Minimizing functions with Gradient Descent

Fundamentals of Artificial Intelligence

Fabien Cromieres

Kyoto University

http://lotus.kuee.kyoto-u.ac.jp/~fabien/lectures/IA/

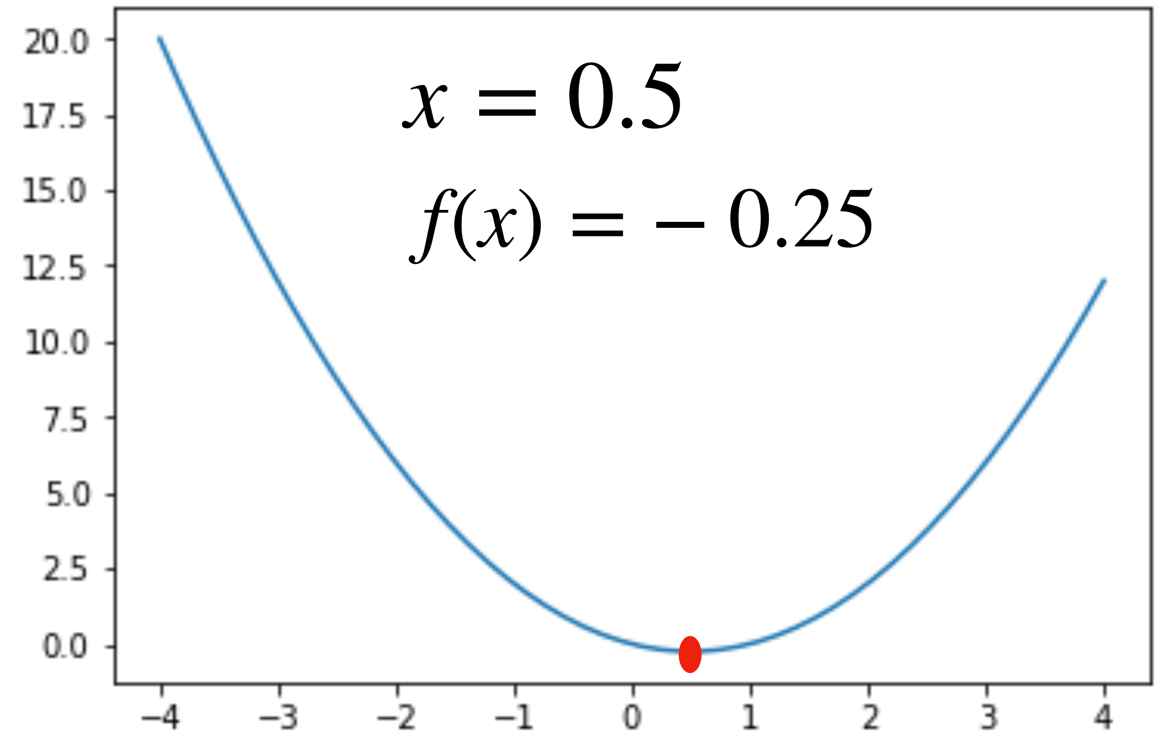
What we are going to do

- Given a function of **one variable**, find practically the value for which it is minimum
  - a.k.a "univariate function"
  - You should have seen how to do that for simple functions in High School
- Given a function of **several variables**, find the value for which it is minimum
  - a.k.a "multivariate function"

$$f : \mathbb{R} \rightarrow \mathbb{R}$$

$$f(x) = x^2 - x$$

$$\arg\min_x f(x)$$



$$f : \mathbb{R}^3 \rightarrow \mathbb{R}$$

$$f(x, y, z) = (x - y)^2 + z^2 - z$$

$$\arg\min_{x,y,z} f(x, y, z)$$

Why we do it?

- Actually, **almost all** algorithms of **supervised machine learning** can be reduced to finding the **minimum** of a function of several variable

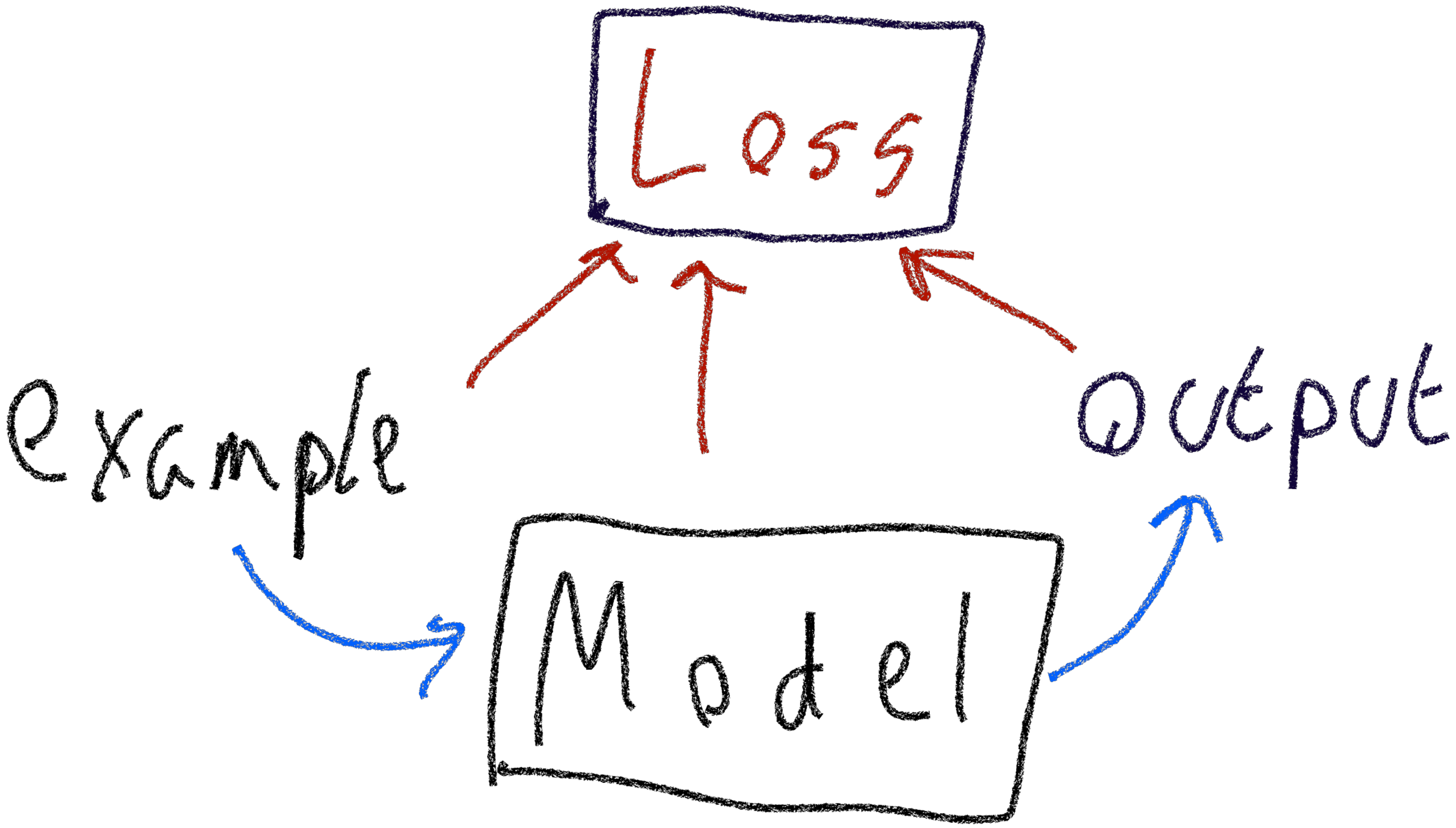
- The idea is that we will have a function called the ***loss*** that measure the difference between our system output and the training example

- By <u>minimizing the *loss*</u>, we make our system learn to reproduce the examples correctly

- Note that "function of several variables" can mean "<u>millions of variables</u>"

  - Some extremely big neural networks might have a loss function with <u>billions of variable</u>

