Convex Optimization 10-725, Lecture 18: Introduction to non-convex optimization: Bayesian Optimization

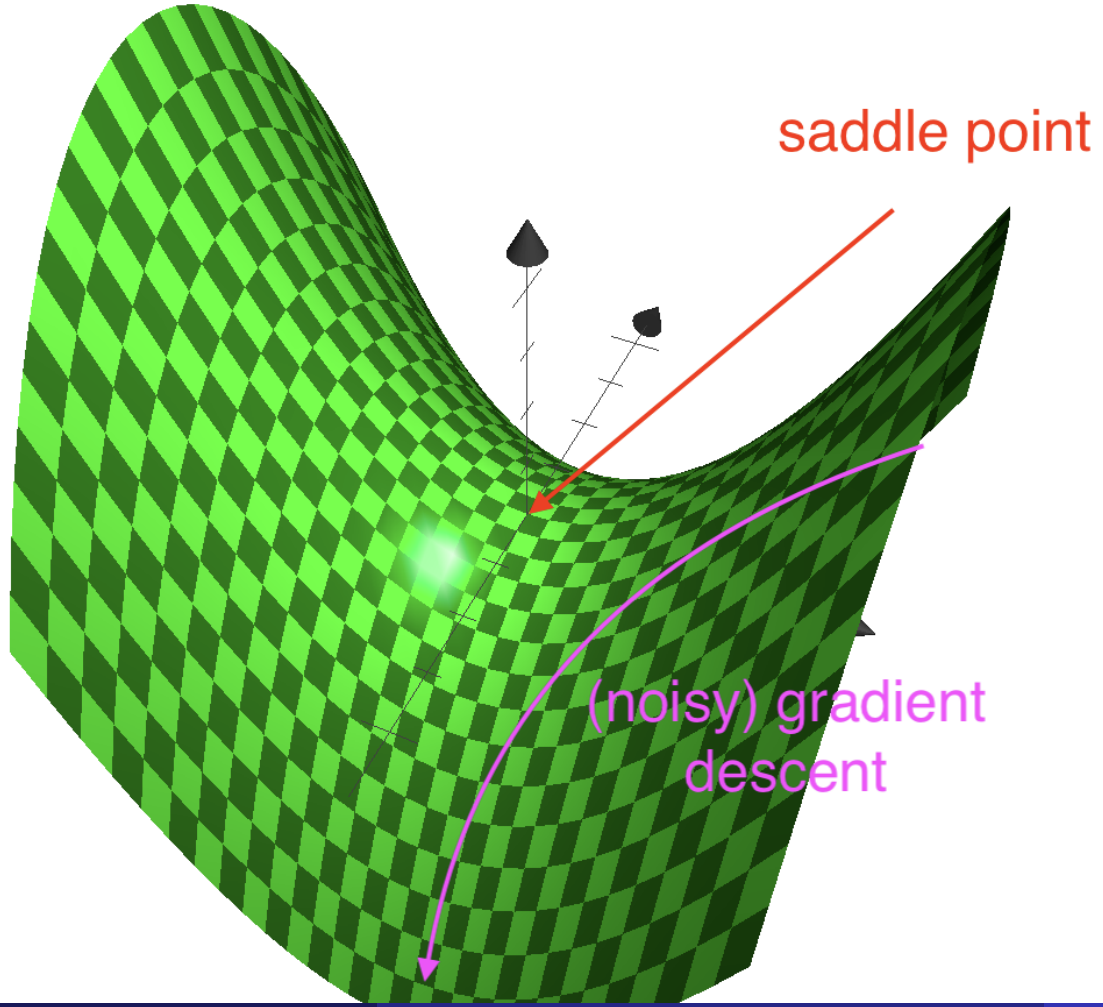
Yuanzhi Li

Assistant Professor, Carnegie Mellon University
Visiting Researcher, Microsoft

Today

Last lecture

- We learnt the definition of (second-order) local minima, saddle points.
- We learnt the algorithm Hessian Descent and the algorithm Noisy Gradient Desent to find a (second order) local minima.



