Gradient Descent

[optimisation]

- -> Multivariate calculus
- -> GD Theory
- -> GD Code

Recap

↳ We are trying to optimize

$$\max_{\vec{w}, w_0} = \sum_{i=1}^{n} y_i \frac{w^T x_i + w_0}{\|w\|}$$ → Grain Function

→ maximise distance of all points from decision boundary in the same direction as the label

in literature, we use:

$$\min_{\vec{w}, w_0} - \sum_{i=1}^{n} y_i \frac{w^T x_i + w_0}{\|w\|}$$ → Loss Function

Most data science problems are formulated as minimisation problems.

$$\therefore \min_{\vec{w}, w_0} f(w, w_0)$$

$$= \min_{w_0, w_1 \cdots w_n} f(w_1, w_2 \cdots w_n)$$

Need to perform multivariate optimisation

Multivariate Differentiation

$y = f_1(x)$ → single var

$y = f_2(x_1, x_2 \cdots x_n)$ → multivariate

$$\frac{d f(x)}{dx} = f'(x) \quad \text{[using rules etc]}$$

$$\frac{d f(x_1, x_2 \cdots x_n)}{dx_1} = ??$$

Consider,

$z = f(x, y)$ | find $\frac{dz}{dx}$

so z depends on x & y. but, does x and y depend on each other?

Can be:

Imagine 2 features

weight, height

income, age

diameter, weight, ect.

Hence $y = g(x)$

$\therefore z = f(x, g(x))$

↳ In order to fully differentiate z w.r.t 'x' I need to know the relatn of all other

variables with 'x'.

Example

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

$$\therefore z = x^2 + [g(x)]^2 \quad \because \text{Assume } y = g(x)$$

$$\therefore \frac{dz}{dx} = 2x + 2g(x) \cdot g'(x) \quad \because \text{chain rule}$$

$$\therefore z' = 2x + 2y \cdot y'$$

In order to proceed i need to know y' or $\frac{dy}{dx}$ which can be hard!!

Sometimes all inputs can be a funcⁿ of a single external var also.

$$\rightarrow z = f(x, y)$$
$$x = g(t), \quad y = h(t)$$

So we noticed that full differenc^n can be hard.

But even if we did it, to find the max / min we need to simultaneously solve

$$\frac{dz}{dx} = 0 \quad , \quad \frac{dz}{dy} = 0 \quad \text{... and so on}$$

Partial derivatives

↳ A derivative wrt one variable keeping others constant.

$$\frac{\partial z}{\partial x} = \frac{df(x,\ y=\text{const})}{dx}$$

$$\frac{\partial z}{\partial y} = \frac{df(x=\text{const},\ y)}{dy}$$

Partial vs Full

→ Partial : Take other var as const

→ Full : Take other var as func of x;

Example:

$$f(x,y) = 2x^2y + 3y^3x^2 + 3y$$

$$\frac{\partial f}{\partial x} = 4xy + 6y^3x + 0$$

$$\frac{\partial f}{\partial y} = 2x^2 + 9y^2x^2 + 3$$

Now, we need to design a way to use partial derivatives to find optimas

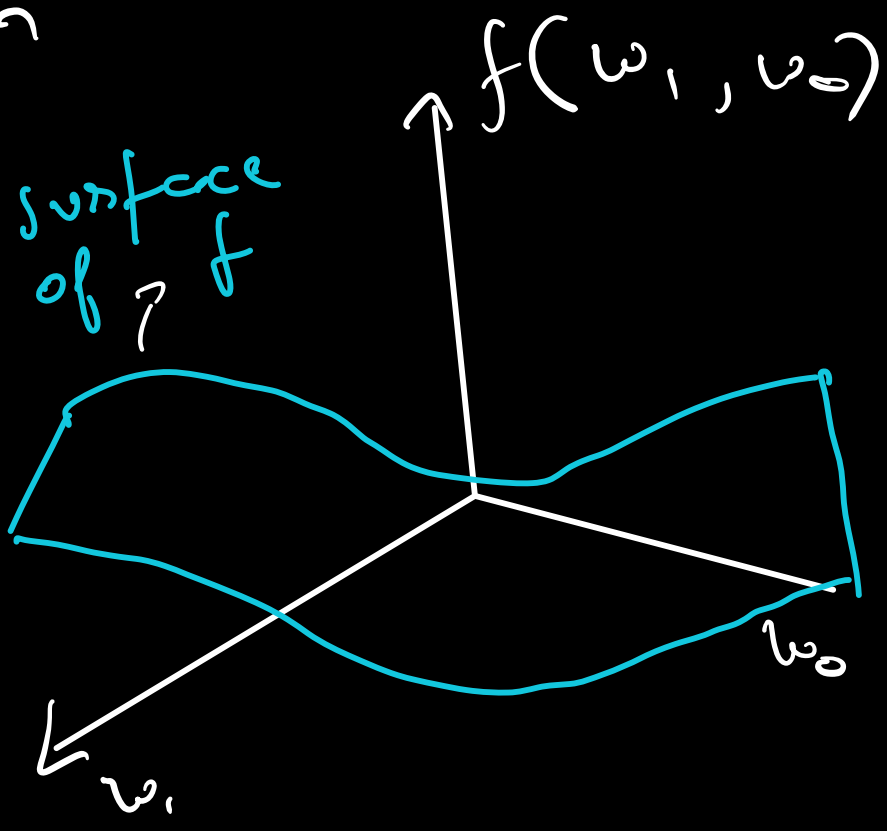
Gradient [Extra]

Goal:

$$\min_{w_0, w_1 \ldots w_n} f(w_0, w_1 \ldots w_n)$$

What is the slope in each direction?

Q: If there are 3 features, how many dimensions will this visualisation be?



a) 2 b) 3 c) 4 d) 5