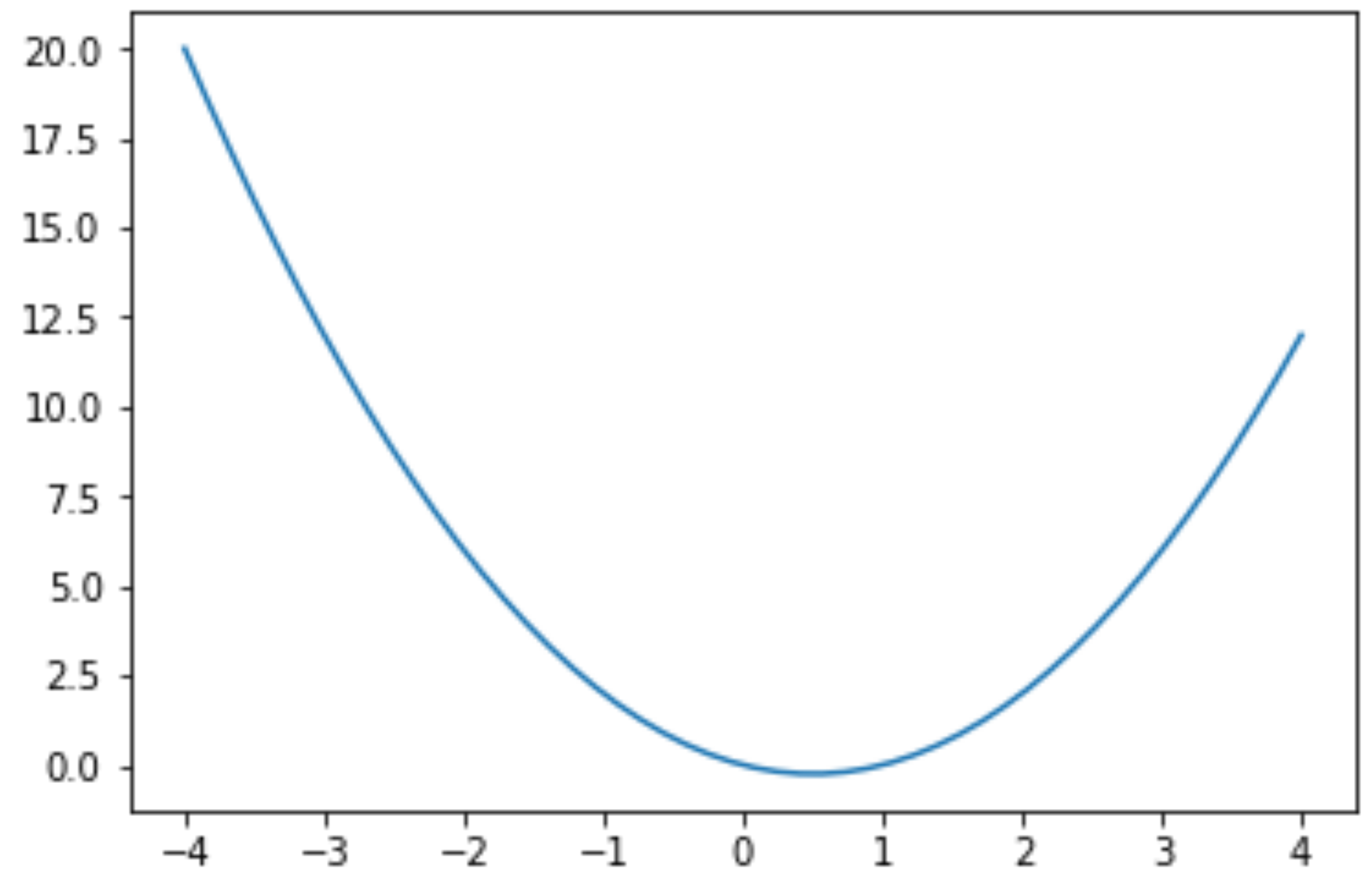
  - Fortunately, it is conceptually not more difficult to minimize a function of 2 variables or a function of one million variables



Minimizing a function of one variable

The “High School” view of minimization

- Let us start by recalling what we learn in high school
    - What you learned exactly might vary depending on your country of education and majors, but hopefully you all have some experience with this



To minimize f(x):

1. Compute **first derivative** f’(x)
2. Compute **second derivative** f”(x)
3. Find x0 such that f’(x0) = 0
4. If f”(x0) > 0 then x0 is a **local minimum** of f

$$f : \mathbb{R} \to \mathbb{R}$$

$$f(x) = x^2 - x$$

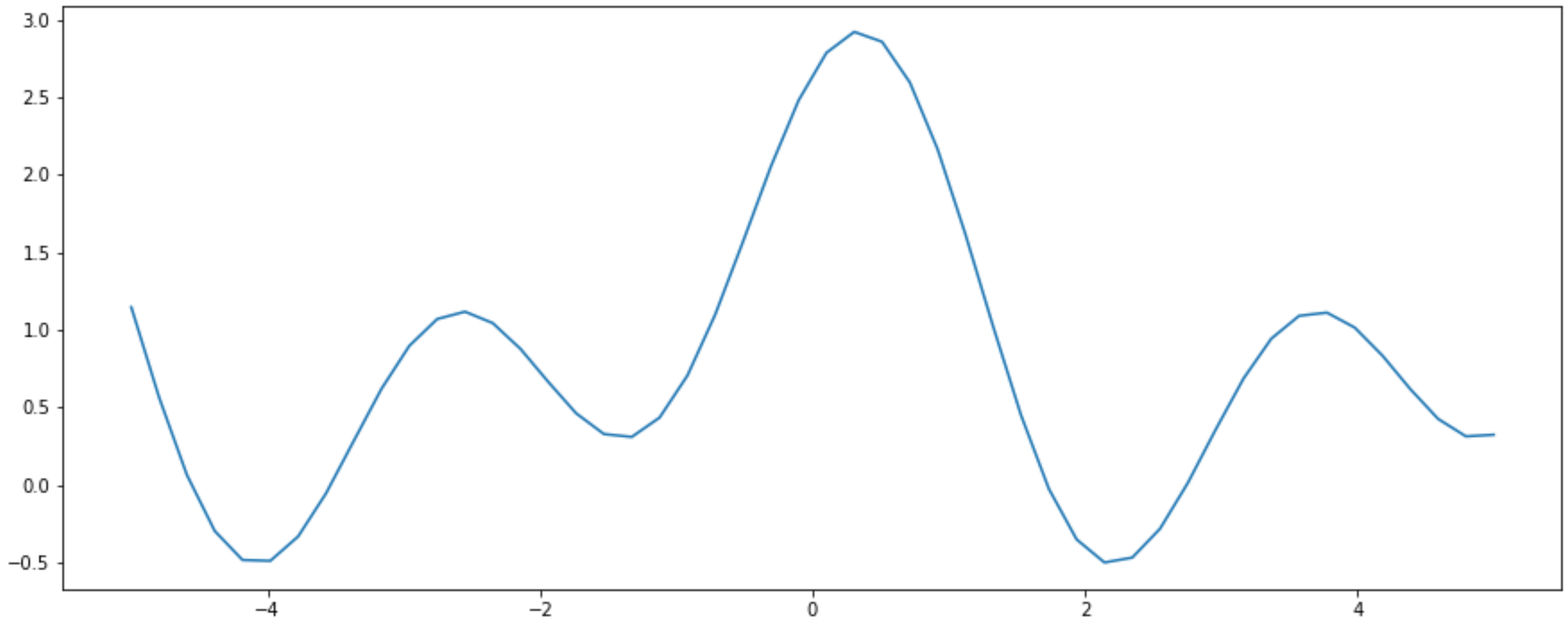
$$f'(x) = 2x - 1 \rightarrow x_0 = 0.5$$

$$f''(x) = 2$$

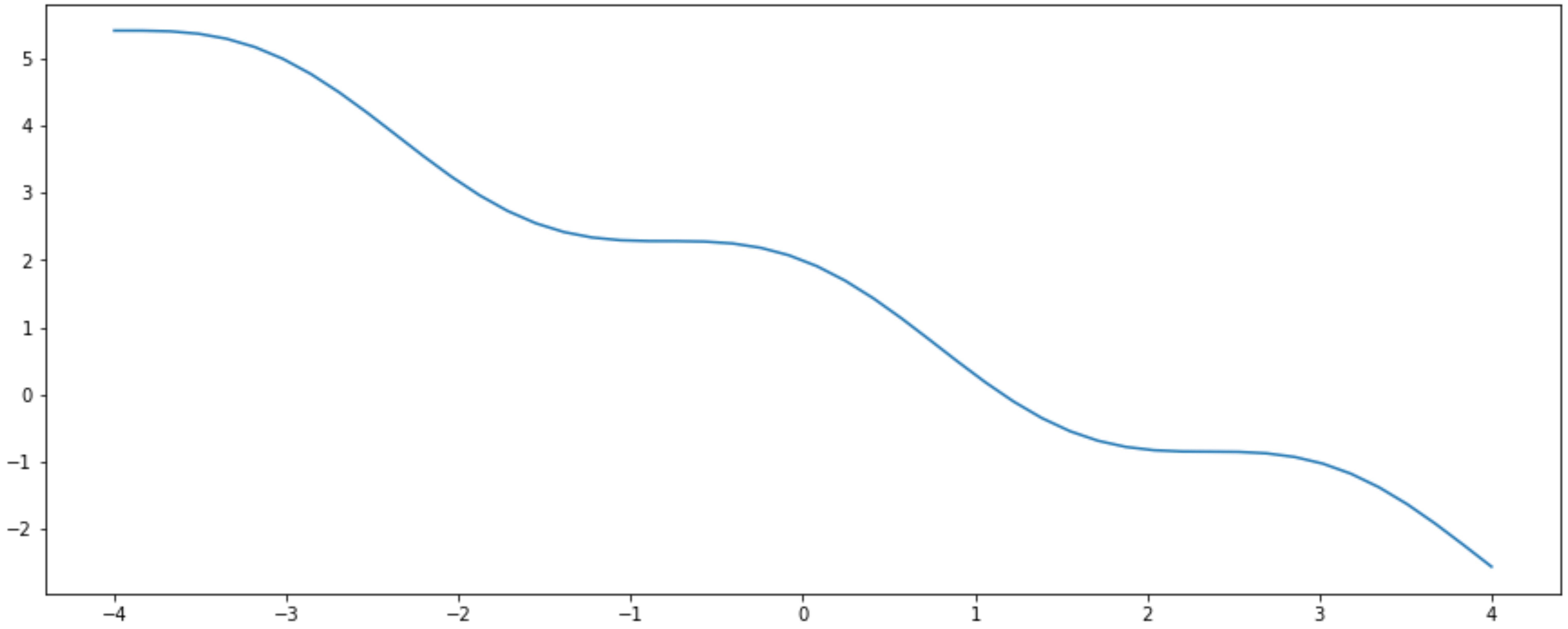
Note: It is not how we will minimize functions in practice

Local minimum, local maximum

- Note that the condition on the second derivative is important to distinguish **minimums** from **maximum**
- Also, the solution could be only a **local minimum**



- It is even possible for derivative to be 0 even if the function has no minimum



About derivatives

- Does everybody remember how to compute derivatives?
- Do not panic if you don't.
  - In practice, we will have functions that can compute the derivatives automatically for us
  - Still, you should understand at least how they work
  - we will review briefly the basics

Computing derivatives

Exercise:

$sin(x) + log(x)$

$2 \times log(x + 1)$

$sin(2x)$

$\dfrac{e^x}{x}$

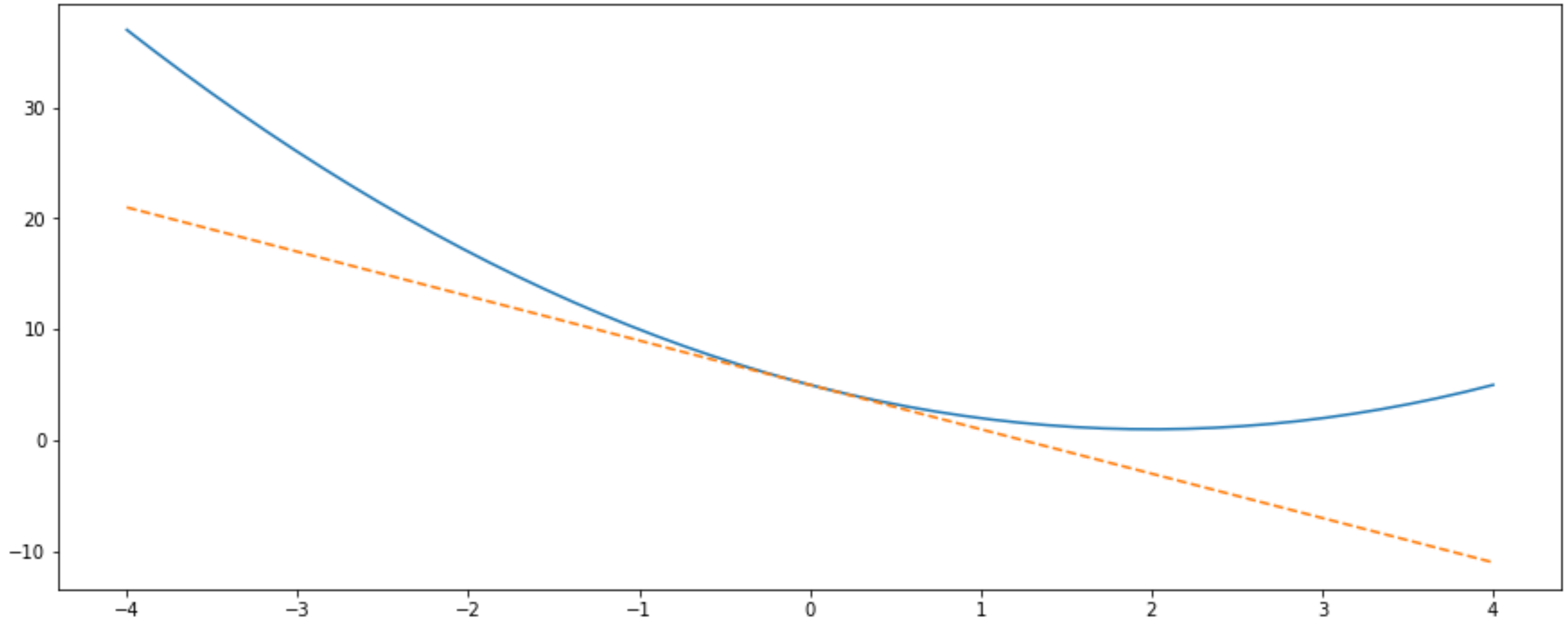
f(x)	**f'(x)**	
sin(x)	cos(x)	
cos(x)	-sin(x)	
x^n	nx^{n-1}	
log(x)	$\frac{1}{x}$	
e^x	e^x	
$g(h(x))$	$h'(x) \times g'(h(x))$	**Composition rule**
$g(x) \times h(x)$	$g'(x) \times h(x) + g(x) \times h'(x)$	**Leibniz rule**
$g(x) + h(x)$	$g'(x) + h'(x)$	**Linearity I**
$\alpha \cdot h(x)$	$\alpha \cdot h'(x)$	**Linearity II**

What is a derivative?

- One <u>definition</u>: the coefficient of the best linear approximation of a function at x
- If h is small: $f(x+h) \approx f(x) + h \cdot f'(x)$
- Example:
  - if we know that $log(2.3) = 0.832909...$
  - How much is $log(2.4)$?
    - Supposing we cannot compute a log again
  - 2.4 = 2.3 + 0.1
  - We can approximate: $log(2.4) \approx log(2.3) + 0.1 \times \frac{1}{2.3}$
  - Which gives: $log(2.3) + 0.1 \times \frac{1}{2.3} = 0.876387...$
  - The true value is: $log(2.4) = 0.875468...$

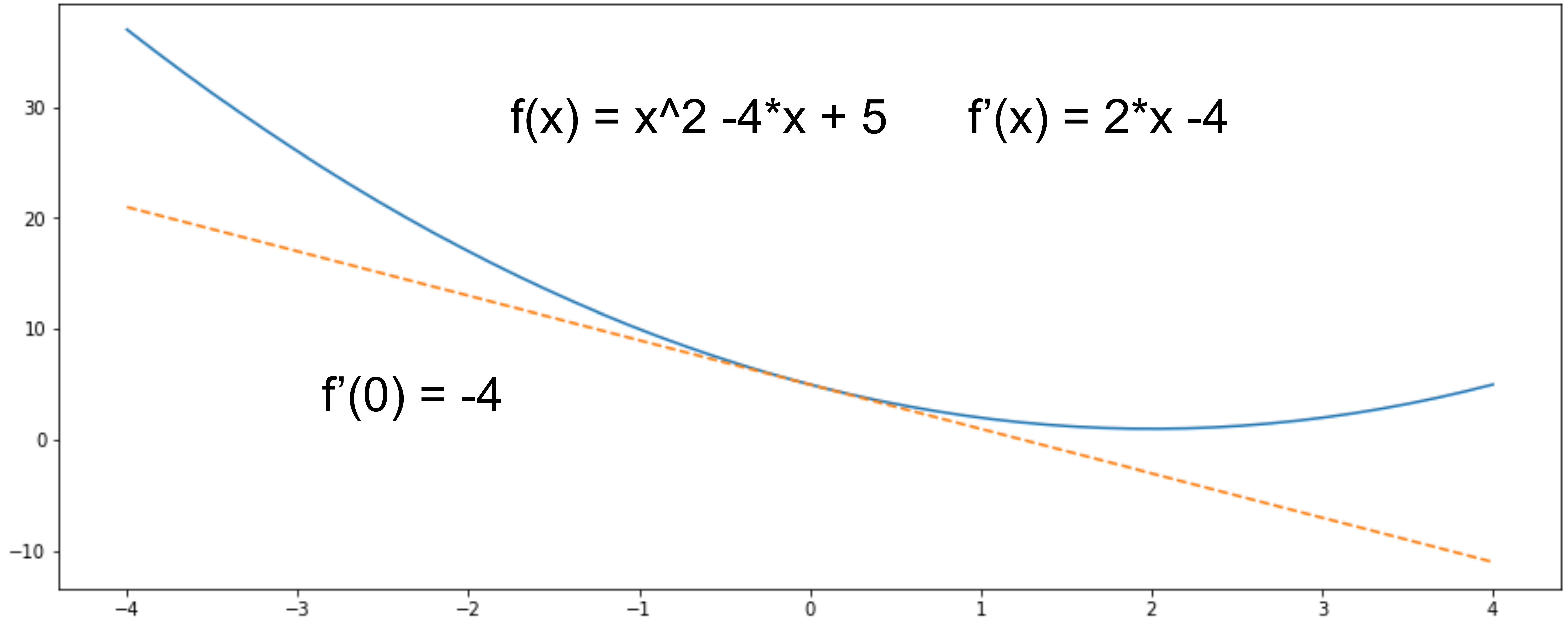
What is a tangent?

- The line that best approximate a line at a point



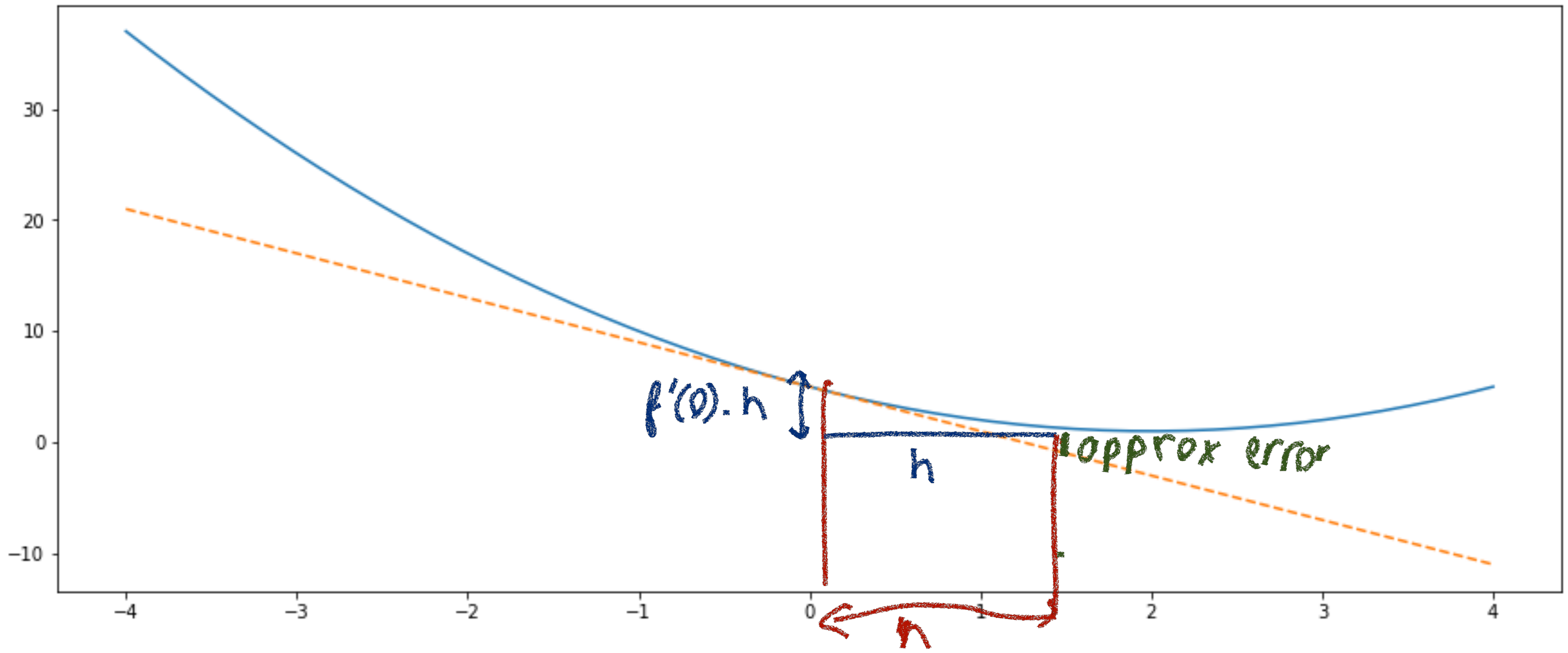
What is a derivative?