Last lecture

- Clarification: There is "another definition" of local minima: x is called a local minima for function f if there is a $\delta > 0$ such that for every y with $\|y - x\|_2 \leq \delta$, $f(y) \geq f(x)$.
- Clarification: There is "yet another definition" of local minima: x is called a local minima for function f if for every $\varepsilon > 0$, there is a $\delta > 0$ such that for every y with $\|y - x\|_2 \leq \delta$, $f(y) \geq f(x) - \varepsilon$.
- These are not (or equivalent to) the definition of second order local minima, these "local minima" CANNOT CANNOT CANNOT CANNOT be found EFFICIENTLY EFFICIENTLY EFFICIENTLY EFFICIENTLY.
- When you read research papers, be very careful which local minima is the author referring to.

This lecture

- We are going to learn a new, completely different type of optimization algorithm: The Bayesian optimization.
- It is a non-convex optimization algorithm, and it is fundamentally different from gradient descent.
- We will learn the spirit of Bayesian optimization: The GSD algorithm: The Graduate Student Descent algorithm.
- Next lecture we will go back to non-convex optimization in deep learning. We will see the very first lecture: The power of over-parameterization in non-convex optimization. — This is where we start looking at the specific structure of the function f, instead of treating it as a black box (with gradient oracle etc.) as we did in convex optimization.

The difference between Bayesian optimization and gradient descent

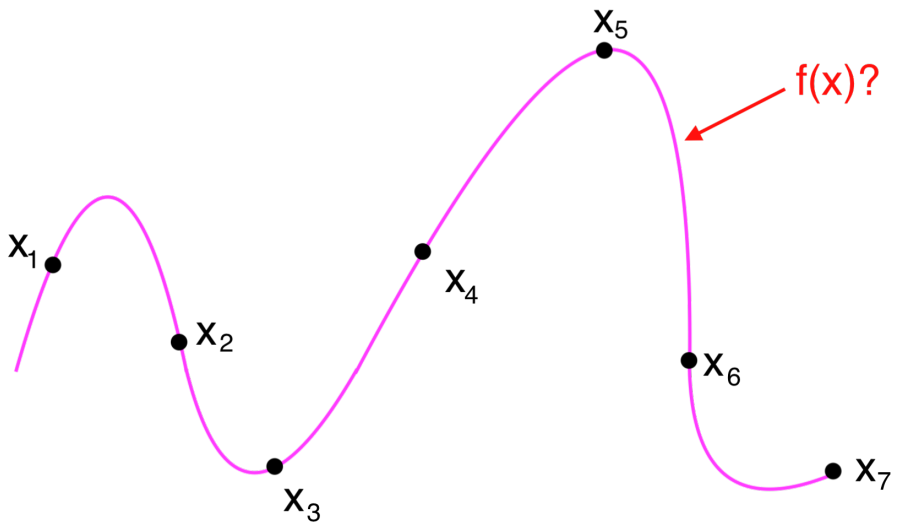
- Bayesian optimization does not require computing the gradient of f, it only requires knowing the function value.
- Bayesian optimization is a global optimization, it looks at the global structure of f. Gradient descent is local.
- Bayesian optimization is a low dimension non-convex optimization algorithm, it is ineffective in high dimension. It is an analog of Ellipsoid algorithm, but for *non-convex* optimization.
- Key applications: Obtaining higher quality solution comparing to gradient descent in low dimension optimization problem, and avoiding computing gradient (when computing the gradient is expensive). Such as hyper-parameter tuning, architecture search etc.