So we need to find the gradient of $f(w_0 .. w_n)$ axis, w.r.t $w_0, w_1 \ldots w_n$ axis

This is called Gradient vector.

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial w_0} \\ \vdots \\ \frac{\partial f}{\partial w_n} \end{bmatrix}$$

Gradient Descent

Goal: Develop some algo to find optima of multivariate function w/o full diffn

Let's do it for 1 variable 1st

$y = x^2 - 30$ Q: min value?

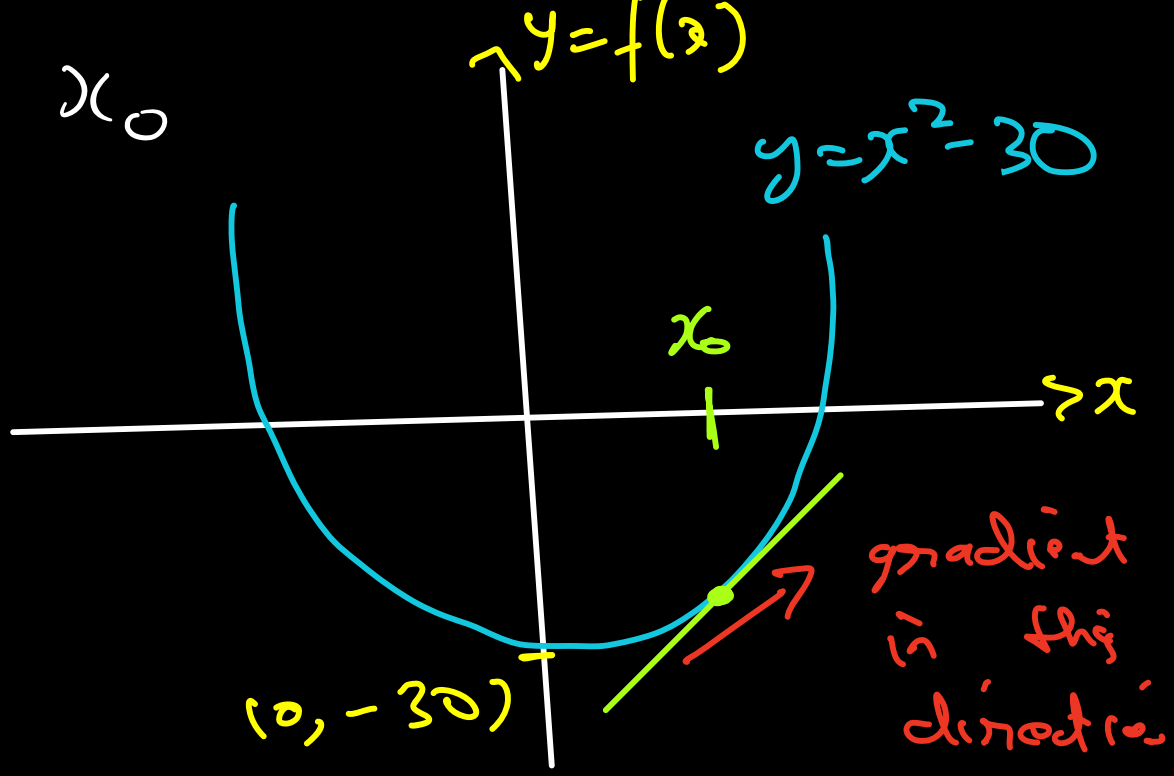
$\frac{\partial y}{\partial x} = f'(x) = 2x$