- The derivative is also the coefficient of the tangent to the graph of the function.



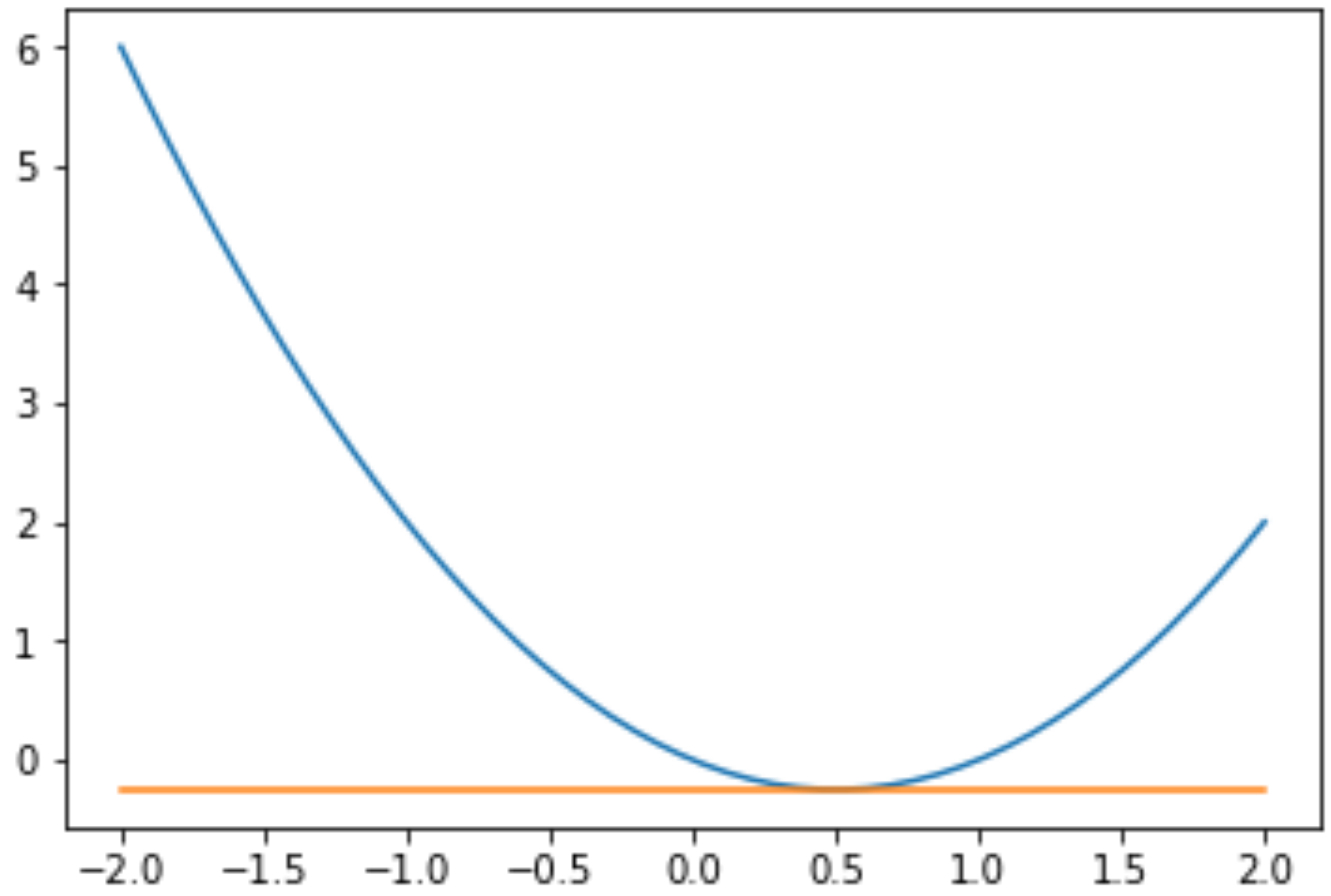
Tangent and derivative

$$f(x+h) \approx f(x) + h \cdot f'(x)$$



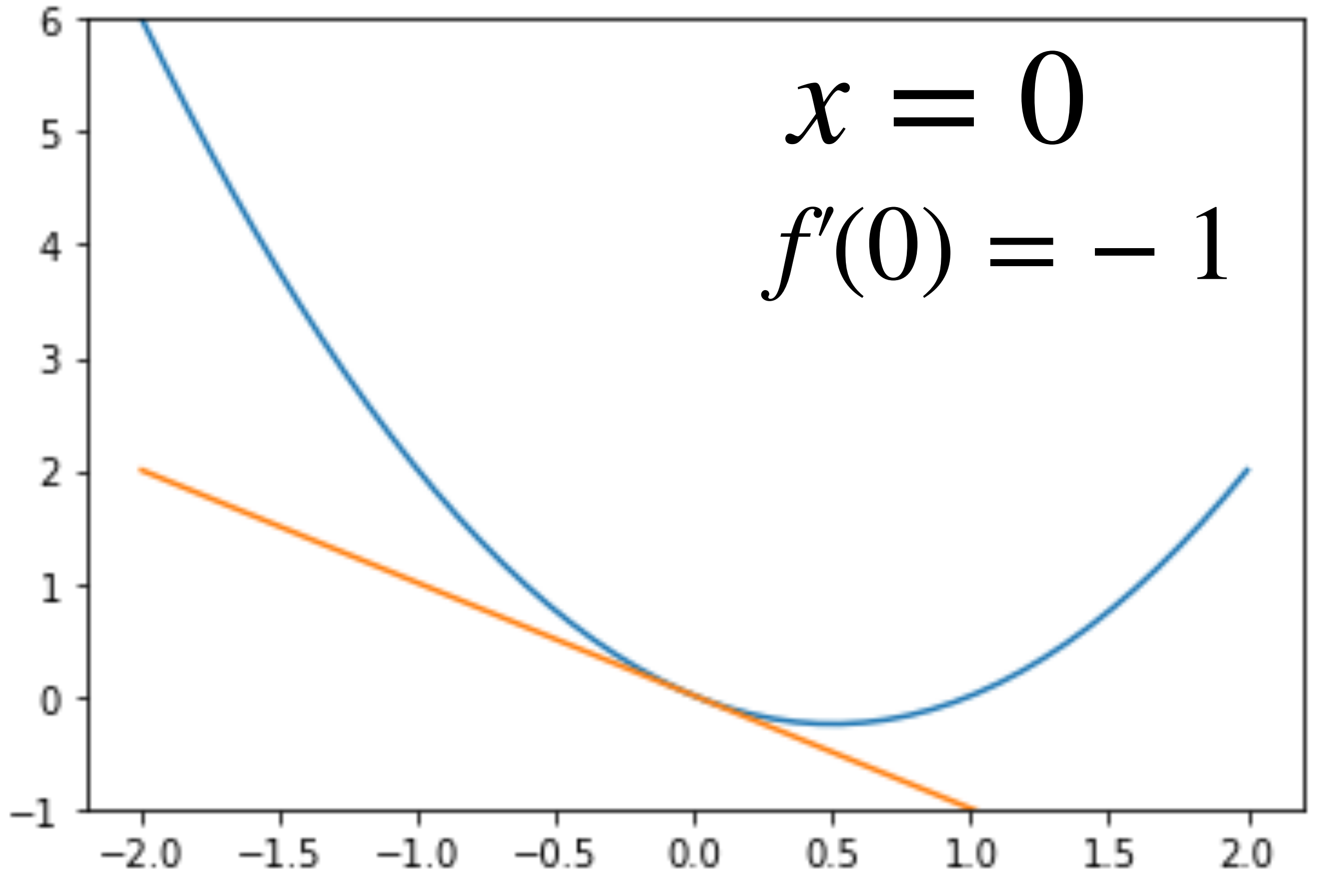
Derivative and minimum

- Intuitively, this shows you why the derivative should be zero at a minimum

