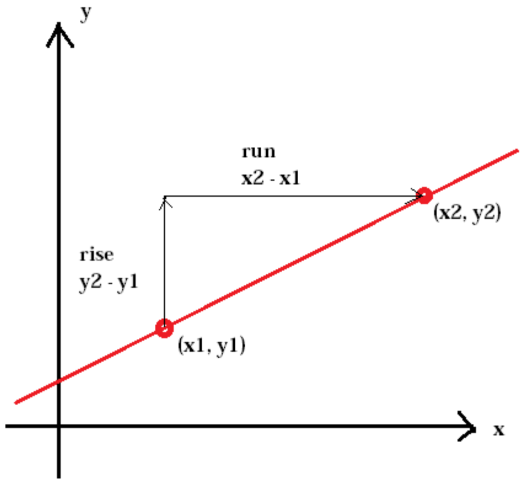
Gradients, Gradient Descent and Convexity

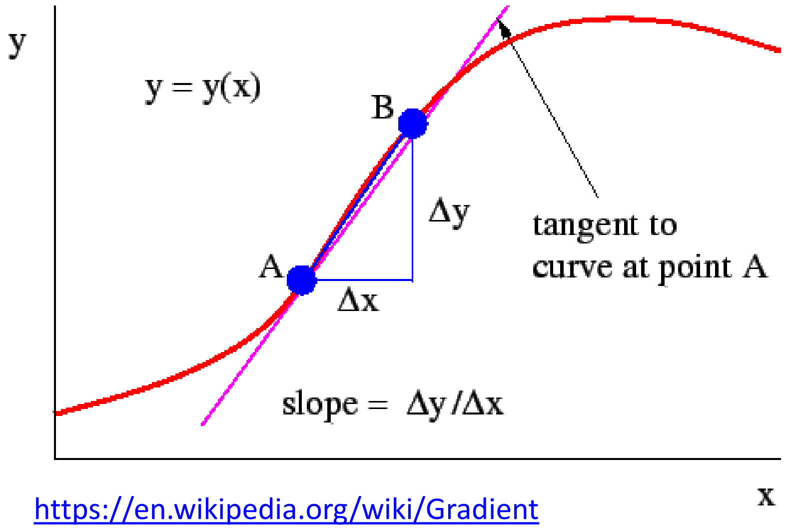
Dr. Fayyaz Minhas

Department of Computer Science
University of Warwick
https://warwick.ac.uk/fac/sci/dcs/teaching/material/cs909/

