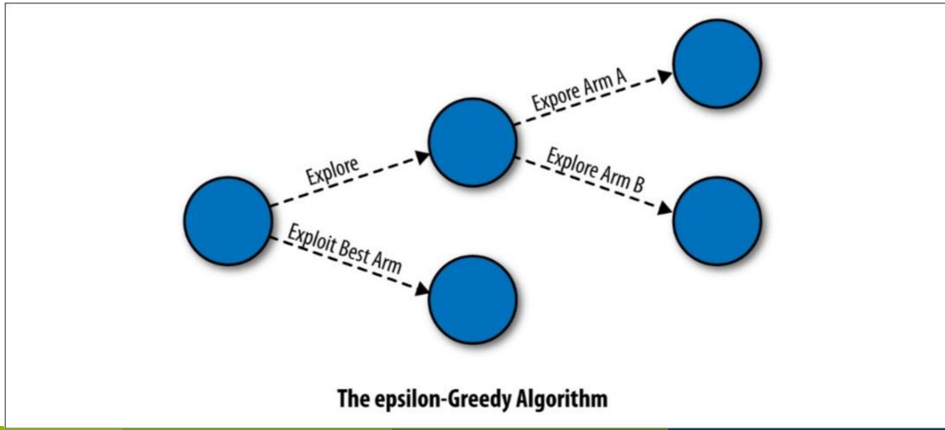
We cannot guarantee that it will ever learn the correct strategy!

Overview of Bandits: ε - Greedy Bandit

It will eventually converge to the optimal solution, but it takes time to learn

Similar to greedy, but we include a small probability ε, where we pick an arm at random.

```
For each time step t
    Estimate $\hat{f}_t = \mathbb{E}[f|H_t]$
    With probability $\varepsilon$ choose $x_t \sim Unif(\chi)$, else $x_t \in argmax_x \widehat{f_t}$
```



The epsilon-Greedy Algorithm

Overview of Bandits: ε - Greedy Bandit

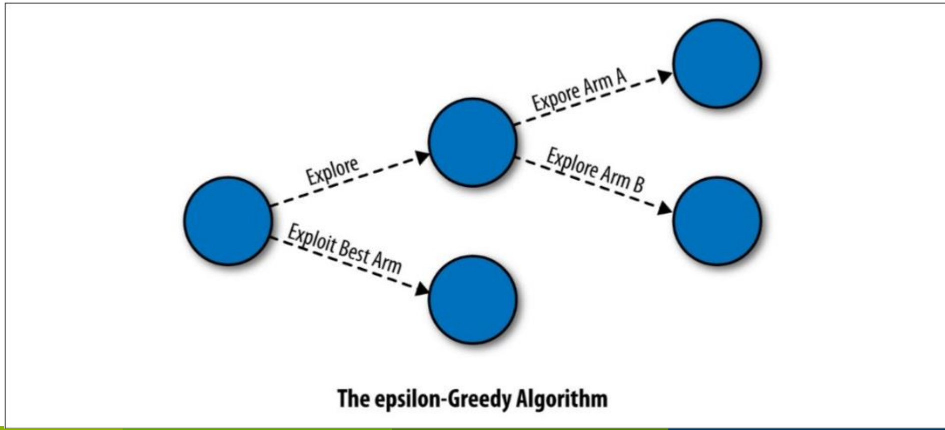
It will eventually converge to the optimal solution, but it takes time to learn

- A very popular algorithm for many settings of interest.
- If we choose a small enough ϵ we can guarantee that the algorithm will converge on the optimal solution:

$$Regret(T,\pi) = o(T)$$

Problem: In the casino setting it grows $\sim e^k$

It is slow, because it does not explore efficiently



The epsilon-Greedy Algorithm

Overview of Bandits: Bayesian Bandit

In some cases, the optimal solution can be stated using Bayesian statistics

Agent formulates its posterior belief and solve for the action with the highest expected long term rewards.

For each time step t:

$$Evaluate\ posterior\ \phi(\cdot|H_t)\ for\ f$$

$$Solve\ for\ x_t\ \in\ max_x\ \mathbb{E}_\phi \mathbb{E}\left[\sum_{j=t}^{T} f(x_j)\right]$$

Problem:

- In setting with generalization the problem is NP-hard

Overview of Bandits: Bayesian Bandit

It works how profitable each action will be in the long run and picks action with highest EV

Why it is good?

- It gives priority to exploration of arms that can be either promising or informative or both
- Exploration is not wasteful
- In some cases it can be computed effectively with Gittins indices (a real scalar value associated to the state of a stochastic process with a reward function and a probability of termination)
- In other cases Monte Carlo Tree Search can be applied to approximate to the Bayes-optimal solution

Overview of Bandits: Bayesian Bandit

It is hard because you need to consider how your action affects all subsequent choices

- Why is computing the Bayesian bandit so difficult?
  - We need to consider not only the chosen action, but also what we might learn about the function from this action and all subsequent choices (exponential search tree with all possible outcomes)
- Even with infinite computing power, our regret bound will never be better than:

$$Regret(T,\pi) = \Omega\left(\sqrt{KT}\right) \; \forall \; possible \; arm \; configurations \; 1, \ldots, K$$

Overview of Bandits: UCB-Bandit

It is a greedy heuristic, computationally efficient and guarantees bounded regret

For each time step t:

1. Form a set F_t containing the true f with high probability given data H_t
2. Choose the most optimistic outcome $x \in$ $arg_x max_{x \in X, f \in F_t} f(x)$

Remarks:

- We do not care about the whole function f, only about finding its maximum
- At each time step we use the data we have seen to narrow down the possible f to consider
- “Optimistic”: Pick the action with the highest expected return in the best possible case.

Overview of Bandits: UCB-Bandit

It is a greedy heuristic, computationally efficient and guarantees bounded regret

Why is it good?

- It offers a tractable solution with theoretical guarantees
- It incentivizes exploration but quickly focus on only the first arms
    - Instead of the arm with highest expected return, we pick the one with the highest mean plus three standard deviations

We can guarantee to be close to the optimal solution:

$$Regret(T, \pi) = O\left(\sqrt{KT}\right) \forall\, possible\ arm\ configurations\ 1, \dots, K$$

Overview of Bandits: UCB-Bandit

It is a greedy heuristic, computationally efficient and guarantees bounded regret

Problems:

- Two points require deep expertise and might be computationally intractable:
    - Constructing the optimistic set F_t given data H_t can be a lot of hard work and potentially cannot be used in practice.
    - Solving the double maximization $arg_x max_{x \in X, f \in F_t} f(x)$

Overview of Bandits: Thomson Sampling

The algorithm chooses an action randomly, according to the probability that it is optimal

Quantify our belief about the system in terms of a distribution rather than a point estimate. (Similar to the Bayesian solution but lower cost)

For each time step t:

Take a single sample $f_t \sim \phi(\cdot|H_t)$ posterior distribution for f

Solve for $x_t \in max_x f_t(x)$ maximum for that sample.

Intuitively: “We instantiate our beliefs randomly in each round and act optimally according to them”

Remark: Also known as posterior sampling or probability matching in the literature

ϕ is often a Beta distribution in the literature

Overview of Bandits: Thomson Sampling

The algorithm chooses an action randomly, according to the probability that it is optimal

Why is it good?

- It is computationally cheap & works well in practice
- It is simple to explain
- It can be easily implemented
- It has good guarantees
- The true function f and the single sample (imagined function) f_t are drawn from exactly the same distribution

We can guarantee to be close to the optimal solution:

$$Regret(T, \pi) = O\left(\sqrt{KT}\right) \forall \, possible \; arm \; configurations \; 1, \ldots, K$$

Epsilon First: A/B Test

A/B testing is a bandit called epsilon-first

Why not use A/B Test instead?

A/B test is a bandit strategy called ε – First: “A pure exploration phase is followed by a pure exploitation phase”

Exploration phase:

Choose $x_t \sim Unif(\chi)$

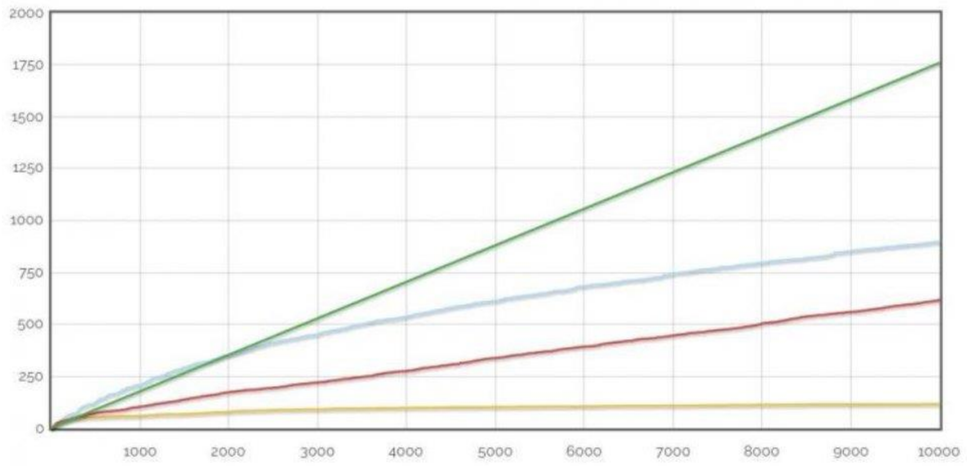
Exploitation phase:

$$Estimate\ \widehat{f_t} = \mathbb{E}[f|H_t]$$
$$Choose\ x_t \in argmax_x\ \widehat{f_t}$$

Bandit Benchmarking Example

The benefits of efficient experimentation are visible for large problems

Regret after n pulls



Green: Greedy
Blue: UCB
Red: E-Greedy (E = 0.1)
Yellow: Thomson

Number of pulls

- Being greedy is almost never a good strategy
- E-Greedy is suitable for small problems but deteriorates quickly
- UCB starts becoming interesting for large problems
- Thomson Sampling is one of the best options

Bayesian Optimization: Motivation

It is a relaxation of the assumption of binary payoffs

In the classic formulation, each slot machine is independent, sampling one arm tells us nothing about any other arm: We need to try every arm at least once.

