Optimization and Gradient Descent

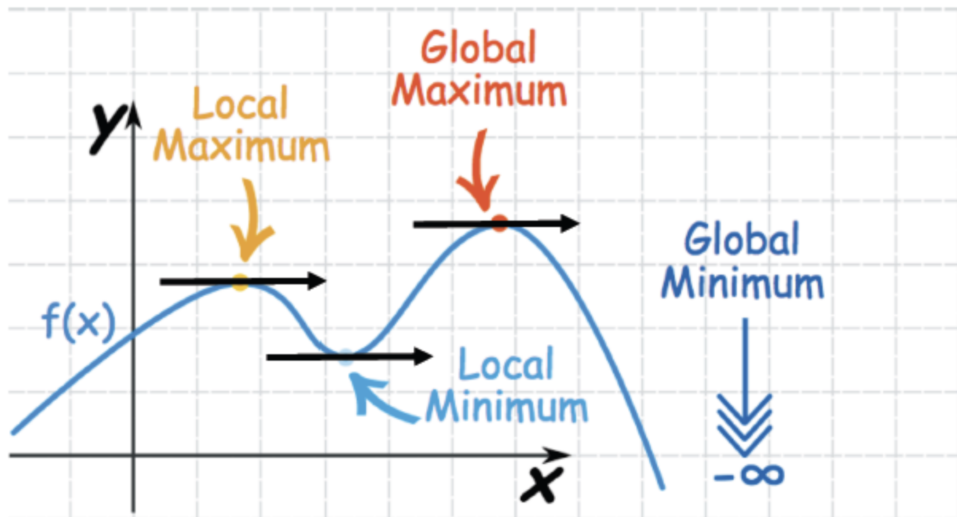
Optimization in the context of machine learning

- Given some sample output points y_i for *i = 1, …….,N and some* sample input points x_i for i = 1, …….,N and the function f() of the form $f(x_i,w_k)$, for k = 1,…, K where K is the number of parameters. Find w_k for k = 1,…, K that make $f(x_i,w_k)$ as close as possible to y_i for all *i = 1, …….,N*

- $\mathrm{j}(w_k) = \sum_{i=1}^{i=N} |(y_i \ - f(x_i, w_k) \ |^2$

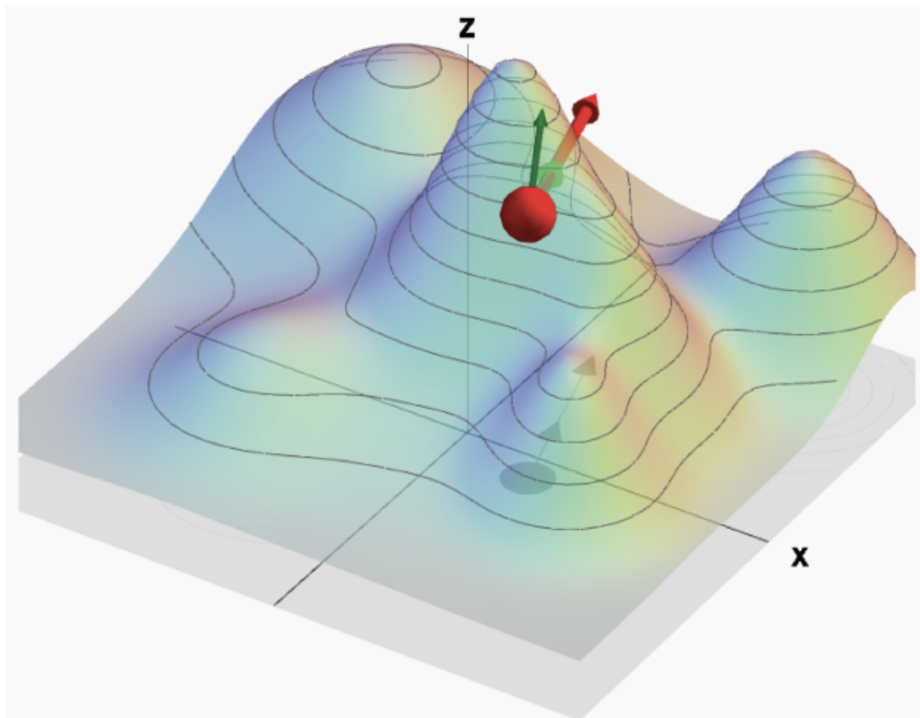
- *How to find w_k that minimize this cost function* $\mathrm{j}(w_k)$

Finding minia of $j(w_k)$



The derivative is *zero* at any local maximum or minimum.