Preliminaries

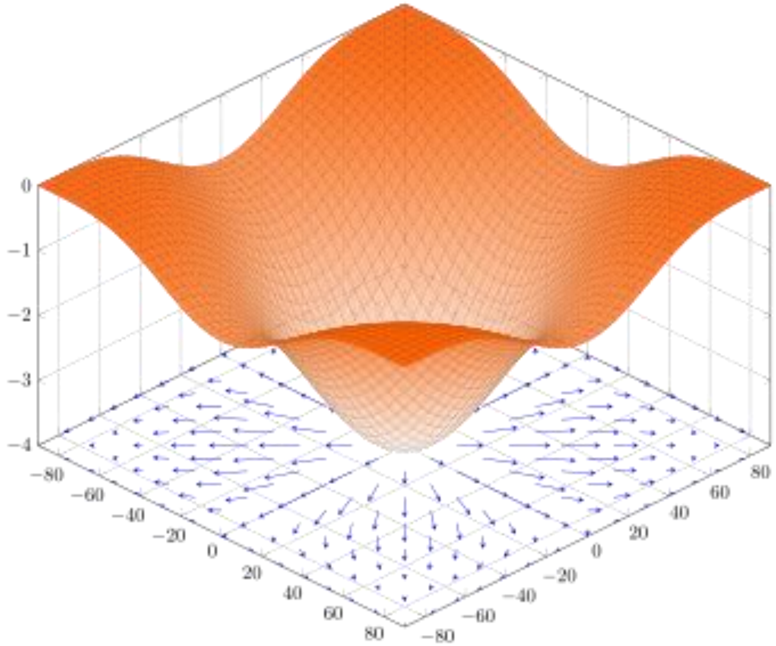
- Gradients





https://en.wikipedia.org/wiki/Gradient

$$\nabla f(\boldsymbol{x}) = \begin{bmatrix} \frac{\partial f(\boldsymbol{x})}{\partial x^{(1)}} \\ \frac{\partial f(\boldsymbol{x})}{\partial x^{(2)}} \\ \vdots \\ \frac{\partial f(\boldsymbol{x})}{\partial x^{(d)}} \end{bmatrix}$$



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

Finding minima and maxima of functions

- Given a function f(x)
- Take the derivative
- Substitute the derivative to zero
- Solve for x when $\frac{df}{dx} = 0$

$$f(w) = (w - 0.5)^2$$
$$\frac{df}{dw} = 2(w - 0.5) = 0$$
$$w^* = 0.5$$

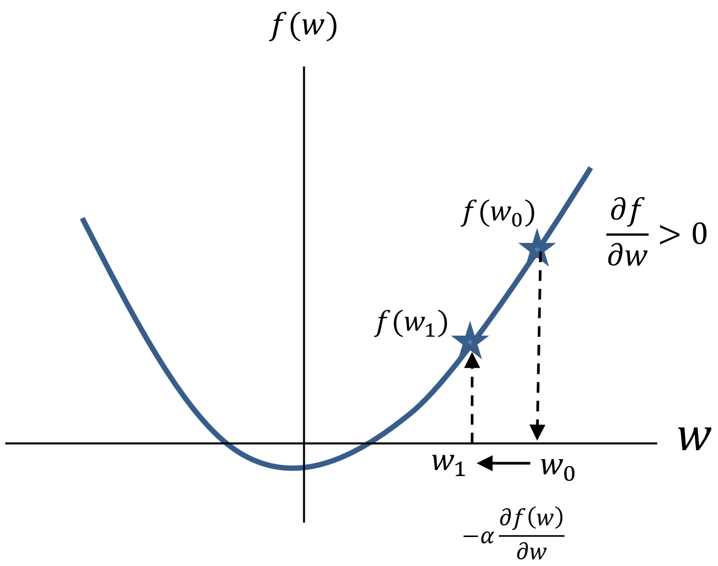
- Works when we can solve for x

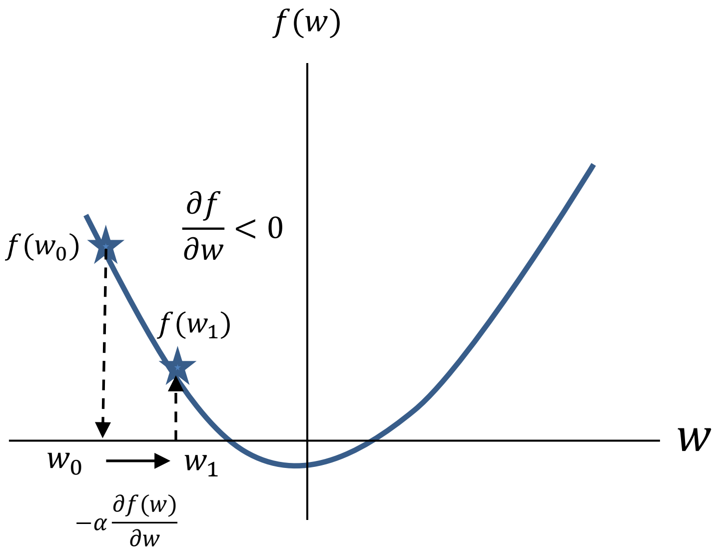
$$f(w) = (w - 0.5)^2 + sin(4w)$$
$$\frac{df}{dw} = 2(w - 0.5) + 4\cos(4w) = 0$$
$$w^* =?$$

