COMPSCI 389
Introduction to Machine Learning

Days: Tu/Th. **Time:** 2:30 – 3:45 **Building:** Morrill 2 **Room:** 222

Topic 7.0: Gradient Descent

Prof. Philip S. Thomas (pthomas@cs.umass.edu)

Optimization Perspective

- Recall:
$$\text{argmin}_w \; L(w, D)$$
- Viewing $L(w, D)$ as a function, f, of just the weights (and a fixed data set):
$$\text{argmin}_w \; f(w)$$
- Note that this is equivalent to maximizing a different function, where $g = -f$
$$\text{argmax}_w \; g(w)$$
- We could also write x instead of w:
$$\text{argmin}_x \; f(x)$$
- The function being optimized (minimized or maximized) is called the **objective function** (optimization terminology).
    - In this case, our objective function is a **loss function** (machine learning terminology).
- **Question**: How do we find the input that minimizes a function?

Local Search Methods

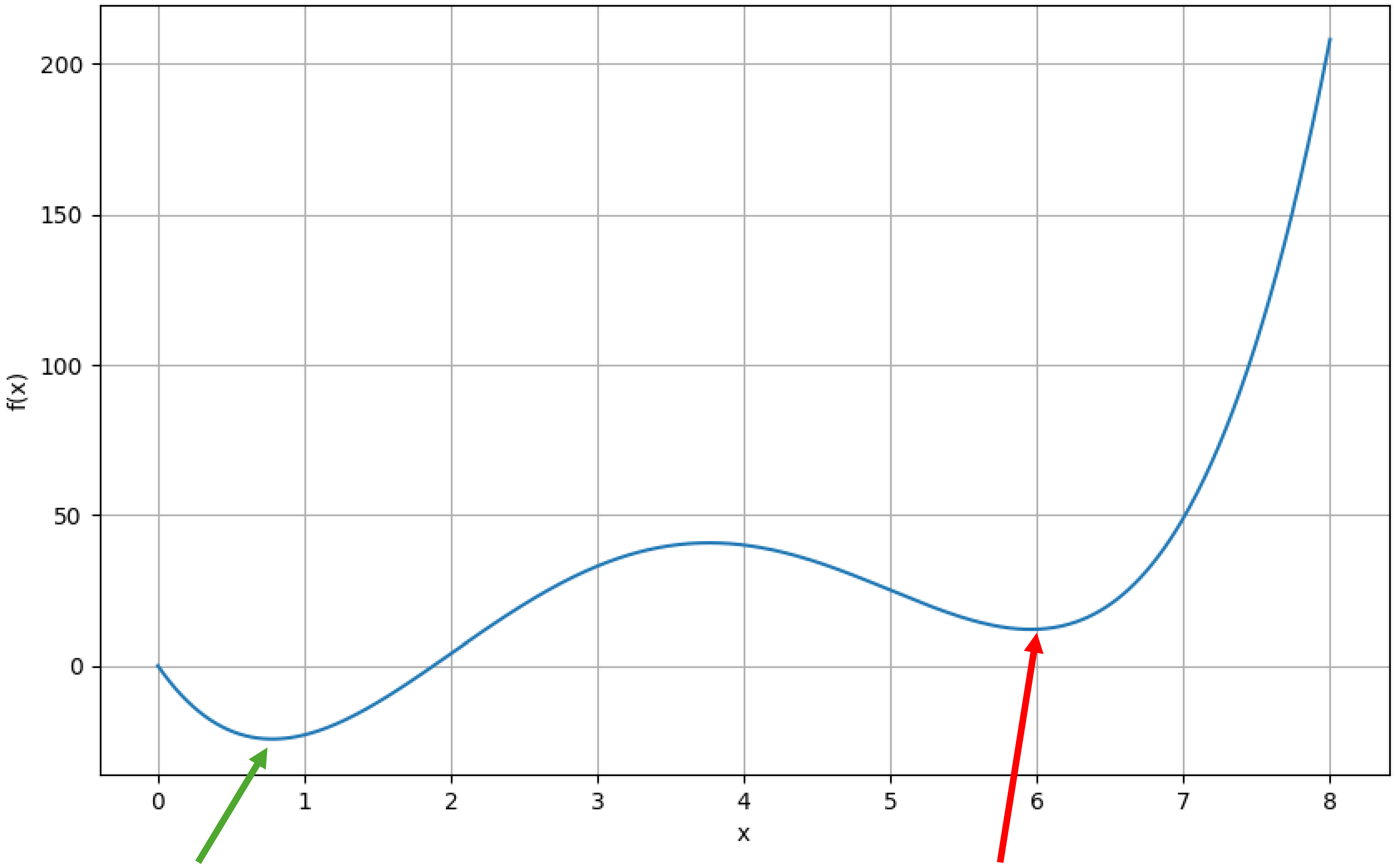
- Start with some initial input, x_0
- Search for a nearby input, x_1, that decreases f:
$$f(x_1) < f(x_0)$$
- Repeat, finding a nearby input x_{i+1} that decreases f (for each iteration i):
$$f(x_{i+1}) < f(x_i)$$
- Stop when:
  - You cannot find a new input that decreases f
  - The decrease in f becomes very small
  - The process runs for some predetermined amount of time
- Called "local search methods" because they search locally around some current point, x_i.