We must relax our model to allow for shared information and the assumption of binary payoffs

Bayesian Optimization: Intuition

Bayesian optimization is an efficient model in situations where evaluations are costly

Intuitively:

It is a sequential model-based approach to solving problems, where we prescribe a prior belief over the possible objective functions and then sequentially refine this model as data are observed via Bayesian posterior updating.

Bayesian Optimization: Historical overview

Bayesian optimization is an efficient model in situations where evaluations are costly

- The idea dates back to Thomson in 1933 where he addressed the exploration vs exploitation trade-off with the likelihood that one unknown Bernouilli probability is greater than another given observational data.
- The actual term was coined in the 70s
- The technique is also known as Gaussian process bandits
- In the mid 2000’s, the interest in BO resurrected with the increase in computing power and the realization that it can be used for finding good hyperparameters

Bayesian Optimization: Definition

It is a relaxation of the assumption of binary payoffs

Environment described by an unknown (expensive) function

$$f\colon \chi \longrightarrow \mathbb{R}$$

$$x_{opt} = argmax_{x \epsilon \chi} f(x)$$

- Where χ is some design space of interest and often but not necessarily $\chi \subset \mathbb{R}^d$.
- At each time step t we choose an action $x_t \in \chi$ and receive in return a noise corrupted (stochastic) output $r_t = f^*(x_t) + w_t \in \mathbb{R}$ where w_t is some zero-mean noise such that $\mathbb{E}[r_t | f(x)] = f(x)$

Bayesian Optimization: Definition

It is a relaxation of the assumption of binary payoffs

Environment described by an unknown (expensive) function

$$f\colon \chi \longrightarrow \mathbb{R}$$

$$x_{opt} = argmax_{x \epsilon \chi} f(x)$$

- Function f can be stochastic, non-convex or even non-continuous. Often in literature is assumed that it is linear $f^*(x) = x^T\theta^*$ or drawn from some Gaussian process. It can be evaluated at any arbitrary query point x within the domain

Bayesian Optimization: In Summary

It is a relaxation of the assumption of binary payoffs

BO consists of two key pieces:

1. A probabilistic surrogate mode consisting of:
    1. A prior distribution capturing our beliefs about the behavior of the unknown objective function
    2. An observation model describing the data generation mechanism
2. A loss function describing how optimal a sequence of queries are: Regret (simple or cumulative).
    1. The loss is minimized to select an optimal sequence of queries
    2. After observing the output of each query, prior updated to produce a more informative posterior distribution over the space of objective functions.

Bayesian Optimization: Algorithm

It is a relaxation of the assumption of binary payoffs

The algorithm looks as following:

For each time step t:

- Select new x_{n+1} by optimizing acquisition function α
    - $x_{n+1} = argmax_x = \alpha(x; \mathbb{D}_n)$
- Query objective function to obtain y_{n+1}
- Augment data $\mathbb{D}_{n+1} = \{\mathbb{D}_n, (x_{n+1}, y_{n+1})\}$
- Update statistical model

- end

Bayesian Optimization: Definition

It is a relaxation of the assumption of binary payoffs

Problem:
In the minimum expected risk framework described previously, the true sequential risk up to the full evaluation budget is typically computationally intractable.

Solution:
Use myopic heuristics (acquisition functions)

Bayesian Optimization: Acquisition Function (1)

The representation of exploration vs exploitation

- Defines a balance between exploring new areas in the objective space and exploiting areas already known to have favorable values.
- Optima located where the uncertainty in the surrogate model is large (exploration) and/or where the model prediction is high (exploitation)
- Has to be cheap to evaluate (in relation to the expense of evaluating blackbox f) or approximate
    - As a consequence, it is much easier to optimize than the original objective function.

Bayesian Optimization: Acquisition Function (2)

The representation of exploration vs exploitation

- Common acquisition functions:
  - Probability of improvement
  - Expected improvement
  - Upper Confidence Bounds
  - Thomson Sampling
  - Entropy search

Bayesian Optimization: Acquisition Function (3)

A closer look to the 3 most popular acquisition functions

$\mu^+ = argmax_{x_i \in x_{1:t}} \mu(x_i)$ where μ^+ is the best observed value

- Probability of improvement (Kushner 1964)
    - We try to maximize probability that function evaluated at the point we will try next is higher than the one we have seen so far.
    - $PI(x) = P(f(x) \geq \mu^+ + \xi) = \Phi(\frac{\mu(x) - \mu^+ - \xi}{\sigma(x)})$ where Φ cumulative area under the curve (since Gaussian) & ξ is small in case we sample an area with low variance(control ex-expl)
    - Find the x that maximizes Φ
    - One that is least used in practice, but the simplest we can do
    - Intuitively: PI chooses the point with the largest CDF of the Gaussian greater than our current maxima

Bayesian Optimization: Acquisition Function (4)

A closer look to the 3 most popular acquisition functions

$\mu^+ = argmax_{x_i \in x_{1:t}} \mu(x_i)$ where μ^+ is the best observed value

- Expected improvement (Mockus 1978)
  - Maximize expected utility (minimize expected cost): $EU(a) = \sum_x u(x,a) P(x \mid data)$
  - Equivalent to choosing point minimizing the distance to the objective evaluated at max x^*: $x_{n+1} = argmin_x \mathbb{E}\left(||f_{n+1}(x) - f \quad (x^*)|| \,\middle|\, \mathbb{D}_n\right)$ where f is the true function & f_{n+1} the functions sampled from our GP
  - Catch: We do not know the true objective at the maximum, so we have to do maximum expected improvement.

Bayesian Optimization: Acquisition Function (5)

A closer look to the 3 most popular acquisition functions

$\mu^+ = argmax_{x_i \in x_{1:t}} \mu(x_i)$ where μ^+ is the best observed value

- Expected improvement (Mockus 1978)
  - To overcome this, Mockus proposed an alternative:
    - $x = argmax_x \mathbb{E}(\max\{0, f_{n+1}(x) - f^{max}\} | \mathbb{D}_n)$ *and* f^{max}*is* $\mu^+ + \xi$
  - Analogously, we can obtain an analytical expression:
    - $EI(x) = (\mu(x) - \mu^+ - \xi)\Phi(Z) + \sigma(x)\phi(Z)$ if $\sigma(x)$>0 where $\phi(\cdot)$ is the PDF and $\Phi(\cdot)$ the CDF of the standard Normal and $Z = \frac{\mu(x) - \mu^+ - \xi}{\sigma(x)}$
  - Very similar to probability of improvement

Bayesian Optimization: Acquisition Function (6)

A closer look to the 3 most popular acquisition functions

$\mu^+ = argmax_{x_i \in x_{1:t}} \mu(x_i)$ where μ^+ is the best observed value

- Expected improvement (Mockus 1978)

  - Intuitively: A weighted combination of the CDF and PDF by the expected increase & uncertainty. Mixture helps balance ex-expl problem. New points will be selected from both unknown regions with high variance & known areas that we think are likely to contain maxima.

Bayesian Optimization: Acquisition Function (7)

A closer look to the 3 most popular acquisition functions

$\mu^+ = argmax_{x_i \in x_{1:t}} \mu(x_i)$ where μ^+ is the best observed value

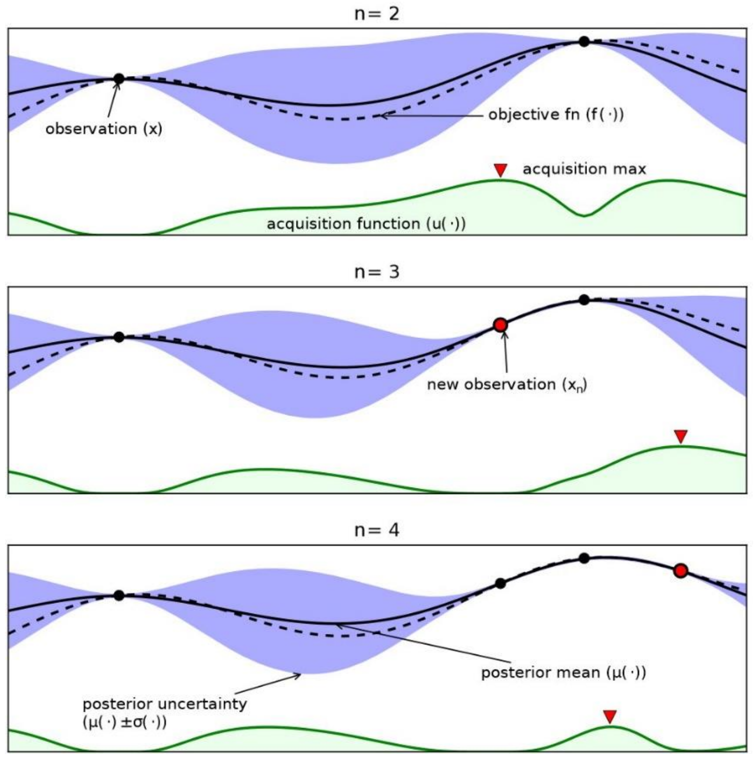
- Upper Confidence Bounds (Srinivas et al. 2010)
  - We define a regret and a cumulative regret:
    - $r(x) = f(x^*) - f(x)$ and $R_T = r(x_1) + \cdots + r(x_T)$
  - $GP - UCB(x) = \mu(x) + \sqrt{\nu\tau_t}\sigma(x)$ as the tradeoff between the mean and the variance

  - Intuitively: We pick an arbitrary bound on the CDF of the GP & calculate cost on this bound. Sample from the point with the highest bound.

