Applied Optimization

Alexander G. Ororbia II

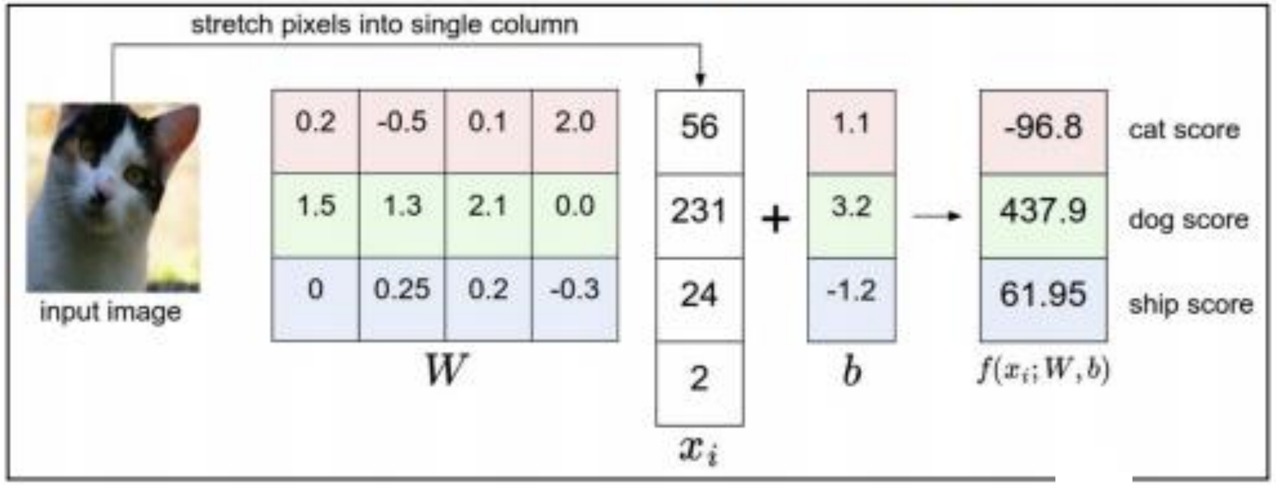
Introduction to Machine Learning

CSCI-335

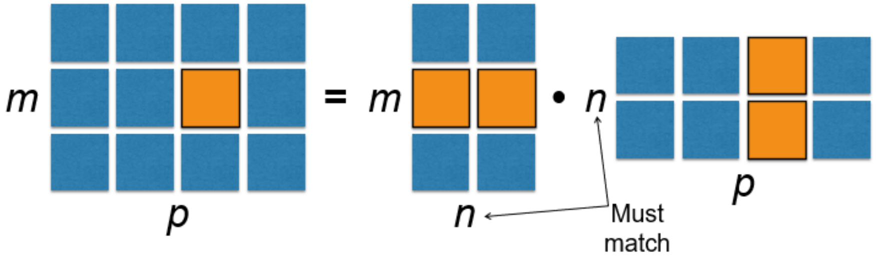
1/23/2026

Tensors in Statistical Learning

A linear classifier $y = Wx^{\mathrm{T}} + b$



Vector x is converted into vector y by multiplying x by a matrix W



Optimization and Decision-Making Problems

1. Get a precise definition of problem, all relevant data and information on it
   - Uncontrollable factors (“random” variables)
   - Controllable inputs (“decision” variables)

2. Construct a mathematical (optimization) model of problem
   - Build objective functions and constraints

3. Solve model
   - Apply most appropriate algorithms for given problem

4. Implement solution

Mathematical Optimization in the "Real World"

Mathematical Optimization is a branch of applied mathematics which is useful in many different fields. Here are a few examples:

- Manufacturing
- Production
- Inventory control
- Transportation
- Scheduling
- Networks
- Finance
- Engineering
- Mechanics
- Economics
- Control engineering
- Marketing
- Policy Modeling

Optimization Vocabulary

Your basic optimization problem consists of…

- The objective function, f(x), which is the output you're trying to maximize or minimize.

Optimization Vocabulary

Your basic optimization problem consists of…

- The objective function, $f(x)$, which is the output you're trying to maximize or minimize.
- Variables, x_1 x_2 x_3 and so on, which are the inputs – things you can control. They are abbreviated x_n to refer to individuals or x to refer to them as a group.

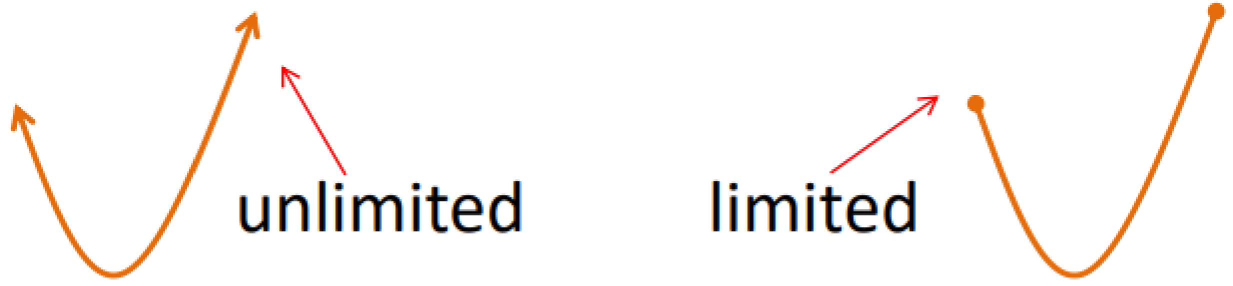
Optimization Vocabulary

Your basic optimization problem consists of…

- The objective function, $f(x)$, which is the output you're trying to maximize or minimize.
- Variables, x_1 x_2 x_3 and so on, which are the inputs – things you can control. They are abbreviated x_n to refer to individuals or x to refer to them as a group.
- Constraints, which are equations that place limits on how big or small some variables can get. Equality constraints are usually noted $h_n(x)$ and inequality constraints are noted $g_n(x)$.

Types of Optimization Problems

- Some problems have constraints and some do not.



Types of Optimization Problems

- Some problems have constraints and some do not.
- There can be one variable or many.

x_3

x_1

x_4 x_2

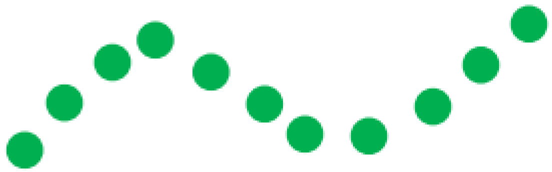
x_6 x_5

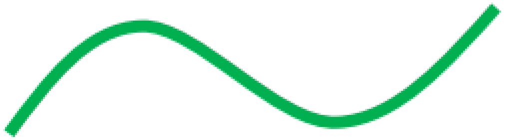
x_7

x_8

Types of Optimization Problems

- Some problems have constraints and some do not.
- There can be one variable or many.
- Variables can be discrete (for example, only have integer values) or continuous.





Types of Optimization Problems

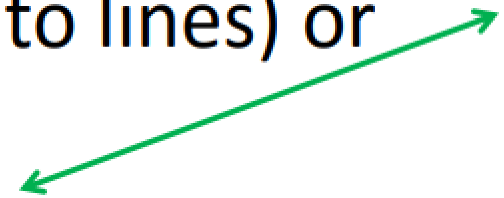
- Some problems have constraints and some do not.
- There can be one variable or many.
- Variables can be discrete (for example, only have integer values) or continuous.
- Some problems are static (do not change over time) while some are dynamic (continual adjustments must be made as changes occur).

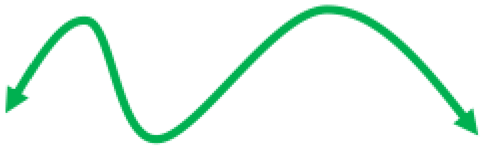
Types of Optimization Problems

- Some problems have constraints and some do not.
- There can be one variable or many.
- Variables can be discrete (for example, only have integer values) or continuous.
- Some problems are static (do not change over time) while some are dynamic (continual adjustments must be made as changes occur).
- Systems can be deterministic (specific causes produce specific effects) or stochastic (involve randomness/ probability).

