Partial Derivatives

- We can also have functions that have more than one variable.

  Example: $f(x, y) = x^2 + y^2$

  A **partial derivative** of a function of several variables is its derivative with respect to one of those variables, with the others held constant.

  For example, the partial derivatives of f(x,y) with respect to x and y are given as:

$$\frac{\partial f}{\partial x} = 2x + 0 = 2x \qquad \text{and} \qquad \frac{\partial f}{\partial y} = 0 + 2y = 2y$$

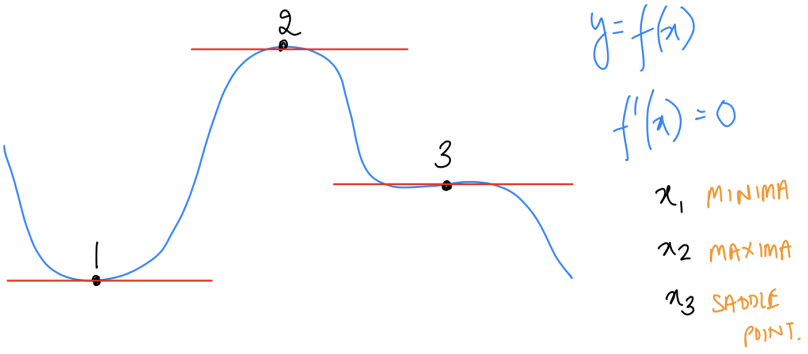
- ∇ is called a **delta operator**. It is a 2D vector that consists of derivatives w.r.t single variables also called partial derivatives.

  Let us assume a function **$f(w_0, w_1, w_2, \ldots, w_n)$** that has more than one input variable. Then,

$$\nabla f(w0, w1, w2, ..., wn) = \begin{bmatrix} \frac{\partial f}{\partial w_0} \\ \frac{\partial f}{\partial w_1} \\ \frac{\partial f}{\partial w_2} \\ . \\ . \\ \frac{\partial f}{\partial w_n} \end{bmatrix}$$

  Optima for f can be found by putting ∇f equal to a **Null matrix** of the same dimensions as ∇f.

  We can have points where ∇f = 0, but those points may not be global maxima or minima. Those are called **Saddle points.**



- **Gradient descent** is an iterative algorithm to reach the optima of a function.

Let's say we have a function **z = f(x,y)** and are trying to find the minimum of this function

We will start by initializing x_0 and y_0 randomly. Then we will keep updating x and y till the point where the partial derivatives are very close to 0 or some fixed number of iterations

Algorithm:

Step 1: Initially, pick x_0 and y_0 randomly.

Step 2: Compute $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$ at x = x_0 and y = y_0 respectively.

Step 3: The new values of x_0 and y_0 which are closer to the optima are given as:

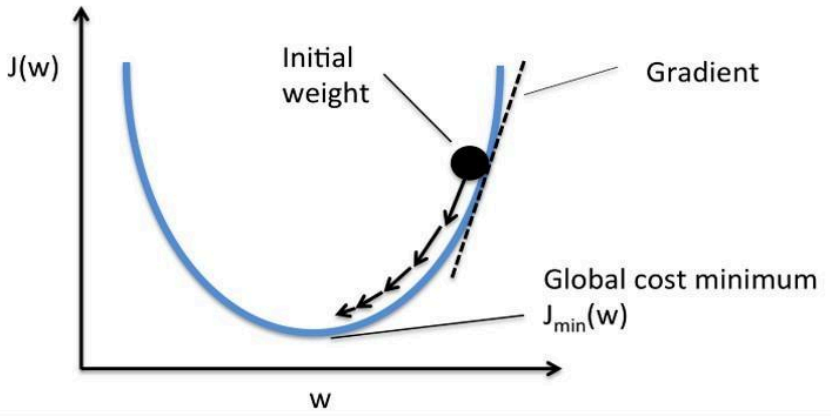
$$x_1 = x_0 - \eta . \frac{\partial f}{\partial x} \quad \text{and} \quad y_1 = y_0 - \eta . \frac{\partial f}{\partial y}$$

Step 4: Repeat step 3 either till some k(constant) iterations or till a point where $\delta f / \delta x \approx 0$ and $\delta f / \delta y \approx 0$.

Here, η (eta) is the **learning rate** and decides the step size of our iterations. If we set its value to very small, then the updates will happen **very slowly**. If we set it to a large value, it may **overshoot** the minima.

Therefore, the **update rule** of the Gradient descent algorithm is:

$$x_{i+1} = x_i - \eta \frac{\partial f}{\partial x} \Big|_{x=i}$$



If we want to **maximize** some function. Then we can convert the maximum function into a minimum function by adding a **negative** sign i.e $max\ f(x, y) = min - f(x, y)$