Gradient descent algorithm

Step -1: Choose random x_0

$$f(x_0) = x_0^2 - 30$$

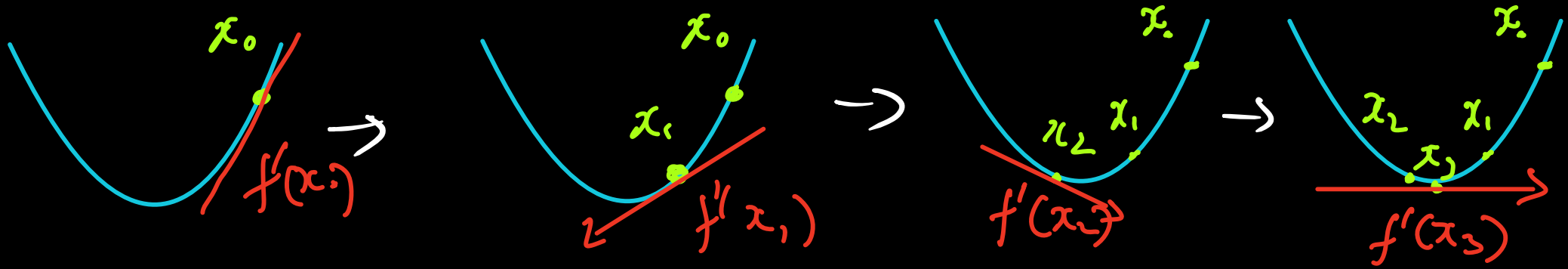
$$f'(x_0) = 2x_0$$



'Descent' refers to minimisation

Step -2: Move in the direction of minimisation

$$x_1 = x_0 - \eta \cdot f'(x_0)$$



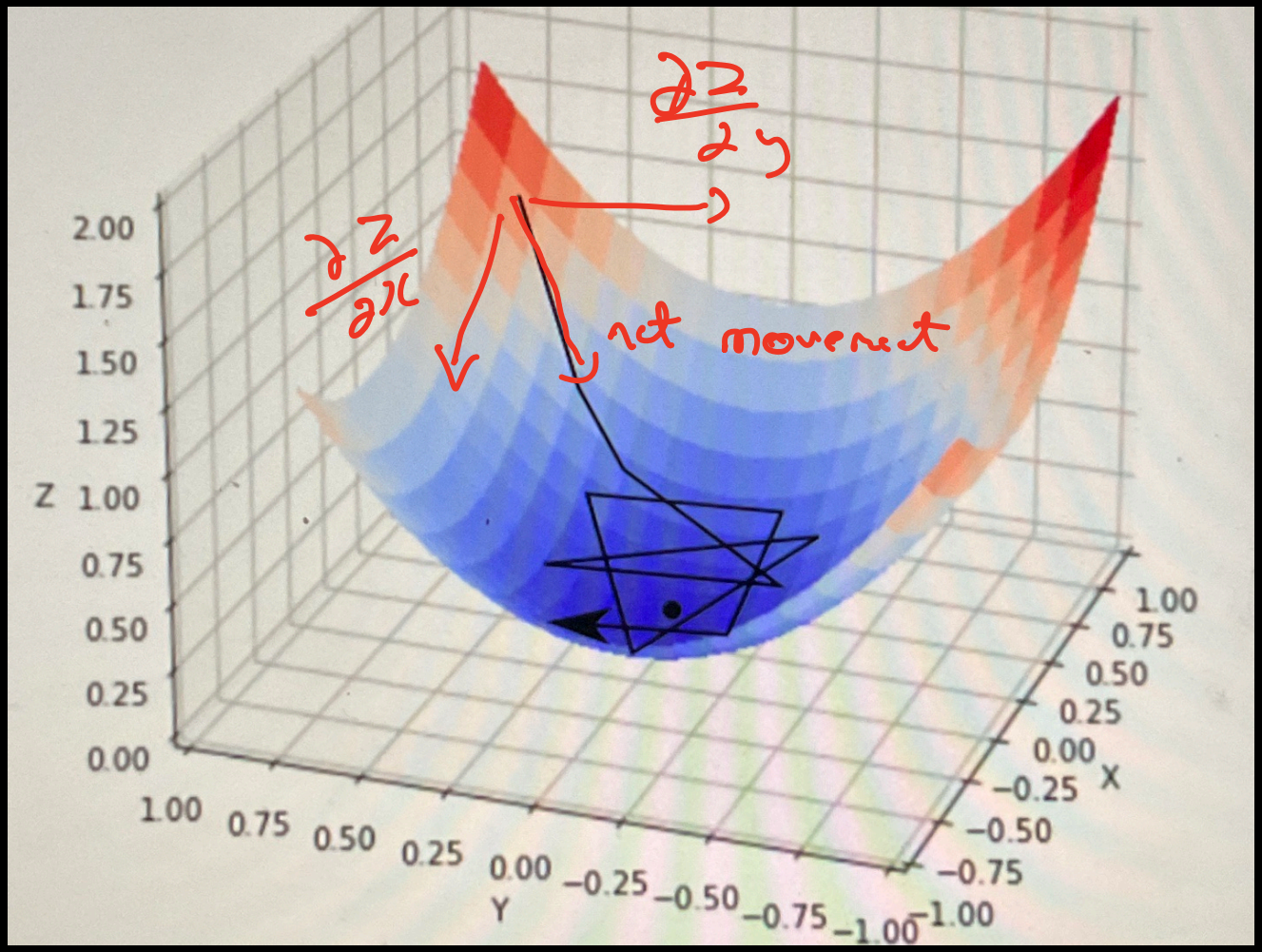
$$\Delta x = -\eta f'(x_0)$$

Algorithm converged. Found minima

⇒ Imagine throwing a ball in a bowl, it will eventually settle down at the bottom
  → Rolls faster when steeper
  → slows down when flatter.