Types of Optimization Problems

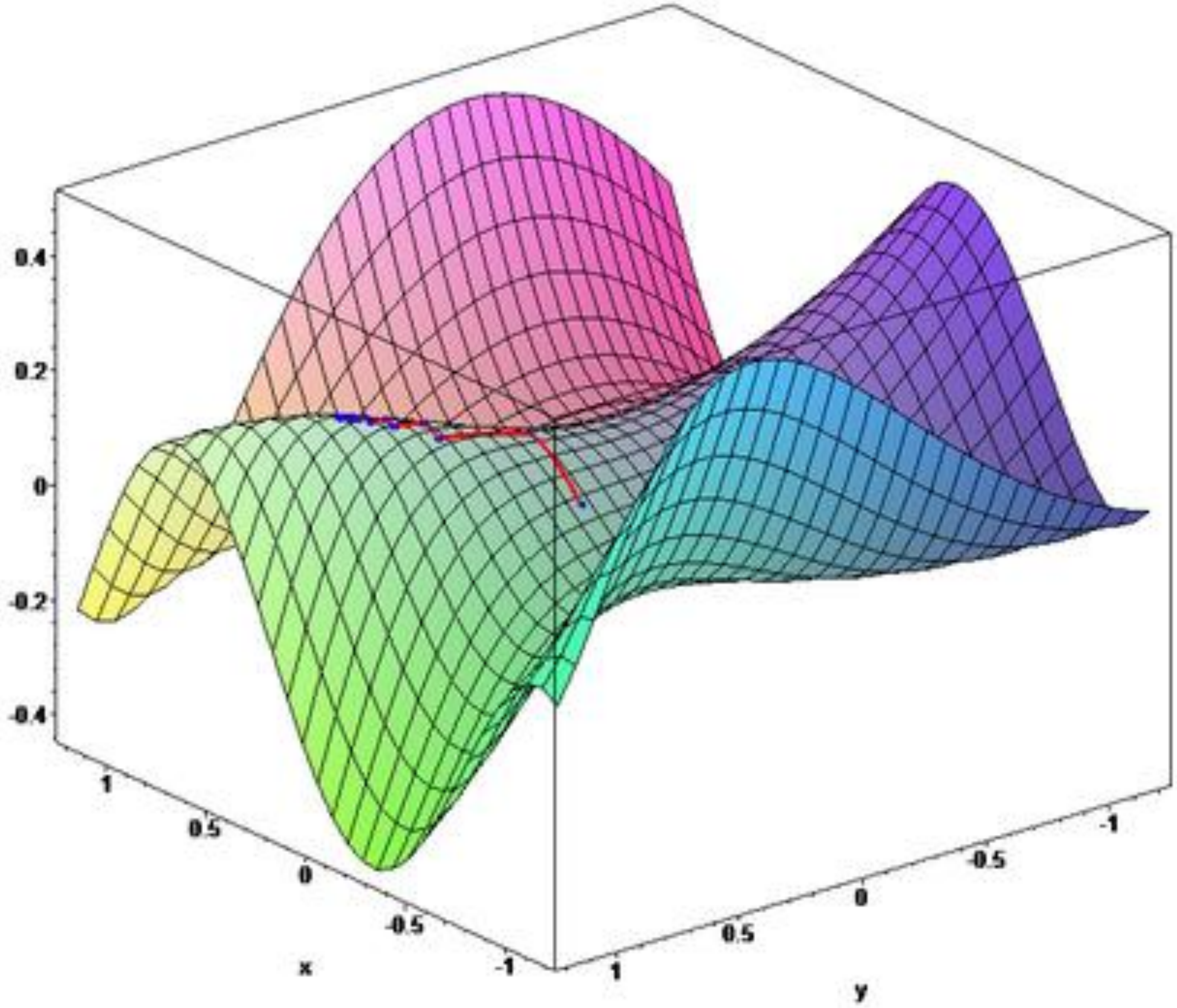
- Some problems have constraints and some do not.
- There can be one variable or many.
- Variables can be discrete (for example, only have integer values) or continuous.
- Some problems are static (do not change over time) while some are dynamic (continual adjustments must be made as changes occur).
- Systems can be deterministic (specific causes produce specific effects) or stochastic (involve randomness/ probability).
- Equations can be linear (graph to lines) or nonlinear (graph to curves)





Gradient-Based Optimization

- Most ML algorithms involve optimization
- Minimize/maximize a function $f(\boldsymbol{x})$ by altering $\boldsymbol{x}$
  - Usually stated a minimization
  - Maximization accomplished by minimizing $-f(\boldsymbol{x})$
- $f(\boldsymbol{x})$ referred to as objective function or criterion
  - In minimization also referred to as loss function cost, or error
  - Example is linear least squares $f(x) = \frac{1}{2} || Ax - b ||^2$
  - Denote optimum value by $\boldsymbol{x}^* = \text{argmin}\, f(\boldsymbol{x})$



0.4
0.2
0
-0.2
-0.4
1
0.5
0
-0.5
-1
x
1
0.5
0
-0.5
-1
y

Problem Specification

Suppose we have a cost function (or ***objective function***)

$$f(\mathbf{x}) : \mathbb{R}^N \longrightarrow \mathbb{R}$$

Our aim is to find values of the parameters (decision variables / ***features***) $\mathbf{x}$ that minimize this function

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} f(\mathbf{x})$$

(Often) subject to the following ***constraints***:

- **equality**: $c_i(\mathbf{x}) = 0$
- **nonequality**: $c_j(\mathbf{x}) \geq 0$

We will handle these a bit more "softly" (to be discussed later)!

If we seek a maximum of $f(\mathbf{x})$, it is equivalent to seeking a minimum of $-f(\mathbf{x})$

Equivalence between Minimum and Maximum

- If a point x^* corresponds to the minimum value of the function $f(x)$, the same point also corresponds to the maximum value of the negative of the function, $-f(x)$.
  - This means optimization can be re-interpreted to mean minimization since the maximum of a function can be found by seeking the minimum of the negative of the same function.

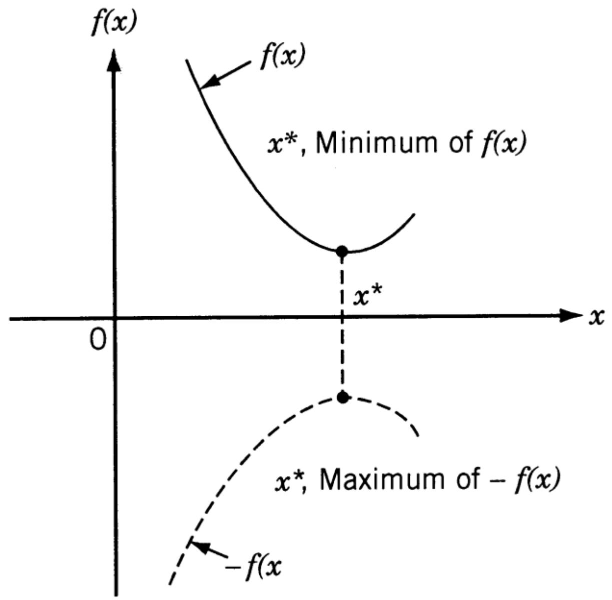
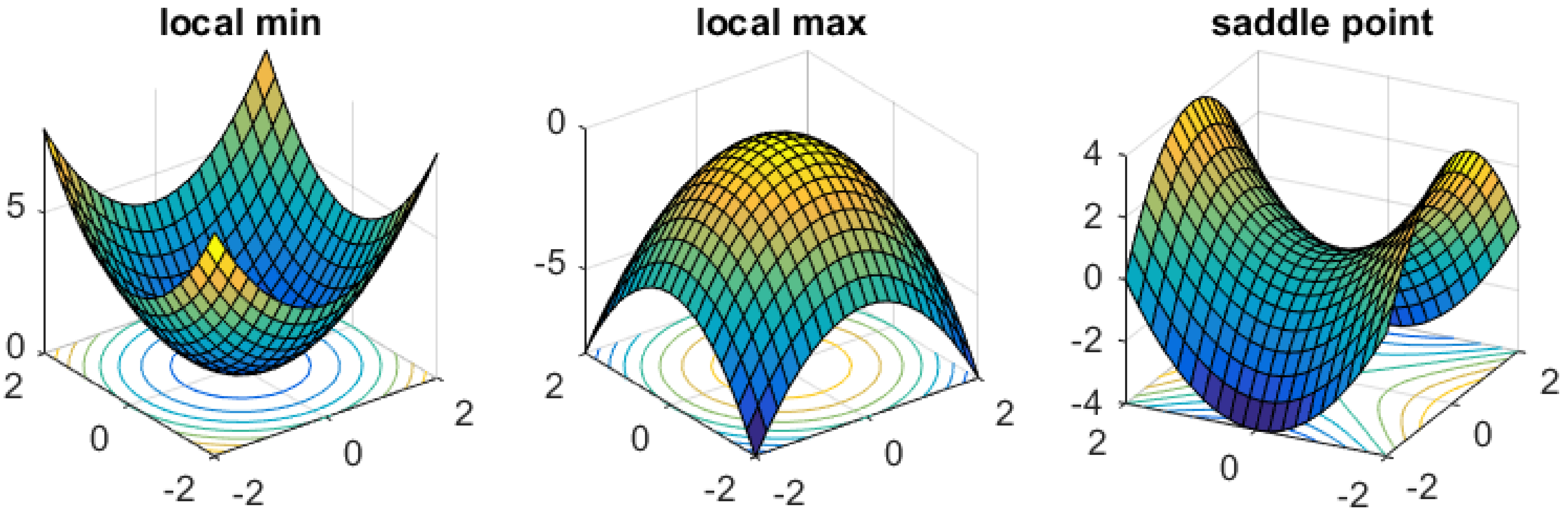