Finding minia of $j(w_k)$



Assumption

- Let assume the cost function is f(x) where x is some parameters.
- How to find the best x that minimizes the cost function?

The derivative is *zero* at any local maximum or minimum.

One way to find a minimum: set f'(x)=0 and solve for x.

$f(x) = x^2$

$f'(x) = 2x$

$f'(x) = 0$ when $x = 0$, so minimum at $x = 0$

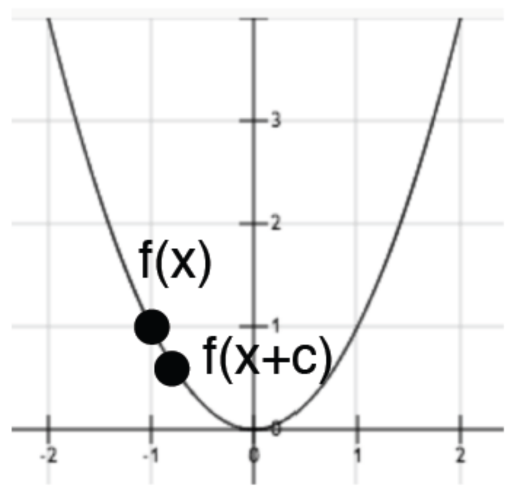
Rate of change in derivative

The slope at a point is called the **derivative** at that point

Intuition: Measure the slope between two points that are really close together

$$\frac{f(x + c) - f(x)}{c}$$

Limit as *c* goes to zero



The derivative is *zero* at any local maximum or minimum.

One way to find a minimum: set $f'(x)=0$ and solve for x.

- For most functions, there isn't a way to solve this.
- Instead: algorithmically search different values of x until you find one that results in a gradient near 0.

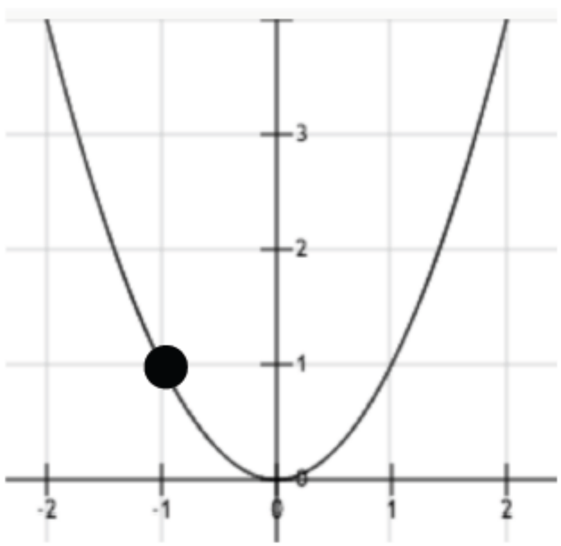
Finding minia

If the derivative is positive, the function is **increasing.**

- Don’t move in that direction, because you’ll be moving away from a trough.

If the derivative is negative, the function is **decreasing.**

- Keep going, since you’re getting closer to a trough

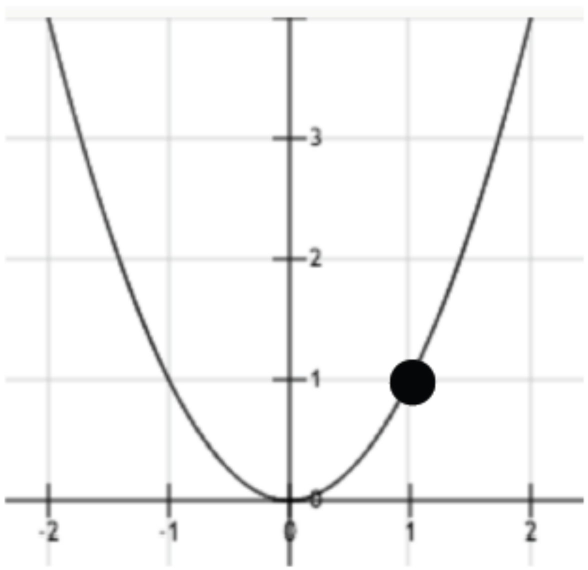


f'(-1) = -2

At x=-1, the function is decreasing as x gets larger. This is what we want, so let's make x larger.