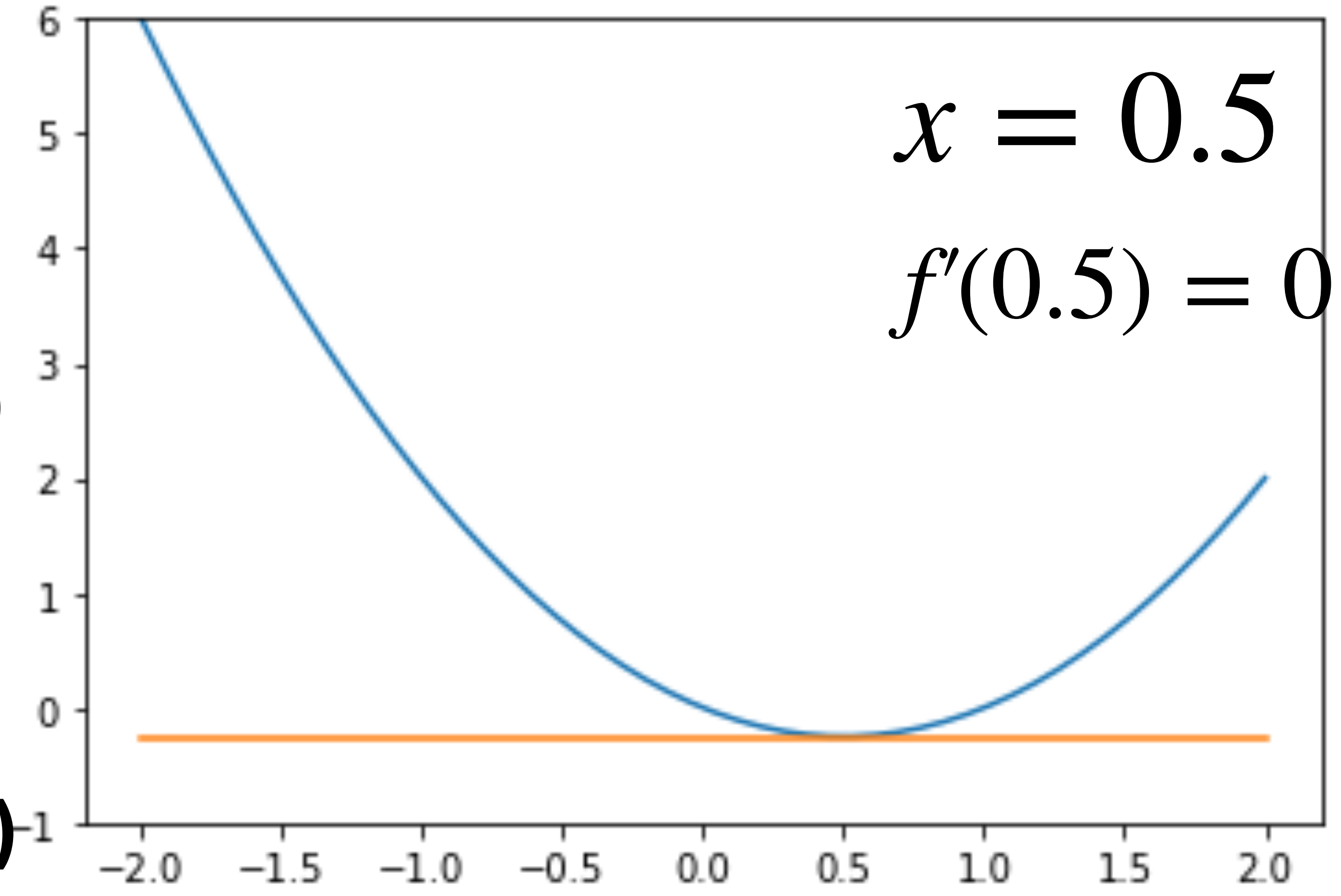


Derivative and minimum

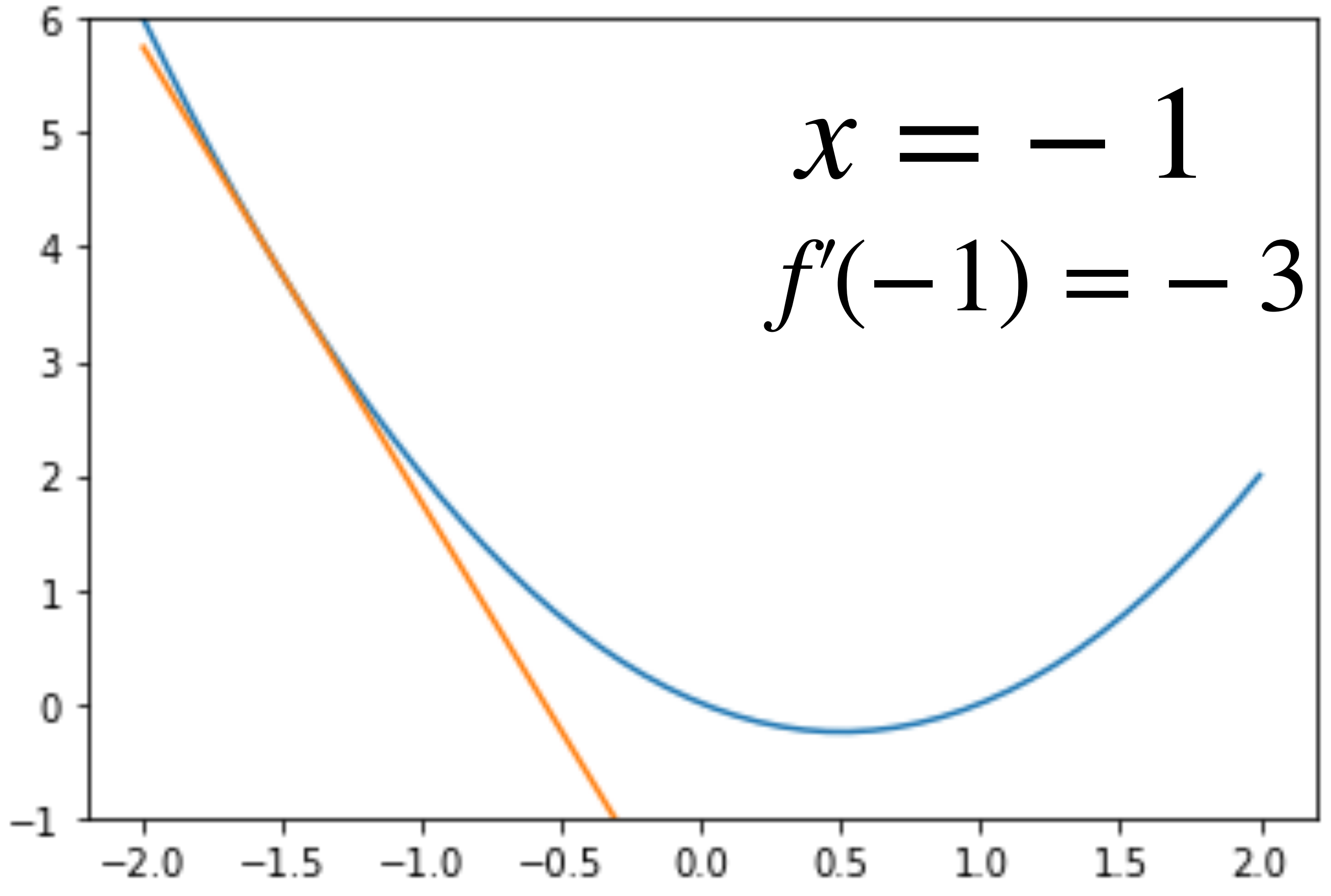
- The derivative tells us in which direction move to find the minimum



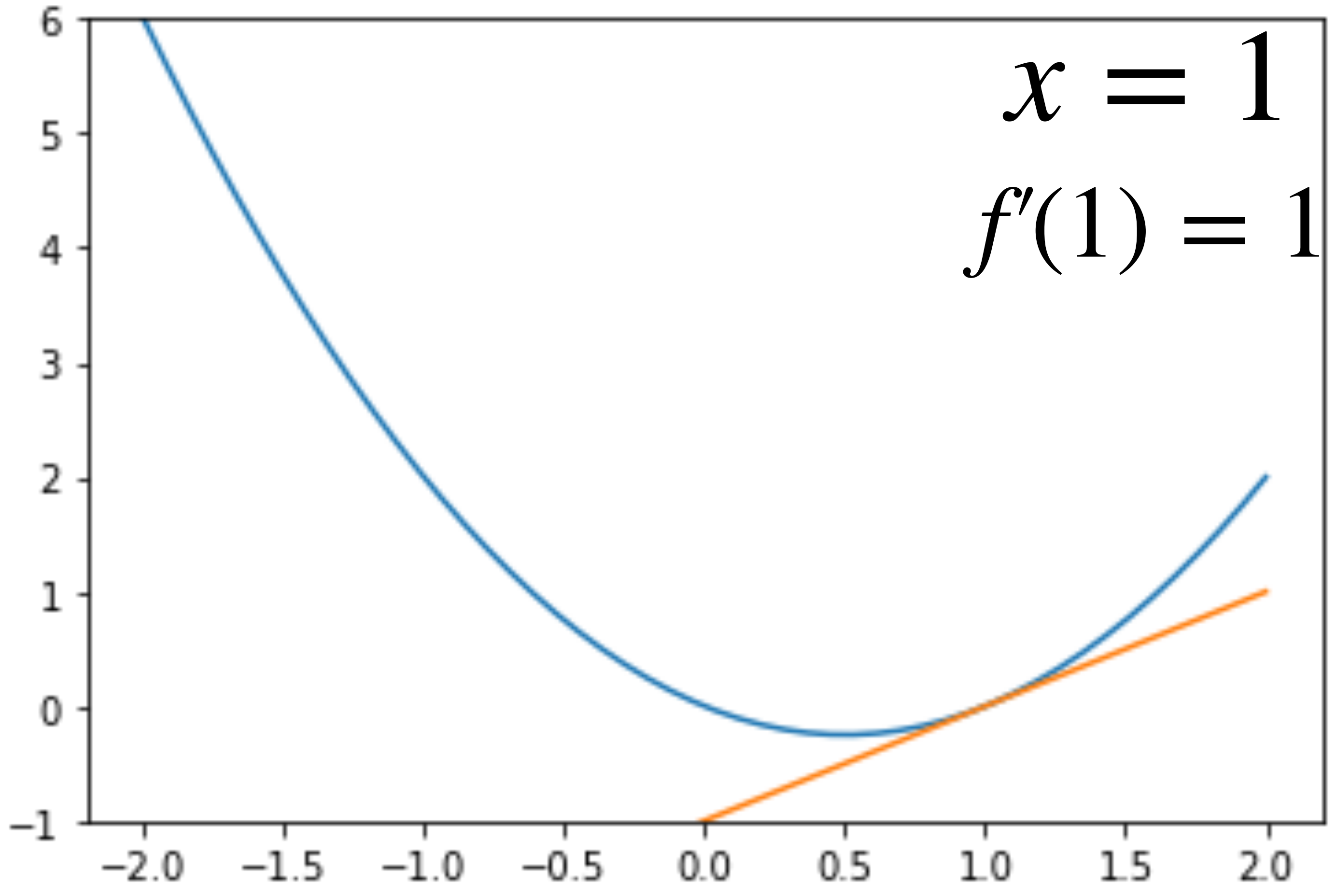
If derivative at x is negative, minimum is on the right
(we need to increase x to get closer to the minimum)