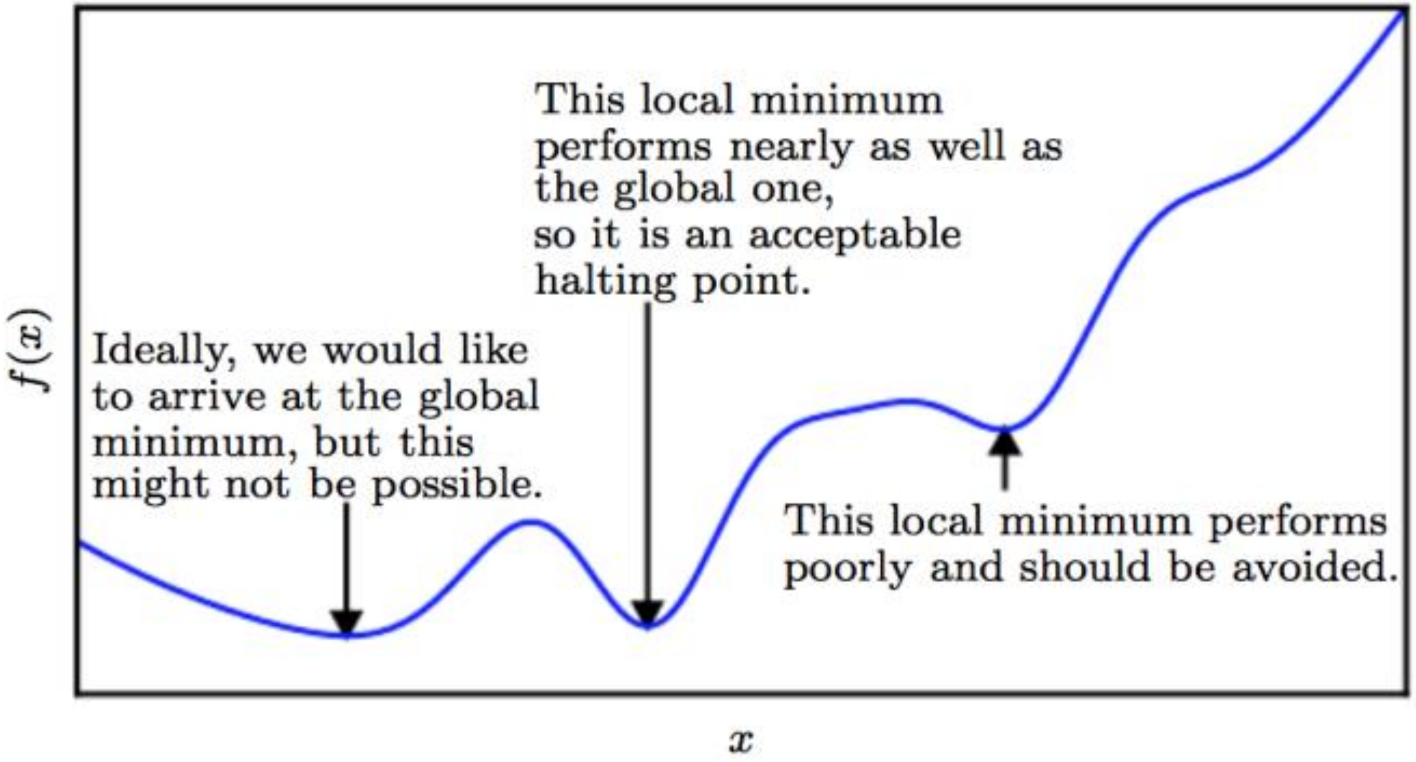
Figure 1.1 Minimum of $f(x)$ is same as maximum of $-f(x)$.

The Many Faces of Optimization!!



Presence of Multiple Optima (Minima)

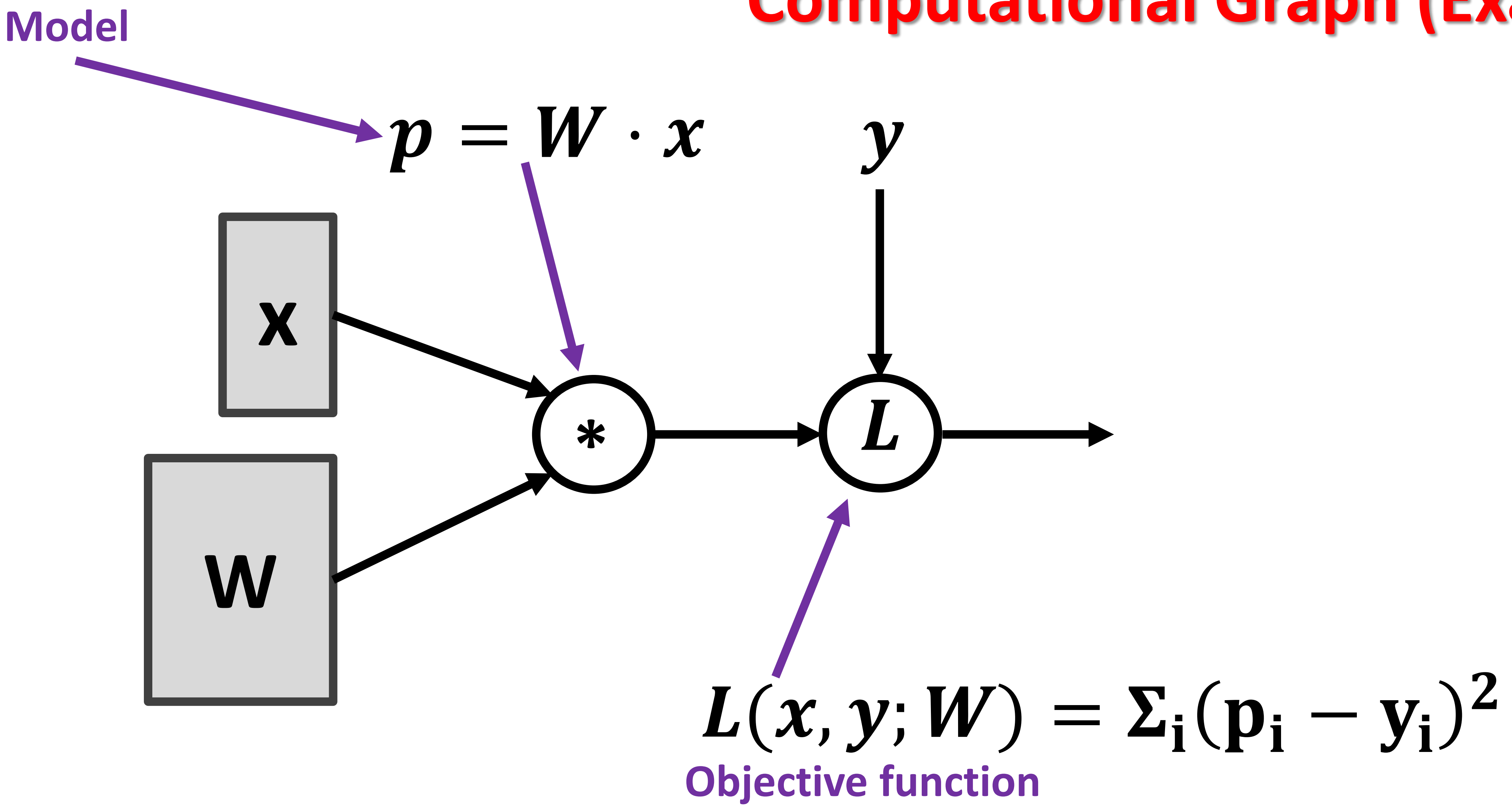
- Optimization algorithms may fail to find global minimum
- Generally accept such solutions