Preliminaries: Gradient Descent

- In order to find the minima of a function, keep taking steps along a direction opposite to the gradient of the function

$$\boldsymbol{w}^{(\boldsymbol{k}+\boldsymbol{1})} = \boldsymbol{w}^{(\boldsymbol{k})} - \alpha \nabla f(\boldsymbol{w}^{(\boldsymbol{k})})$$





GD Implementation

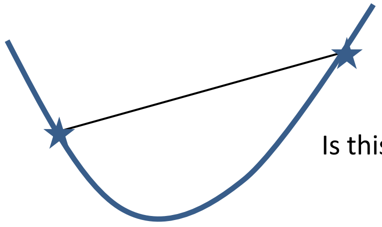
```python
import numpy as np

def gd(fxn,dfxn,w0=0.0,lr = 0.01,eps=1e-4,nmax=1000, history = True):
    """
    Implementation of a gradient descent solver.
        fxn: function returns value of the target function for a given w
        dfxn: gradient function returns the gradient of fxn at w
        w0: initial position [Default 0.0]
        lr: learning rate [0.001]
        eps: min step size threshold [1e-4]
        nmax: maximum number of iters [1000]
        history: whether to store history of x or not [True]
    Returns:
        w: argmin_x f(w)
        converged: True if the final step size is less than eps else false
        H: history
    """
    H = []
    w = w0
    if history:
        H = [[w,fxn(w)]]
    for i in range(nmax):
        dw = -lr*dfxn(w) #gradient step
        if np.linalg.norm(dw)<eps: # we have converged
            break
        if history:
            H.append([w+dw,fxn(w+dw)])
        w = w+dw #gradient update
    converged = np.linalg.norm(dw)<eps
    return w,converged,np.array(H)
```

```python
if __name__=='__main__':
    import matplotlib.pyplot as plt
    def myfunction(w):
        z = (w-0.5)**2#+np.sin(4*w)
        return z
    def mygradient(w):
        dz = 2*(w-0.5)#+4*np.cos(4*w)
        return dz


    wrange = np.linspace(-3,3,100)
    #select random initial point in the range
    w0 = np.min(wrange)+(np.max(wrange)-np.min(wrange))*np.random.rand()

    w,c,H = gd(myfunction,mygradient,w0=w0,lr = 0.01,eps=1e-4,nmax=1000, history = True)

    plt.plot(wrange,myfunction(wrange)); plt.plot(wrange,mygradient(wrange));
    plt.legend(['f(w)','df(w)'])
    plt.xlabel('w');plt.ylabel('value')
    s = 'Convergence in '+str(len(H))+' steps'
    if not c:
        s = 'No '+s
    plt.title(s)
    plt.plot(H[0,0],H[0,1],'ko',markersize=10)
    plt.plot(H[:,0],H[:,1],'r.-')
    plt.plot(H[-1,0],H[-1,1],'k*',markersize=10)
    plt.grid(); plt.show()
```

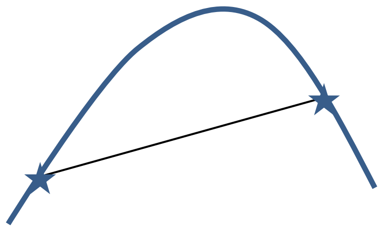
Convex vs. non-convex functions

- If you draw a line between "any" two points on a function and the line always remains above or on the function, then that function is called convex function
  - Strict Convexity

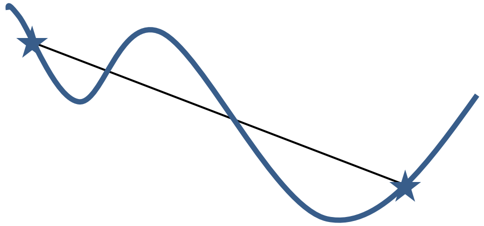
- Convex functions will have a single minima



Is this convex?



Is this convex?



Is this convex?

https://en.wikipedia.org/wiki/Convex_function