Bayesian optimization: Motivation

- Suppose we know the value of the function f at several points $x_1, x_2, \cdots, x_n$.
- Can we infer the value of a point x?
- Idea: We fit a function f_n such that

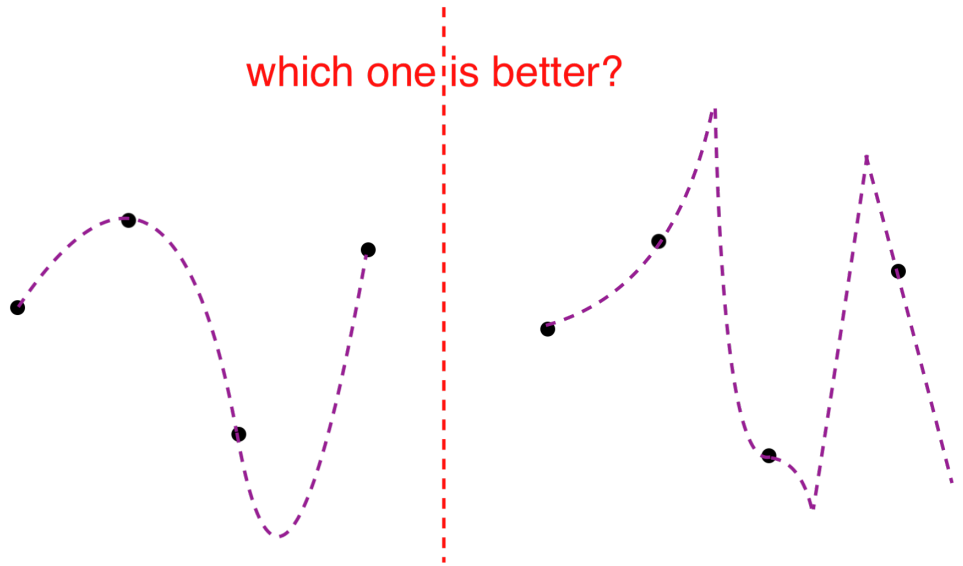
$$f_n(x_i) = f(x_i), \forall i \in [n]$$

- And we output $f_n(x)$.



Bayesian optimization: Motivation

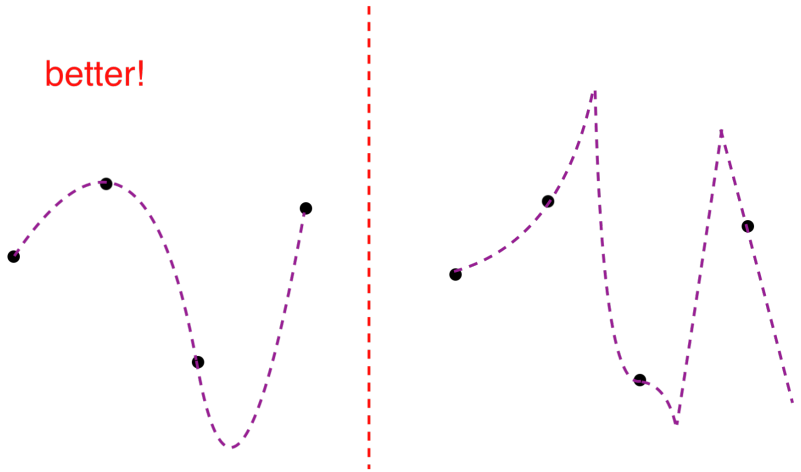
- However, the space of all functions $f_n : \mathbb{R}^d \to \mathbb{R}$ is even un-countable. How do we parameterize the space of f_n so we can search through efficiently?
- More importantly: What is a good function f_n?



- This is the key issue to be solved in Bayesian optimization.