Minimizing with Multiple Inputs

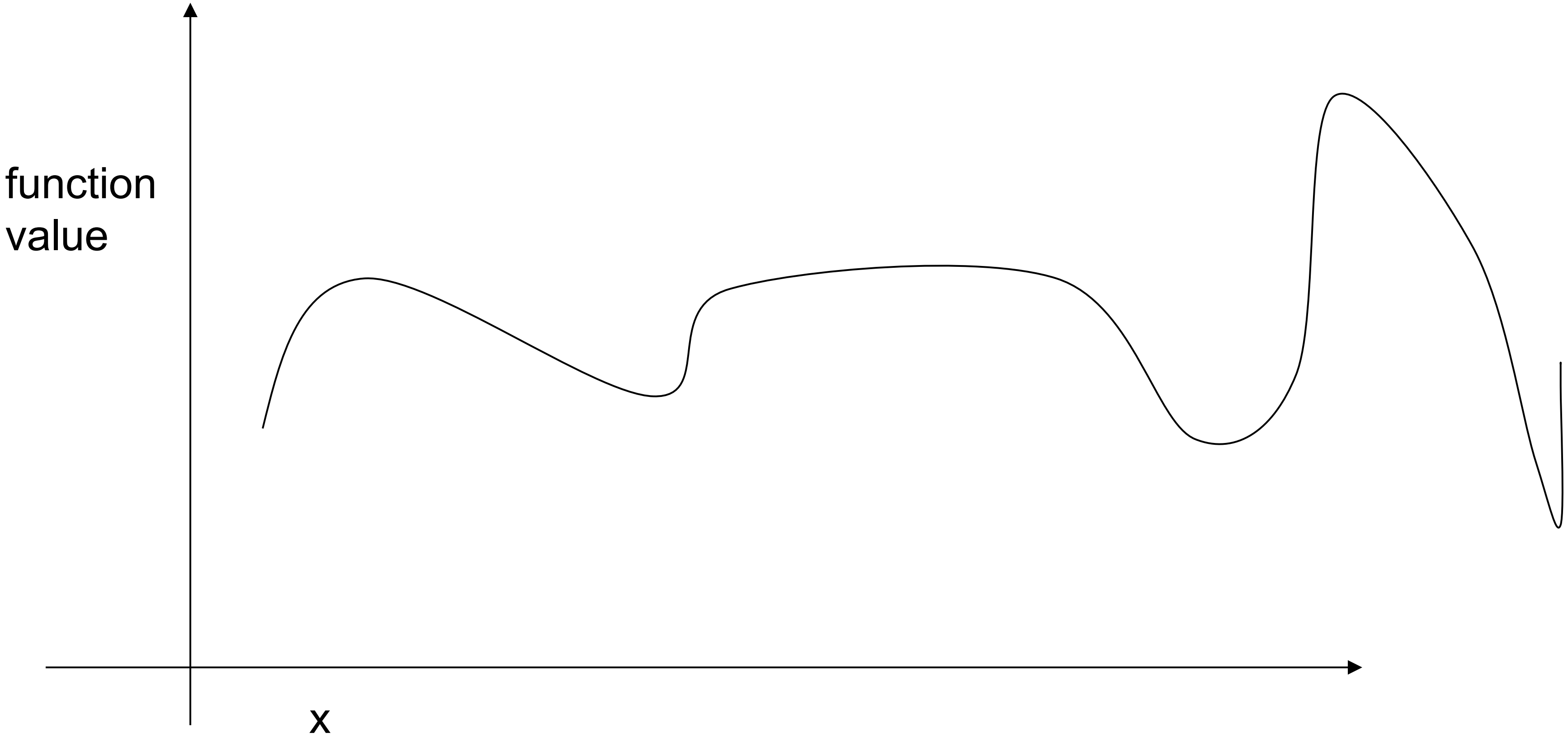
- We often minimize functions with multiple inputs: $f: R^n \rightarrow R$
- For minimization to make sense there must still be only one (scalar) output

Computational Graph (Example)



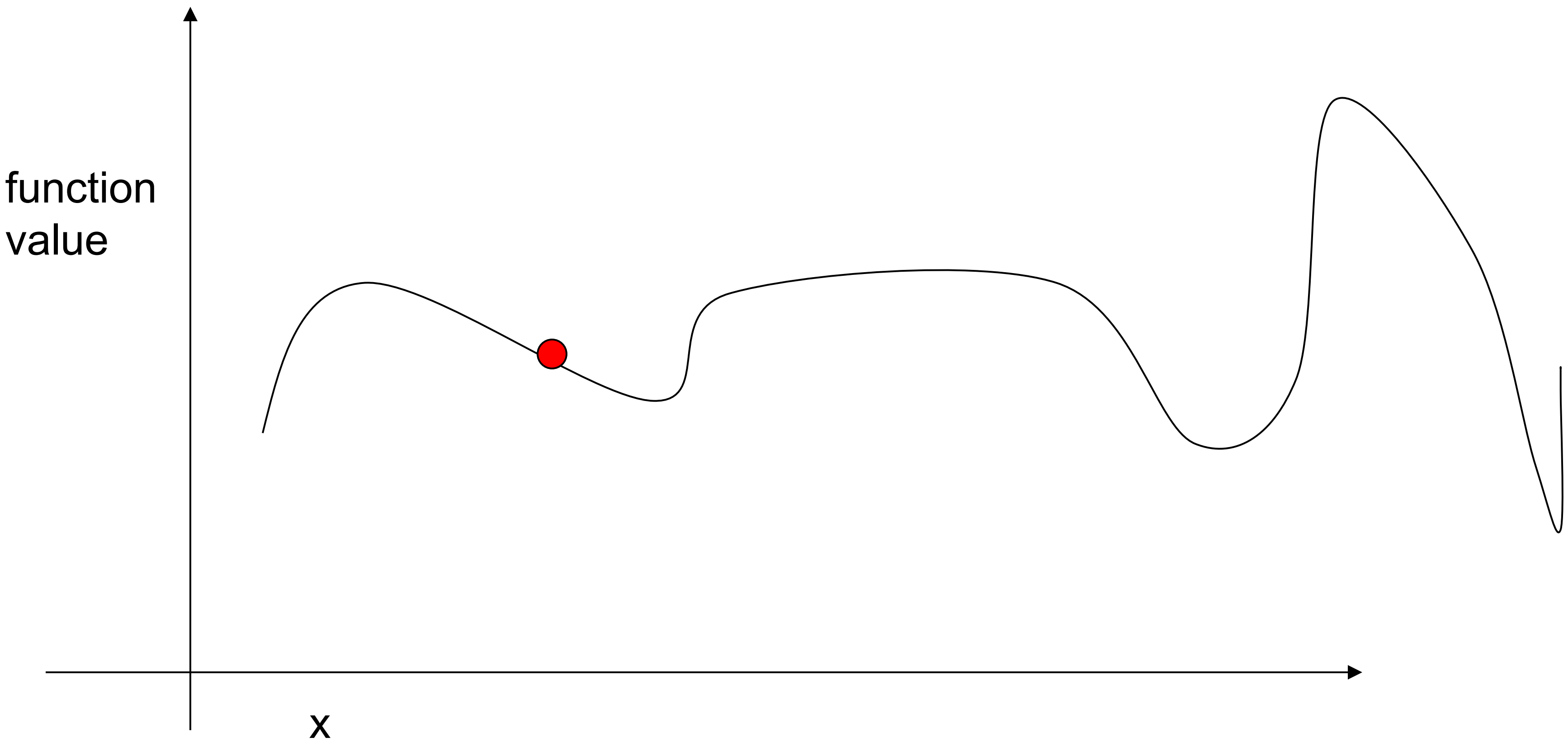
Simple Example: The Idealized Climb

- One dimension (typically use more):



