

Iterative Algorithms

Song Liu (song.liu@bristol.ac.uk)

GA 18, Fry Building,

Microsoft Teams (search "song liu").

Search Problem

$$\{i^*\} = \arg \min_{i \in \mathcal{F}} f(i)$$

- where \mathcal{F} defines a search space,
- f defines the search criterion.

Greedy Algorithm

Greedy algorithm is the name of a **set of search algorithms**.

1. Greedy algorithm finds the best solution(s) to a smaller and more manageable **subproblem**.
2. From the solution, it **revises** the subproblem and solves a new subproblem.
3. Until a certain stopping criterion is satisfied.

Hill Climbing

$$x_{t+1}^* = \arg \max_{x \in [x_t^* - \epsilon, x_t^* + \epsilon]} f(x),$$

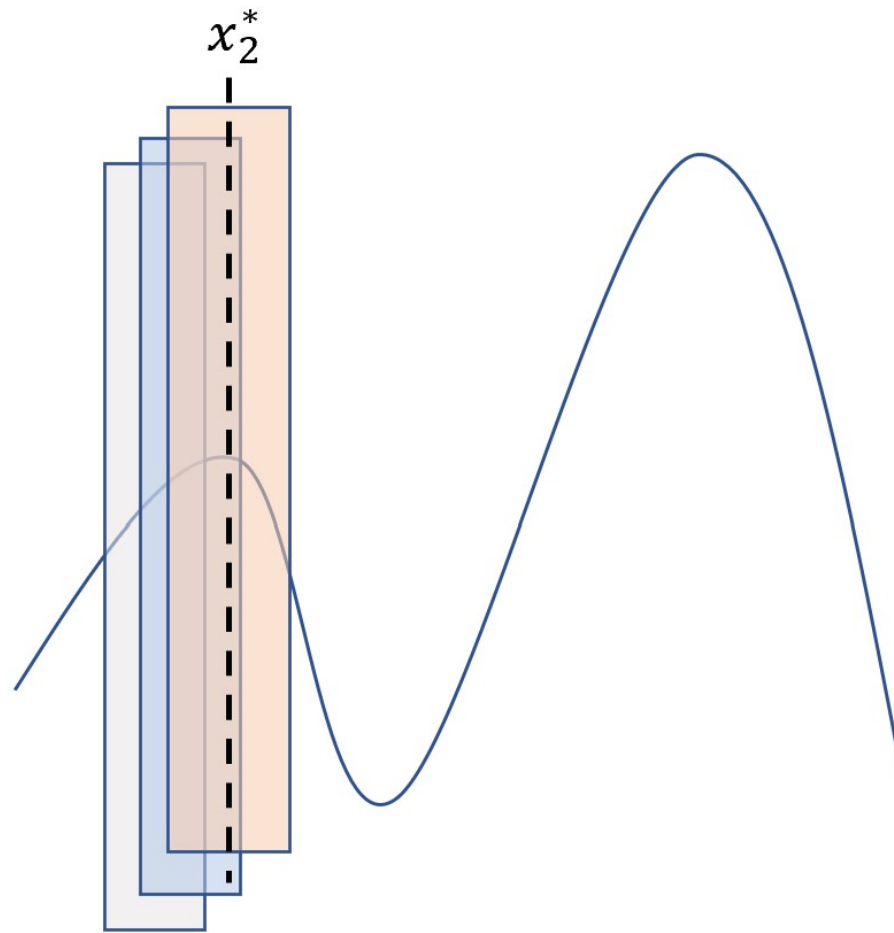
- Show Demo.

Limitations of Greedy Algorithm

Greedy Algorithm is a myopic algorithm: It may not lead to the global optimal solution.

- Hill-climbing does not find the global maximum if the search space of each subproblem is too small.
- TicTacToe does not anticipate opponent moves leading to a suboptimal move.

Local Optimum



Algorithm stuck at x_2^* !

A Suboptimal Move

```
X O *
```

```
X X O
```

```
* * *
```

AI plays...

```
X O O
```

```
X X O
```

```
* * *
```

It does not anticipate your opponent's move (X checkmate!)