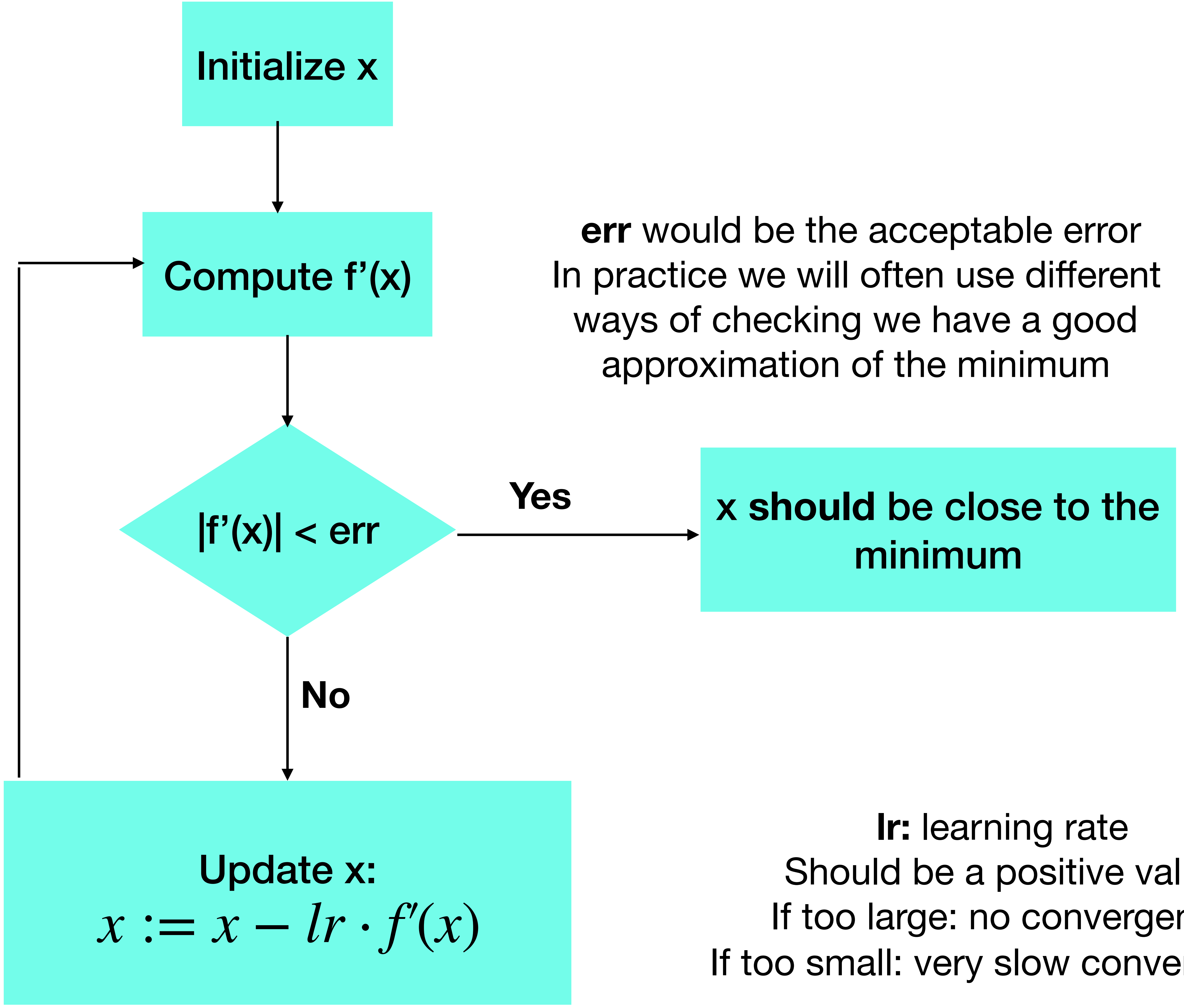
If derivative at x is positive, minimum is on the left
(we need to decrease x to get closer to the minimum)



If derivative at x is zero, x should be a minimum
(not necessarily in theory, but in our cases, it will be)



Gradient Descent algorithm



err would be the acceptable error
In practice we will often use different ways of checking we have a good approximation of the minimum

lr: learning rate
Should be a positive value
If too large: no convergence
If too small: very slow convergence

$$f(x) = x^2 - x$$

$$f'(x) = 2x - 1$$

$$\arg\min_x f(x) = 0.5$$

lr = 0.2

x = 0
f'(x) = -1

x=0.2
f'(x) = -0.6

x=0.32
f'(x) = -0.36

x = 0.392
…
…
…
…

x=0.493
f'(x) = -0.014

STOP?

Gradient Descent Algorithm

- Gradient descent works well **even** when we have functions of millions of variable
  - This is why it is so useful for Machine Learning and Neural Networks
  - Other methods will not be practical in such settings
- Convergence will depend on the choice of a good **learning rate**
  - In experiments, a good deal of time is often spent finding an optimal learning rate
  - **Too large** learning rate: **no** convergence (ie. the system learn nothing)
  - **Too small** learning rate: **slow** convergence (ie. the system takes a long time to learn)

Minimizing a function of several variables

http://lotus.kuee.kyoto-u.ac.jp/~fabien/lectures/IA/

(Lecture 3-2nd part)

Functions of several variables

- A function of several variables is just that: a function which has several variables

$$f : \mathbb{R}^3 \to \mathbb{R}$$

$$f(x, y, z) = (x - y)^2 + z^2 - z$$

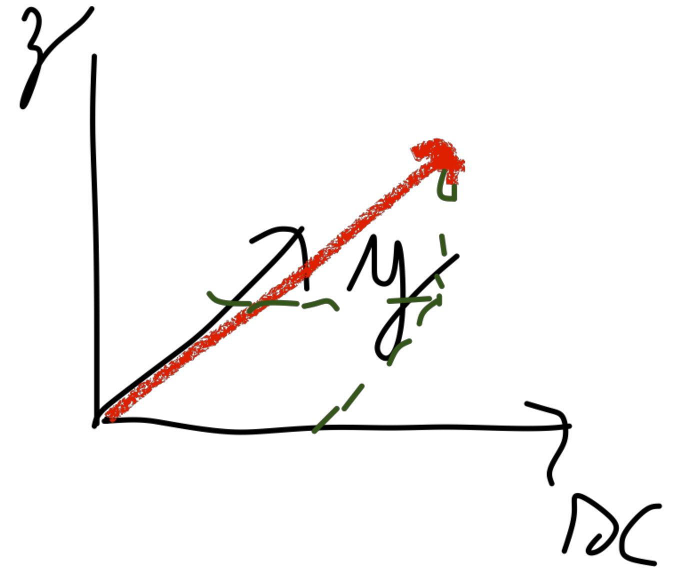
$$f(0,0,0) = 0$$
$$f(1,2,3) = 7$$
$$f(-1,2,2) = 11$$
$$f(0,1,1) = ?$$
$$f(2,2,0) = ?$$

- Like before, we want to find its minimum:

$$\arg\min_{x,y,z} f(x, y, z) = (0,0,0.5)$$

Vectors

- What are vectors?
- You probably have used vectors in Physics classes to represent force and speed
    - 3-dimensional vectors: [2.3, 4.5, -1]



- In Machine Learning, we also use them a lot
- Except that they can have more than 3 dimensions
    - 5-dimensional vector: [-1, 3, 4.1 ,5.2, 4]
    - We often note the set of all n-dimensional vectors $\mathbb{R}^n$

$$[1,2.1,4.1,-1,-1] \in \mathbb{R}^5$$

Vectors (Continued)

- For now, we only need to know the following about vectors:
  - A n-dimensional Vector is a list of n numbers
  - We can add 2 vectors (if they have the same dimension)

$$[2.1, 3.4, 1.1, 3.2] + [-1, 2.1, 3.1, -2] = [1.1, 5.5, 4.2, 1.2]$$

$$[2.1, 3.4] + [-1, 2.1, 3.1, -2] = \text{✘}$$

  - We can multiply a vector by a number

$$0.5 \times [2, 3, -1, -2] = [1, 1.5, -1.5, -1]$$

Vectors(Continued)

- We will usually denote a vector by a letter with an arrow on it: $\overrightarrow{x}$
- We denote the i^{th} component of $\overrightarrow{x}$ by x_i
- If $\overrightarrow{x} = [1, 2.2, -1, 4]$
  - Then we have $x_0=1$, $x_1=2.2$, $x_2=-1$, $x_3=4$

Vectors: Exercise

$$\overrightarrow{x} = [1,5,-2,0.5]$$

$$\overrightarrow{y} = [2,2,10,10]$$

$$\vec{z} = [3,-3,0]$$

- Dimensions of $\overrightarrow{x}, \overrightarrow{y}, \vec{z}$?
- Values of x_1, y_2, z_0, y_0?
- Compute:

$$\overrightarrow{x} + \overrightarrow{y}$$
$$\overrightarrow{x} + 0.5 \times \overrightarrow{y}$$
$$\overrightarrow{y} + \vec{z}$$