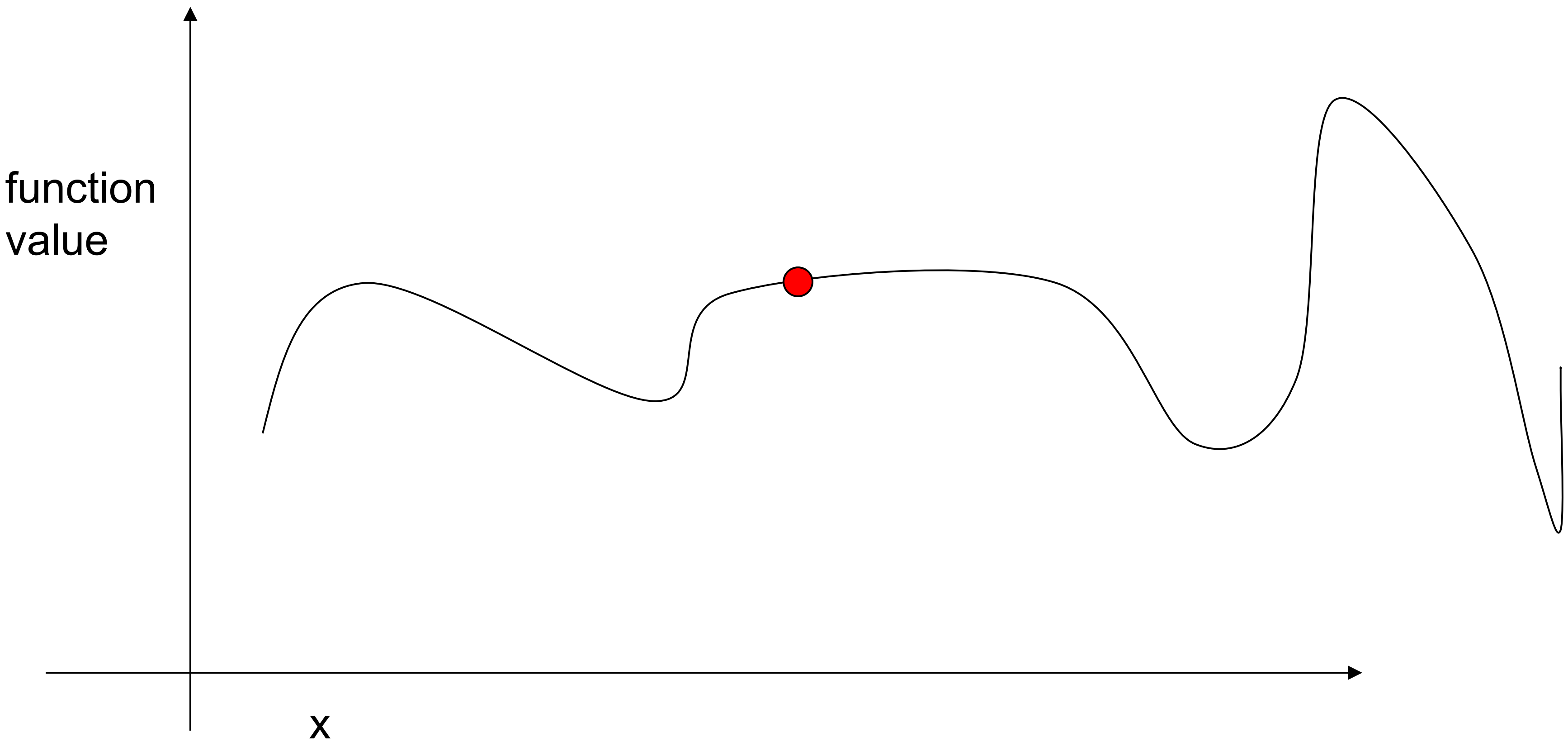
Simple Example: The Idealized Climb

- Start at a valid state, try to maximize



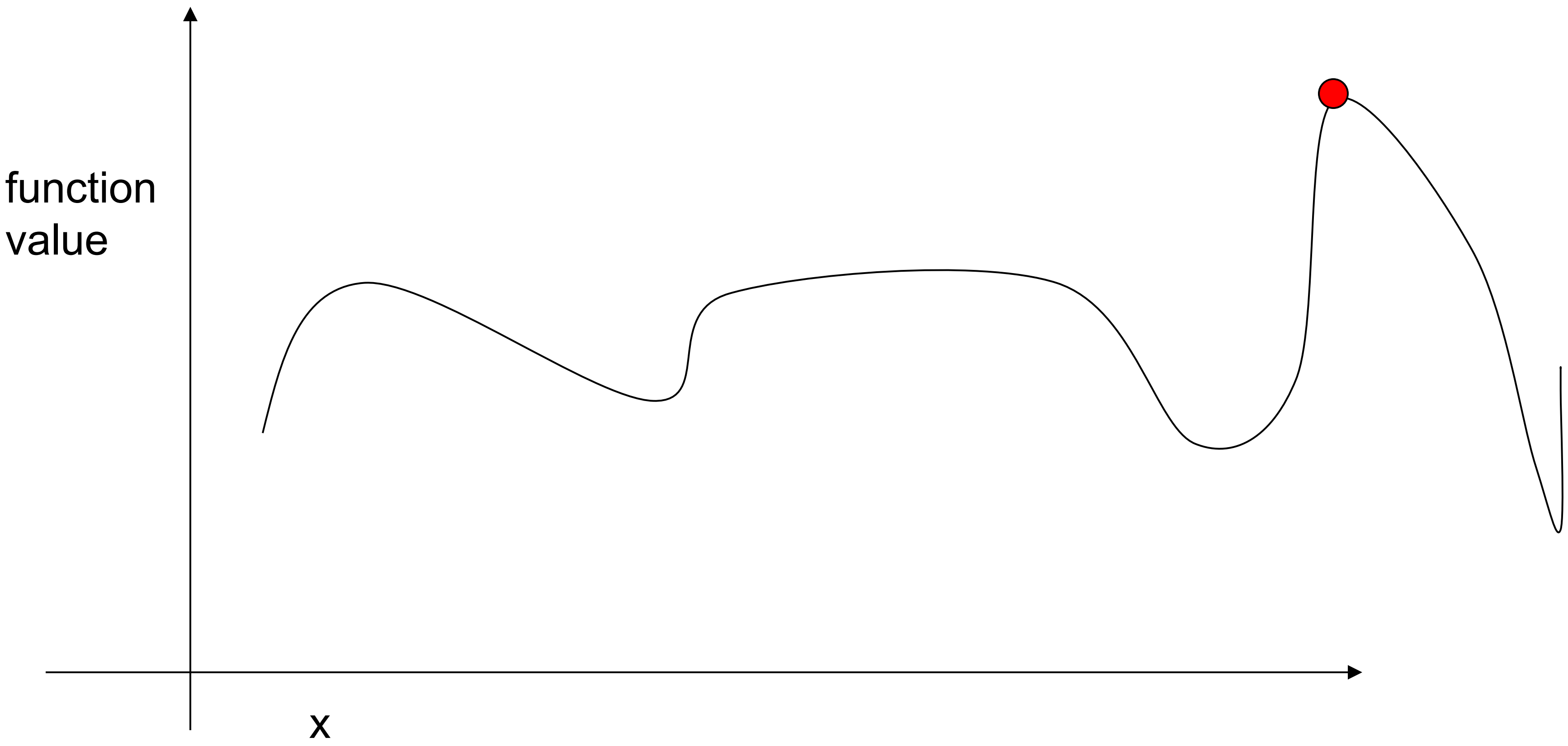
Simple Example: The Idealized Climb

- Move to better state



Simple Example: The Idealized Climb

- Try to find maximum



Stochastic Hill-Climbing Search

- Steepest ascent, but random selection/generation of neighbor candidates/positions (*<u>variations</u>*: first-choice hill climbing, **random-restart hill-climbing**)

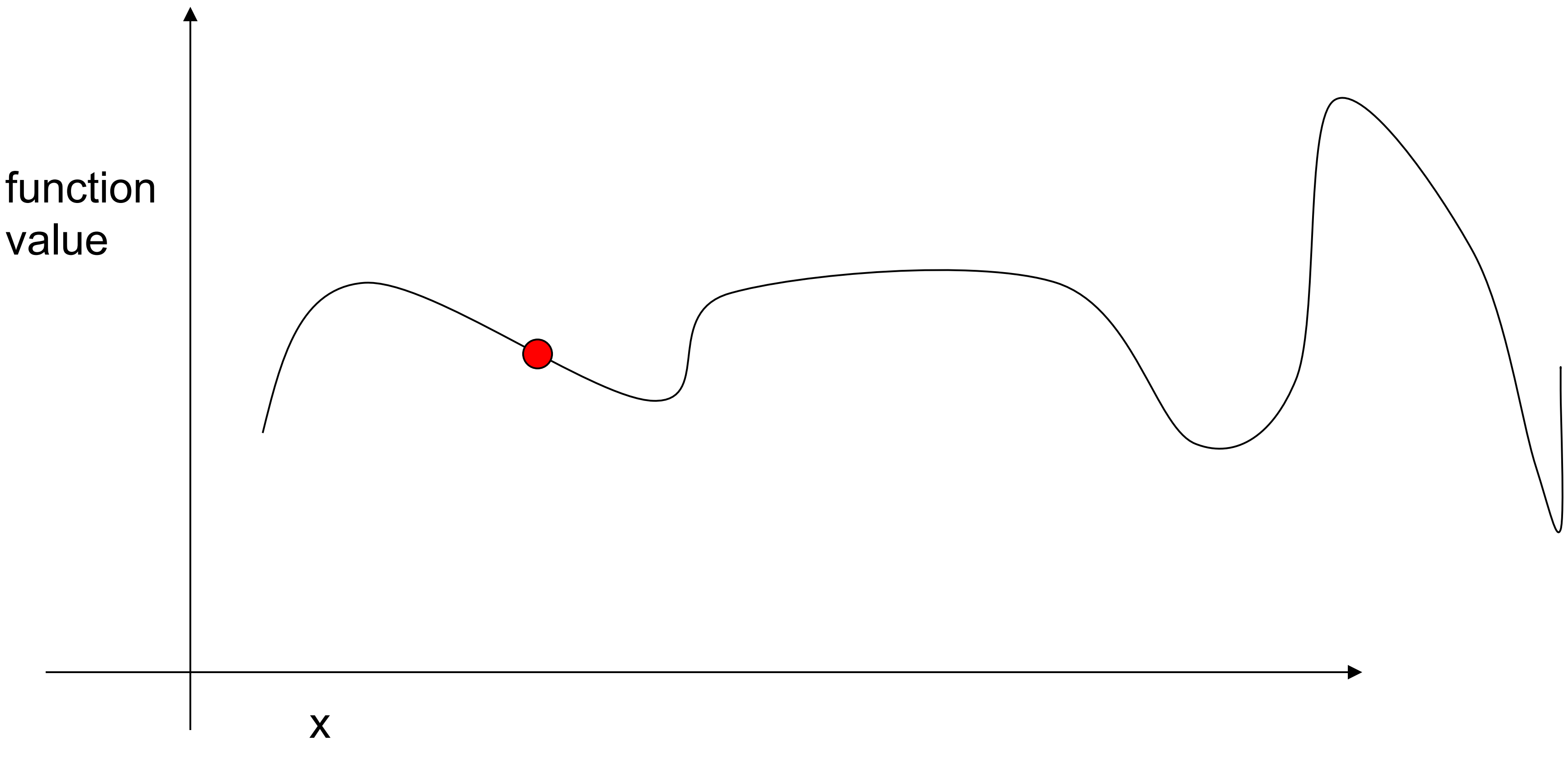
```
function HILL-CLIMBING(problem) returns a state that is a local maximum
   inputs: problem, a problem
   local variables: current, a node
                    neighbor, a node

   current ← MAKE-NODE(INITIAL-STATE[problem])
   loop do
      neighbor ← a highest-valued successor of current
      if VALUE[neighbor] ≤ VALUE[current] then return STATE[current]
      current ← neighbor
```

Generate a random sample/set of neighbors around *current*, choose highest-valued among them