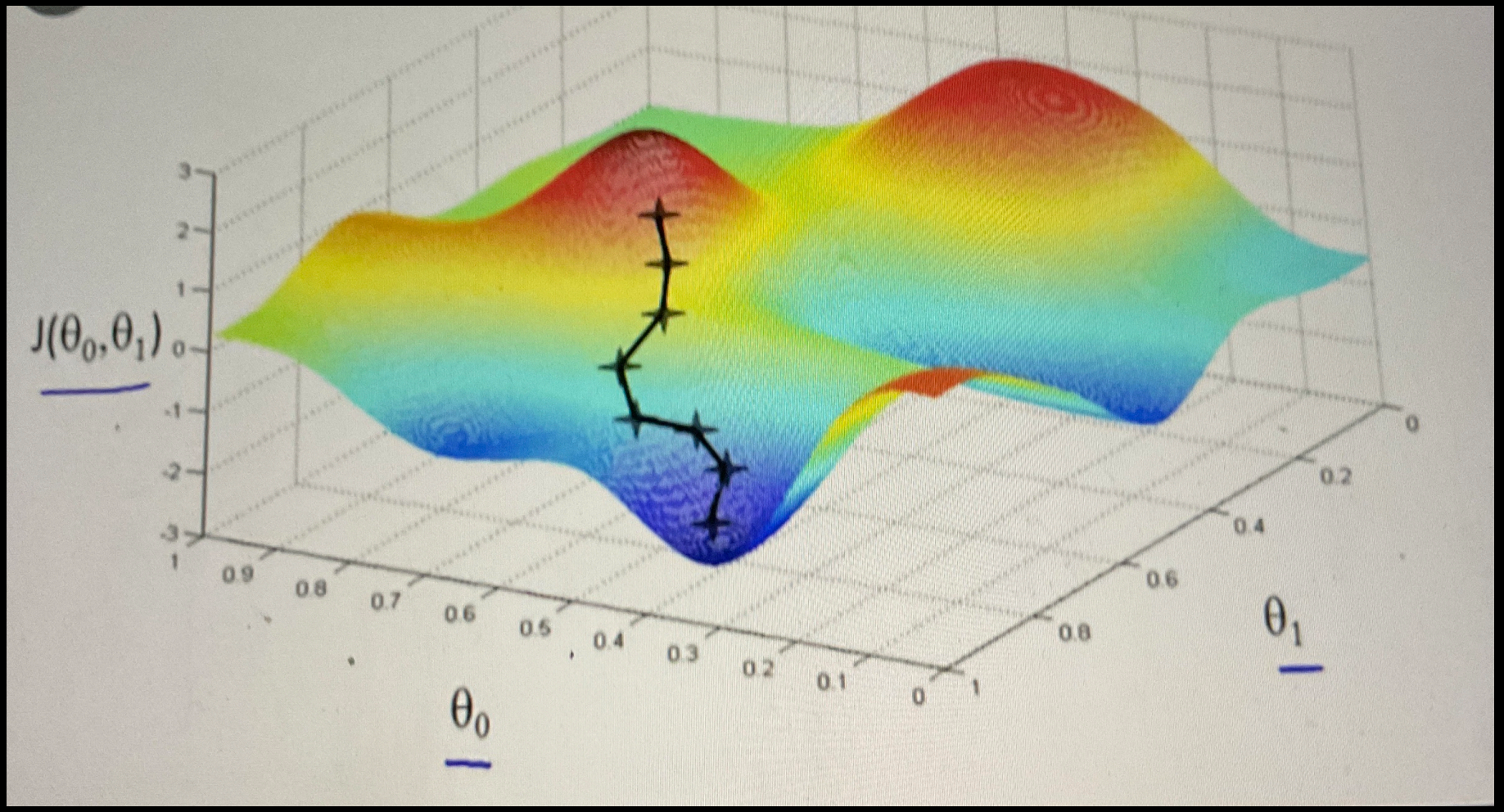
→ code

→ In 3-d this would look as follows:



We use partial derivatives, which give us how much to move along each axis

This is one of the reasons, that in machine learning it is recommended to have independent features. However, G.D still works because we move very little steps so partial derivatives hold.



Sometimes we run G.D with multiple initialisat^n because many results might be local minima.

For multiple dimensions:

$$w_1[\text{next}] = w_1[\text{current}] - \eta \frac{\partial f}{\partial w_1}$$

$$\vdots$$

$$w_n[\text{next}] = w_n[\text{current}] - \eta \frac{\partial f}{\partial w_n}$$

η = is called the learning rate