“Find a nearby point that decreases f”

- We will consider gradient-based optimizers.
- At any input/point x, we can query:
  - $f(x)$: The value of the objective function at the point
  - $\frac{df(x)}{dx}$: The derivative of the objective function at the point
    - This is the **gradient**, and is also written as $\nabla f(x)$

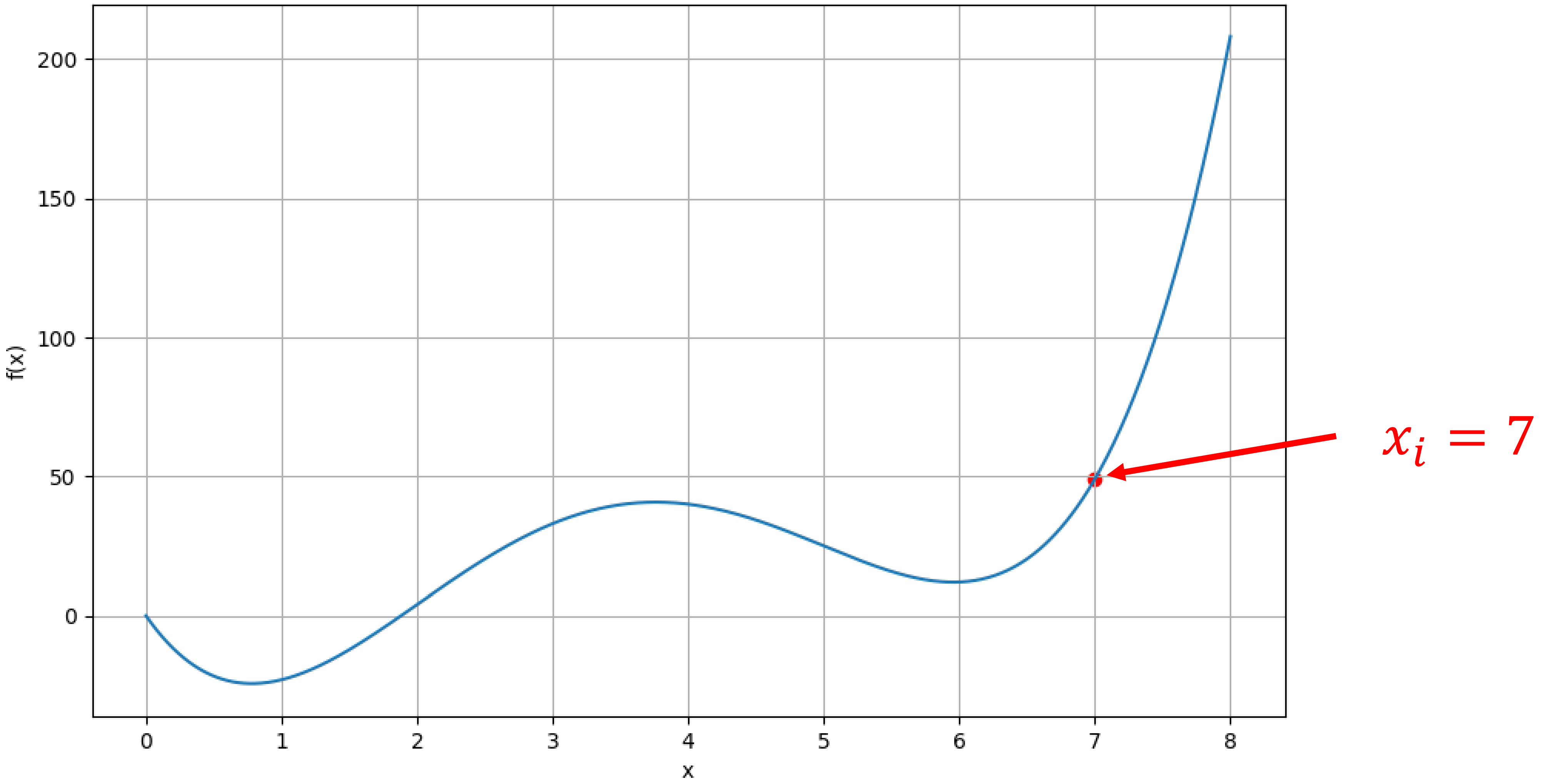
Question: Is a global minimum a local minimum?

Answer: Yes!

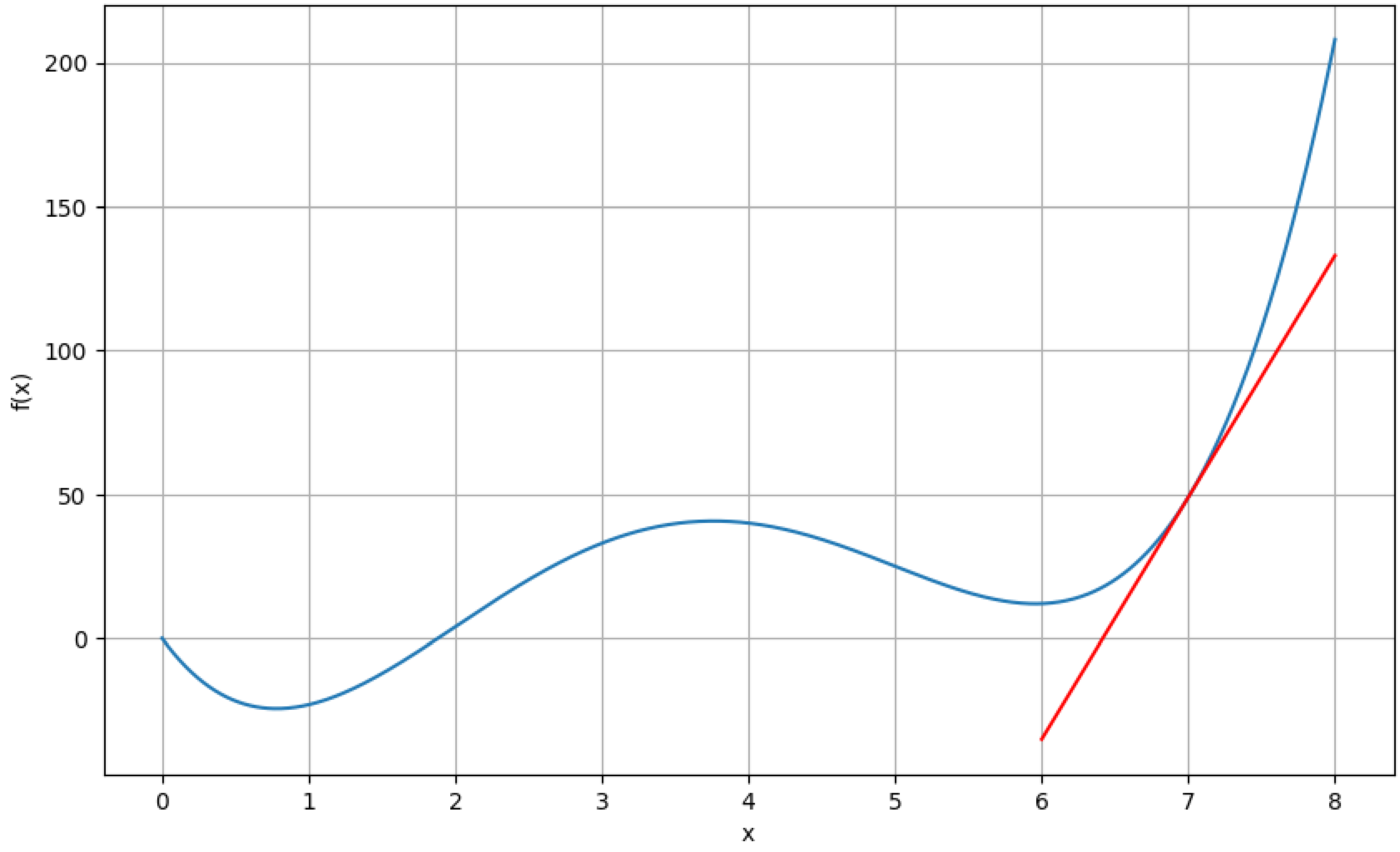


Global minimum: A location where the function achieves the lowest value (the argmin).

Local minimum: A location where all nearby (adjacent) points have higher values.

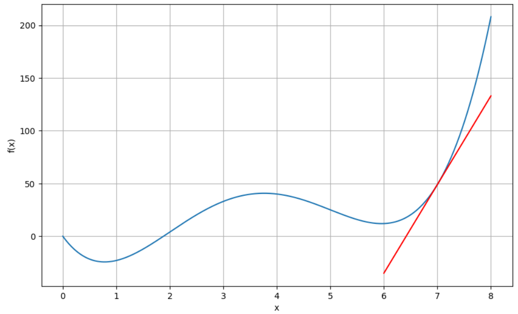


Question: How can we find a point x_{i+1} such that $f(x_{i+1}) < f(x_i)$? That is, a point that is “lower”?
Idea: Move a small amount “downhill”



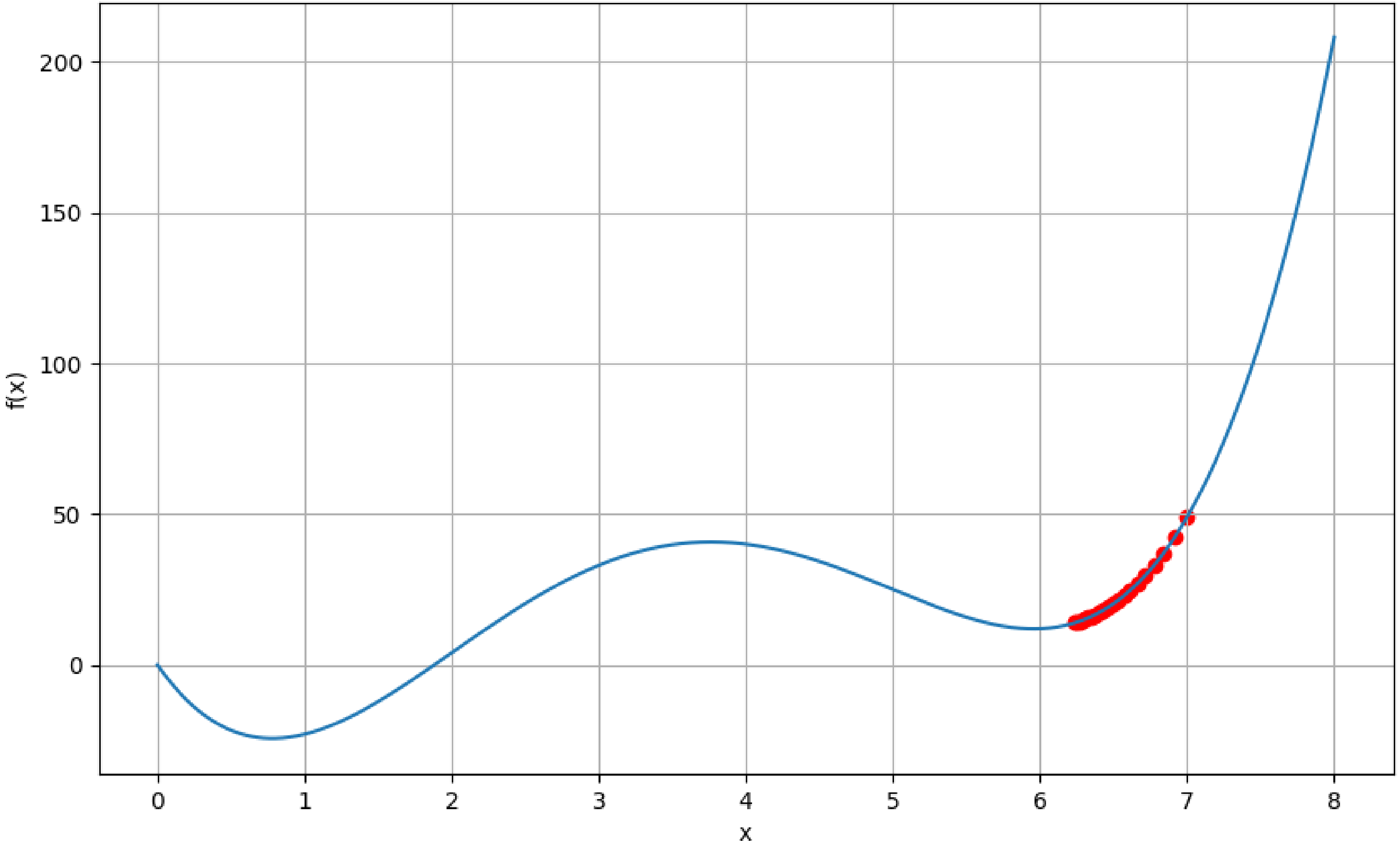
Notice: The slope of the function tells us which direction is uphill / downhill.
Positive slope: Decrease x_i to get x_{i+1}. **Negative slope**: Increase x_i to get x_{i+1}.

Gradient Descent



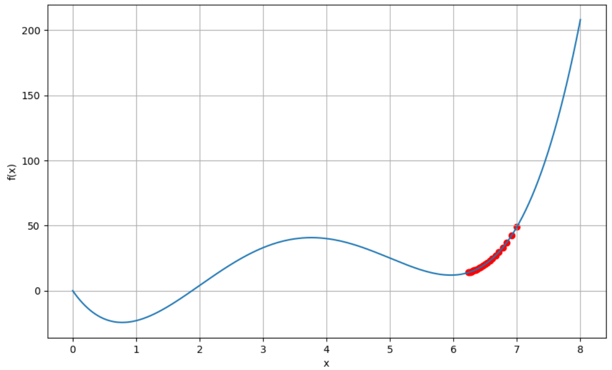
- Take a step of length α (a small positive constant) in the opposite direction of the slope:
$$x_{i+1} = x_i - \alpha \times \text{slope}.$$
- **Note**: The slope is $\frac{df(x)}{dx}$, so we can write:
$$x_{i+1} = x_i - \alpha \frac{df(x)}{dx}.$$
- α is a hyperparameter called the **step size** or **learning rate**.

Gradient descent, $x_0 = 7, \alpha = 0.001$
$f(x) = x^4 - 14x^3 + 60x^2 - 70x$



Question: Why do the points get closer together when we use the same step size, α?

Why do the points get closer together when we use the same step size, α?



$$x_{i+1} = x_i - \alpha \frac{df(x)}{dx}$$

- As x_i approaches a local optimum, the slope goes to zero.
- This allows for "convergence" to a local optimum.
- Gradient descent can still overshoot the (local) minimum.
- If the step size is small enough (or decayed appropriately over time), gradient descent is guaranteed to converge to a local minimum.
  - If it overshoots a minimum by a small amount, it will reverse direction and move back towards the minimum.
- If the step length was always constant, it could forever over-shoot the (local) minimum, not making progress towards the (local) minimum.