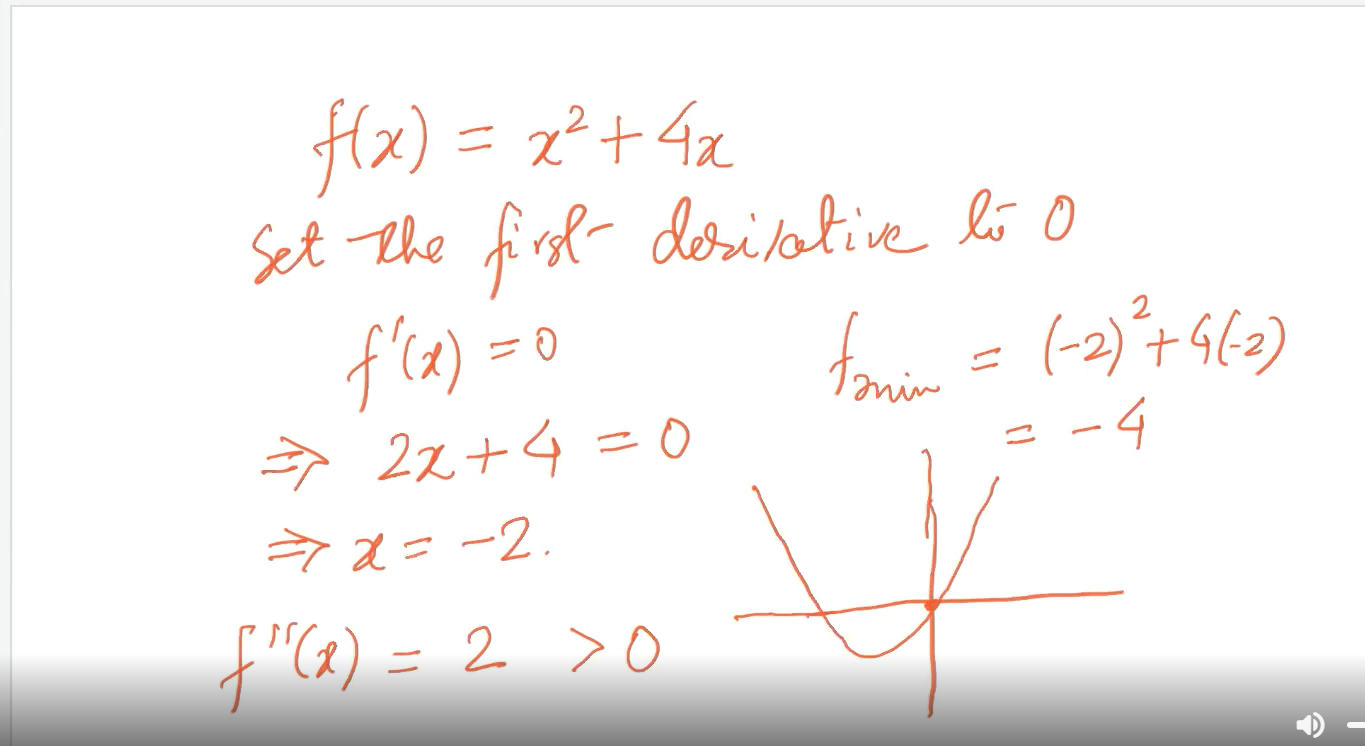
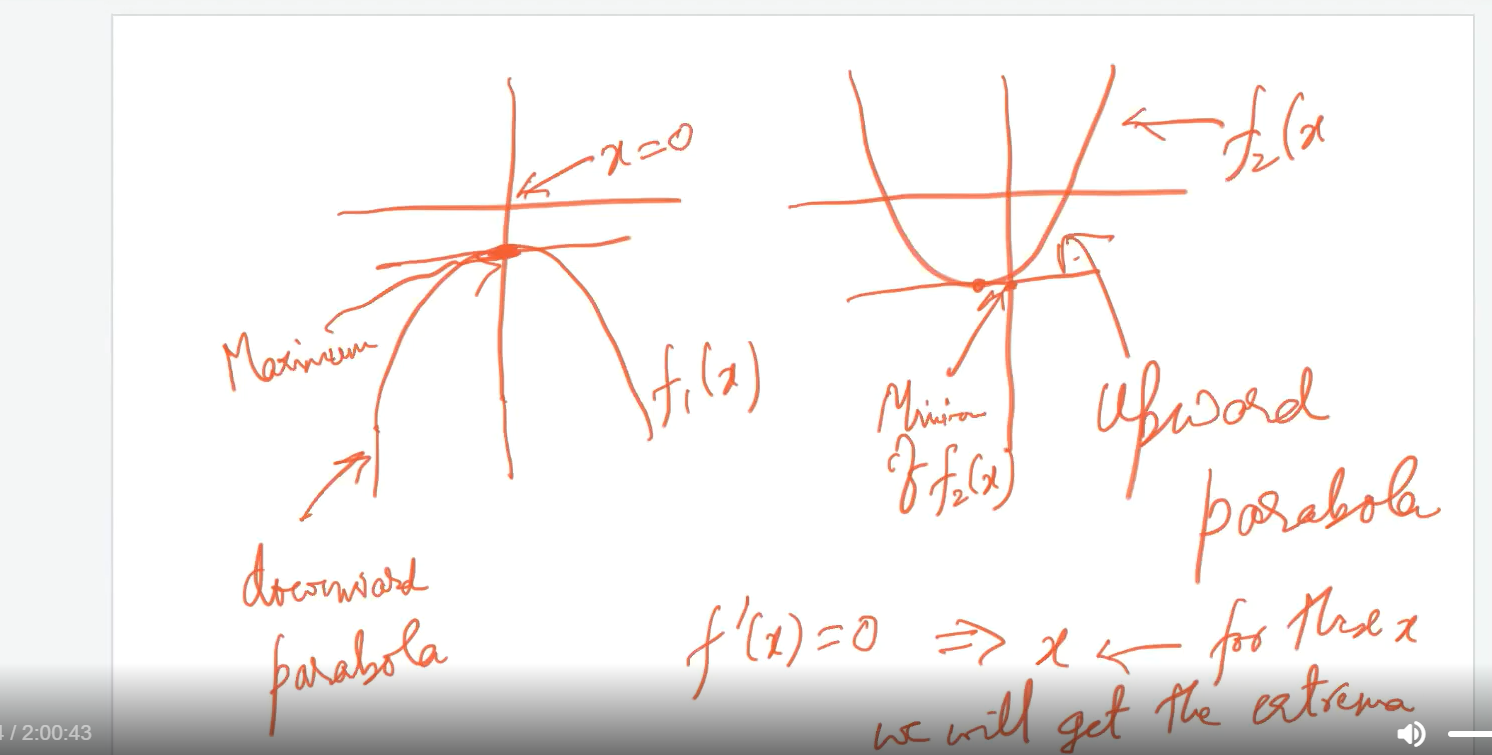
# Gradient descent:

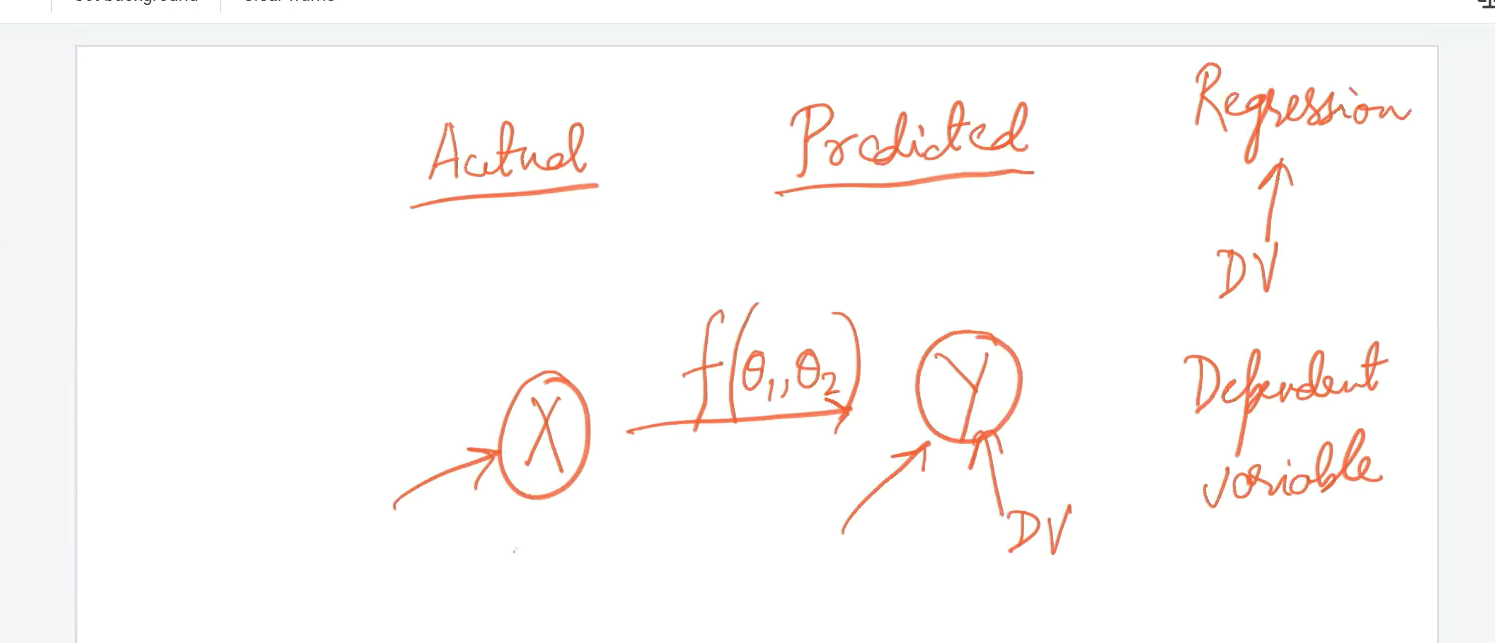
Gradient descent is defined as the local minima of any differentiable function.



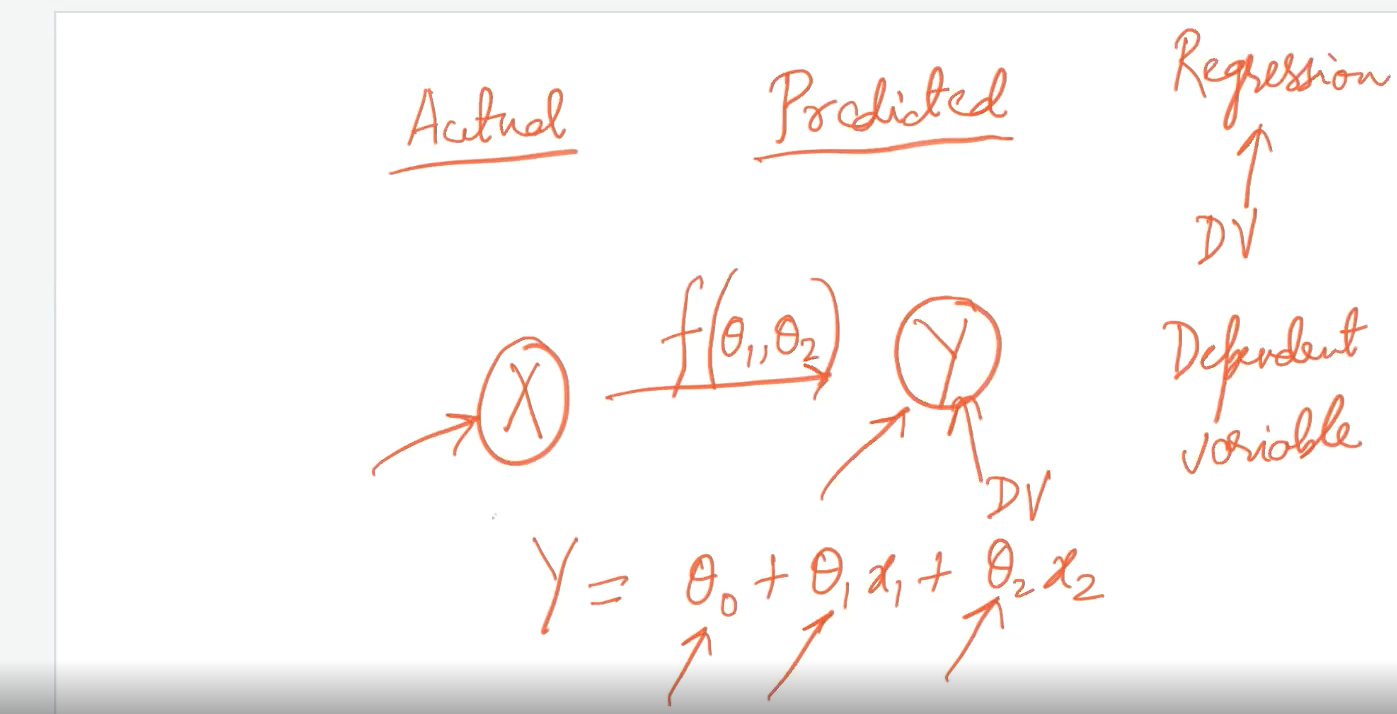
In this we can see that at x=-2 the minimum of the differentiable function exist.



Here you can see by f\_dash(x) we get values of x and by f\_double\_dash(x) we get local minima if its greater than 0 and local maxima if its less than 0.



The work of any ML algorithm is to learn about the optimal parameter and there in between function to find out the relation.



For ex if we take the example of regression algorithm the task of the algorithm is to find out he theta1,theta2,theta3, so on so that we can find out the mapping between x and y.

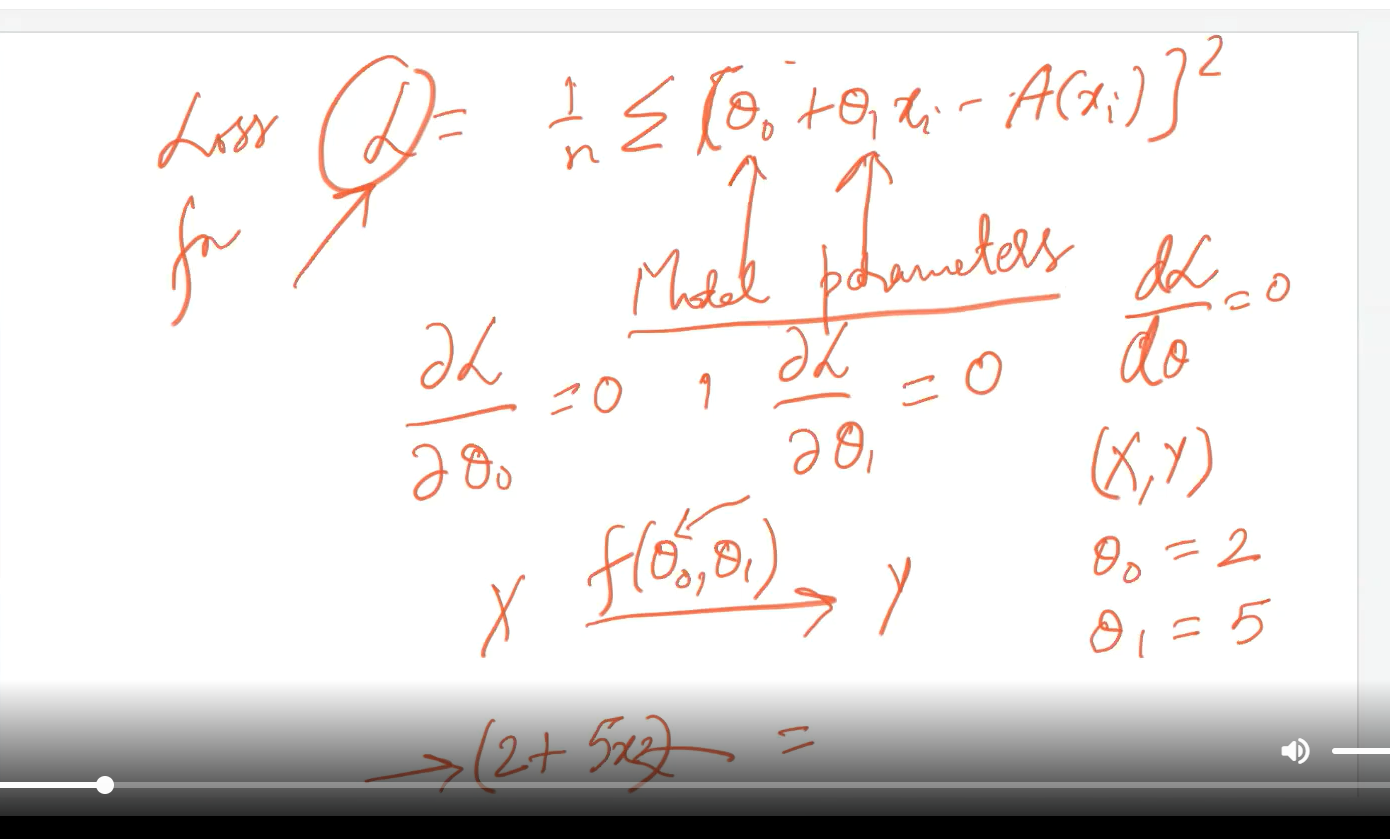
# How to find the mapping:



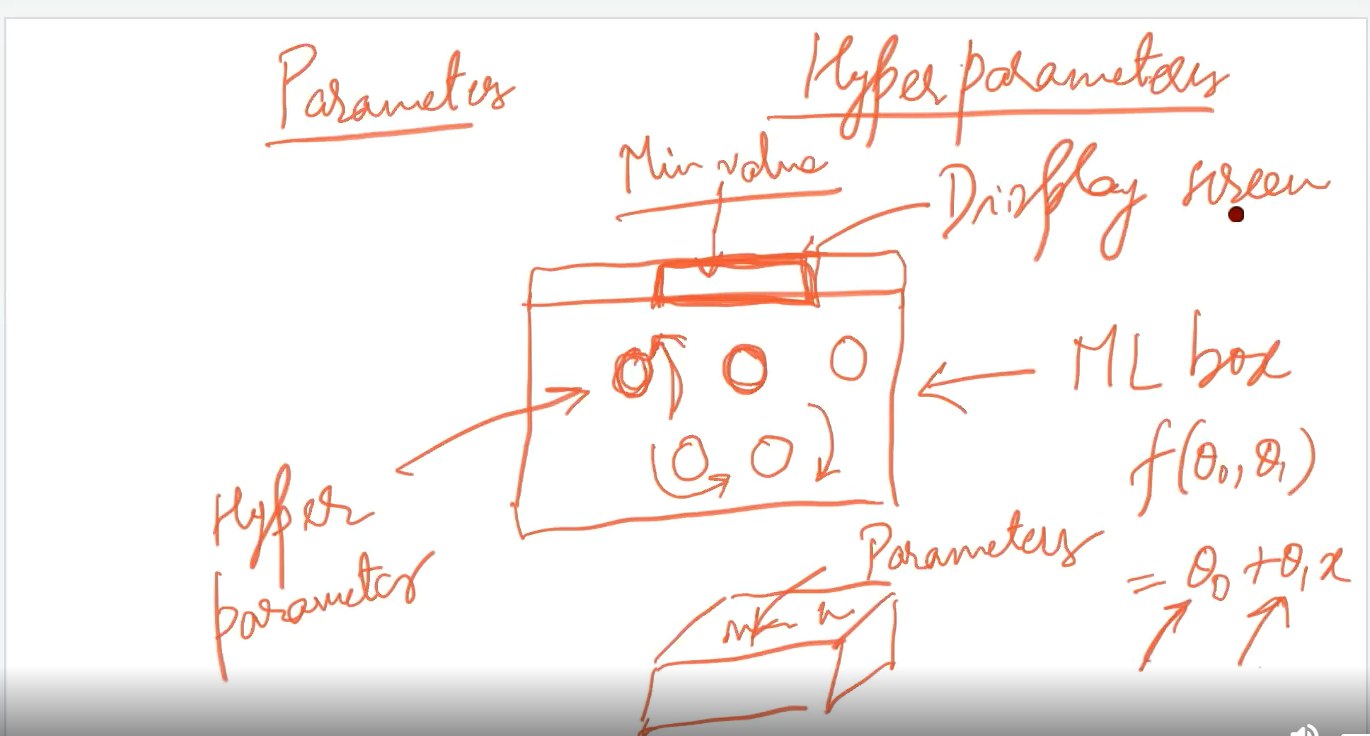
Above the function might be different for different algos and just we need to find out the optimal values(theta1,theta2,etc) so that our loss function will be as less as possible.

# Model Parameter/Goal of any ML algorithm:

The goal of any ML algorithm is to find the mapping between dependent variable and independent variable in such a way that to get optimal parameter and minimum loss function.