Bayesian Optimization: Three iterations

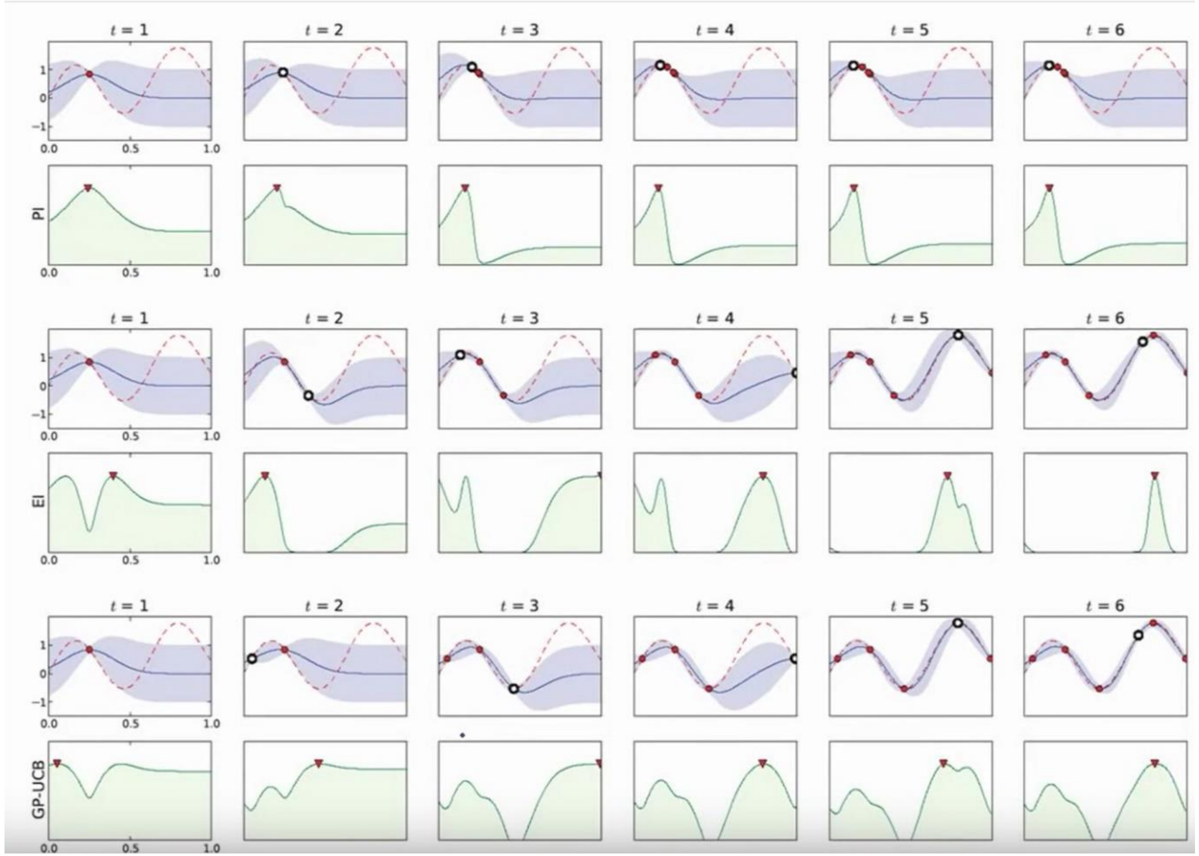
Acquisition is high where high objective is predicted and uncertainty is high

- Mean & confidence intervals estimated with a probabilistic model of the objective function.
- Objective function shown, in practice unknown.
- Acquisition is high where:
  - The model predicts a high objective (exploitation)
  - The prediction uncertainty is high (exploration).
- Remark: Area on the far left is not sampled - High uncertainty but little improvement.



Bayesian Optimization: Comparing AFs

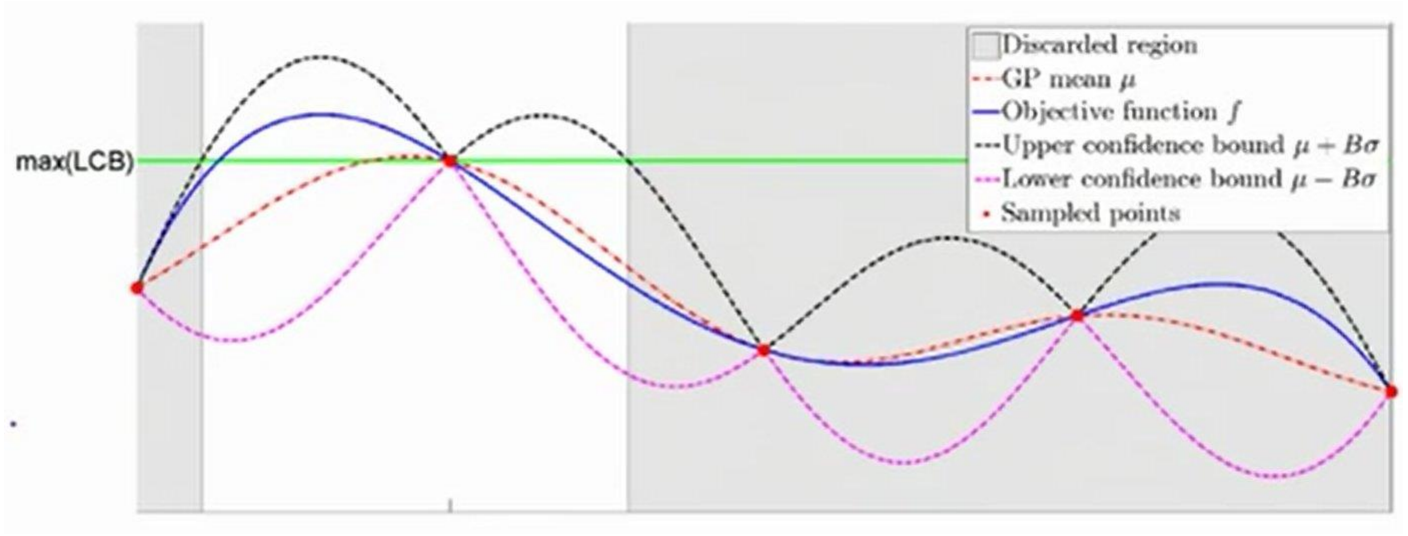
The three most common acquisition functions behave differently



- Some Afs tend to be more greedy than others
- UCB & EI behave similarly
- PI does much worse.
  In certain cases PI performs better (e.g., you know the maximum)

Bayesian Optimization: Why it works?

The three most common acquisition functions behave differently



- The upper bound tells us “this is the best I can do”
- If the best I can do, is worse than the worse I can do, ignore, even if there are regions with high variance.
- Under these assumptions, we can ignore a lot of the space & focus in the relevant areas.

Bayesian Optimization: Applications

It is a very flexible framework applied in a myriad of settings

- Some use cases for BO are:
  - Web optimization (A/B Testing) (Minimize opportunity costs)
  - Recommender systems (Optimize multiple suggestions)
  - Reinforcement Learning (Tune parameters of neural network policy)
  - Environmental monitoring & sensor network (Expensive to activate so where should be located & when activated?)
  - Preference learning (Set parameters in a model & ask for feedback)
  - Automatic machine learning & hyperparameter tuning (Tune deep belief networks, MCMC methods, convolutional neural networks)

Challenges of Bayesian Optimization

Bayesian optimization is promising, but why is not mainstream?