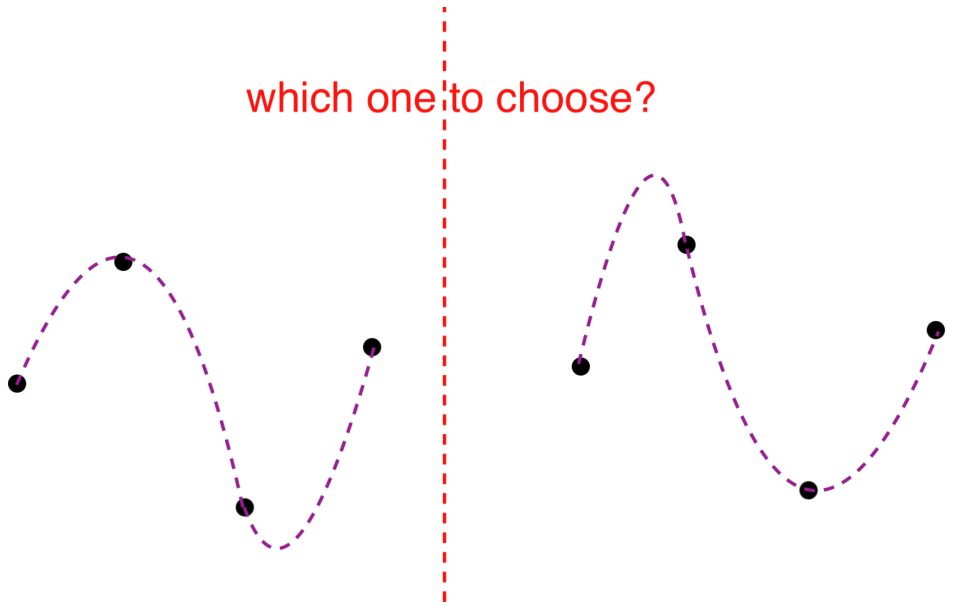
Good functions f_n

- A function f_n is considered as "good" for a given set of observations $f(x_1), \cdots, f(x_n)$, if
- When x is close to one of the x_i,
- Then $f_n(x)$ is close to $f_n(x_i)$.
- f_n is "smooth".
- These are essentially the only properties we need.



Good functions f_n

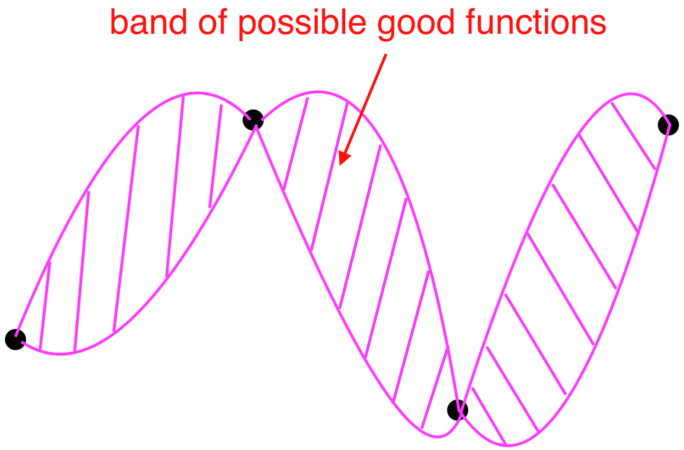
- Now we "sort of" know which functions are good, however,
- Given a set of observations $f(x_1), \cdots, f(x_n)$, there might be multiple "good functions".



- Which one do we choose?
- Bayesian optimization: We do not stick to any of them, but we define a distribution over them.

Bayesian optimization

- The first step of Bayesian optimization is:
- Given a set of observations $f(x_1), \cdots, f(x_n)$, output a distribution over *functions f_n* such almost all of the f_n sampled from this distribution, f_n is good.



- How do we define such a distribution? How do we infer $f(x)$ from this distribution?
- We are going to present the generic routine for Bayesian optimization, and then we are going to define this distribution and define how do we infer $f(x)$ from such a distribution.

Bayesian optimization: The generic routine

- For a function $f : \mathbb{R}^d \to \mathbb{R}$, convex or not:
- Bayesian optimization to find an approximate *maximizer* of f is given as:
- At every step t, compute a distribution P_t over functions $\mathbb{R}^d \to \mathbb{R}$ (haven't defined yet), using $f(x_0), f(x_1), \cdots, f(x_t)$.
- Find the x_{t+1} as the maximizer of the acquisition function associated with P_t (haven't defined yet).
- Obtain value $f(x_{t+1})$.
- Now we look at each term separately: The distribution and the acquisition function.

Bayesian optimization: The distribution P_t

- We will focus on a special type of Bayesian optimization: The Gaussian Process Regression.
- Given a kernel function $K(x, y) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$:
- Recall: A kernel function is a function satisfies for any set of points $x_1, \cdots, x_t$, the $t \times t$ matrix M defined as:

$$M_{i,j} = K(x_i, x_j)$$

  is PSD.

- Example: $K(x, y) = \langle x, y \rangle$ (inner product kernel), $K(x, y) = e^{-\frac{\|x-y\|_2^2}{2\sigma^2}}$ (Gaussian Kernel).
- Gaussian Kernel: $K(x, y)$ measures some inverse distance between x and y (the closer x is to y, the larger K is).

Bayesian optimization: The distribution P_t

- Given a kernel function $K(x, y) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$:
- Define

$$v_t(x) = (K(x, x_0), K(x, x_1), \cdots, K(x, x_t))^\top, F_t = (f(x_0), f(x_1), \cdots, f(x_t))^\top$$

- Define matrix $M_t : \mathbb{R}^{(t+1)\times(t+1)}$ such that

$$[M_t]_{i+1,j+1} = K(x_i, x_j)$$

- Define $P_t(x)$ as a Gaussian distribution:

$$P_t(x) \sim \mathcal{N}(\mu_t(x), \sigma_t^2(x))$$

- Where

$$\mu_t(x) = v_t(x)^\top M_t^{-1} F_t$$

- Where

$$\sigma_t^2(x) = K(x, x) - v_t(x)^\top M_t^{-1} v_t(x)$$

Bayesian optimization: The distribution P_t

- What ****** are these formula?
- Define

$$v_t(x) = (K(x, x_0), K(x, x_1), \cdots, K(x, x_t))^\top, F_t = (f(x_0), f(x_1), \cdots, f(x_t))^\top$$

- Define matrix $M_t : \mathbb{R}^{(t+1)\times(t+1)}$ such that

$$[M_t]_{i+1,j+1} = K(x_i, x_j)$$

- Define

$$\sigma_t^2(x) = K(x, x) - v_t(x)^\top M_t^{-1} v_t(x)$$

- Let $M_t(x) = \begin{pmatrix} K(x,x) & v_t(x)^\top \\ v_t(x) & M_t \end{pmatrix}$, by definition of the kernel K, we know that $M_t(x)$ is PSD.
- This implies that for every vector b, we have that

$$K(x, x) + 2b^\top v_t(x) + b^\top M_t b = (1, b)^\top M_t(x)(1, b) \geq 0$$

- Picking $b = -M_t^{-1} v_t(x)$ we show that

$$\sigma_t^2(x) = K(x, x) - v_t(x)^\top M_t^{-1} v_t(x) \geq 0$$

Bayesian optimization: The distribution P_t

- Define

  $v_t(x) = (K(x, x_0), K(x, x_1), \cdots, K(x, x_t))^\top, F_t = (f(x_0), f(x_1), \cdots, f(x_t))^\top$

- Define matrix $M_t : \mathbb{R}^{(t+1)\times(t+1)}$ such that

  $$[M_t]_{i+1,j+1} = K(x_i, x_j)$$

- Key observation: e_j (the j-th basis vector) satisfies

  $$M_t e_j = v_t(x_{j-1})$$

- This implies that $\mu_t(x) = v_t(x)^\top M_t^{-1} F_t$ satisfies

  $$\mu_t(x_j) = e_{j+1}^\top M_t M_t^{-1} F_t = e_{j+1}^\top F_t = f(x_j)$$

- This also that $\sigma_t^2(x) = K(x, x) - v_t(x)^\top M_t^{-1} v_t(x)$ satisfies

  $$\sigma_t^2(x_j) = K(x_j, x_j) - v_t(x_j)^\top M_t^{-1} v_t(x_j) = K(x_j, x_j) - e_{j+1}^\top M_t M_t^{-1} M_t e_{j+1} = 0$$