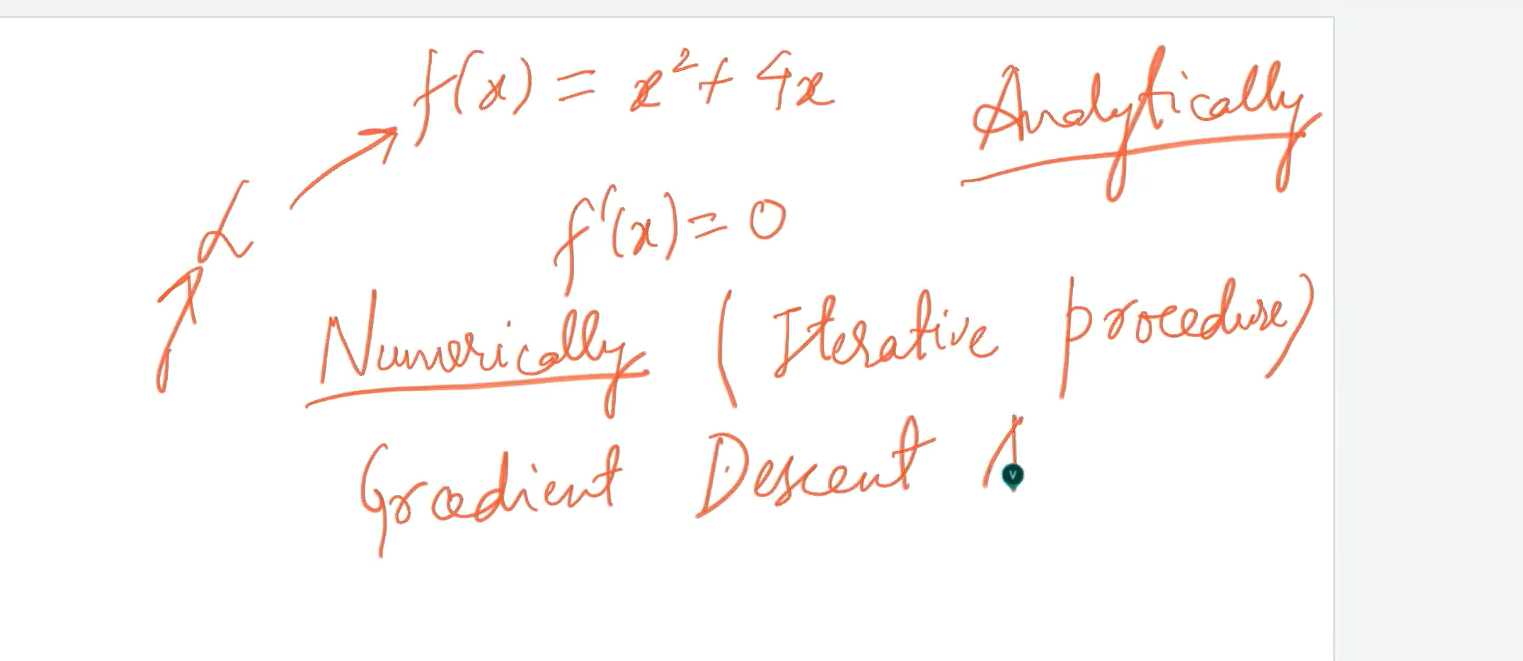


# Hyper parameter and Parameter:

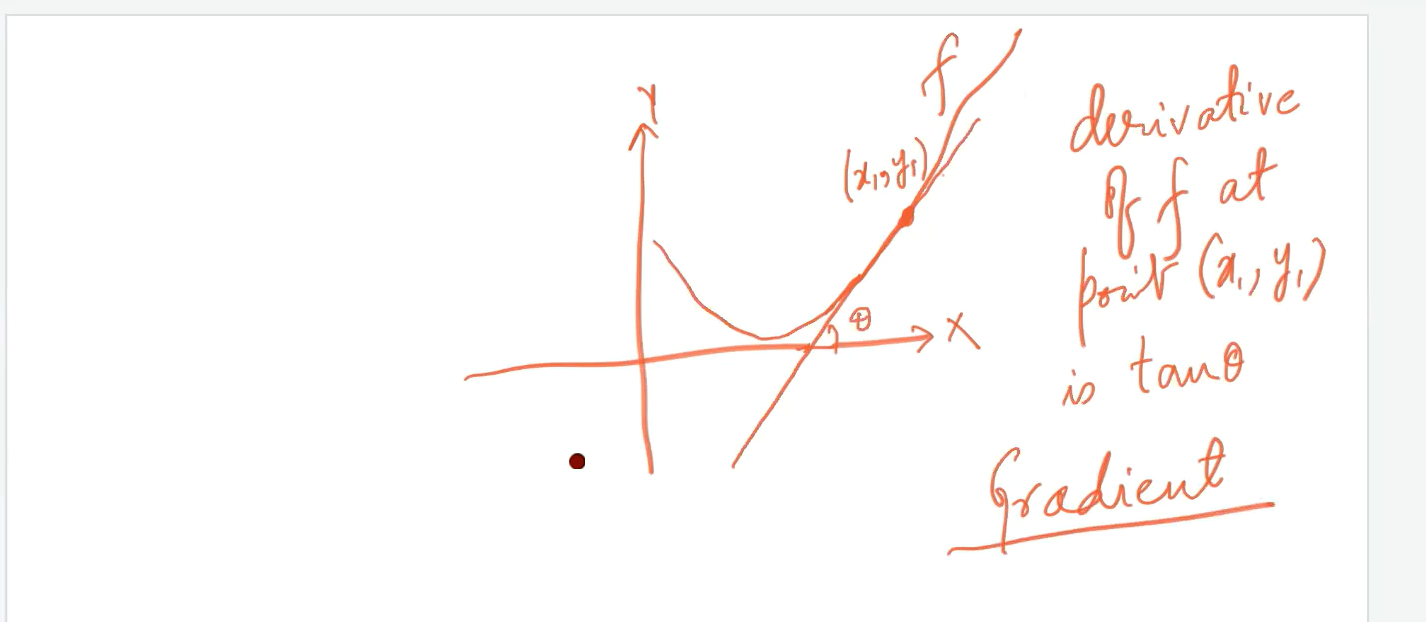


We can’t change the parameter directly, but we can modify hyper parameter in a such a way that it can affect the parameter and from that we can obtain the optimal parameter.

# Analytical v/s Numerical approach:



# Gradient descent function:



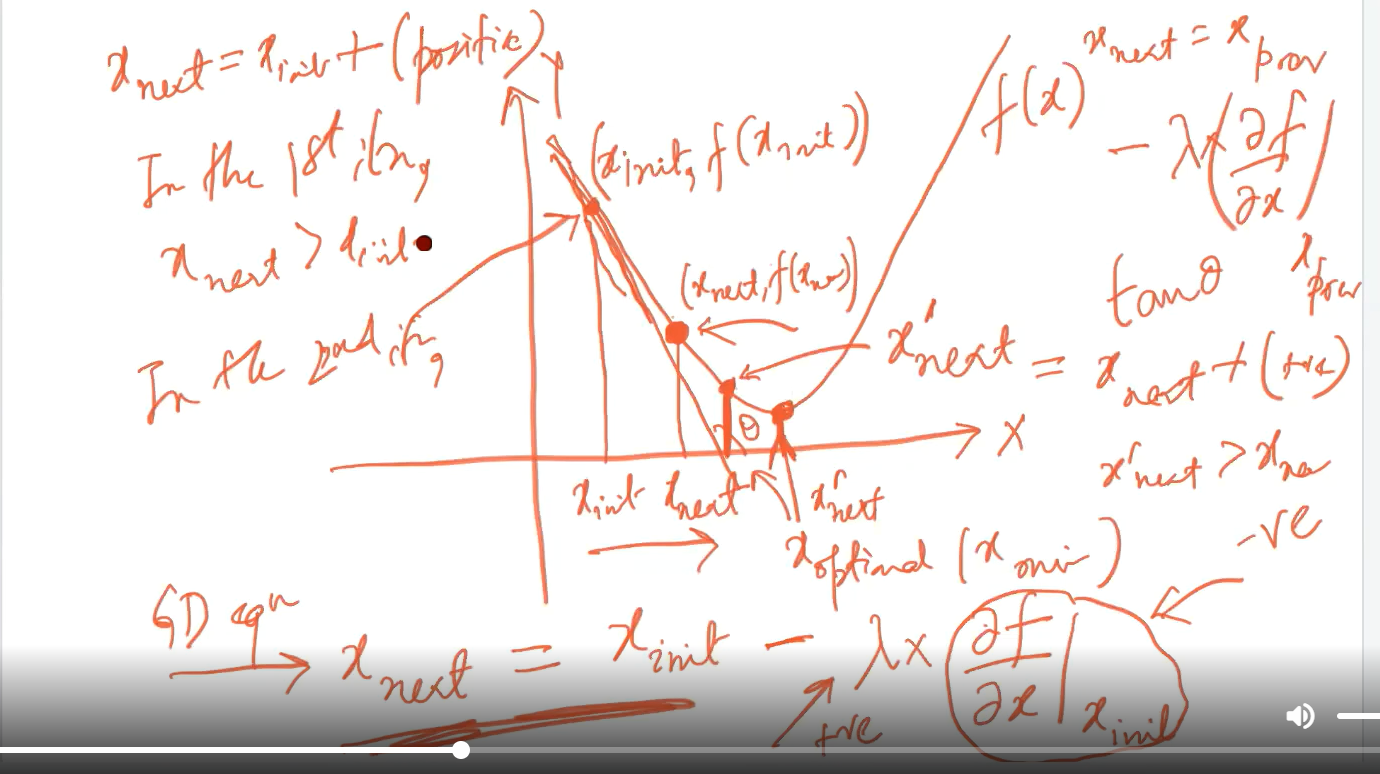
Gradient: tangent of the slope/ differentiation of f at point(x,y)

Descent: gradually decreasing

Optimization: minimizing difference between actual and predicted one.