Iterative Algorithm

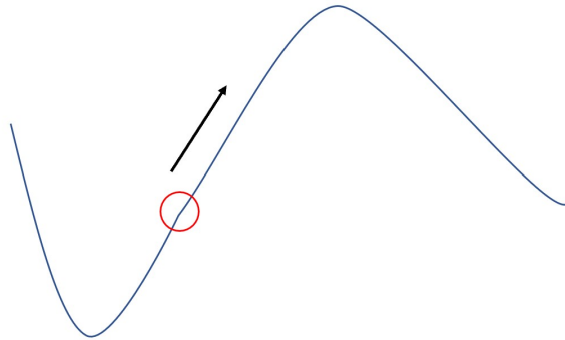
- The hill climbing algorithm is also an iterative algorithm:
 - It starts searching from an initial value x_0 .
 - It solves a sequence of search problems:
 - $x_{t+1}^* = \arg \max_{x \in [x_t^* - \epsilon, x_t^* + \epsilon]} f(x),$
 - where t -th problem is derived from the $t - 1$ -th
 - It terminates when x_{t+1}^* converges.
- **Iterative algorithms** seek to approximate the solution to a numeric problem by **successive improvements**.
 - At each iteration, it revises the current approximation, so that it is closer to the solution.
 - The algorithm stops when a stopping criteria is met.

Iterative Algorithm

- Iterative algorithm **does not have to be greedy** when improving your approximation.

The Slope of a Function

- From the previous example, one can see our hill climbing algorithm is quite naive.
 - It does not realize the **slope** of $f(x)$.

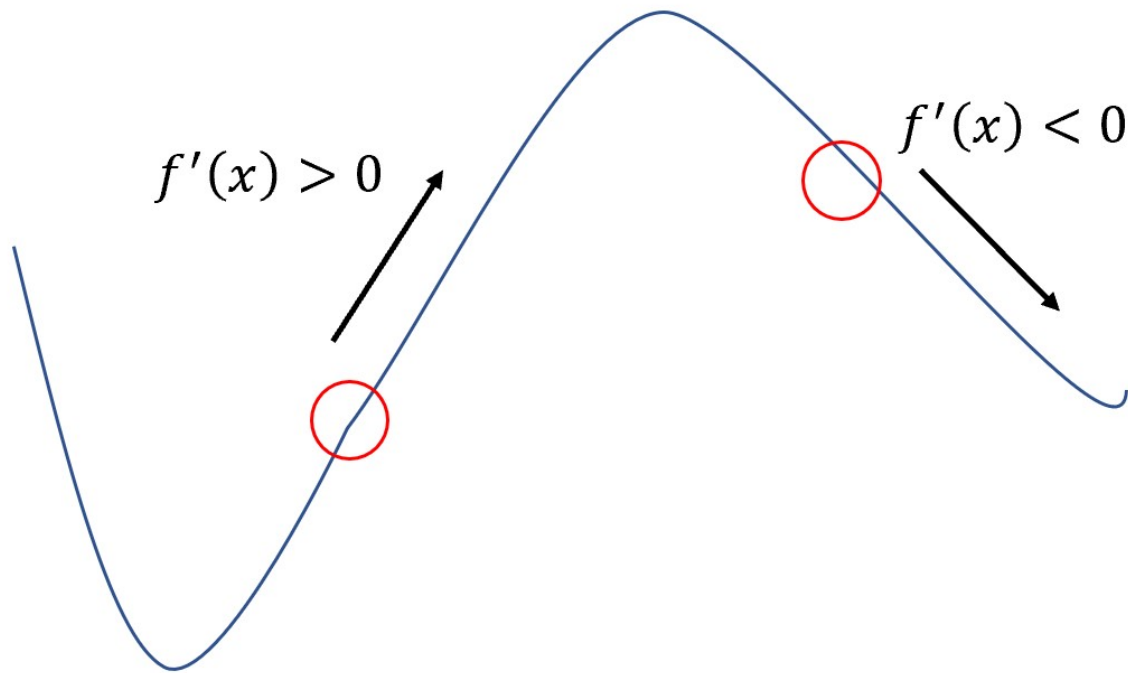


- For a **differentiable function**, its slope indicates the direction you should go to maximize/minimize your function.
 - To maximize, climb up the slope.
 - To minimize, slide down the slope.

The Slope of a Function

What is the slope of a differentiable function?