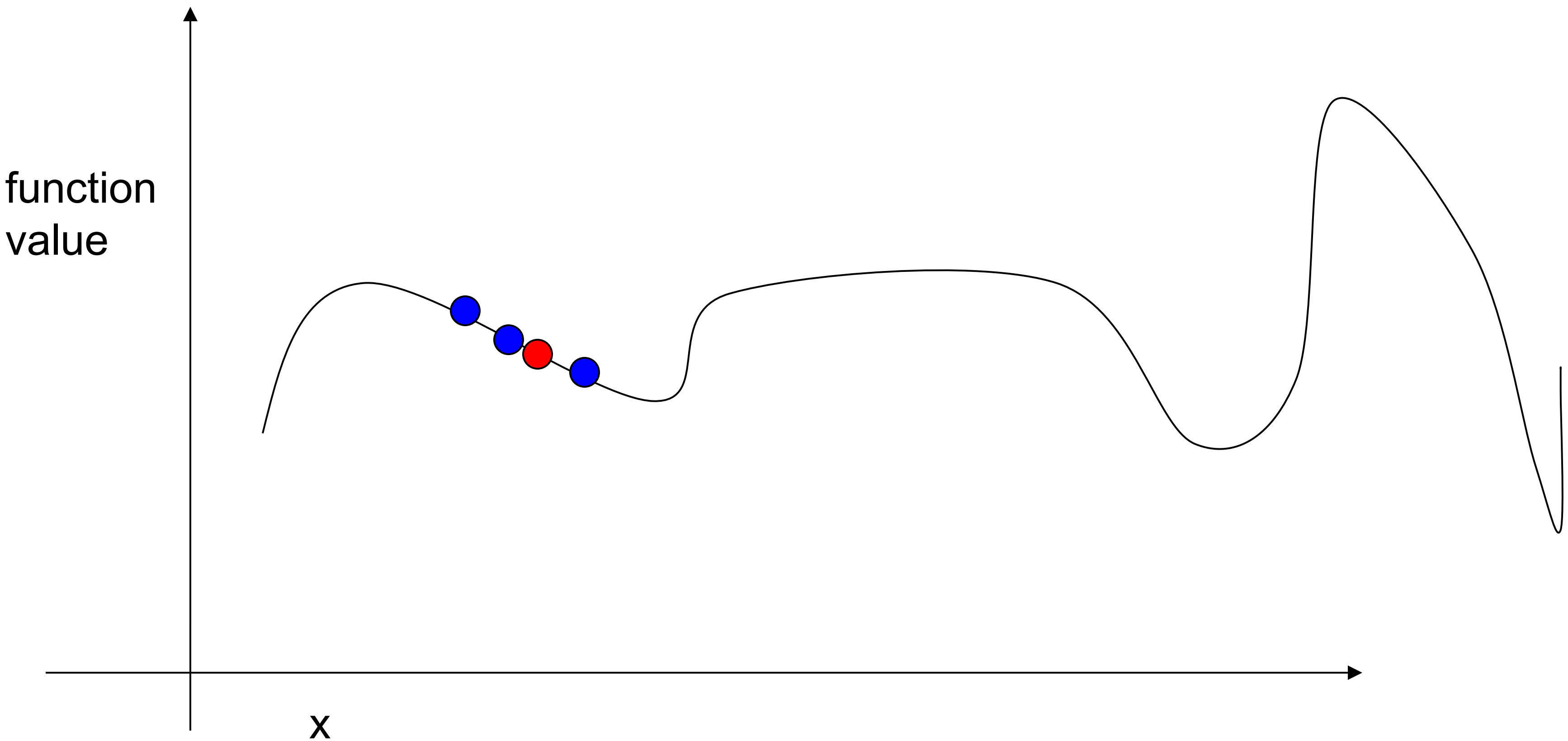
Stochastic Hill Climbing (Ascent)

- Random Starting Point



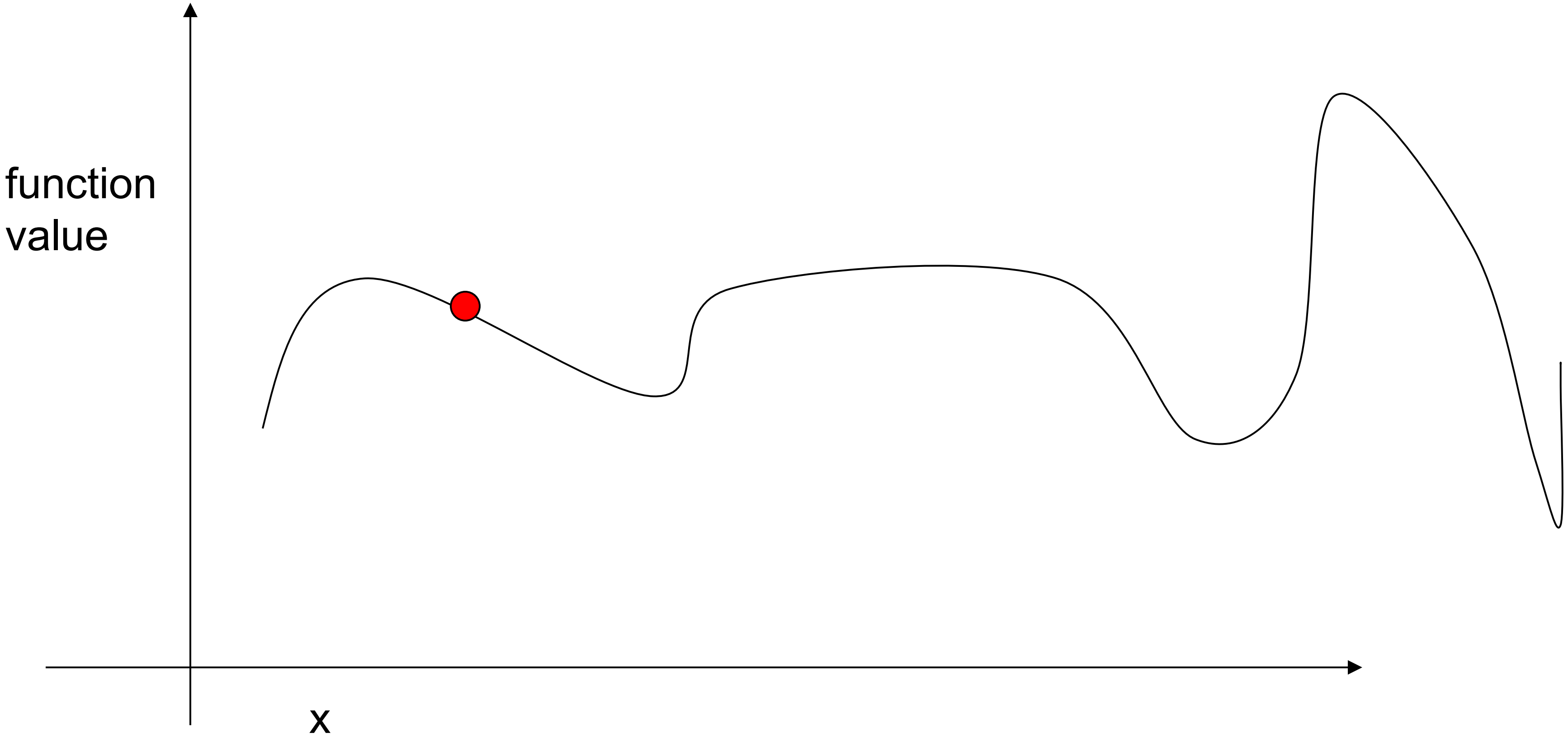
Stochastic Hill Climbing (Ascent)