# Gradient descent strategy:

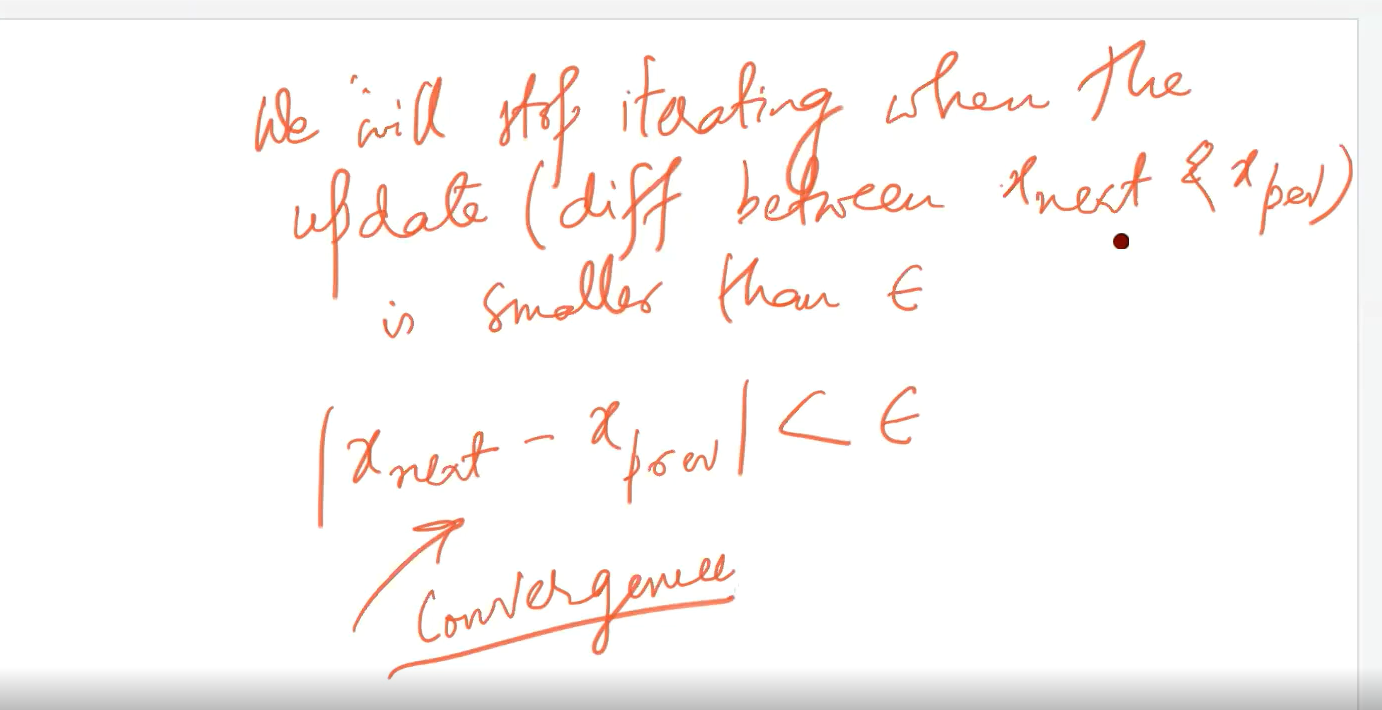
We pass gradient descent function with actual parameters and we get optimal values out of the algorithm.



Here we can see that xnxt=xinint-lambda[(d(f)/d(x)]x(init) as shown above and the tangent is -ve because of the slope (tan(teta)).

Lambda-🡪 Learning rate

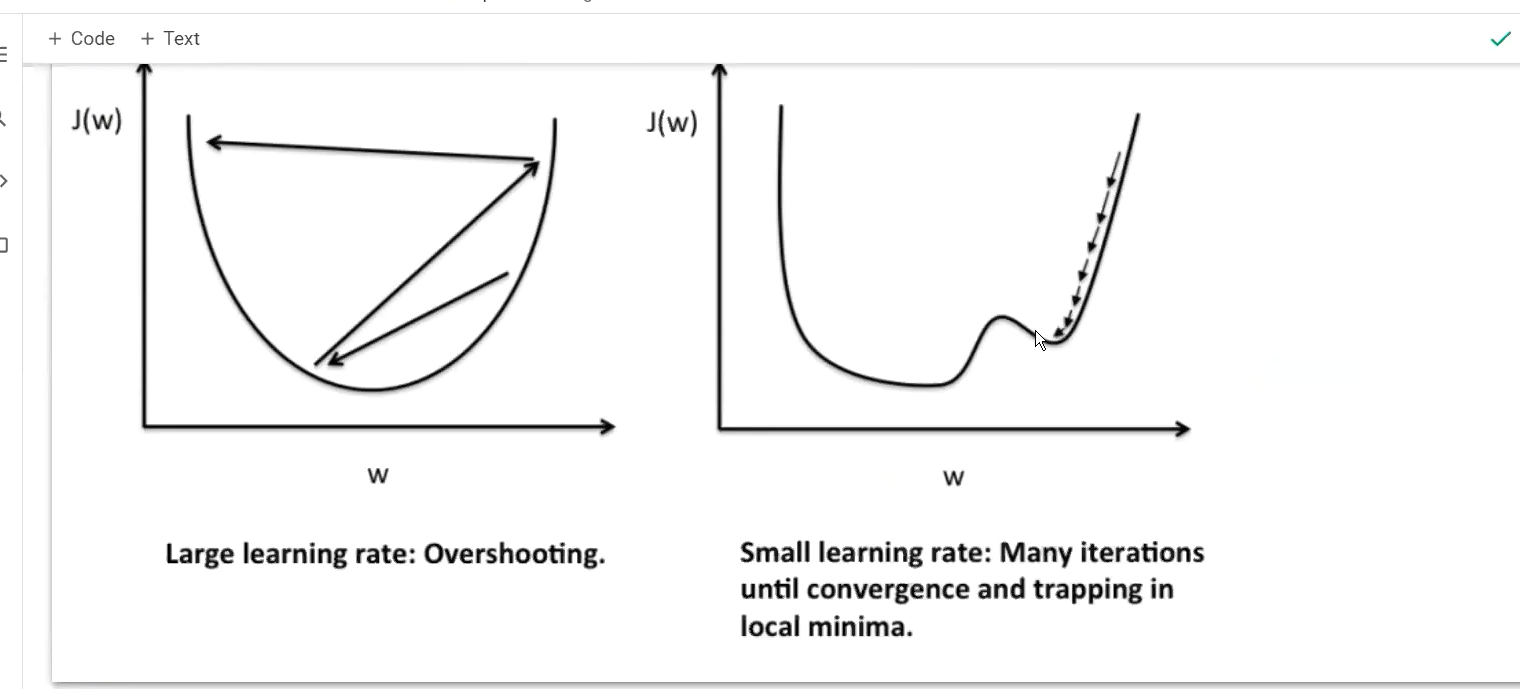
# Convergent:



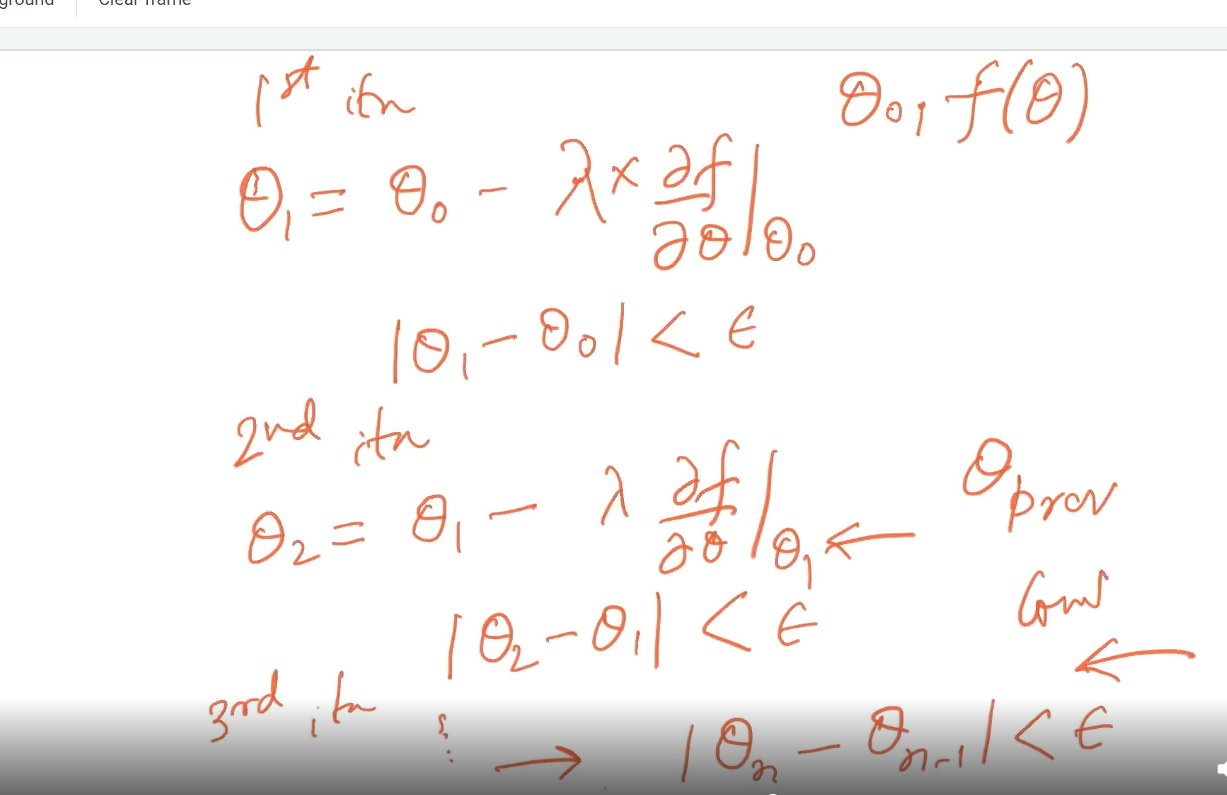
If ebsolone is large then might be learning rate lambda is large and we can jump into minima directly and these descents are not covered.