Bayesian optimization: The distribution P_t

- Define $P_t(x)$ as:

$$P_t(x) \sim \mathcal{N}(\mu_t(x), \sigma_t^2(x))$$

- Where

$$\mu_t(x) = v_t(x)^\top M_t^{-1} F_t, \quad \sigma_t^2(x) = K(x,x) - v_t(x)^\top M_t^{-1} v_t(x)$$

- At point x_j for $j \in \{0, 1, \cdots, t\}$:

$$\mu_t(x_j) = f(x_j), \quad \sigma_t^2(x_j) = 0$$

- This implies that for any function g sampled from distribution P_t,

$$g(x_j) = f(x_j)$$

Bayesian optimization: The distribution P_t

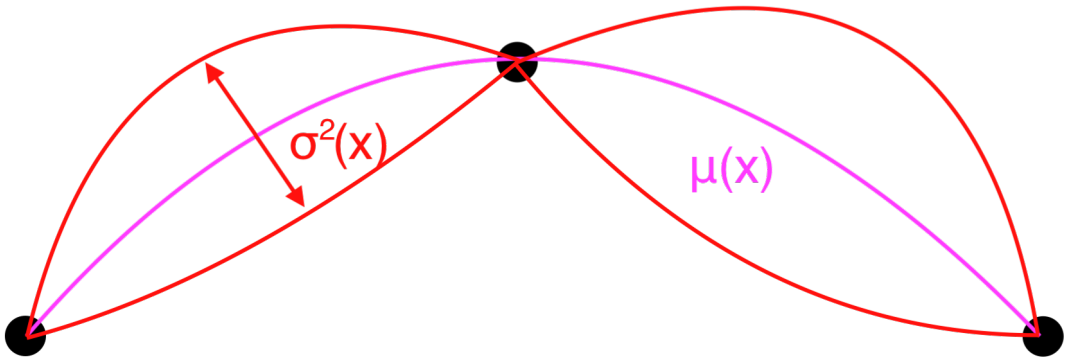
- Define $P_t(x)$ as:

$$P_t(x) \sim \mathcal{N}(\mu_t(x), \sigma_t^2(x))$$

- Recall: for any function g sampled from distribution P_t,

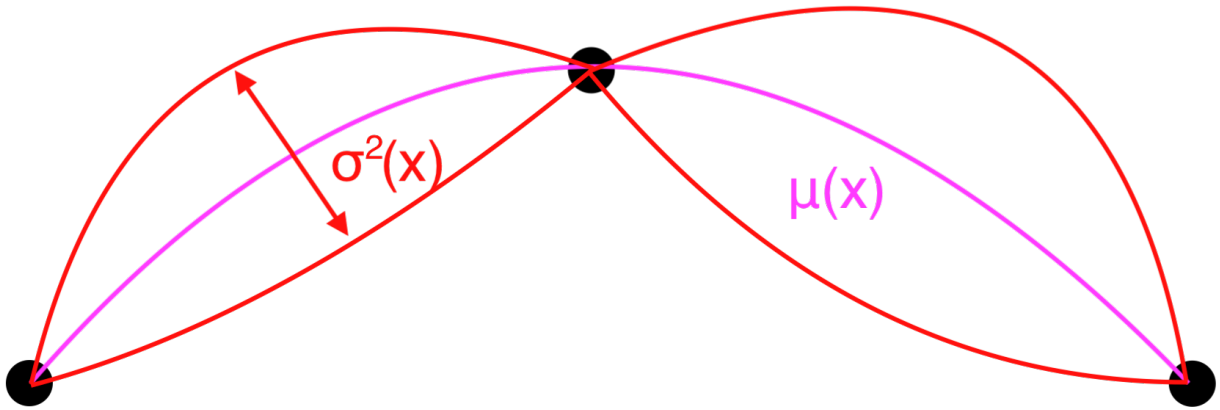
$$g(x_j) = f(x_j)$$

- μ_t, σ_t^2 are "smooth functions", μ_t defines the "mean", σ_t^2 defines the "confidence interval".



Bayesian optimization: The distribution P_t

- Bayesian optimization: Creating a "band of uncertainty":



-
- Why this definition of mean and variance? There is no principle, this is a heuristic choice.

Bayesian optimization: The acquisition function

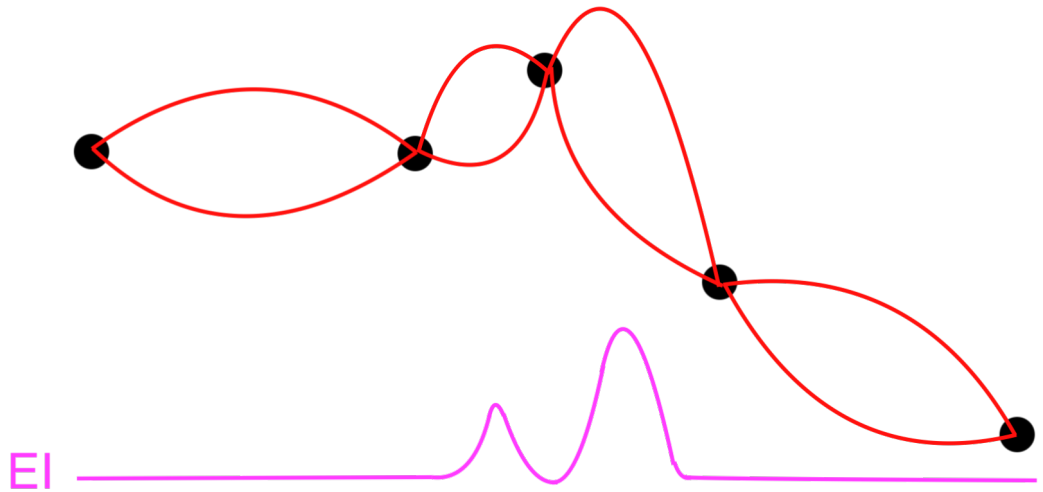
- Now we have learnt what is a distribution P_t over "good functions".
- We begin to answer the question: How to query the next point x_{t+1} using P_t?
- Recall the goal: Find an approximate maximizer of f.
- Key idea: Query the point x_{t+1} such that $f_t(x_{t+1})$ is the largest in expectation, for $f_t \sim P_t$.
- Define $f_t^* = \max_{j=0}^{t}\{f(x_j)\}$, we will find the point x_{t+1} such that

$$x_{t+1} = \text{argmax}_x \mathbb{E}_{g\sim P_t}\left[[g(x) - f_t^*]^+\right]$$

- Here $[z]^+ = \text{ReLU}(z)$.
- Here, $\text{EI}_t(x) = \mathbb{E}_{g\sim P_t}\left[[g(x) - f_t^*]^+\right]$ is called the acquisition function.

Bayesian optimization: The acquisition function

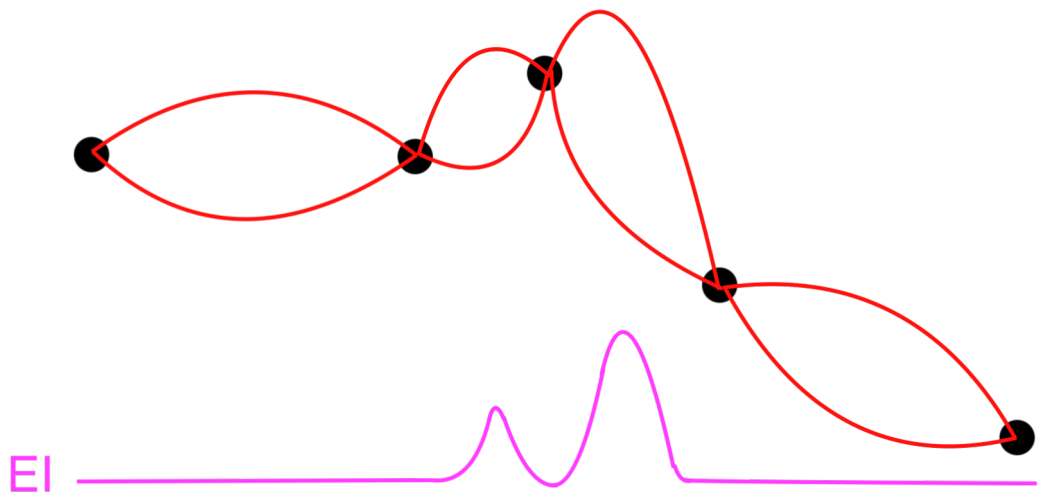
- Recall $\mathrm{EI}_t(x) = \mathbb{E}_{g\sim P_t}\left[[g(x) - f_t^*]^+\right]$ is called the acquisition function.