- Its derivative!



Gradient Ascent Algorithm

- Gradient Ascent is an iterative algorithm for solving the following search problem:

$$x^* = \arg \max_{x \in \mathcal{X}} f(x),$$

where \mathcal{X} is the search space, e.g., \mathbb{R} .

- It starts with a random guess x_0 .
- At iteration t , it revise the current guess x_t by using the derivative (gradient) information $f'(x_t)$.
 - The revision does not have to be greedy!
- Repeat until stopping criteria is met.

Gradient Ascent Algorithm

Pseudo Code: Gradient Ascent

- initialize x_0 with a random guess
- For $t = 0$ to T
 - $x_{t+1} \leftarrow x_t + \epsilon f'(x_t)$.
 - ϵ is the **step size**, a small number, say 0.1.
 - If $|x_{t+1} - x_t| < \eta$, stop the loop.
 - ϵ is a extremely small number, say 1e-5.
- x_t is your approximation to x^* .

Gradient Ascent Algorithm

```
double x0 = -4; //initial guess
int T = 10000; //maximum iteration
double epsilon = .1; // step size
double eta = 1e-5; // stopping threshold

double xt = x0;
for(int t=0; t<T; t++){
    // gradient ascent!
    double xt1 = xt + epsilon*df(xt);
    // do we stop?
    if (fabs(xt1 - xt) < eta){
        xt = xt1;
        break;
    }
    xt = xt1;
}

//NO need to search a grid!
```

Gradient Asecent

$$f(-4.000000) = -0.368995)$$

$$f(-3.999771) = -0.368995)$$

$$f(-3.999537) = -0.368994)$$

...

$$f(2.539099) = 1.760173)$$

$$f(2.539110) = 1.760173)$$

$$f(2.539120) = 1.760173)$$

$$f(2.539130) = 1.760173)$$

$$f(2.539140) = 1.760173)$$

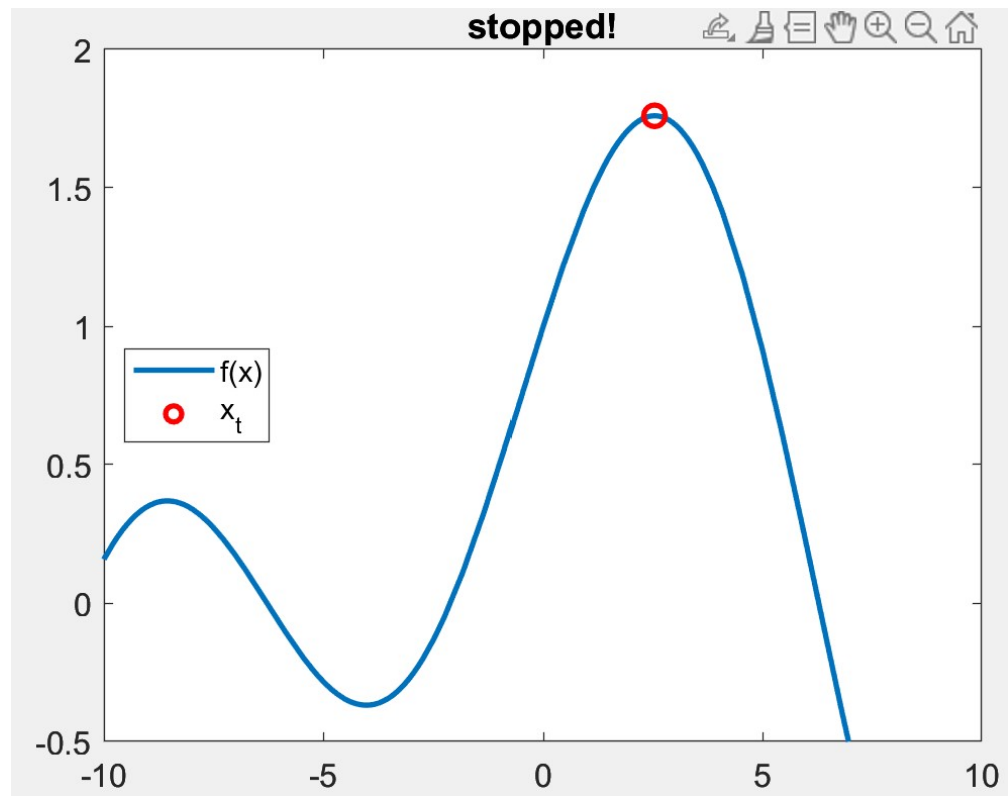
Show demo.

Local Optimum

- Like Hill Climbing algorithm, gradient descent is not guaranteed to return the global maximum solution.
 - Like Hill Climbing algorithm, the gradient ascent is also not aware of the global structure of $f(x)$.
- It may be stuck at a **local optimum**.

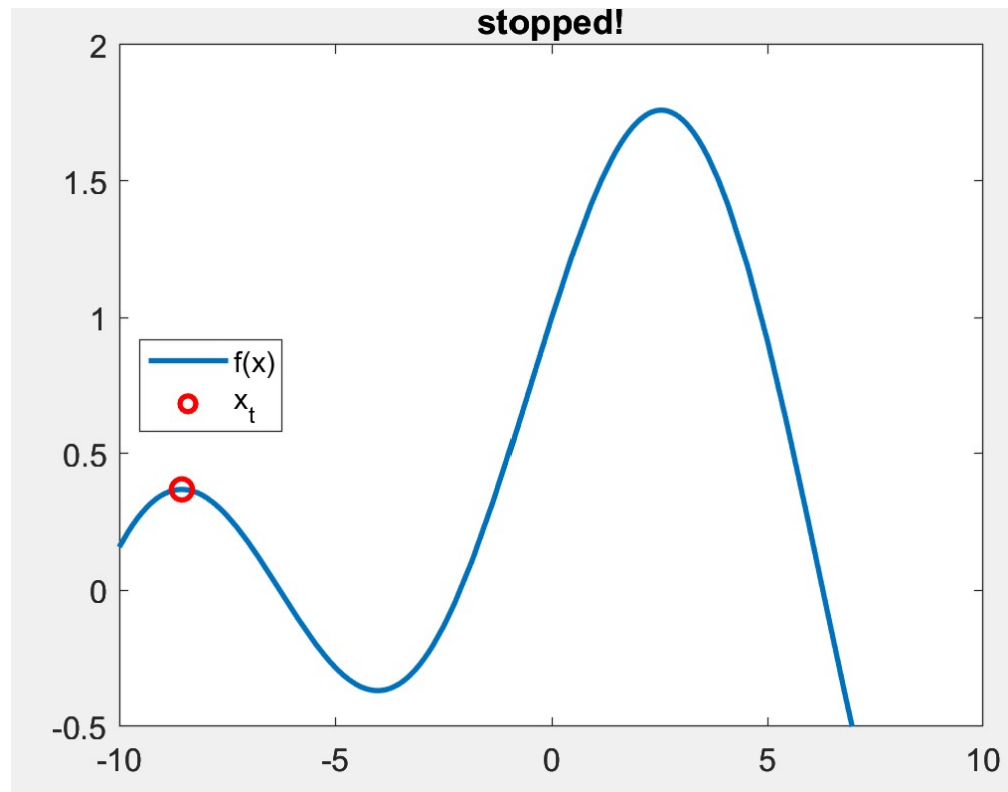
Local Optimum

$$x_0 = -3:$$



Local Optimum

$$x_0 = -5:$$



Gradient Ascent vs. Hill Climbing

Similarities:

- They both search for the maximum of $f(x)$.
- They are both iterative algorithms.
- They both use local structure of function $f(x)$.
 - They both may be stuck at a local optimum.

Dissimilarities:

- Hill climbing evaluates $f(x)$, $x \in [x_t - \epsilon, x_t + \epsilon]$ while gradient ascent evaluates $f'(x_t)$.
 - Gradient ascent cannot be applied to non-differentiable functions (e.g. $f(x) = |x|$), but Hill climbing can.
- Gradient ascent does not seek for "the best" solution of a subproblem. It only makes an improvement.

Gradient Ascent vs. Hill Climbing

- Time Complexity:
- Suppose
 - Evaluating $f(x)$ and $f'(x)$ takes the same amount of time.
 - Both algorithm run for T steps before stop.
 - There are total G grid points in the hill climbing search interval $[x_t - \epsilon, x_t + \epsilon]$.
- Hill Climbing: $O(GT)$
- Gradient Ascent: $O(T)$

Gradient Ascent for Multivariate Functions

- The idea of gradient ascent easily extends to multivariate functions.

$$x^* = \arg \max_{x \in \mathcal{X}} f(x_1, x_2),$$

where $\mathcal{X} := \mathbb{R}^2$.

- The gradient of a function is defined as

$$\nabla f(x_1, x_2) := \begin{bmatrix} \partial_{x_1} f(x_1, x_2) \\ \partial_{x_2} f(x_1, x_2) \end{bmatrix}$$

- The gradient of a multivariate function is a **vector** that points to the direction where f increases the steepest.

Gradient Ascent Algorithm

Pseudo Code: Gradient Ascent (bivariate)

- initialize x_0 with a random guess
- For $t = 0$ to T
 - $x_{t+1} \leftarrow x_t + \epsilon \nabla f(x_1, x_2)|_{(x_1, x_2)=x_t}$
 - ϵ is the **step size**, a small number, say 0.1.
 - If $\|x_{t+1} - x_t\| < \eta$, stop the loop.
 - ϵ is a extremely small number, say 1e-5.
- x_t is your approximation to x^* .

Curse of Dimensionality

- Hill climbing algorithm can also be generalized to multivariate function.
- However, the number of grid points G grows exponentially with dimensionality of your search space.
 - Number of unit length intervals in $[0, 10]$: 10.
 - Number of unit squares in $[0, 10]^2$: 100.
 - Number of unit cubes in $[0, 10]^3$: 1000.
- Time complexity of Hill Climbing grows exponentially with dimensionality of your search space!
- Gradient Ascent does not have such a problem, assuming the gradient of f can be easily evaluated.

Conclusion

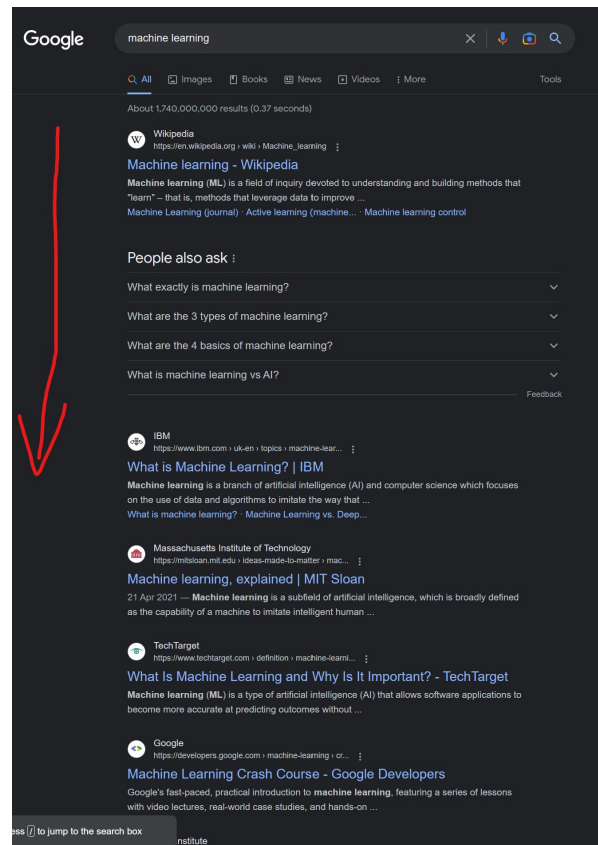
- **Iterative algorithms** seek to approximate the solution to a numeric problem by **successive improvements**.
- Gradient ascent solves the following problem:
$$x^* = \arg \max_{x \in \mathcal{X}} f(x)$$
- Gradient ascent revises the current guess x_t by using the derivative (gradient) information $f'(x_t)$.
- Gradient ascent may also be stuck at the local optimum.
- Gradient ascent easily generalizes to multivariate functions without suffering from the curse of dimensionality.

CW3: Pagerank:

The Secret Weapon of Google

Motivation:

- The common problem of a internet search is: How to rank websites that contains the same keywords?



Motivation:

- You want rank **popular content** before low quality contents.
- How do you determine the popularity of a website?
 - To make things harder, people may try to fool you into thinking some websites are more popular than the others.