# Large v/s small learning rate:

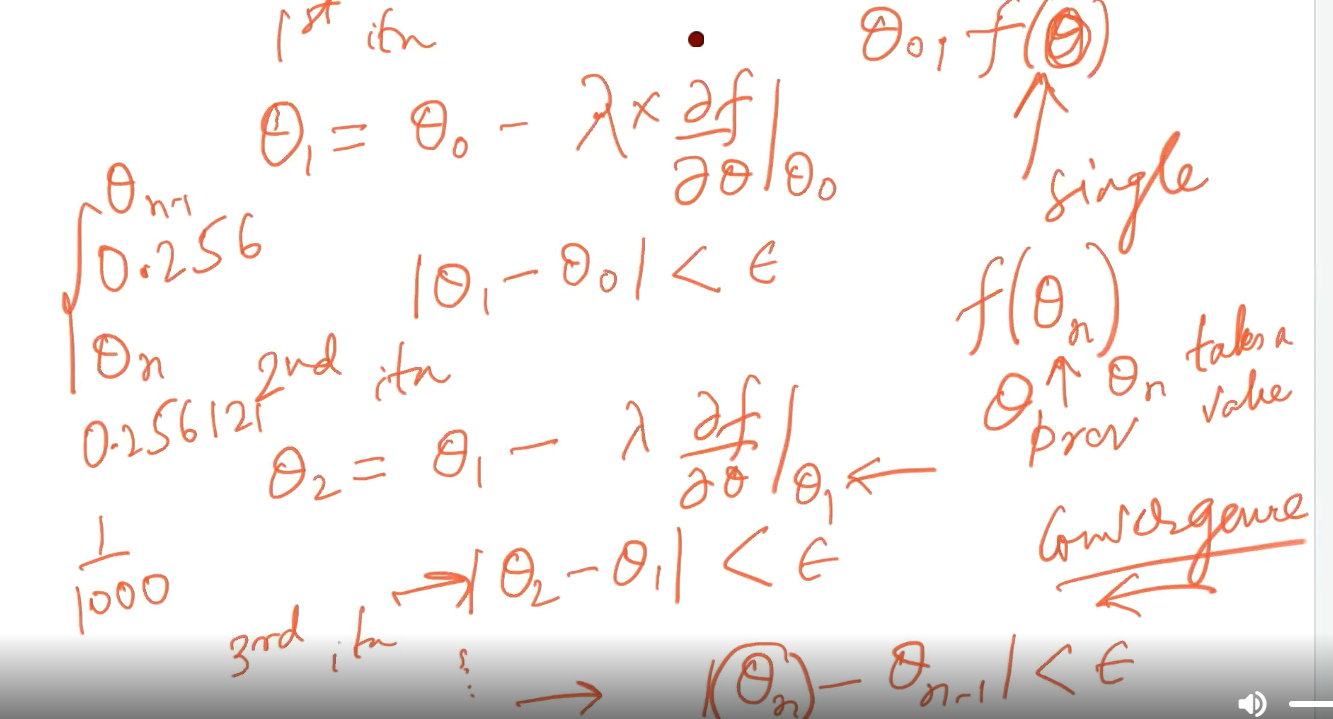
Overshoot and trapping in local minima.\



To get optimal value and to get algorithm result f(theta)=theta(0)+theta(x1)x+theta(2)x^2+rgeta(3)x^3+…