Vectors and Numpy

- In Python, Numpy arrays are a convenient way to represent vectors

$$\overrightarrow{x} = [1,5,-2,0.5]$$

- x = np.array([1, 5, -2, 0.5])
- x[0] == x_0 == 1
- x[1] == x_1 == 5
- Vector operations: x+0.5*y

```
In [981]: x = np.array([1, 5, -2, 0.5])
          y = np.array([2,2,10,10])
          z = np.array([3, -3, 0])
          print(x)
          print(y)
          print(z)

[ 1.   5.  -2.   0.5]
[ 2  2 10 10]
[ 3 -3  0]

In [982]: print(len(x), len(y), len(z))

4 4 3

In [983]: print(x[1], y[2], z[0], y[0])

5.0 10 3 2

In [984]: print(x+y)

[ 3.   7.   8.  10.5]

In [985]: print(x+0.5*y)

[2.  6.  3.  5.5]

In [986]: print(y+z)

---------------------------------------------------------------------------
ValueError
```

Vectors and Multivariate functions

- For now, we have represented the variables of a multivariate function with the letters x, y, z as in: $$f(x, y, z) = (x - y)^2 + z^2 - z$$
- In practice, we can have any number of variables. So it is more convenient to use:
  - x_0 (instead of x) , x_1 (instead of y), x_2 (instead of z), x_3 .. x_n (if we need more than 3 variables) $$f(x_0, x_1, x_2) = (x_0 - x_1)^2 + x_2^2 - x_2$$
- We can also the vectorial notation to represent all of the variables as one vector variable:

$$\overrightarrow{x} = [x_0, x_1, x_2] \qquad f(\overrightarrow{x}) = (x_0 - x_1)^2 + x_2^2 - x_2$$

- So, keep in mind that the 3 following expressions actually refer to the same function:

$$f(x, y, z) = (x - y)^2 + z^2 - z$$

$$f(x_0, x_1, x_2) = (x_0 - x_1)^2 + x_2^2 - x_2$$

- $$f(\overrightarrow{x}) = (x_0 - x_1)^2 + x_2^2 - x_2$$

Partial derivatives

- What is the equivalent of our “high school” derivatives when we have several variables?

- One part of the answer is ***partial derivatives***

- Partial derivatives are computed by choosing one variable and fixing the others

- Indeed, if we choose y, and fix x and z, we can see f(x, y, z) as a function of one variable and compute its derivative

$$f(x, y, z) = (x - y)^2 + z^2 - z$$

- $\dfrac{\partial f}{\partial x} = 2(x - y)$ $\qquad \dfrac{\partial f}{\partial y} = 2(y - x)$ $\qquad \dfrac{\partial f}{\partial z} = 2z - 1$

Computing the partial derivatives

$$f(x, y, z) = (x - y)^2 + z^2 - z$$

$$\frac{\partial f}{\partial x} =$$

$$\frac{\partial f}{\partial y} =$$

$$\frac{\partial f}{\partial z} =$$

Partial derivatives

- Exercise: Compute the partial derivatives

$$f(x, y, z) = xyz - z^2 - y^2$$

$$f(x, y, z) = e^{x+y} - sin(x + z)$$

Gradient

- The partial derivatives become the component of a vector we call the ***gradient***

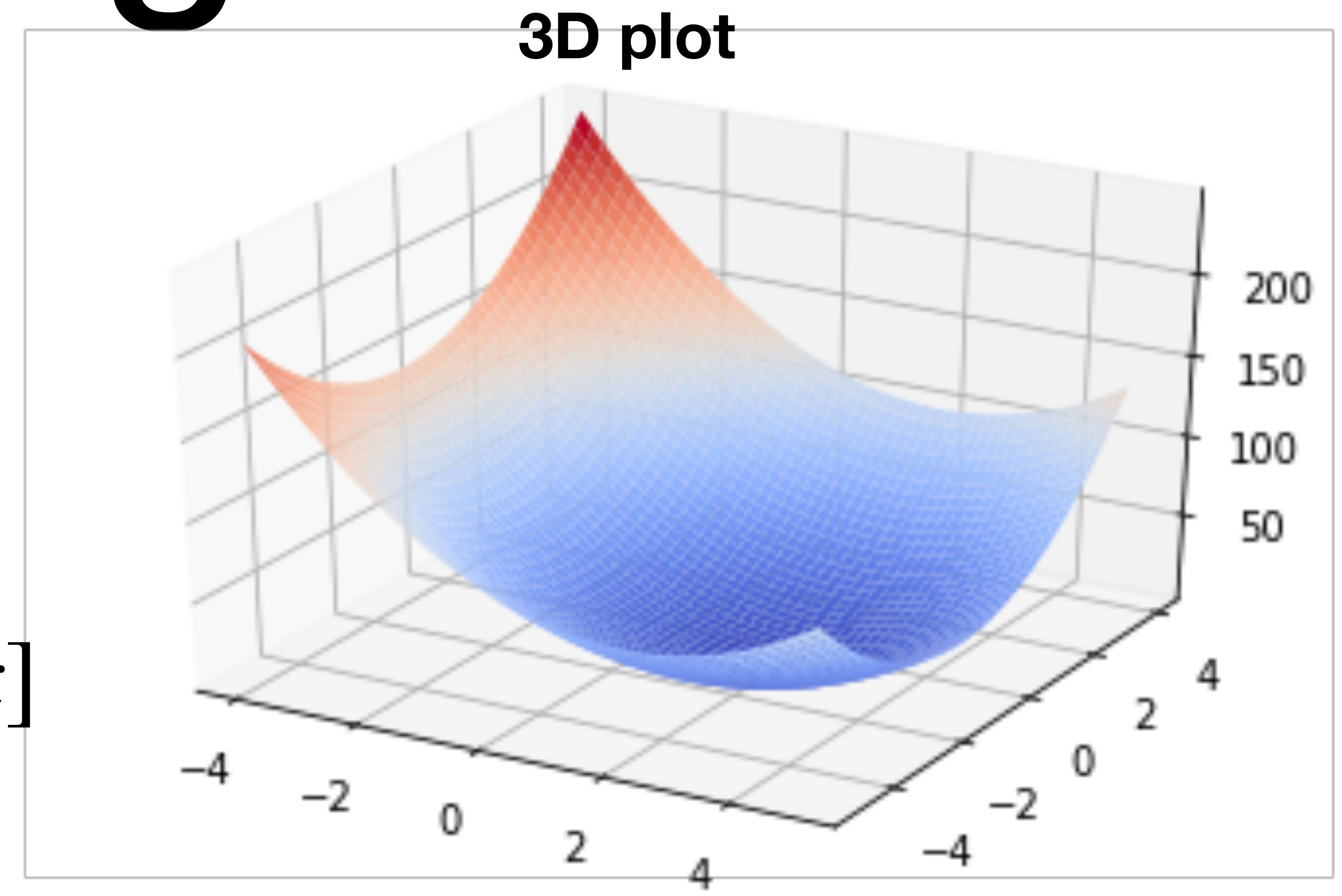
$$grad \cdot f(x, y, z) = [\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}]$$

- For example: $f(x, y, z) = (x - y)^2 + z^2 - z$
- $$grad \cdot f(x, y, z) = [2(x - y), 2(y - x), 2z - 1]$$

Interpreting the gradient

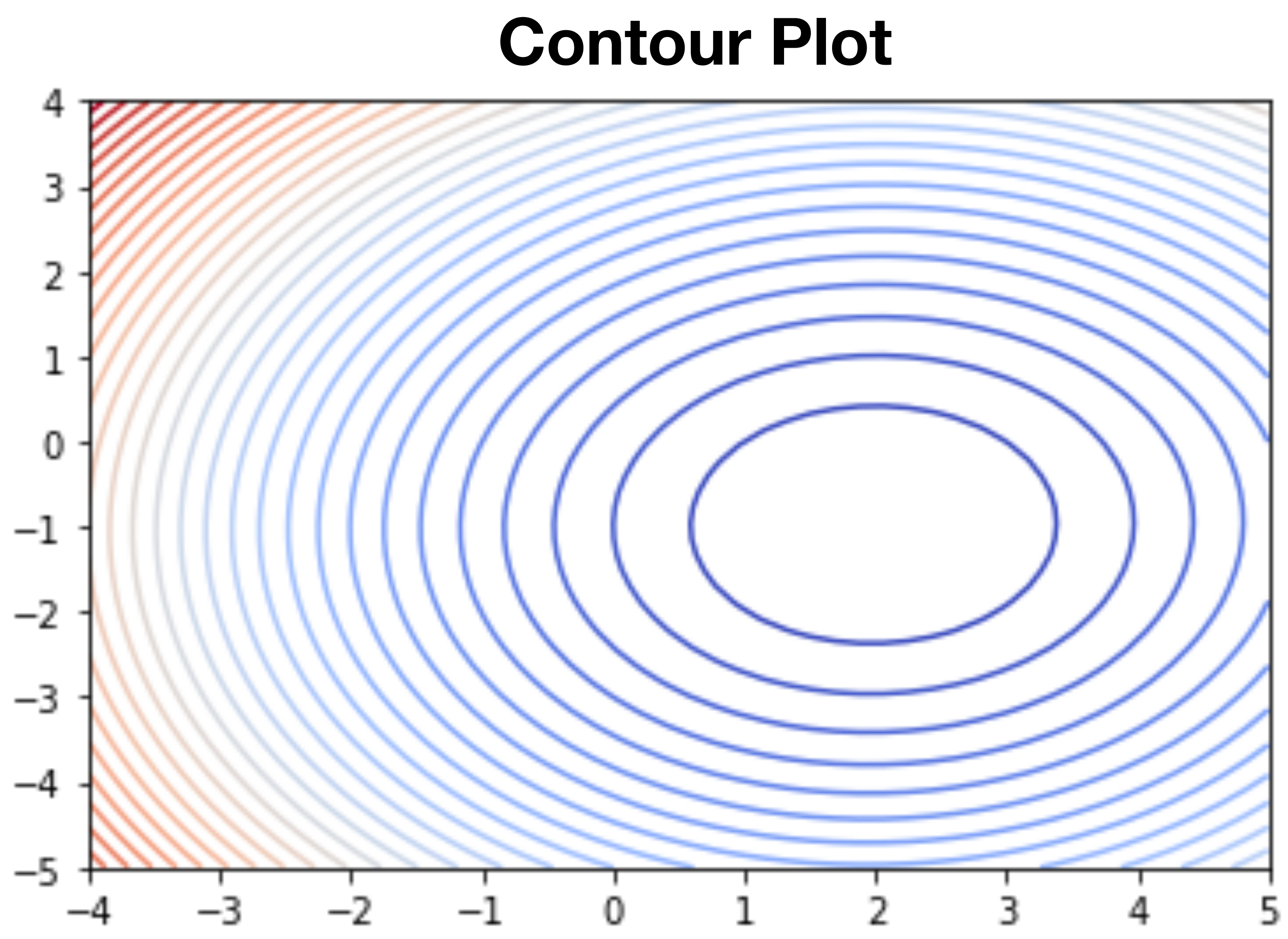
$$f(x, y) = 4(x-2)^2 + 4(y+1)^2 - 0.1xy$$