-
- To compute an approximate (local) maximizer: Using (noisy) gradient ascent. The gradient of $\mathrm{EI}_t(x)$ is cheap to compute, since it only involves $f(x_0), \cdots, f(x_t)$ via P_t.

Bayesian optimization: The acquisition function

- Recall $\mathrm{EI}_t(x) = \mathbb{E}_{g\sim P_t}\left[[g(x) - f_t^*]^+\right]$ is called the acquisition function.



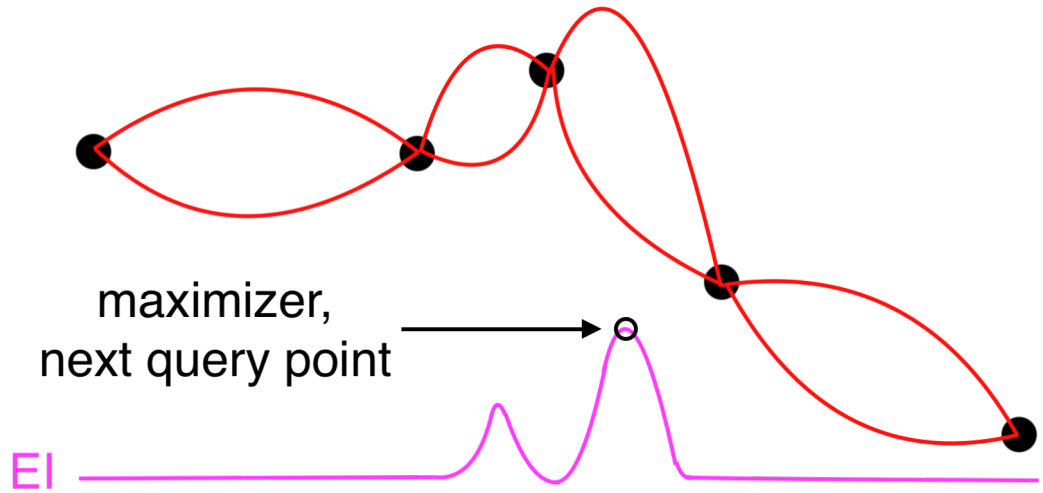
-
- Using $[g(x) - f_t^*]^+$ is often referred to as an "optimistic exploration".
- If $\mu(x)$ is smaller than $\mu(x')$, but $\sigma^2(x)$ is much larger than $\sigma^2(x')$, then the algorithm would prefer x.

Bayesian optimization: The kernel

- The kernel function to define P_t: A common choice is the Gaussian Kernel $K(x, y) = e^{-\frac{\|x-y\|_2^2}{2\sigma^2}}$.
- Another common choice is the Matern kernel (not easy to define, can google after class if interested).
- No matter which kernel we use, the above process is called Gaussian Process Regression due to $P_t(x)$ is defined via a Gaussian Distribution $\mathcal{N}(\mu(x), \sigma^2(x))$.

Bayesian optimization: Convergence rate

- The theory for rate of convergence is in general unknown, this is a heriustic algorithm.



-
- Spirit: Graduate Student Descent: This is exactly how graduate students tune hyper-parameters in deep learning.
- Submit many jobs using different hyper-parameters, observe the TREND, then update the search space of these hyper-parameters.