- Sequentially of experiments: We want to parallelize
- Lack of standard software: We do not have a strong popular open source library for BO
- Lack of robustness to poor defaults: BO is fragile and has its own hyperparameters
- Impact of selected acquisition function: They are not interchangeable & they have to be evaluated based on the given problem

Bayesian Optimization: Open Source Libraries

A list of popular open source software libraries for Bayesian optimization

Package	Language	Model
SMAC	Java	Random forest
Hyperopt	Python	Tree Parzen estimator
Spearmint	Python	Gaussian Process
Bayesopt	C++	Gaussian Process
PyBO	Python	Gaussian Process
MOE	Python / C++	Gaussian Process

Bayesian Optimization: Process

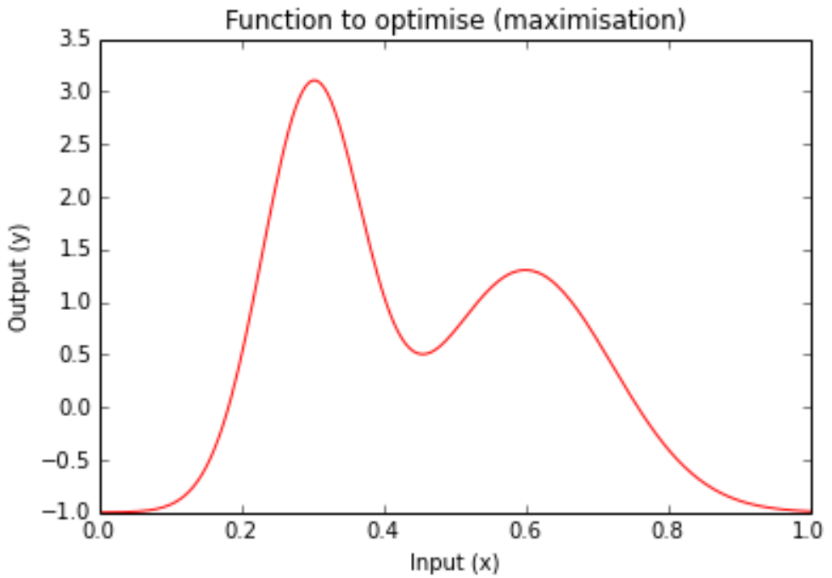
In all these problems, A/B testing does not give satisfactory answers

1. We start with an unknown function (black box)
    1. For the purpose of this example, we define our function.
    2. Any function can be used.
    3. We choose a mixture of two gaussians
2. We want to find its maxima.

Bayesian Optimization: Process

In all these problems, A/B testing does not give satisfactory answers

1. We start with an unknown function (black box)
   1. For the purpose of this example, we define our function.
   2. Any function can be used.
   3. We choose a mixture of two gaussians
2. We want to find its maxima.



Bayesian Optimization: Process

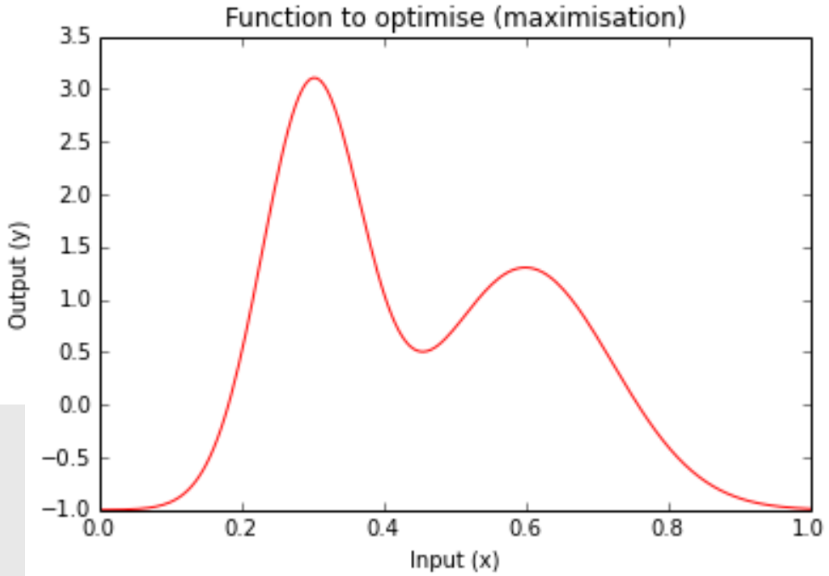
Our goal is to find the global maxima in as few samples as possible

3. We create a Gaussian Process (GP) to model our unknown function.

a) In BO, attributes of the GP such as mean and variance are used to sample successive points.

Suitable for situations where:

- Cost function is costly to evaluate
- MCMC techniques would not work

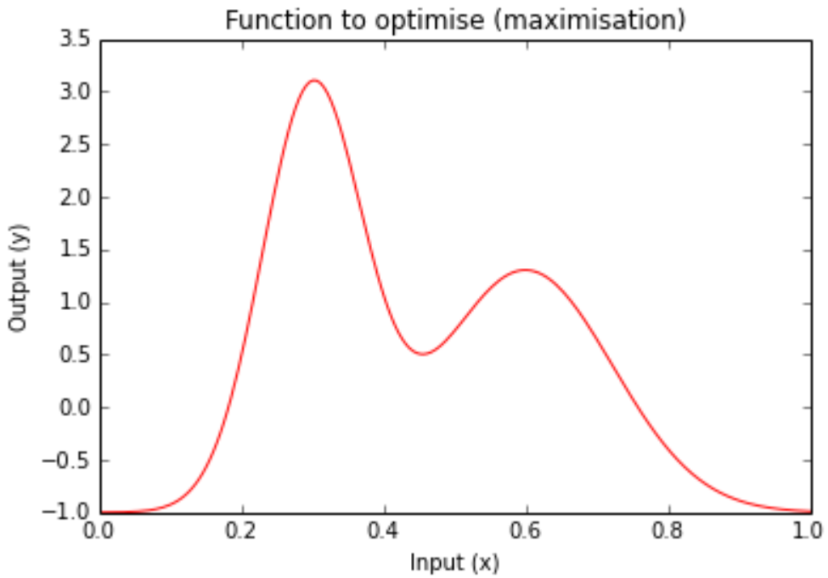


Bayesian Optimization: Exploring AFs

Our goal is to find the global maxima in as few samples as possible

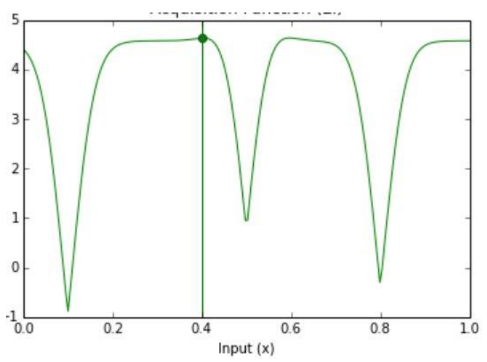
Excursus:
For demonstration purposes, we take some liberties that are rarely known in advance:

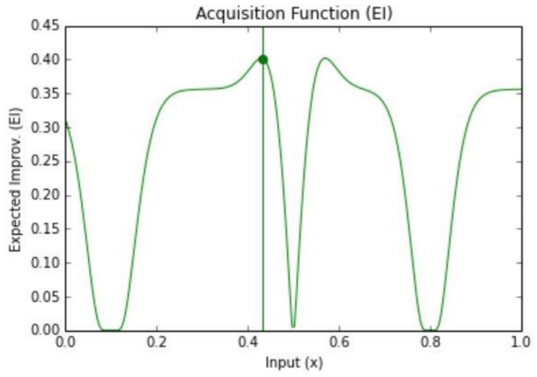
- We initialize with an RBF kernel with constant length scale and output variance.
- Function is noiseless (very often not the case)
- We take 3 arbitrary points.

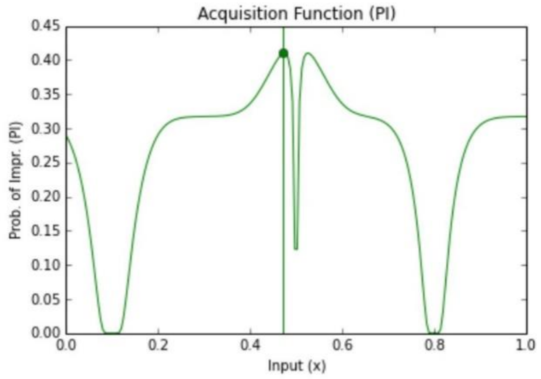


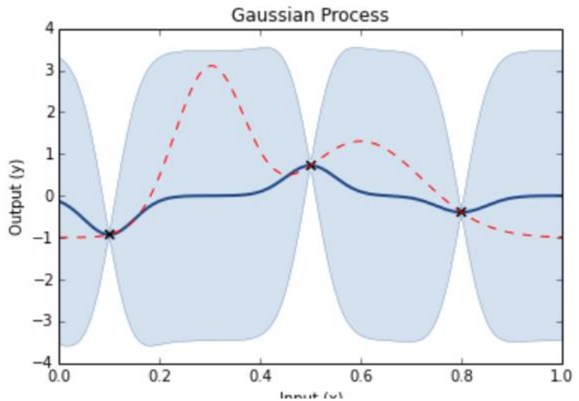
Bayesian Optimization: Exploring AFs (1)

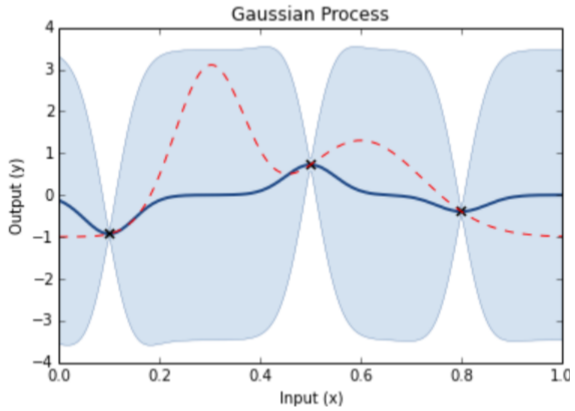
What happens if we do not update our acquisition function?