- Three random steps



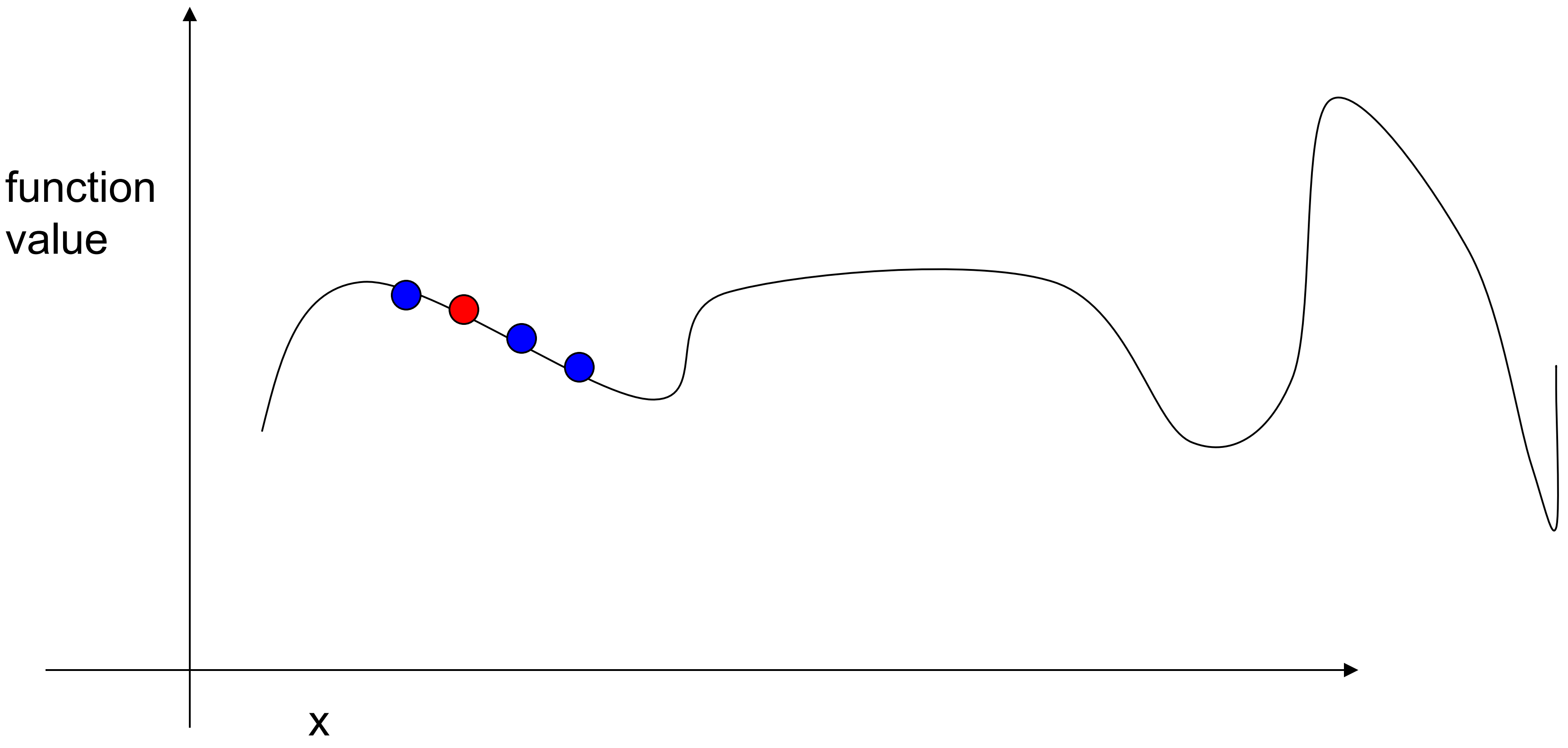
Stochastic Hill Climbing (Ascent)

- Choose Best One for new position



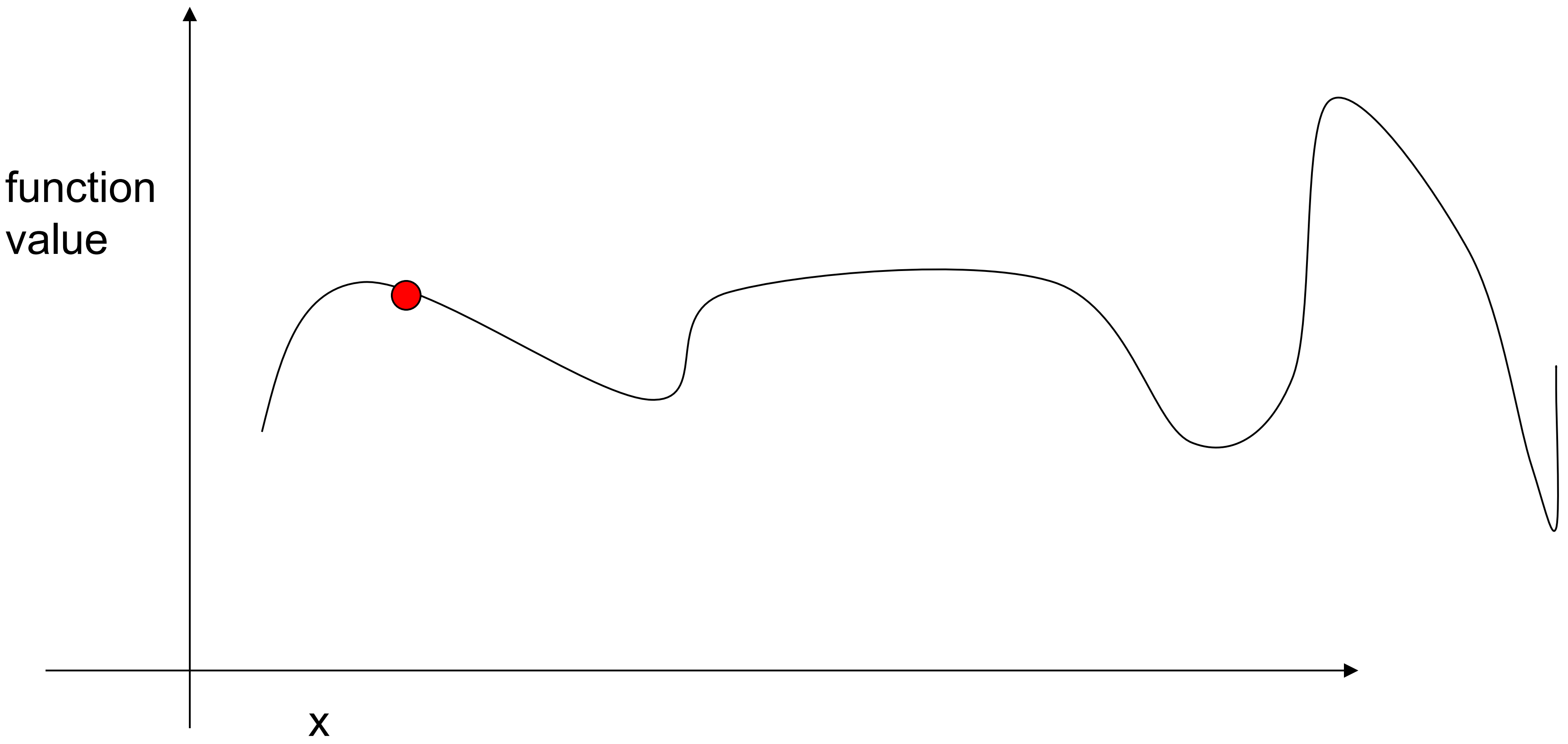
Stochastic Hill Climbing (Ascent)

- Repeat



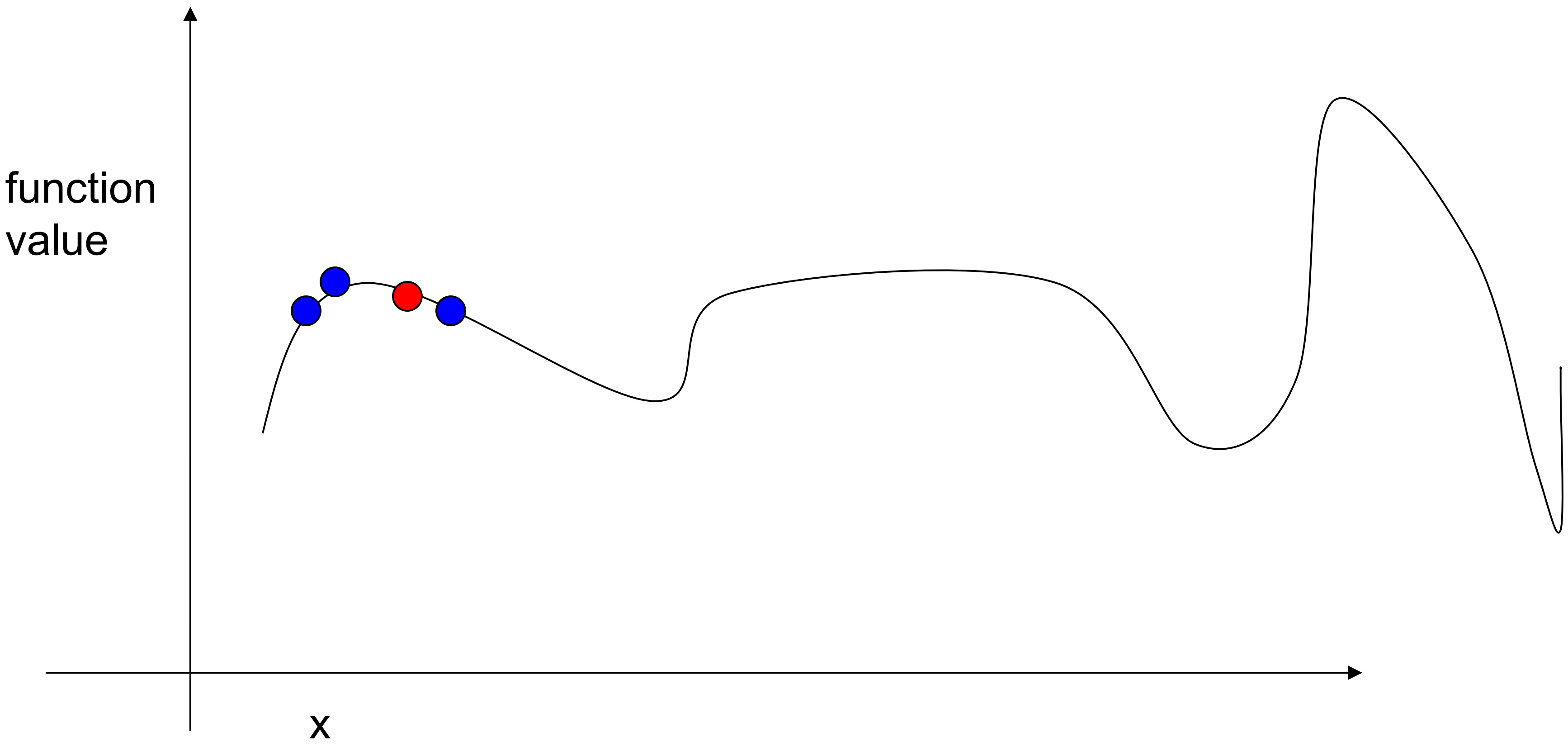
Stochastic Hill Climbing (Ascent)

- Repeat



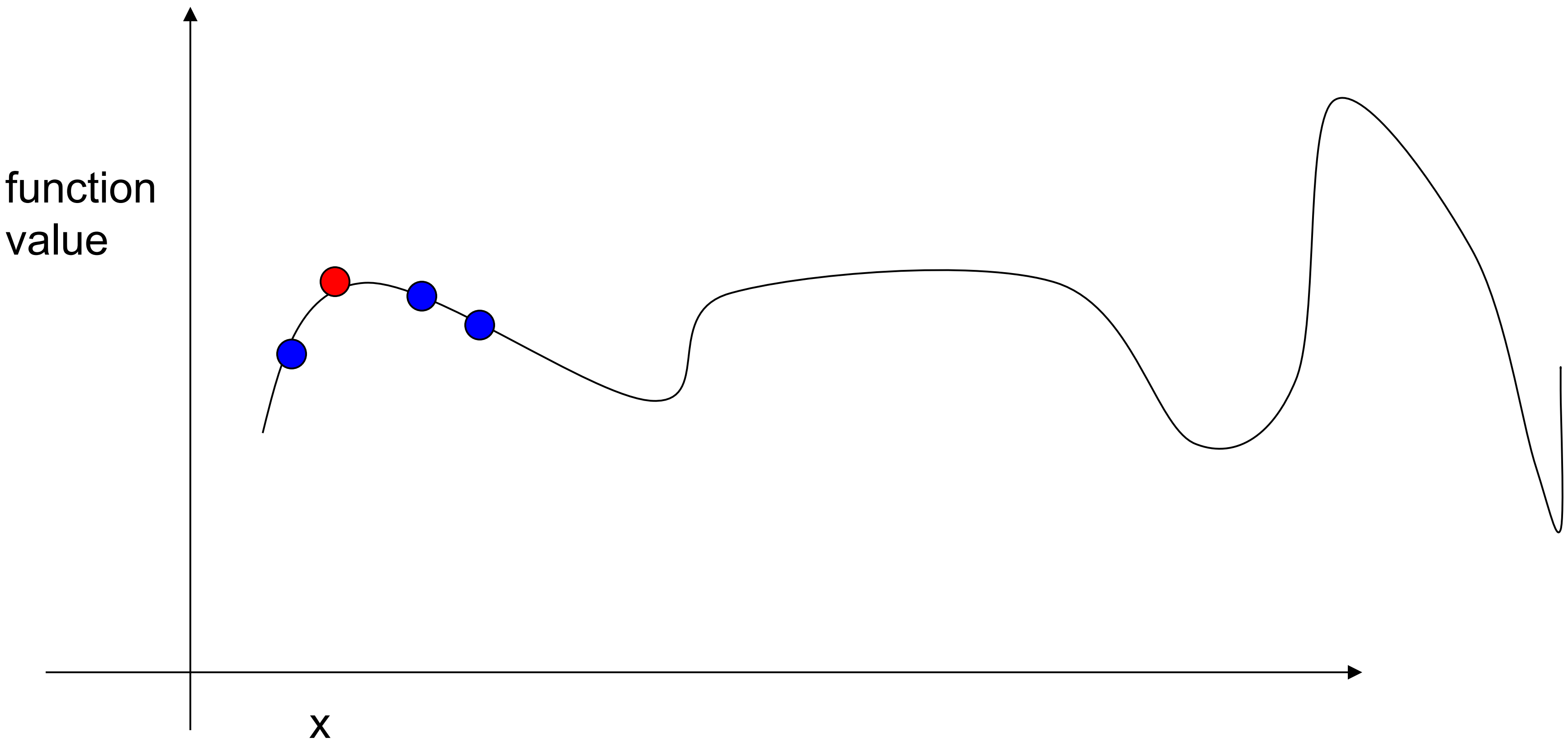
Stochastic Hill Climbing (Ascent)

- Repeat



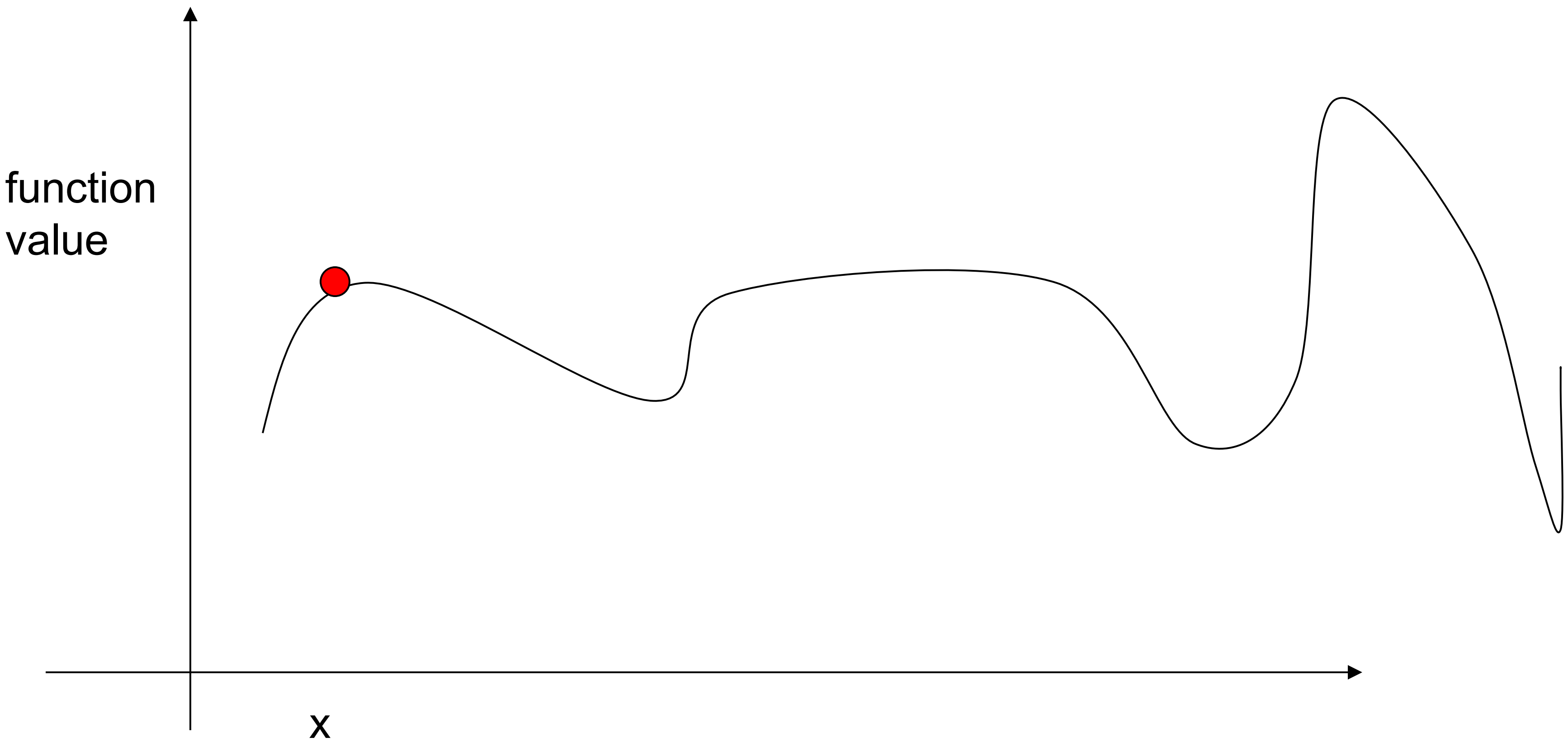
Stochastic Hill Climbing (Ascent)

- Repeat



Stochastic Hill Climbing (Ascent)

- No Improvement, so stop.



Gradient Descent/Ascent (What We Want!)

- Simple modification to hill climbing
  - Generally assumes a continuous state space
- Idea is to take more intelligent steps
- Look at local gradient: the direction of largest change
- Take step in that direction
  - Step size should be proportional to gradient
- Tends to yield (much) faster convergence to maximum

Discretization methods turn continuous space into discrete space, e.g., empirical gradient considers $\pm\delta$ change in each coordinate

Gradient methods compute

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial y_1}, \frac{\partial f}{\partial x_2}, \frac{\partial f}{\partial y_2}, \frac{\partial f}{\partial x_3}, \frac{\partial f}{\partial y_3}\right)$$

to increase/reduce f, e.g., by $\mathbf{x} \leftarrow \mathbf{x} + \alpha \nabla f(\mathbf{x})$

Questions?