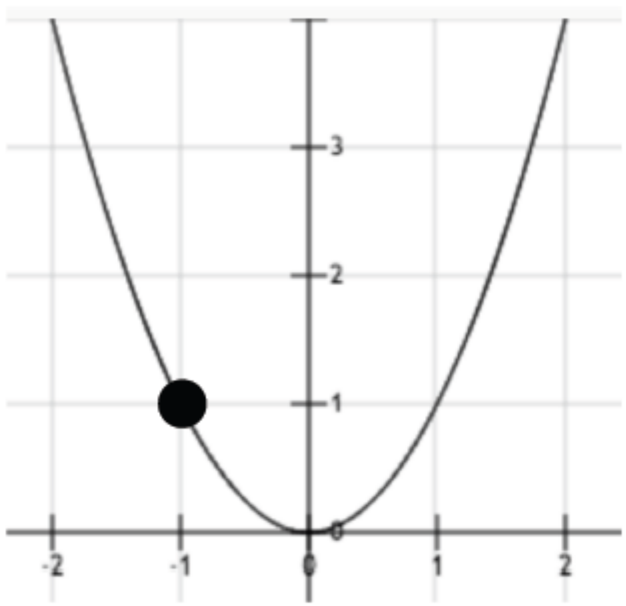
Increase x by the size of the gradient:

-1 + 2 = 1



f’(-1) = -2
At x=-1, the function is decreasing as x gets larger. This is what we want, so let’s make x larger.
Increase x by the size of the gradient:

$$-1 + 2 = 1$$

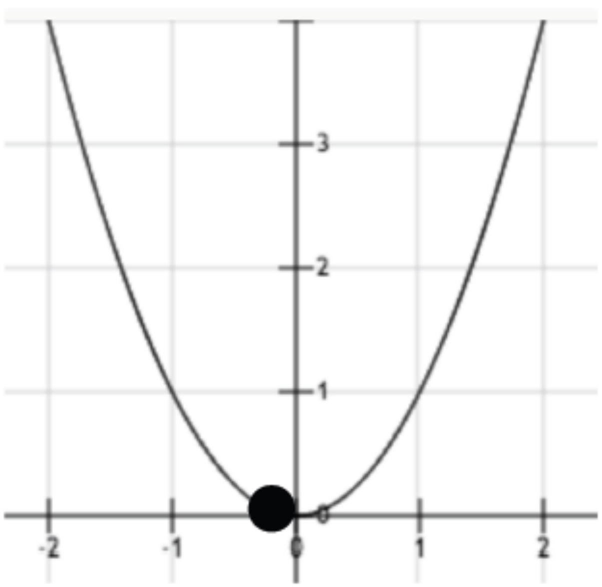


$f'(1) = 2$

At x=1, the function is increasing as x gets larger. This is not what we want, so let's make x smaller. Decrease x by the size of the gradient:

$$1 - 2 = -1$$

Learning rate solves this jumping behavior



f'(-1) = -2
x = -1 + 2(.25) = -0.5
f'(-0.5) = -1
x = -0.5 + 1(.25) = -0.25
f'(-0.25) = -0.5
x = -0.25 + 0.5(.25) = -0.125

Eventually we'll reach x=0.

Learning Rate

In order to guarantee that the algorithm will converge, the learning rate should decrease over time. Here is a general formula.

At iteration t:

$\eta_t = c_1 / (t^a + c_2)$,

where $0.5 < a < 2$

$c1 > 0$

$c2 \geq 0$

Gradient Descent

Gradient descent is guaranteed to eventually find a *local* minimum if:

- the learning rate is decreased appropriately;
- a finite local minimum exists (i.e., the function doesn't keep decreasing forever).

Gradient Descent

1. Initialize the parameters **w** to some guess (usually all zeros, or random values)
2. Update the parameters:
   $$\mathbf{w} = \mathbf{w} - \eta \, \nabla L(\mathbf{w})$$
   $$\eta = c_1 / (t^a + c_2)$$
3. Repeat step 2 until $\|\nabla L(\mathbf{w})\| < \theta$ or until the maximum number of iterations is reached.

Stopping Criteria

For most functions, you probably won't get the gradient to be exactly equal to **0** in a reasonable amount of time.

Once the gradient is sufficiently close to **0**, stop trying to minimize further.

How do we measure how close a gradient is to **0**?

Stopping Criteria

Stop when the norm of the gradient is below some threshold, θ:

$$||\nabla L(\mathbf{w})|| < \theta$$

Common values of θ are around .01, but if it is taking too long, you can make the threshold larger.

Description of Gradient Descent Method

- Algorithm (Gradient Descent Method)
  - **given** a starting point $x \in dom\, f$
  - **repeat**
    1. $\Delta x := -\nabla f(x)$
    2. Line search: Choose step size η via exact or backtracking line search
    3. Update $x := x + \eta\Delta x$
  - **until** stopping criterion is satisfied
- Stopping criterion usually $\|\nabla f(x)\|_2 \leq \epsilon$
- Very simple, but often very slow; rarely used in practice