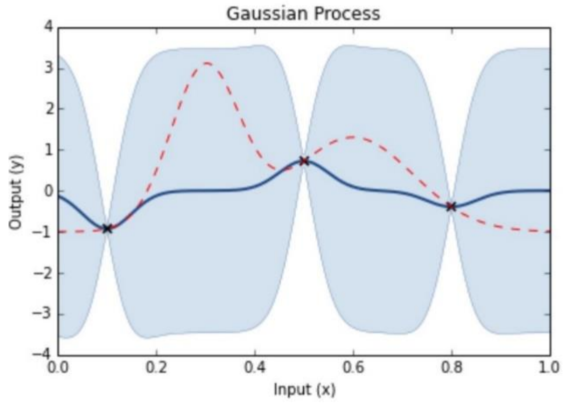






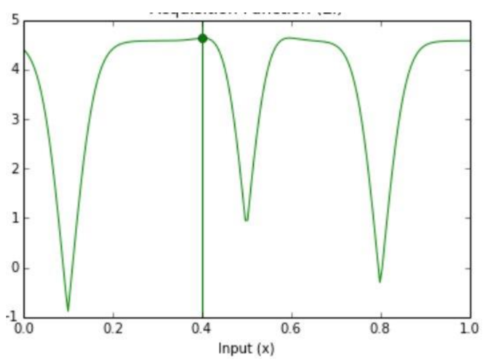


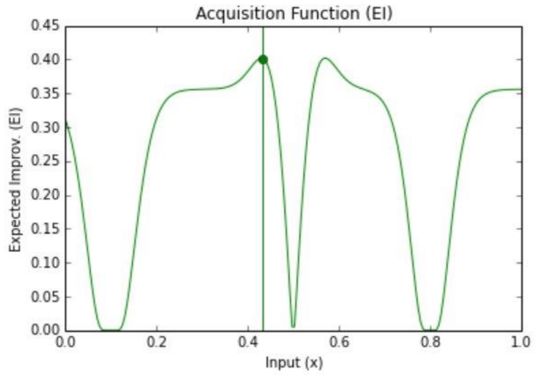


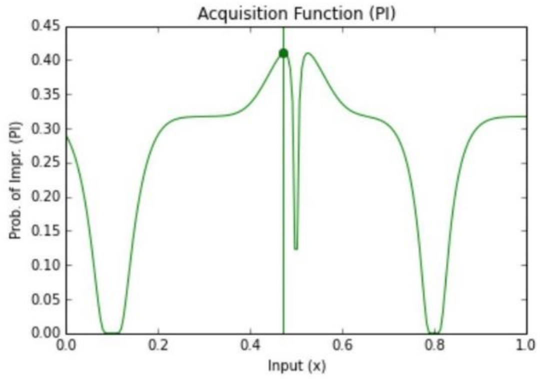


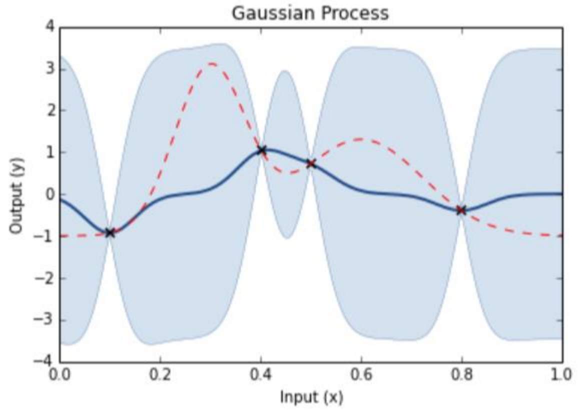
Bayesian Optimization: Exploring AFs (2)

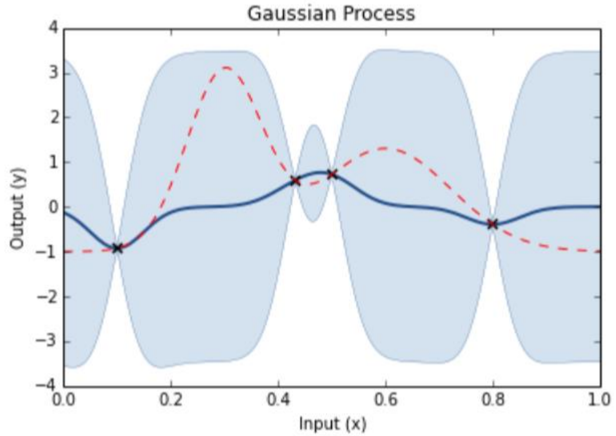
Our goal is to find the global maxima in as few samples as possible

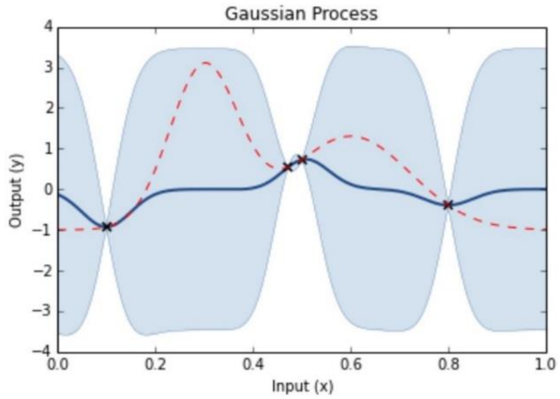






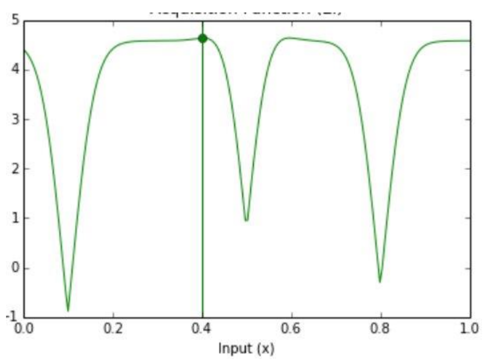


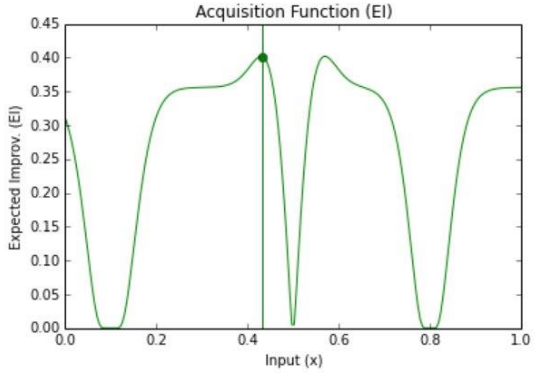


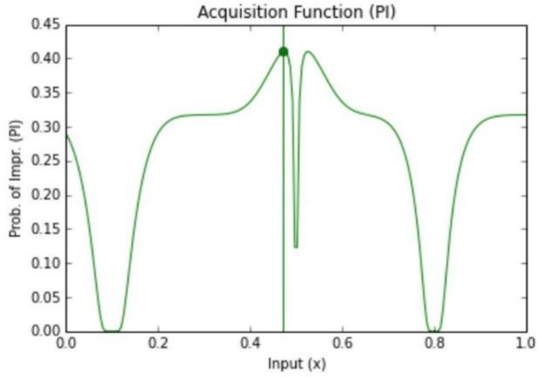


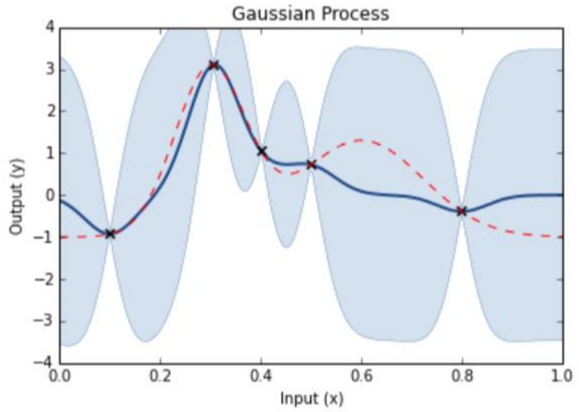
Bayesian Optimization: Exploring AFs (3)

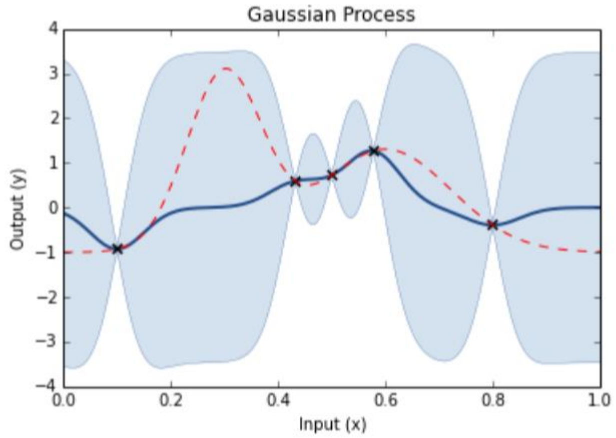
Our goal is to find the global maxima in as few samples as possible

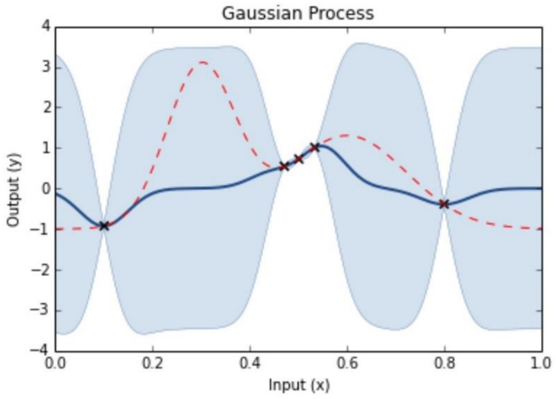






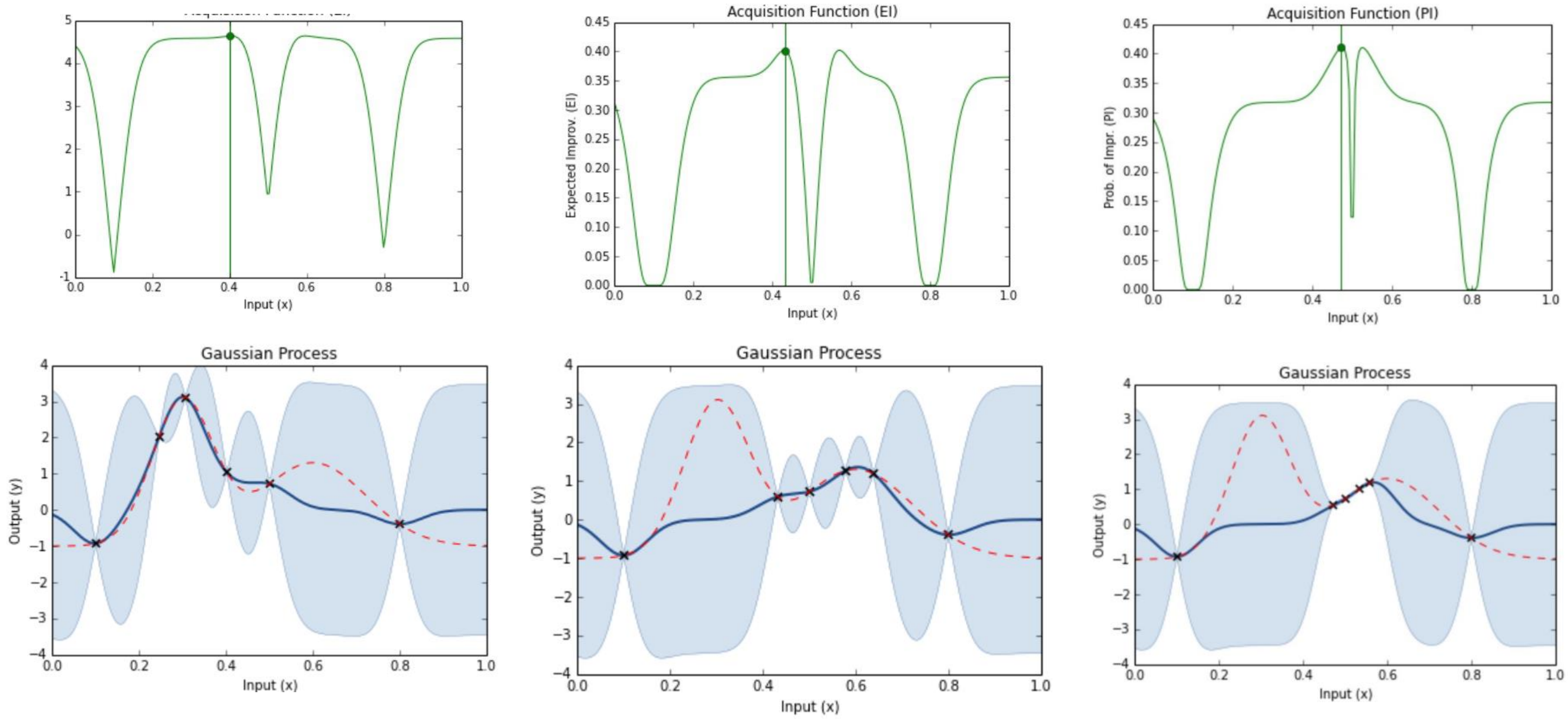






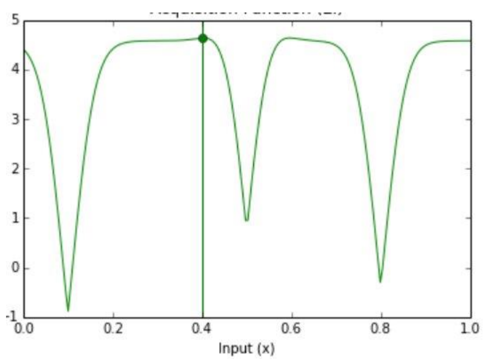
Bayesian Optimization: Exploring AFs (4)

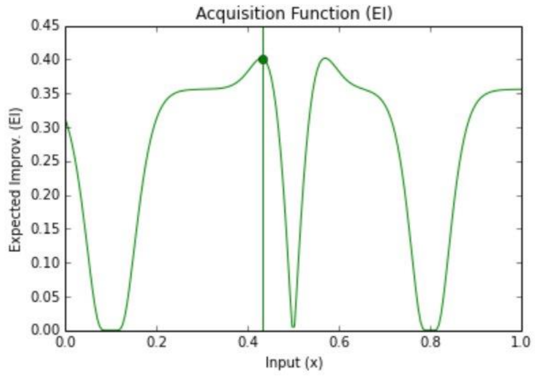
Our goal is to find the global maxima in as few samples as possible

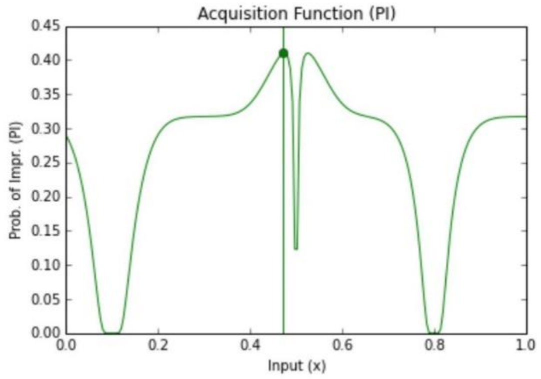


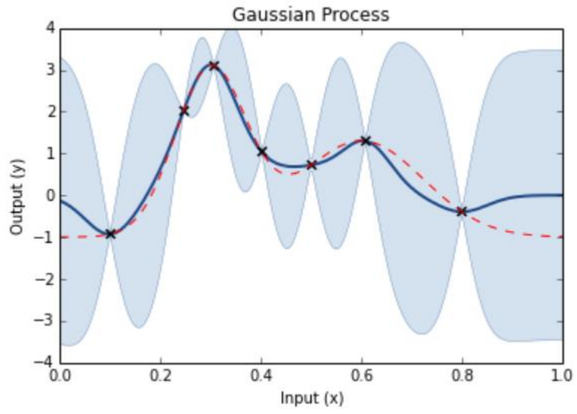
Bayesian Optimization: Exploring AFs (4)

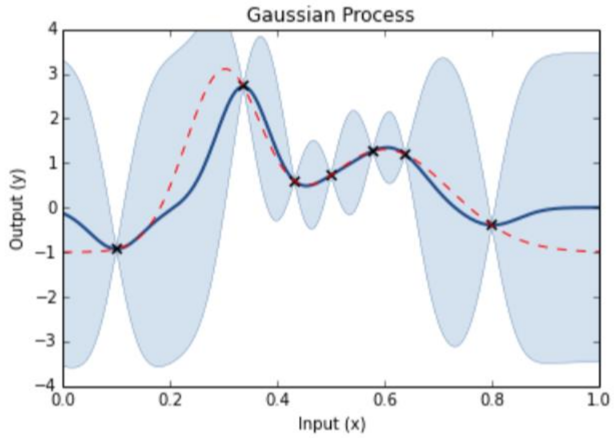
Our goal is to find the global maxima in as few samples as possible

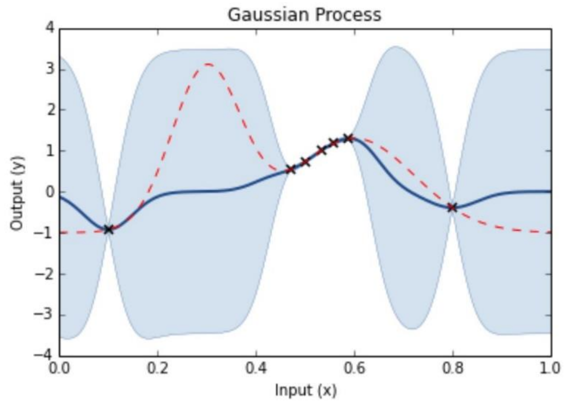






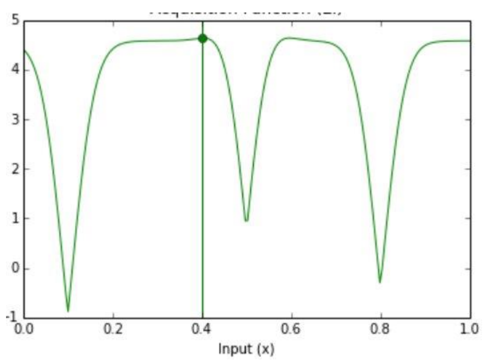


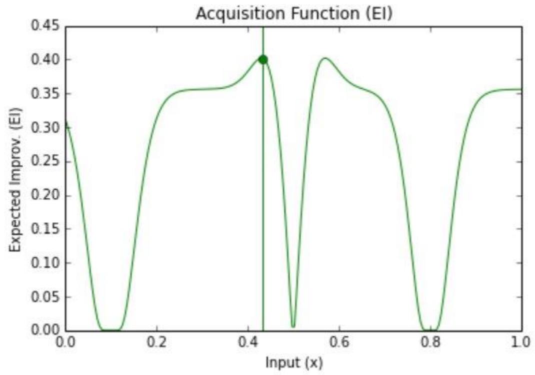


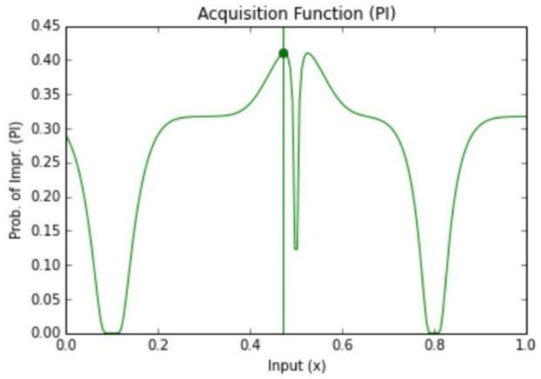


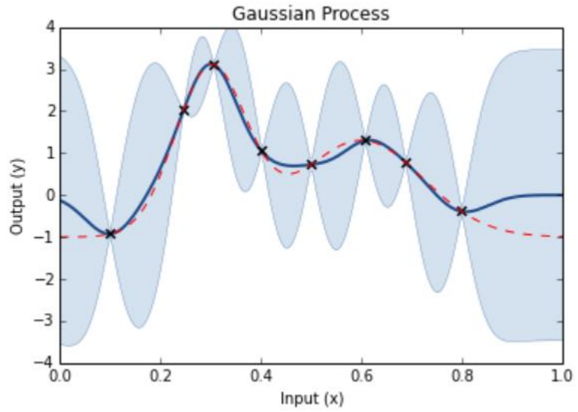
Bayesian Optimization: Exploring AFs (5)

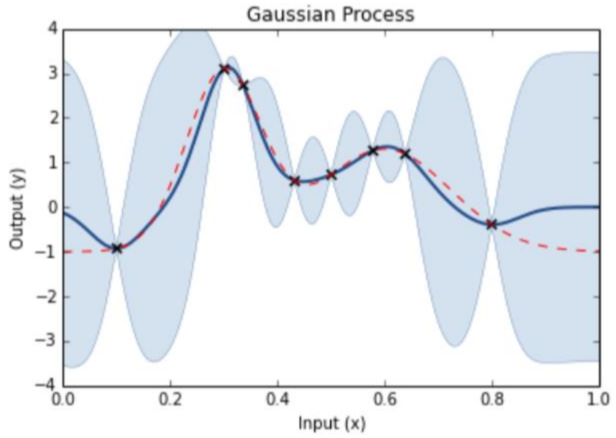
Our goal is to find the global maxima in as few samples as possible

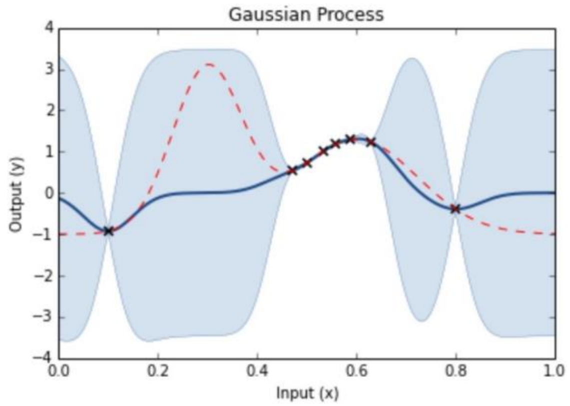






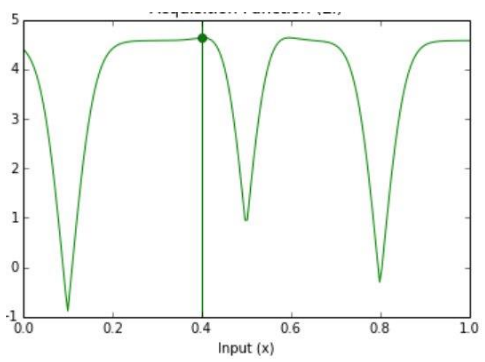


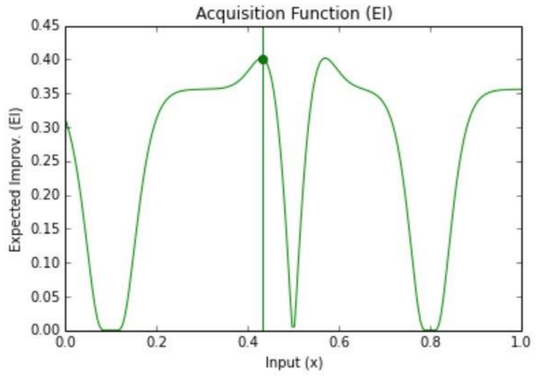


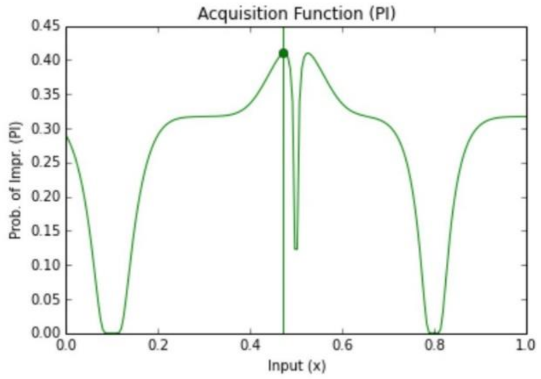


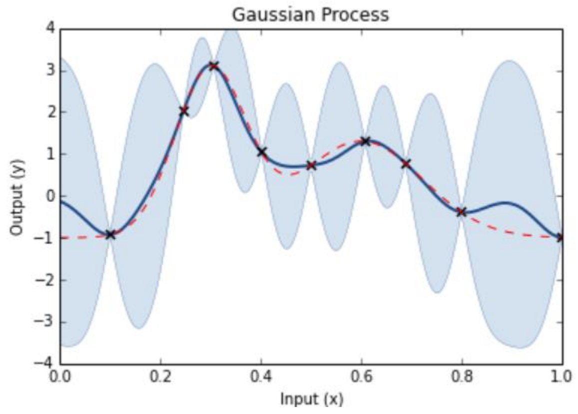
Bayesian Optimization: Exploring AFs (6)

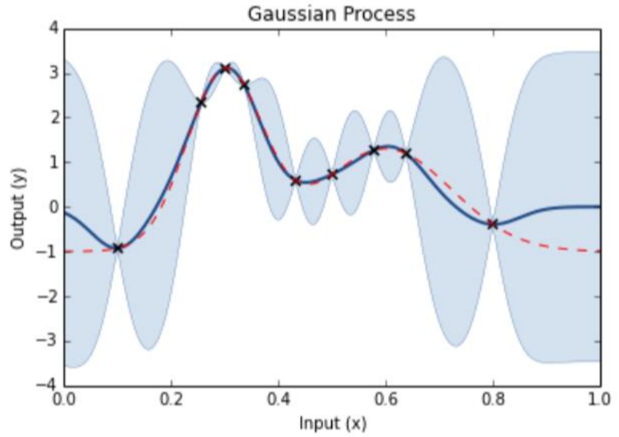
Our goal is to find the global maxima in as few samples as possible

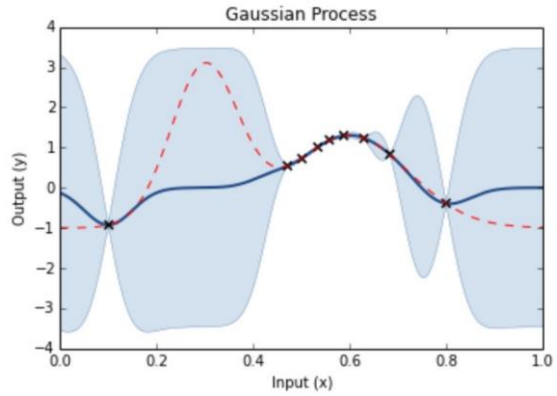






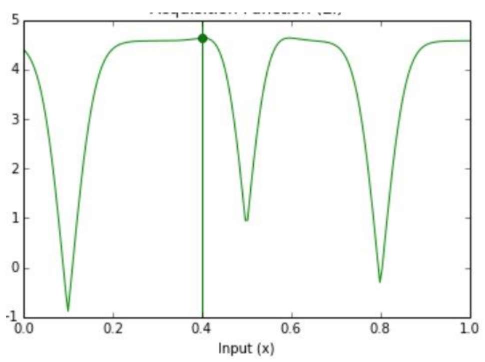


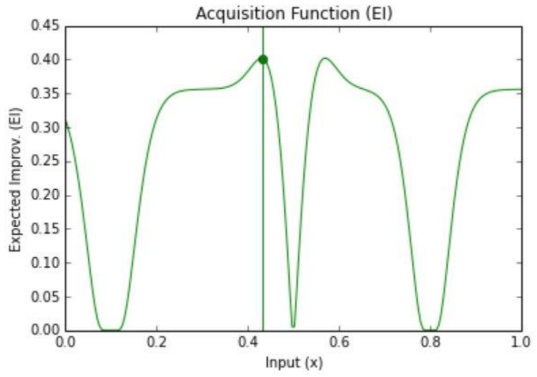


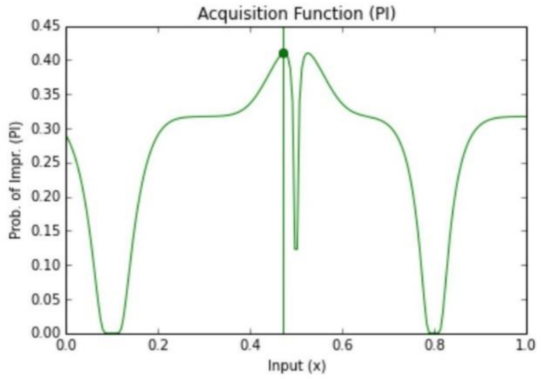


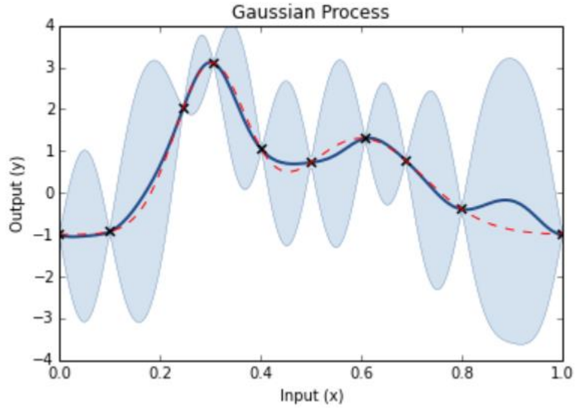
Bayesian Optimization: Exploring AFs (7)

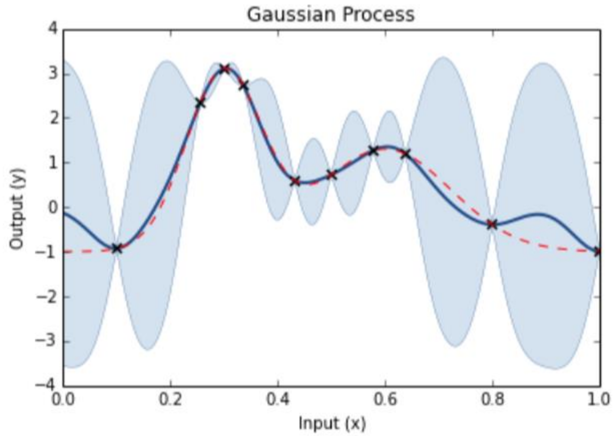
Our goal is to find the global maxima in as few samples as possible

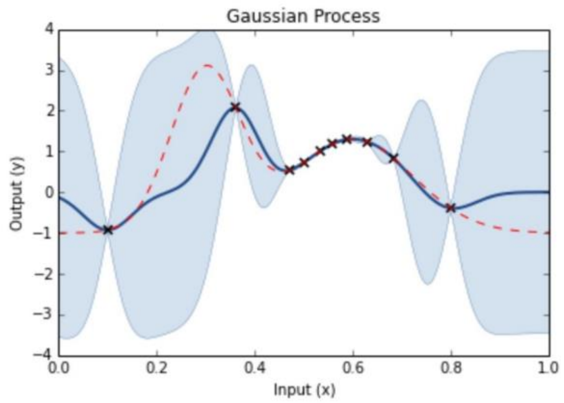






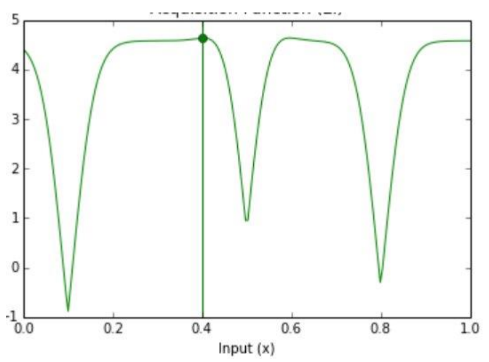


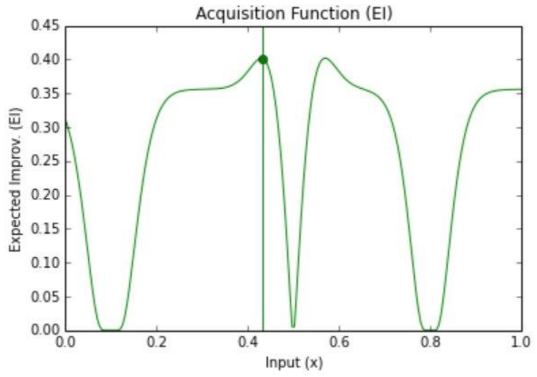


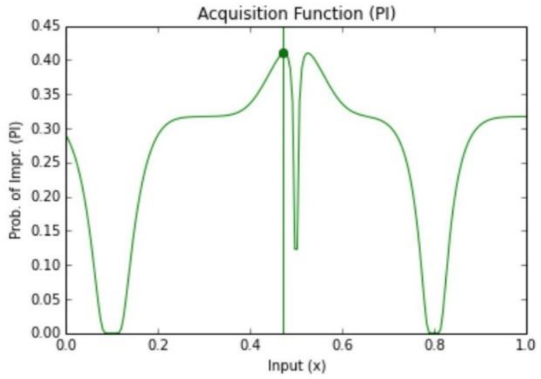


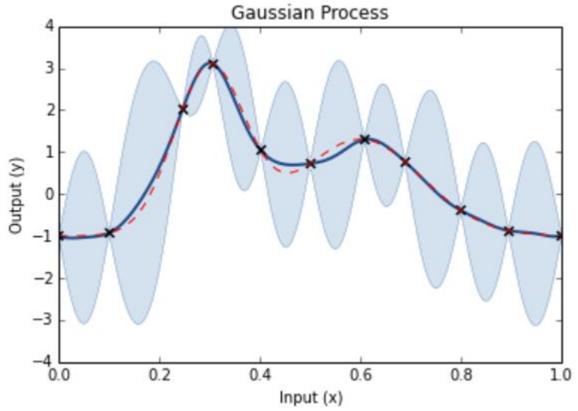
Bayesian Optimization: Exploring AFs (8)

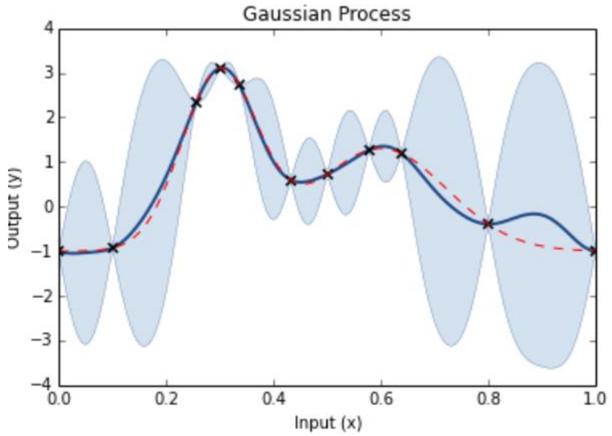
Our goal is to find the global maxima in as few samples as possible

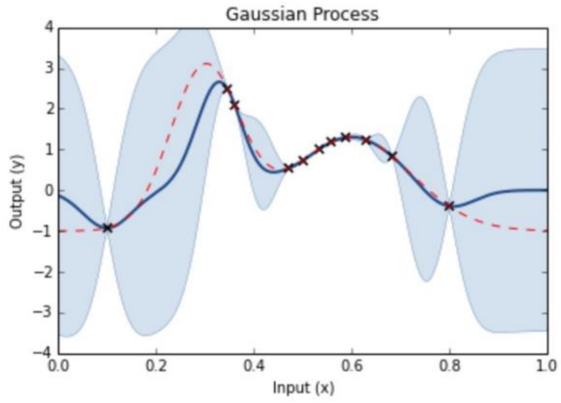












Summary

Many problems in all areas of life can be represented as an exploration-exploitation tradeoff

- Although conceptually simple MABs can be used in many settings to address the exploration-exploitation tradeoff
- Literature on Bandits is very expensive
- E-Greedy is a good alternative for small problems, for the rest, Thomson and UCB are better
- Bayesian optimization tries to address certain limitations of the traditional bandit problem
- It tries to find the optima of an unknown function by sampling, while minimizing the amount of observations required
- BO is an active research field with many industrial applications
- BO has challenges of their own, is not the holy grail, and is not accessible to the layman

References

[] Bobak,
[1] Brochu, Eric, Vlad M. Cora, and Nando De Freitas.
"A tutorial on Bayesian optimization of expensive cost functions,
with application to active user modeling and hierarchical
reinforcement learning."
[] Shipra Agrawal, Navin Goyal.
“Analysis of Thompson Sampling for the Multi-armed Bandit Problem”
[2] Srinivas, Niranjan, et al. "Gaussian process optimization in the
bandit setting: No regret and experimental design."
[] De Freitas, Nando/ Bayesian Optimization and multi-armed bandits
[3] Gittins Index: https://en.wikipedia.org/wiki/Gittins_index
[4] Yelp MOE: http://yelp.github.io/MOE

References

[5] Efficient Experimentation and Multi Armed Bandits http://iosband.github.io/2015/07/19/Efficient-experimentation-and-multi-armed-bandits.html