$$grad \cdot f(x, y) = [8(x-2) - 0.1y, 8(y+1) - 0.1x]$$



$$grad \cdot f(0,0) = [-16, 16]$$

$$grad \cdot f(2, -1) = [0.1, -0.2]$$



Gradient

$$grad \cdot f(x, y, z) = [\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}]$$

- In this case, the function has 3 variables. Therefore the gradient is a vector of size 3
- If the gradient has n variables, it is a vector of size n
- More precisely, the gradient of f is itself a function that return a vector

$$f : \mathbb{R}^n \to \mathbb{R}$$

$$f(x_1, x_2, \ldots, x_n)$$

$$grad \cdot f : \mathbb{R}^n \to \mathbb{R}^n$$

$$grad \cdot f(x_1, x_2, \ldots, x_n) = [g_1, \ldots, g_n]$$

What is the equivalent of second derivative for multivariate functions?

- It is the Hessian Matrix:

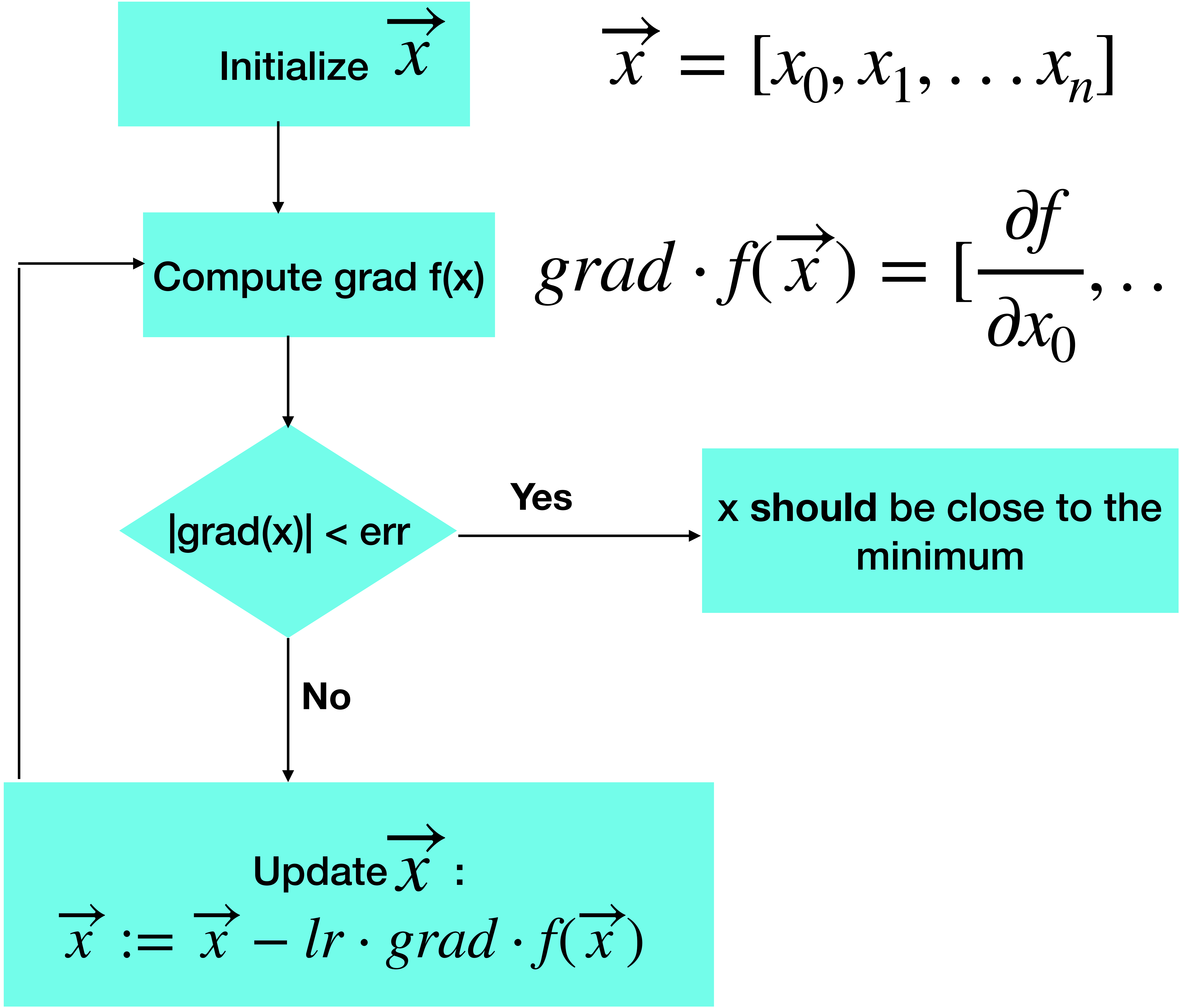
$$\begin{vmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial x \partial z} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} & \frac{\partial^2 f}{\partial y \partial z} \\ \frac{\partial^2 f}{\partial x \partial z} & \frac{\partial^2 f}{\partial y \partial z} & \frac{\partial^2 f}{\partial z^2} \end{vmatrix}$$

- But <u>thankfully, we will not need to use</u> it
- But <u>for your information</u>, this would be the equivalent of the “High School” minimization when we have several variables:
-

To minimize f(x, y, z):

1. **Compute gradient of f(x, y , z)**
2. **Compute hessian of f(x)**
3. **Find x, y, z such that grad f(x,y,z) = 0**
4. **If hessian of f(x,y,z) is definite positive then (x,y,z) is a local minimum of f**

Gradient Descent algorithm



$$\vec{x} = [x_0, x_1, \ldots x_n]$$

$$grad \cdot f(\vec{x}) = [\frac{\partial f}{\partial x_0}, \ldots \frac{\partial f}{\partial x_n}]$$

$$f(\vec{x}) = (x_0 - x_1)^2 + x_2^2 - x_2$$

$$grad \cdot f(\vec{x}) = [2(x_0 - x_1), 2(x_1 - x_0), 2x_2 - 1]$$

$$lr = 0.2$$

$$\vec{x} = (0,1,0)$$

$$grad \cdot f(\vec{x}) = [-2,2,-1]$$

$$\vec{x} = (0.4,0.6,0.2)$$

$$grad \cdot f(\vec{x}) = [-0.4,0.4,-0.6]$$

$$\vec{x} = (0.41,0.43,0.51)$$

$$grad \cdot f(\vec{x}) = [-0.04,0.04,0.01]$$

Gradient Descent Algorithm

- You can see that, in the case of the gardient descent, the algorithm is the same for univariate functions and multivariate functions
- It is a simple algorithm, but it scales very well
- There exists many variations of it:
    - Gradient Descent with momentum
    - Stochastic Gradient Descent
    - Adagrad, Adadelta, Adam, …

Gradient Descent with Momentum

- Compute a “gradient with momentum” at each iteration:
-

$$gm_t = 0.6grad \cdot f(\overrightarrow{x}) + 0.4gm_{t-1}$$

Update $\overrightarrow{x}$:

$$\overrightarrow{x} := \overrightarrow{x} - lr \cdot gm_t$$

Stochastic Gradient Descent

- What happens if the gradient is noisy?
- That is, we can only compute a value that is equal to the true gradient “on average”?
    - A bit like if you are drunk and trying to get home

Stochastic Gradient Descent

- What happens if the gradient is noisy?
- That is, we can only compute a value that is equal to the true gradient "on average"?
  - A bit like if you are drunk and trying to get home
- It turns out it works.
  - But you have to decrease your learning rate over time to stabilize $lr = \frac{lr_0}{\sqrt{(t+1)}}$
  - Convergence will be slower
- Very interesting because a noisy gradient can be million times faster to compute than a "true" gradient

Optimization libraries

- You can also minimize a function by using a specialized library
- It gives you access to more sophisticated minimization algorithms
- However these more sophisticated algorithms do not scale as well as Gradient Descent
  - Which is one Gradient Descent and its variants are still the main tool for large scale Machine Learning (In particular, Deep Learning)

```
In [103]: scipy.optimize.minimize(f_numpy, np.array([0,0]), jac=grad_f_numpy, method="L-BFGS-B")

Out[103]:       fun: 0.1969057665260197
           hess_inv: <2x2 LbfgsInvHessProduct with dtype=float64>
                jac: array([-3.88578059e-16,  2.77555756e-17])
            message: b'CONVERGENCE: NORM_OF_PROJECTED_GRADIENT_<=_PGTOL'
               nfev: 5
                nit: 4
             status: 0
            success: True
                  x: array([ 1.9878106 , -0.97515237])
```