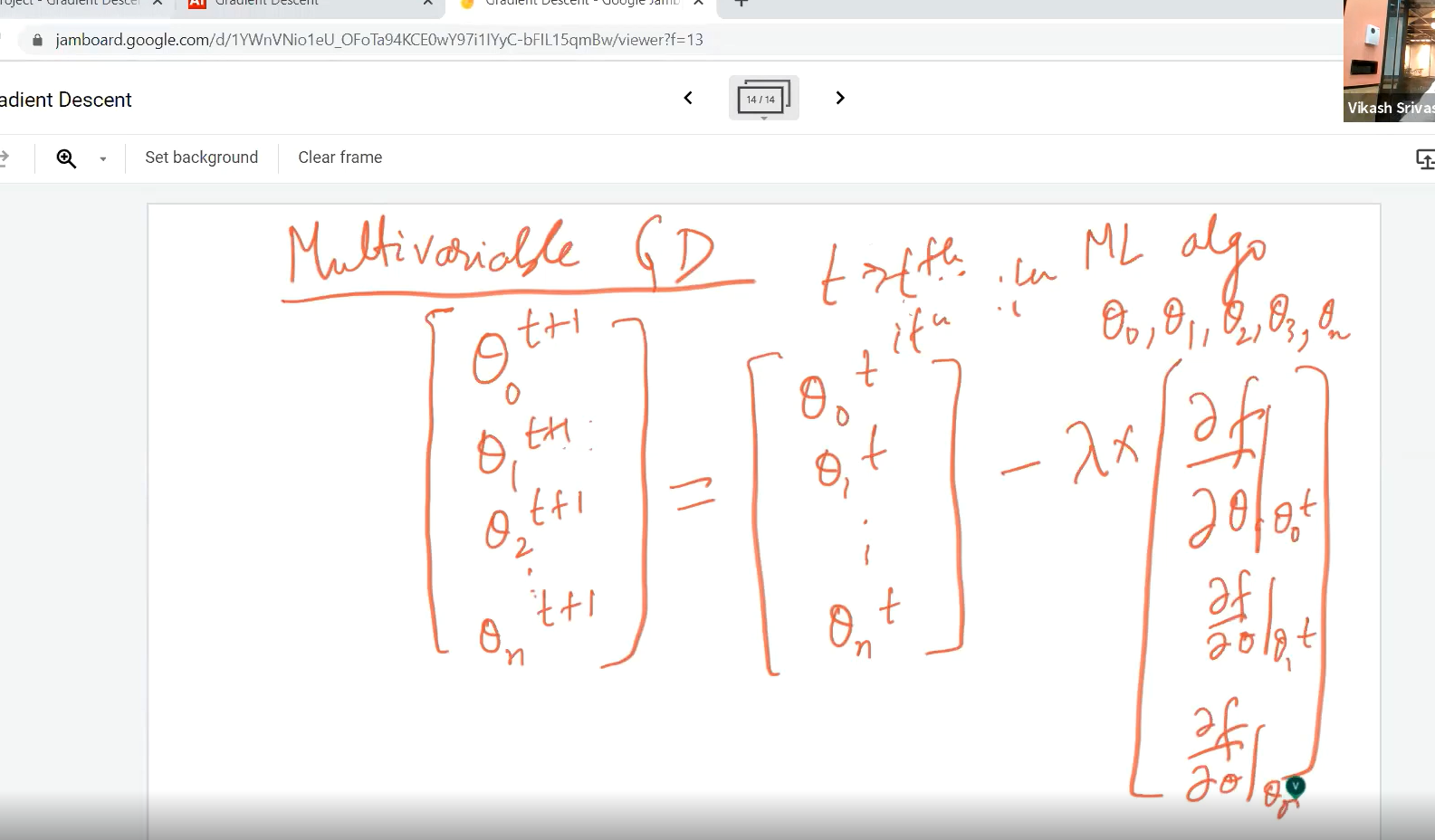


# Convergence by multiple convergence:



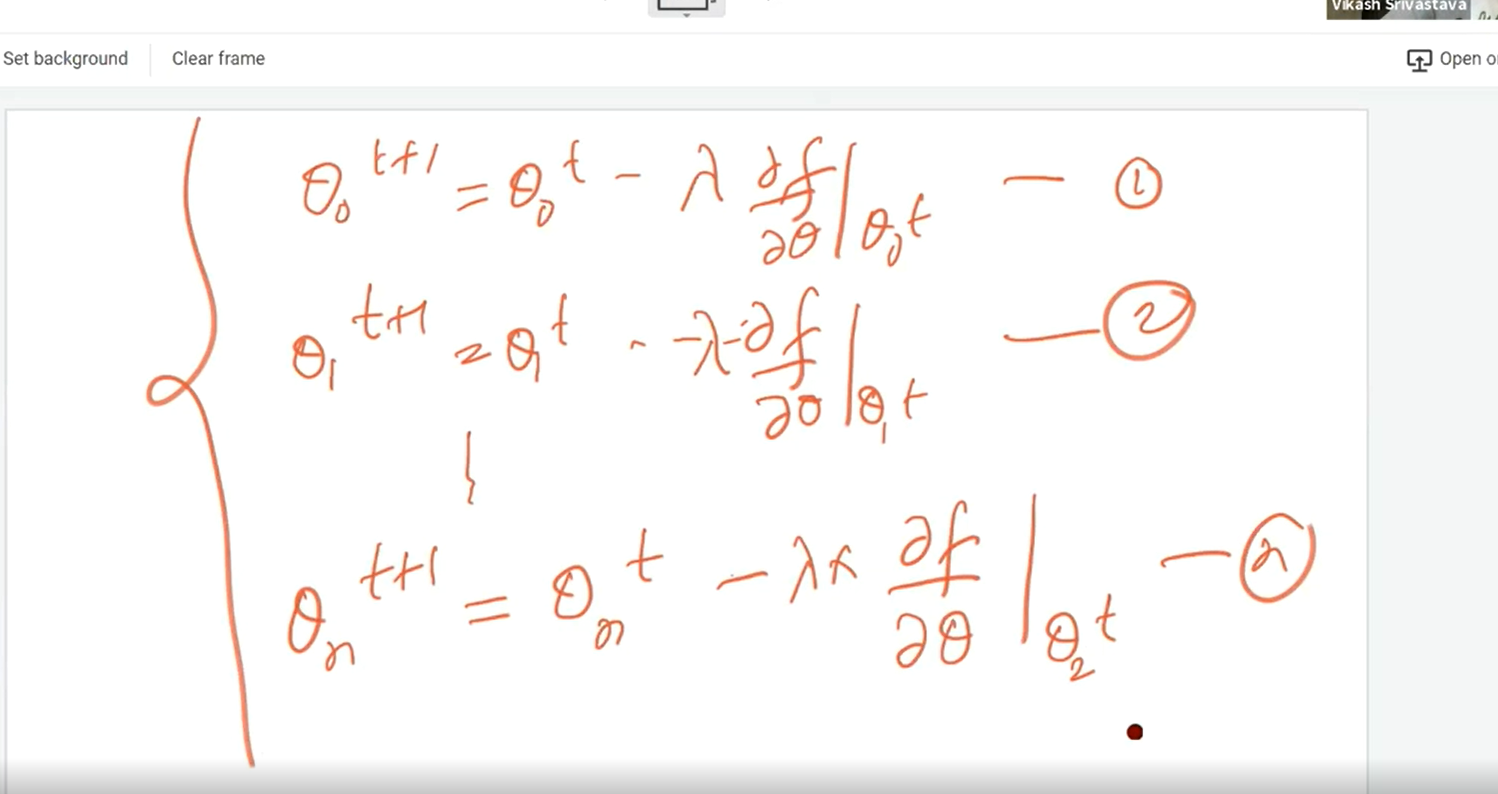
Here we take 1/1000 as a threshold because in thete(n-1) and theta(n) there is hardly some changes greater than 1/1000.

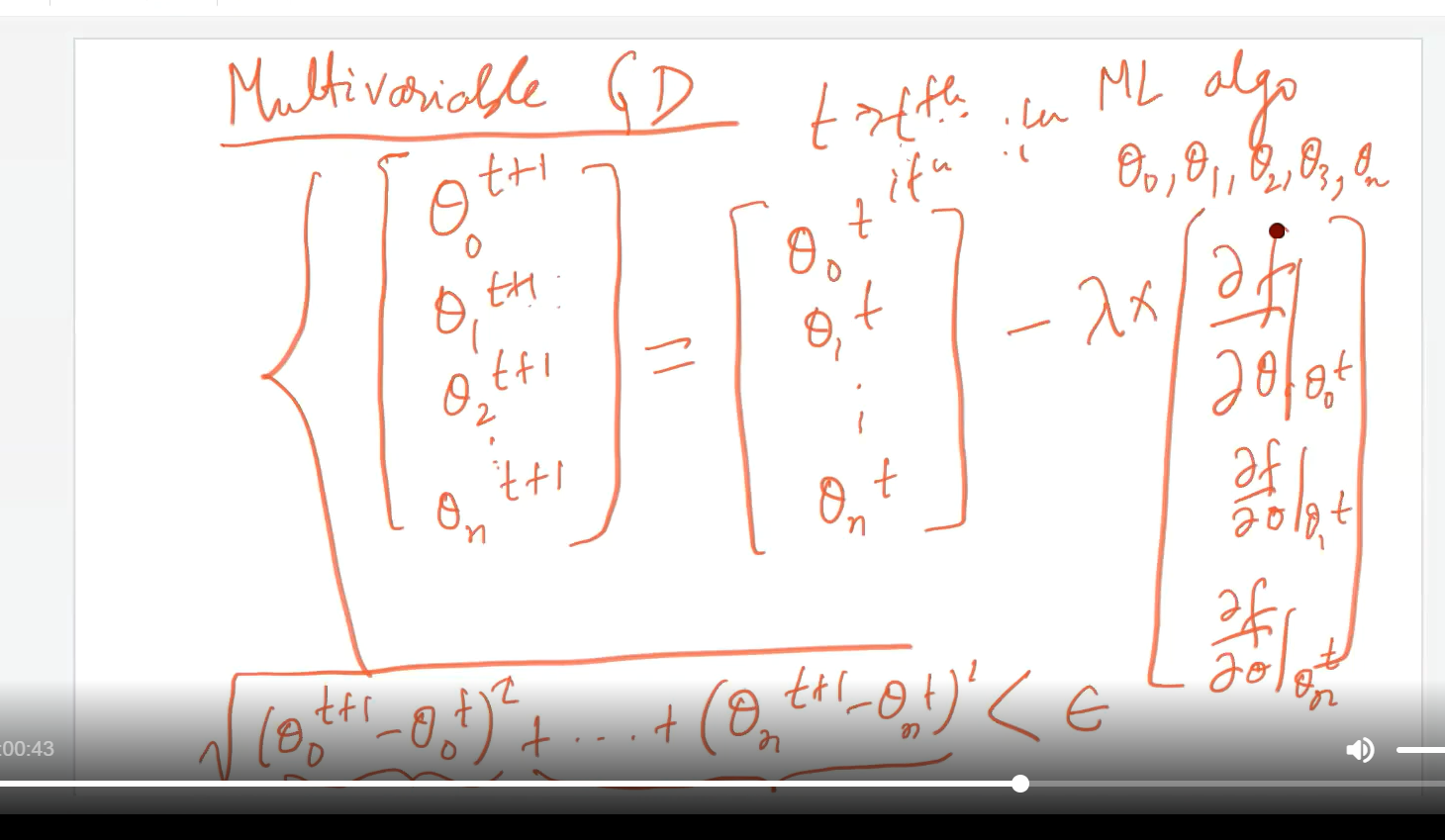
# Multi variable GD:



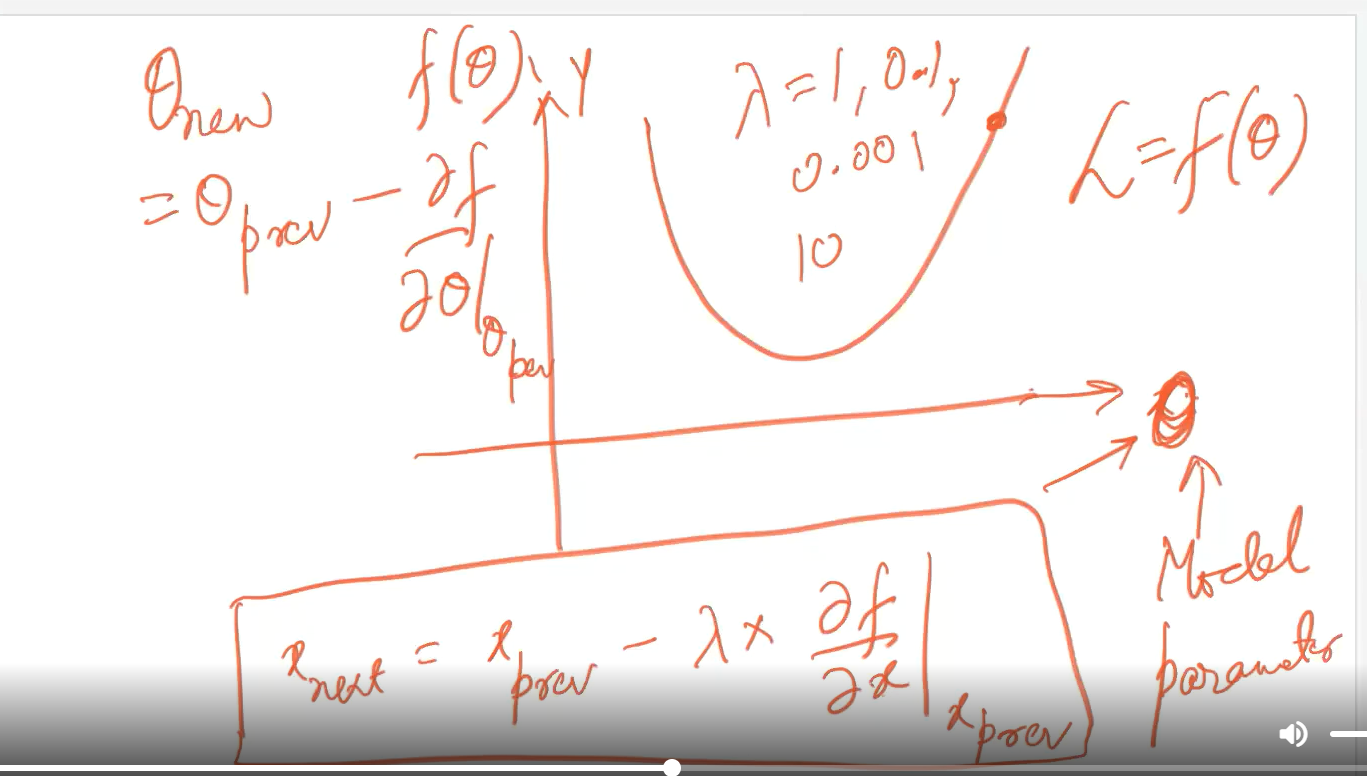
## The updates are done simultaneously for all variables:

Below iterations done simultaneously.

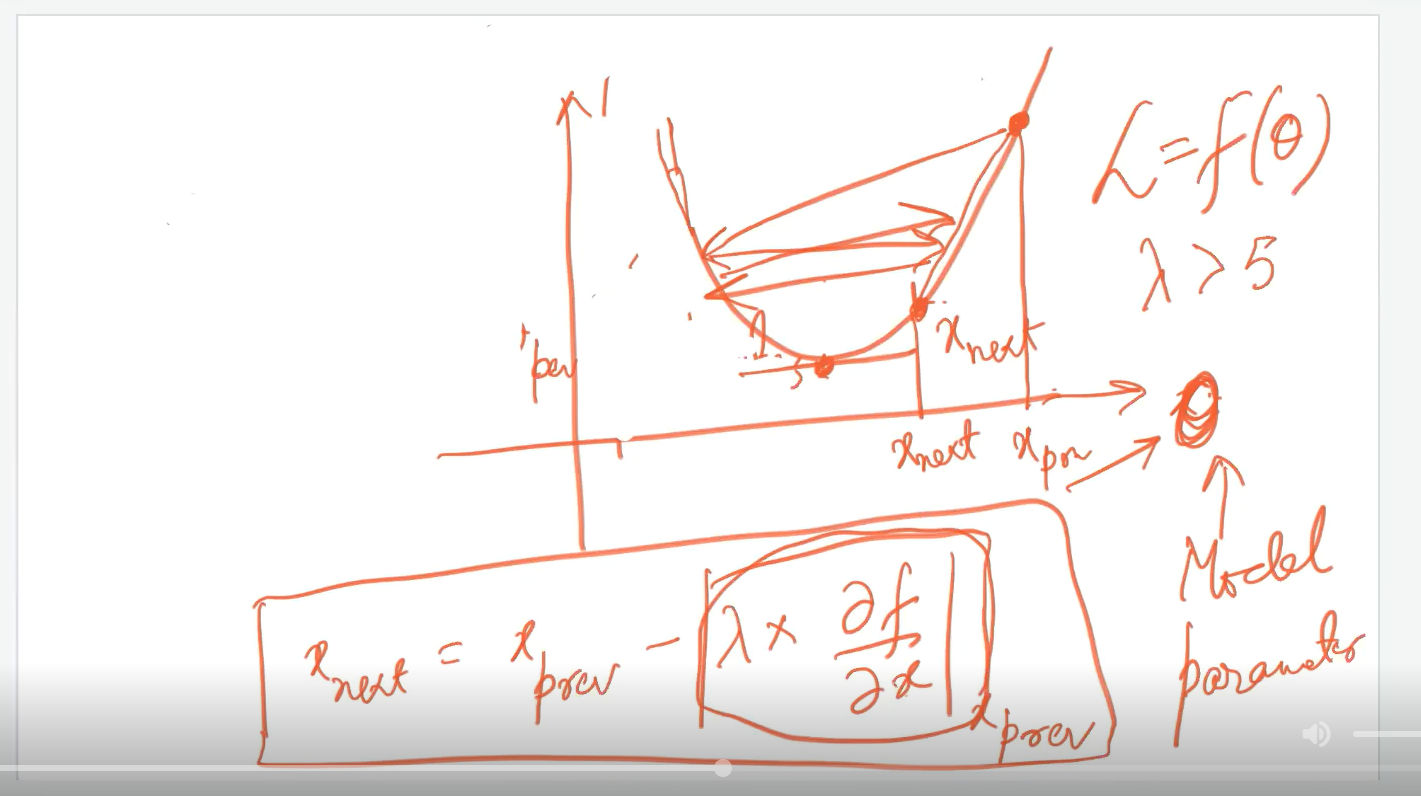




# Model parameter and hyper parameter:



# For larger value of lambda(learning rate) we can overshoot:



If learning rate is large then we cant converge at optimal value.

For multivariate function we have to use partial differentiation to implement:

