Day 84, Feb 22, 2025 (Falgun 10, 2081)

1. Optimization using gradient descent in one Variable
2. Optimization using Gradient Descent in two Variable
3. " " Least Squares with Multiple Observations
4. Numerical Examples on finding Critical points and the minimum points.
5. Optimization using Gradient Descent Instead of Derivative (0/0).

New point = old point − slope

$x_1 = x_0 - f'(x_0)$

Taking big steps

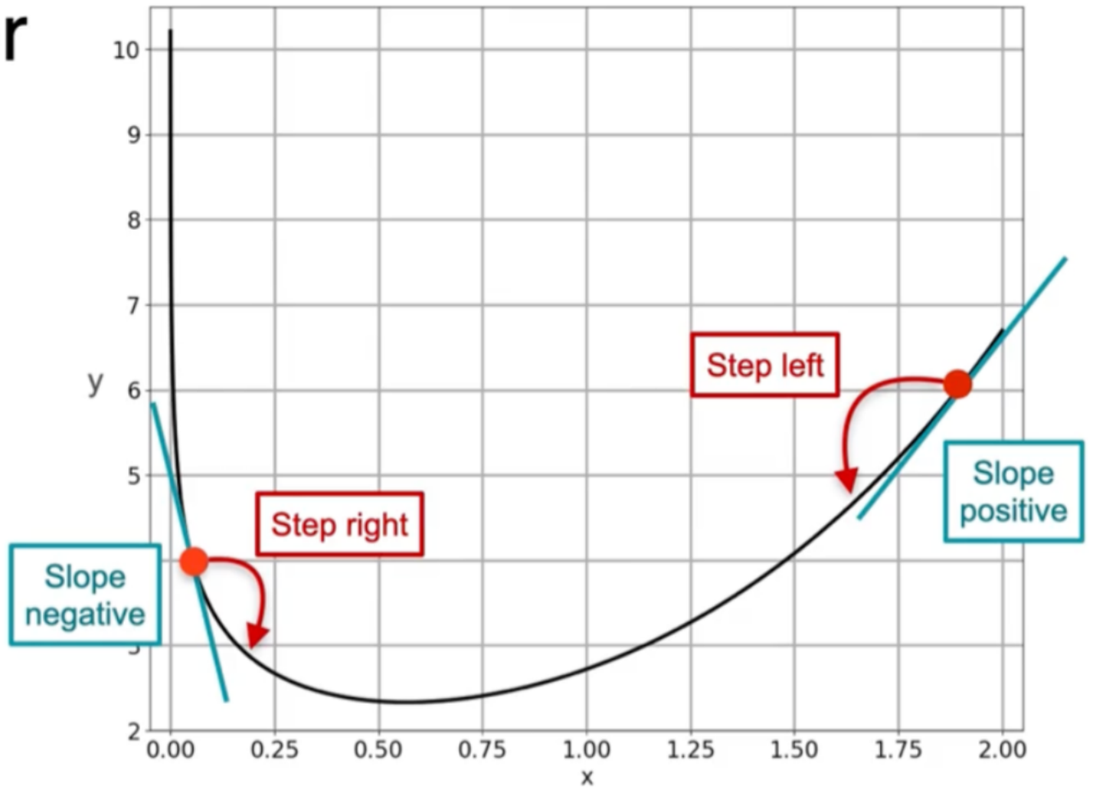
Can be chaotic

And not controable

eg:

Method 2: Be Clever

Try something smarter…



DeepLearning.AI

you need to add a little bit

We use η or α as a learning Rate to control the step size/speed

So, $x_1 = x_0 - \eta f'(x_0)$

Where η can be 0.01 or 0.001 depending on the problem.

It is the learning Rate that has Own Science behind it.

---

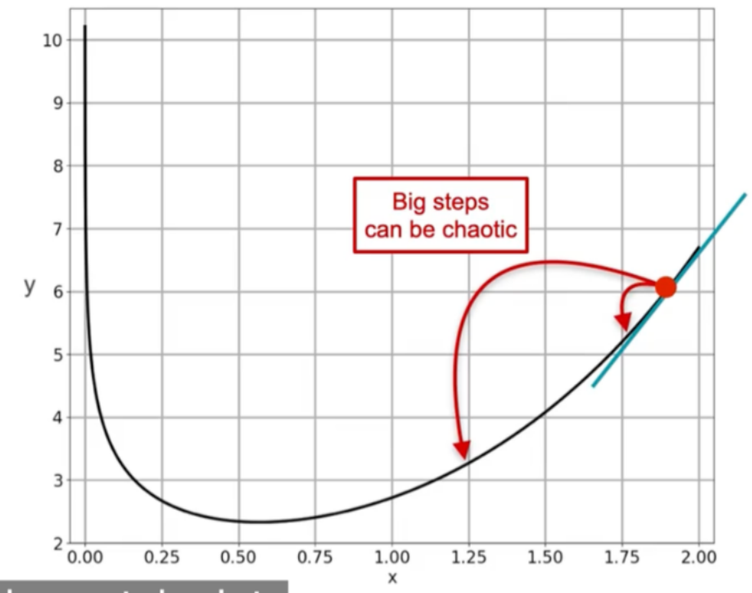
if we play golf and we are far away for away hit really hard to get there closer, if we are not far away we hit carefully and slowly to get the ball into the hole.

Method 2: Be Clever

Try something smarter…

new point = old point - slope

$$x_1 = x_0 - f'(x_0)$$



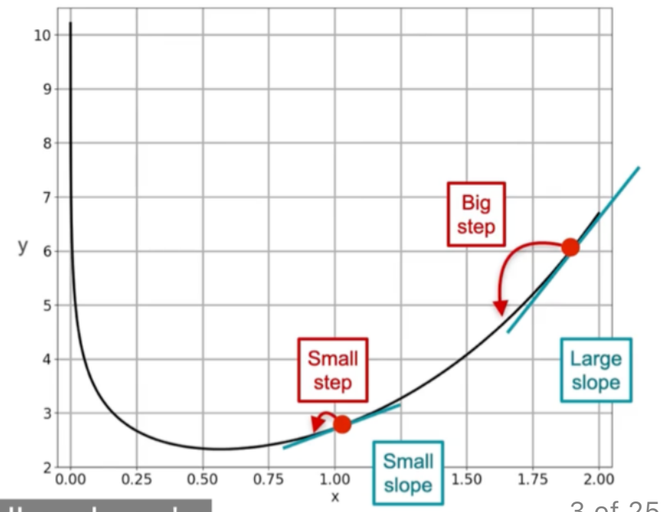
How do we take into account a small step?

DeepLearning.AI

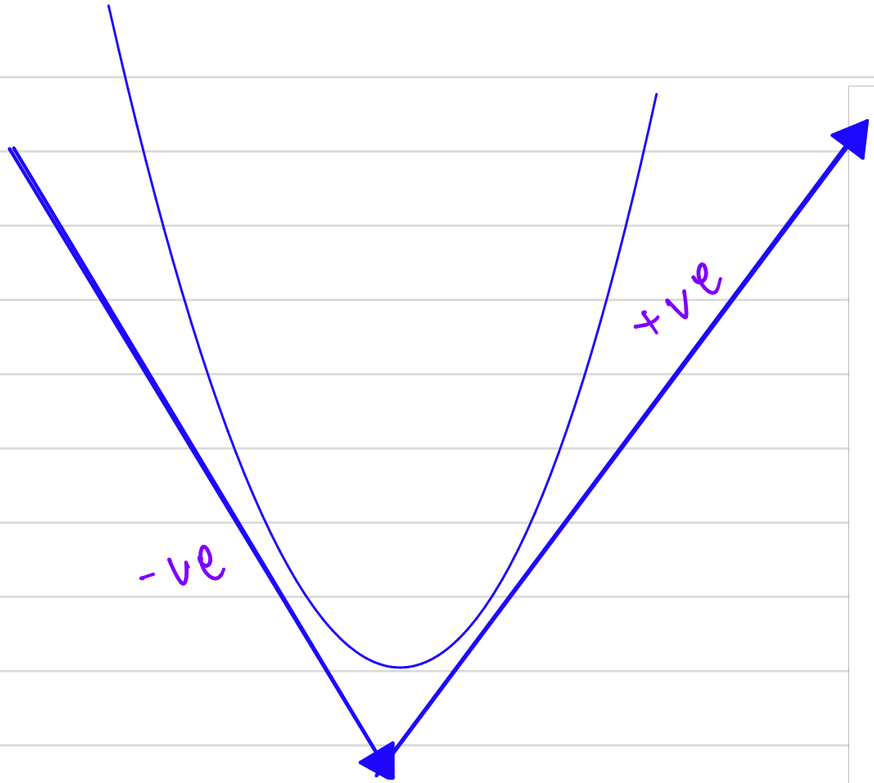
Method 2: Be Clever

Try something smarter…

$$x_1 = x_0 - \alpha f'(x_0)$$



small and you're

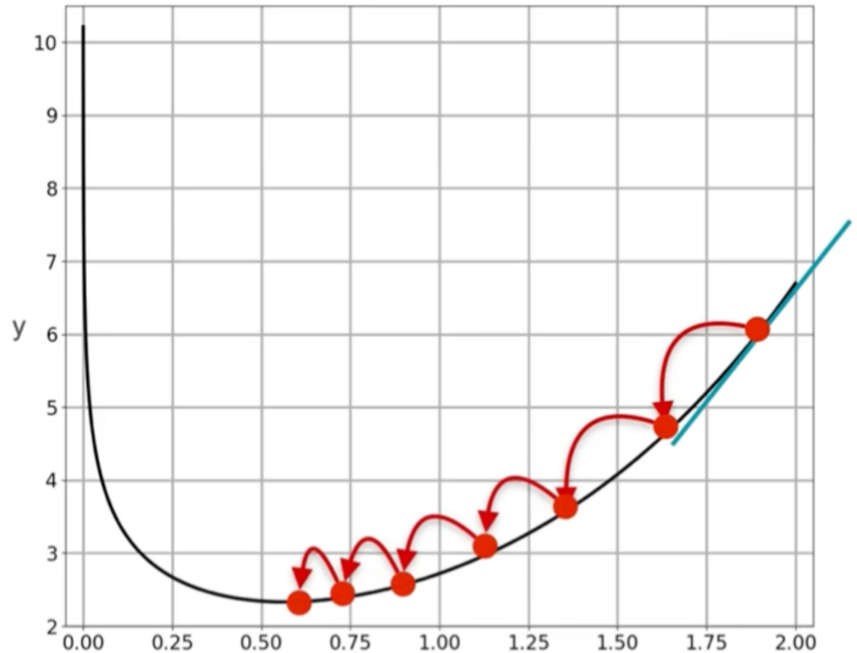


Method 2: Be Clever

Try something smarter…

$$x_{20} = x_{19} - \alpha f'(x_{19})$$

Gradient descent



You can iterate in many times, 20 times,

DeepLearning.AI

Gradient Descent.

$f(x) = e^x - \log(x)$

$f'(x) = e^x - \frac{1}{x}$

(Update Rule $x_k = x_{k-1} - \alpha f'(x_{k-1})$] → Repeat until the true minimum x^*

Update Rule

$$x_k = x_{k-1} - \eta f'(x_{k-1})$$

One benefit is we don't need the derivative to find the local minimum.

Gradient Descent

Function: $f(x)$ Goal: find minimum of $f(x)$

Step 1:

Define a learning rate α

Choose a starting point x_0

Step 2:

Update: $x_k = x_{k-1} - af'(x_{k-1})$

Step 3:

Repeat Step 2 until you are close enough to the true minimum x^*

which is X_k equals X_k minus 1

Gradient Descent

$f(x) = e^x - \log(x)$ $f'(x) = e^x - \frac{1}{x}$

Start: $x = 0.05$ Rate: $\alpha = 0.005$

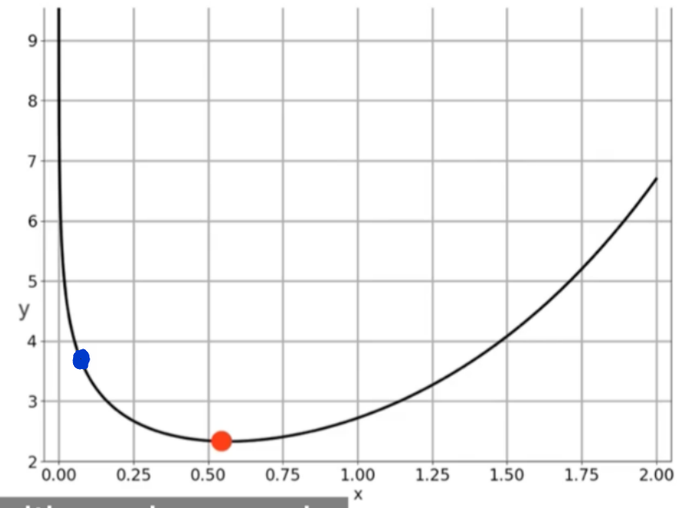
Find:
$f'(0.05) = -18.9$

Move by $-0.005 f'(0.05)$ $x \mapsto 0.1447$

Find:
$f'(0.1447) = -5.7552$

Move by $-0.005 f'(0.1447)$ $x \mapsto 0.1735$

Repeat!



We directly use Gradient Descent formula

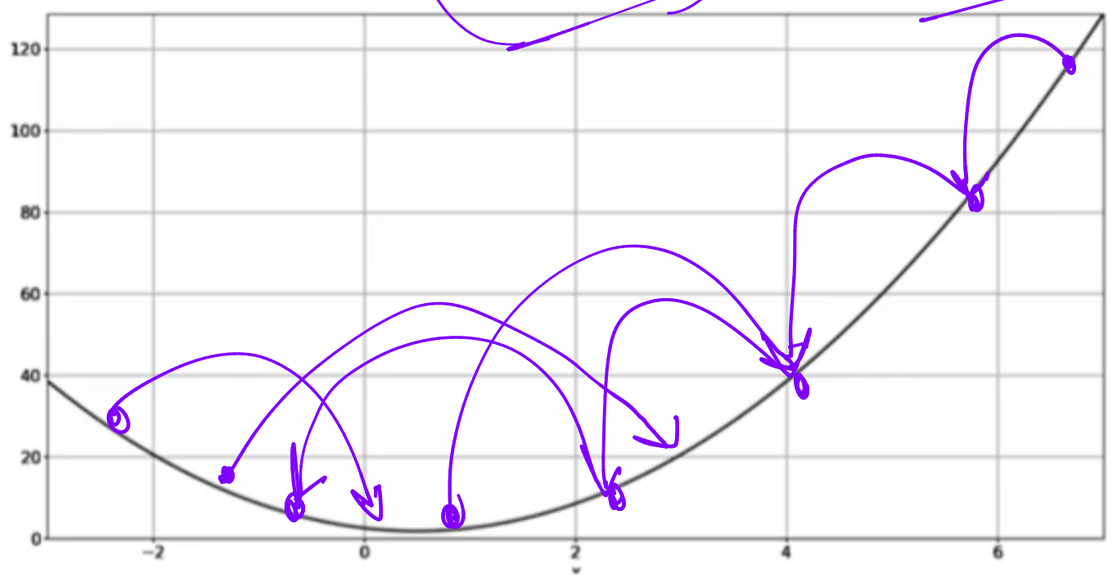
$$\left[w_k = w_{k-1} - \eta \frac{\partial E}{\partial w} \right]$$

it in the algorithm when you're

What Is a Good Learning Rate?

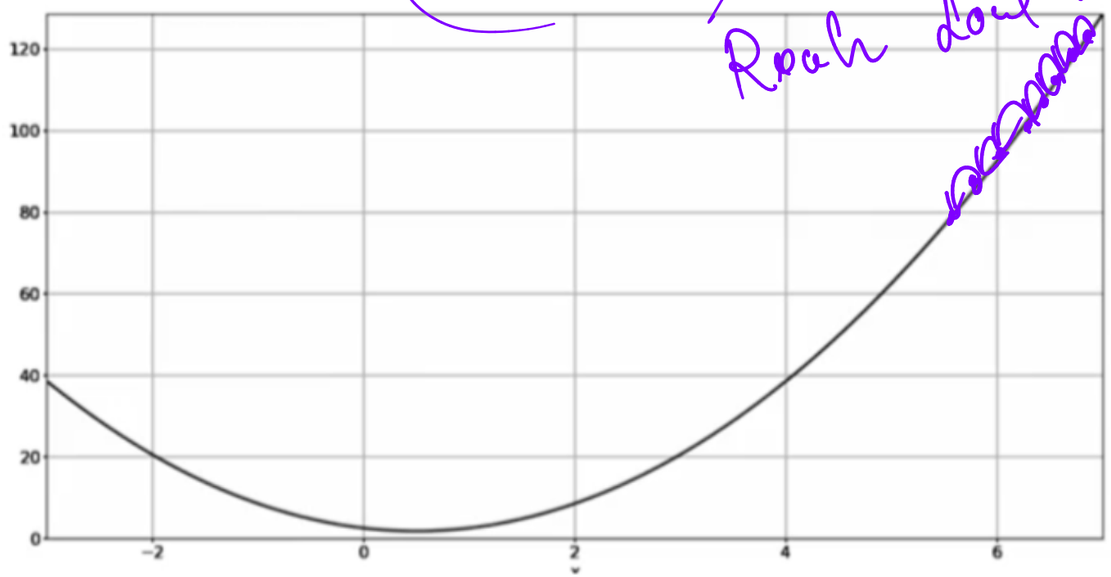
Too large

Chaotic



Too small

takes forever
Reach dowl min.



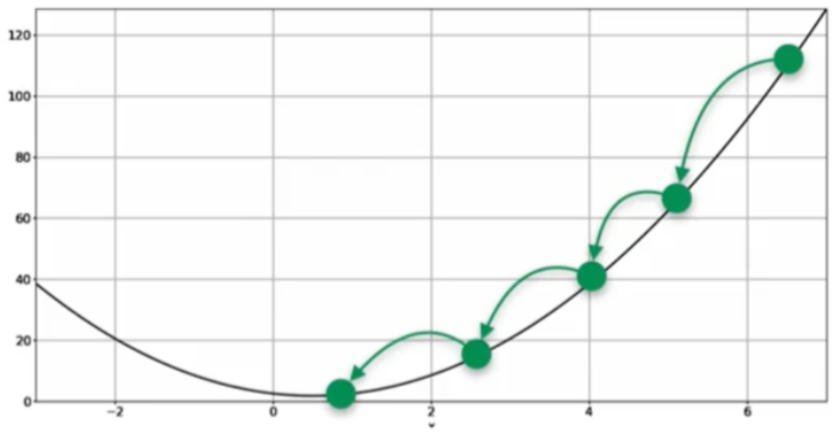
to how well your model does.

DeepLearning.AI

No Definte Method to find the η.

What Is a Good Learning Rate?

Just right



Unfortunately, there is no rule to give the best learning rate α

DeepLearning.AI

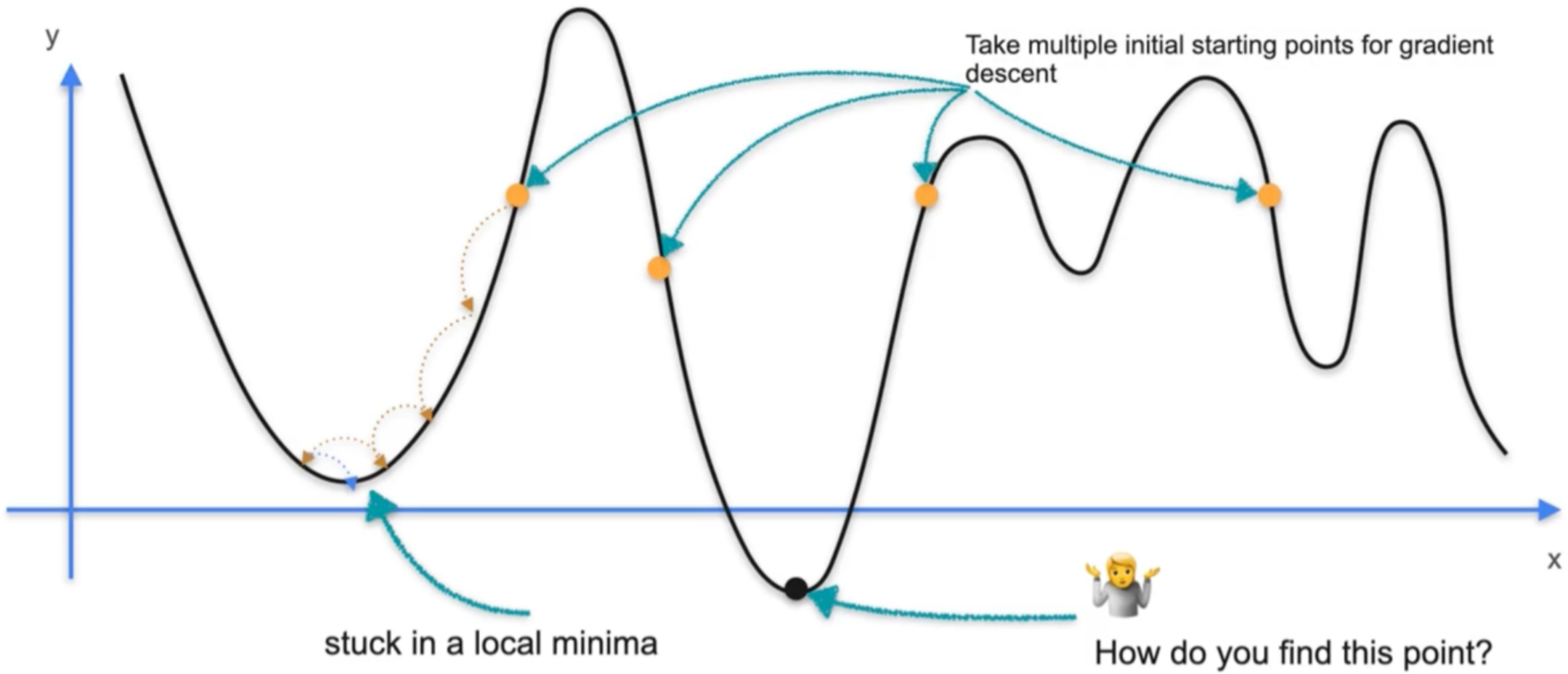
(It's actually a research problem)?

Drawbacks of Gradient Descent.

① Stuck in local minima

Solution: Take Multiple Random points and initalize so that it avoids the local minima.

Drawbacks of Gradient Descent



Once you run it with this,

DeepLearning.AI

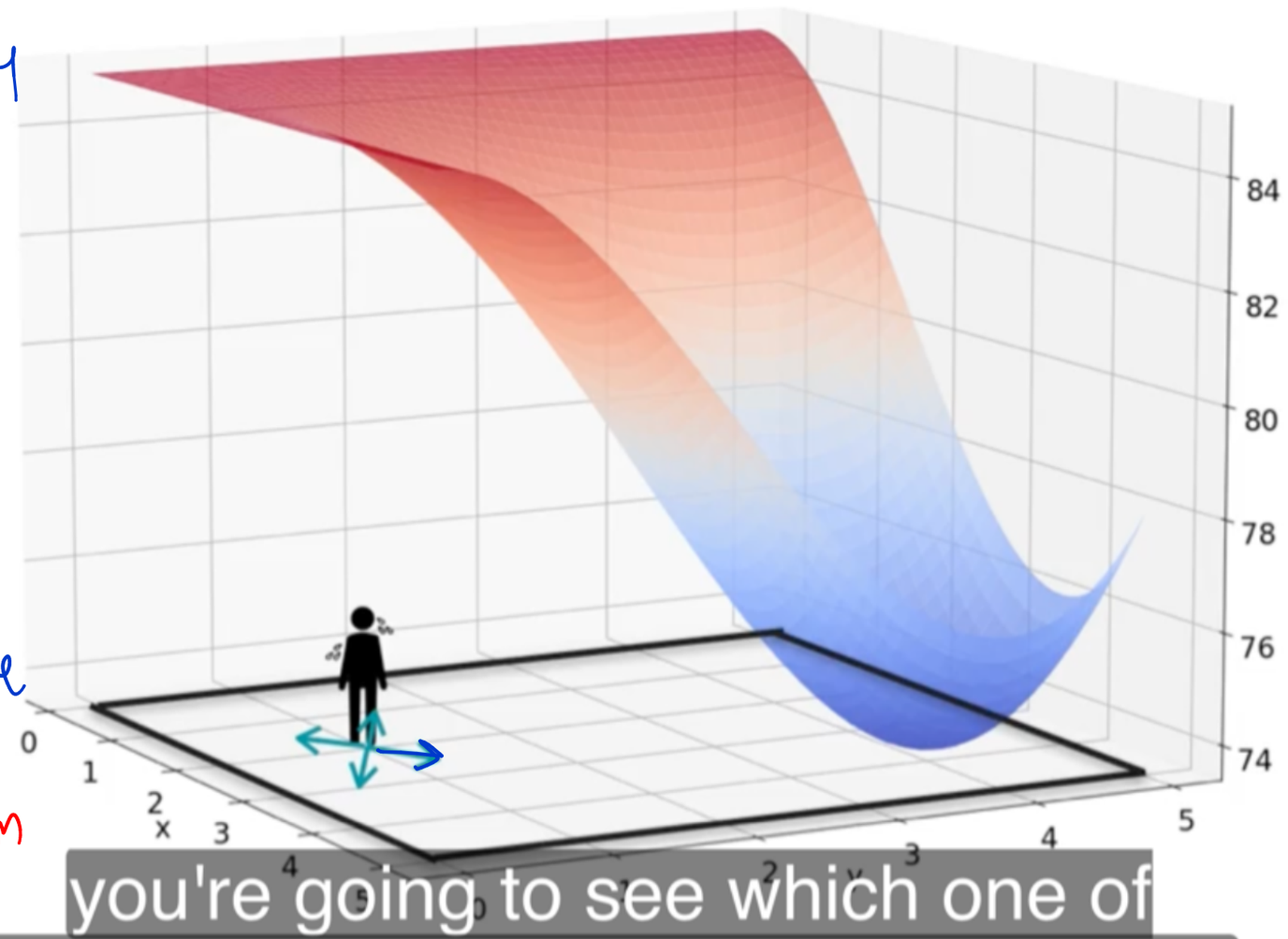
Gradient Descent With Heat Example

(1) take one out of 4 directions

(2) Suppose → then again take 1/4 step direction

(3) Repeat step 2 & Step 3 until reach the coldest place

But How do we know which direction to move?



Gradient Descent also tells us which Direction to move?

DeepLearning.AI

Idea for Gradient Descent

① Initial position: (x_0, y_0)

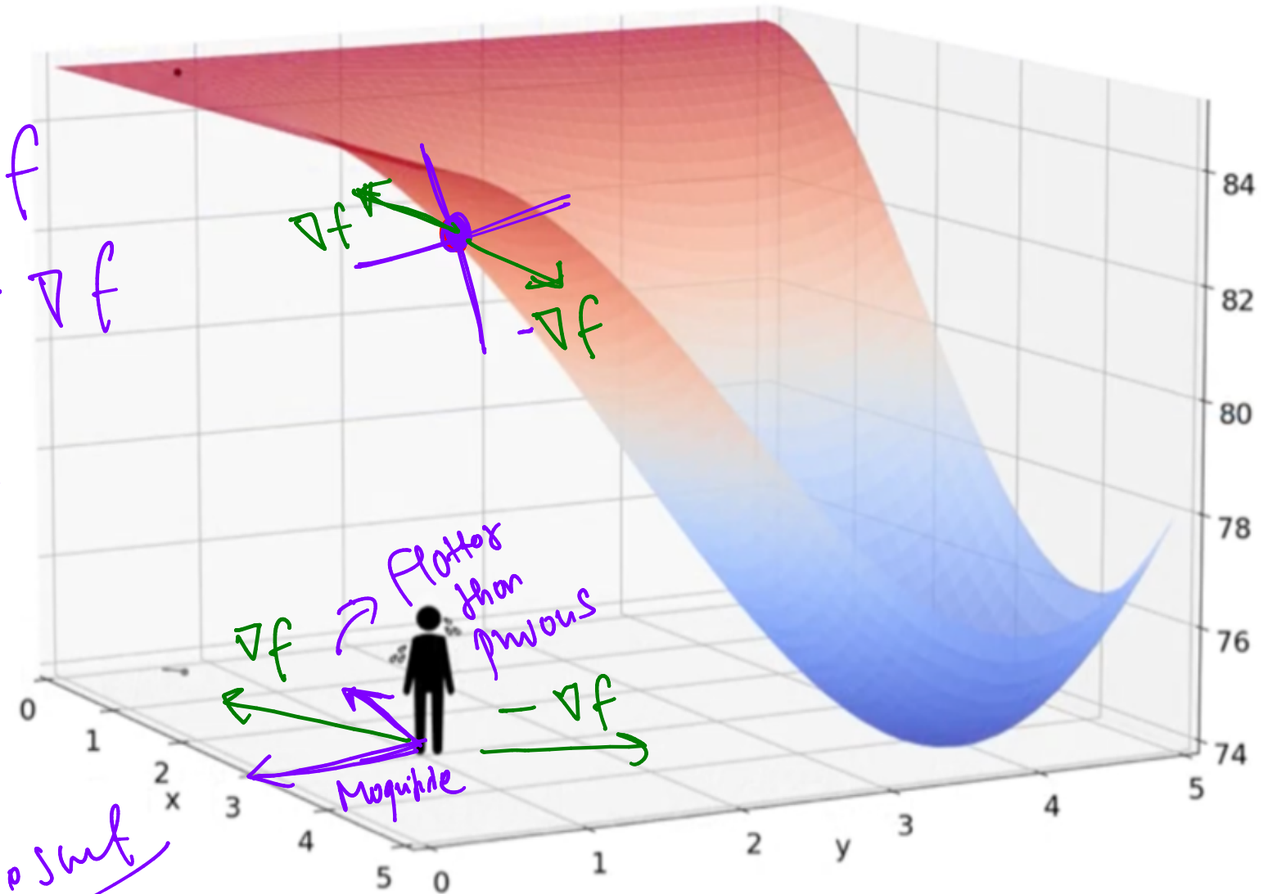
② Direction of greatest ascent: ∇f

③ Direction of greatest descent: $-\nabla f$

④ Updated position: $(x_0, y_0) - \alpha \nabla f$

(x_1, y_1)

Instead of using derivative 0 we use better method Gradient Descent



you saw an okay way to

Idea for Gradient Descent

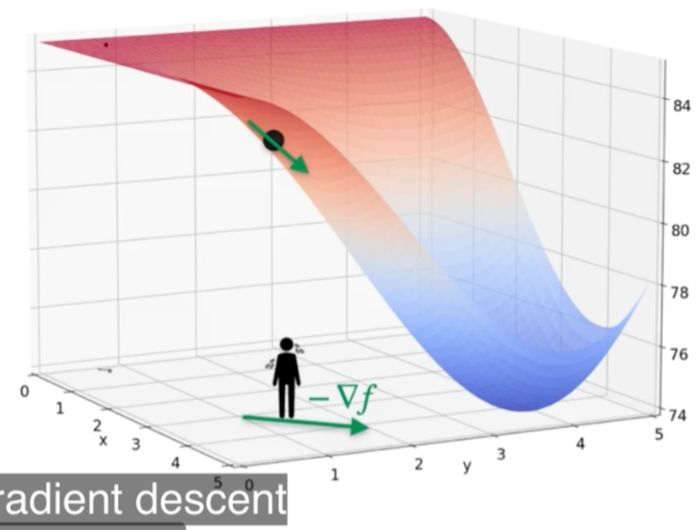
Initial position: (x_0, y_0)

Direction of greatest ascent: ∇f

Direction of greatest descent: $-\nabla f$

Updated position: $\underbrace{(x_0, y_0) - \alpha \nabla f}_{(x_1, y_1)}$

Better point!



same as gradient descent in one variable,

DeepLearning.AI

Move by $\eta = -0.05 \nabla f(0.5, 0.6)$

$x \mapsto 0.5057$

$y \mapsto 0.6047$

Rate $\alpha = 0.05$.

$\nabla f = \begin{bmatrix} -\frac{1}{90} x(3x-12) y^2(y-6) \\ -\frac{1}{90} x^2(x-6) y(3y-12) \end{bmatrix}$

$\nabla f(0.5, 0.6) = \begin{bmatrix} -0.1134 \\ -0.0135 \end{bmatrix}$

Method 2: Gradient Descent

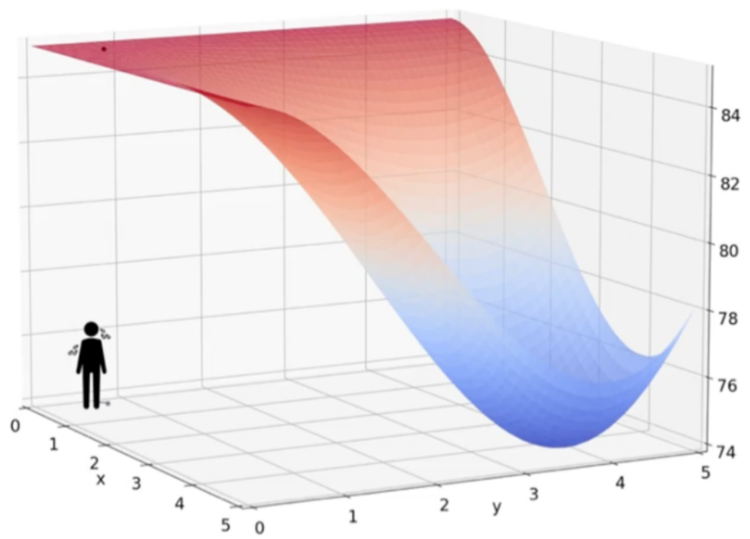
$T = f(x, y) = 85 - \frac{1}{90} x^2(x-6)y^2(y-6)$

Start: $x = 0.5$, $y = 0.6$

$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$

$\frac{\partial f}{\partial x} = -\frac{1}{90} x(3x-12)y^2(y-6)$

$\frac{\partial f}{\partial y} = -\frac{1}{90} x^2(x-6)y(3y-12)$



previously so here they are.

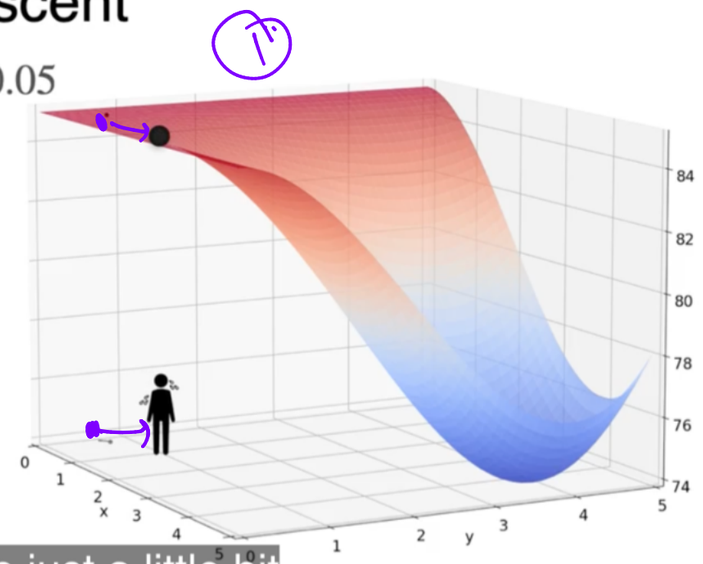
Method 2: Gradient Descent

Start: $x = 0.5,\ y = 0.6$ Rate: $\alpha = 0.05$

$$\nabla f(0.5,0.6) = \begin{bmatrix} -0.1134 \\ -0.0935 \end{bmatrix}$$

Move by
$-0.05\ \nabla f(0.5,0.6)$

$x \mapsto 0.5057$
$y \mapsto 0.6047$



Now you're just a little bit
closer to the minimum poi

DeepLearning.AI

Using ∇ affect the magnitude and direction of both x and y variables.

2

Gradient Descent

Start: $x = 0.5,\ y = 0.6$ Rate: $\alpha = 0.05$

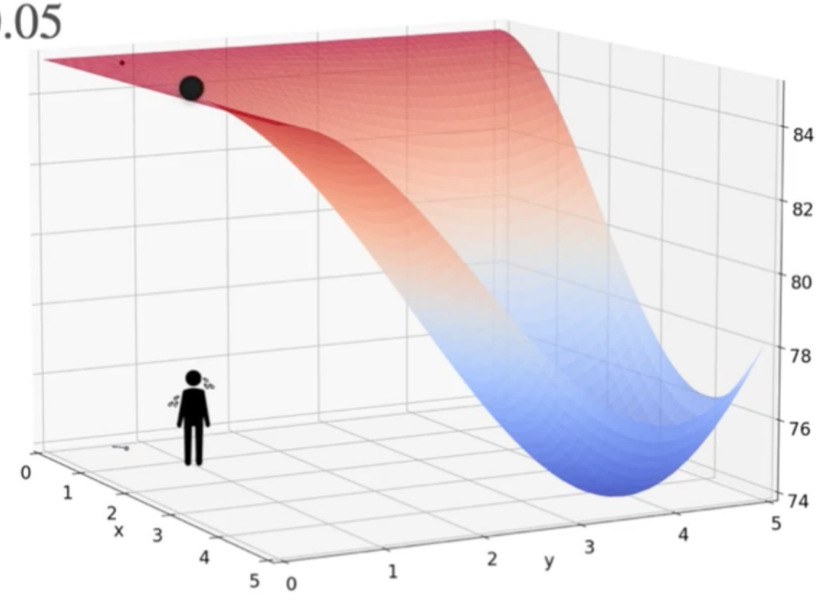
Find:

$$\nabla f(0.5057,0.6047) = \begin{bmatrix} -0.1162 \\ -0.0961 \end{bmatrix}$$

Move by
$-0.05\ \nabla f(0.5057,0.6047)$

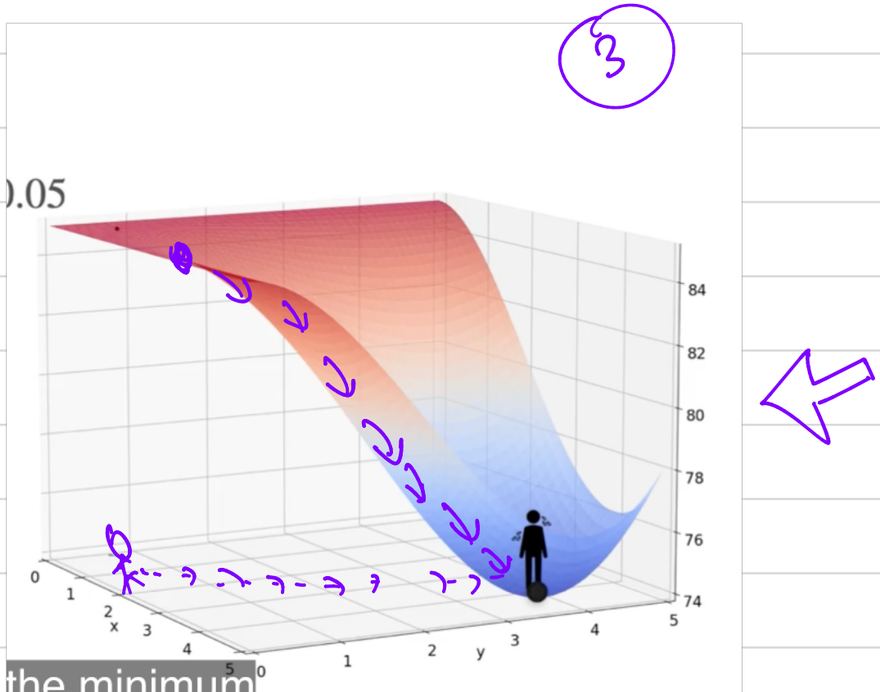
$x \mapsto 0.5115$
$y \mapsto 0.6095$

Repeat!



If you repeat this many times,

3



the minimum
lly fast.

DeepLearning.AI

Gradient Descent

Function: $f(x, y)$ Goal: find minimum of $f(x, y)$

Step 1:

Define a learning rate α

Choose a starting point (x_0, y_0)

Step 2:

Update: $\begin{bmatrix} x_k \\ y_k \end{bmatrix} = \begin{bmatrix} x_{k-1} \\ y_{k-1} \end{bmatrix} - \alpha \nabla f(x_{k-1}, y_{k-1})$

Step 3:

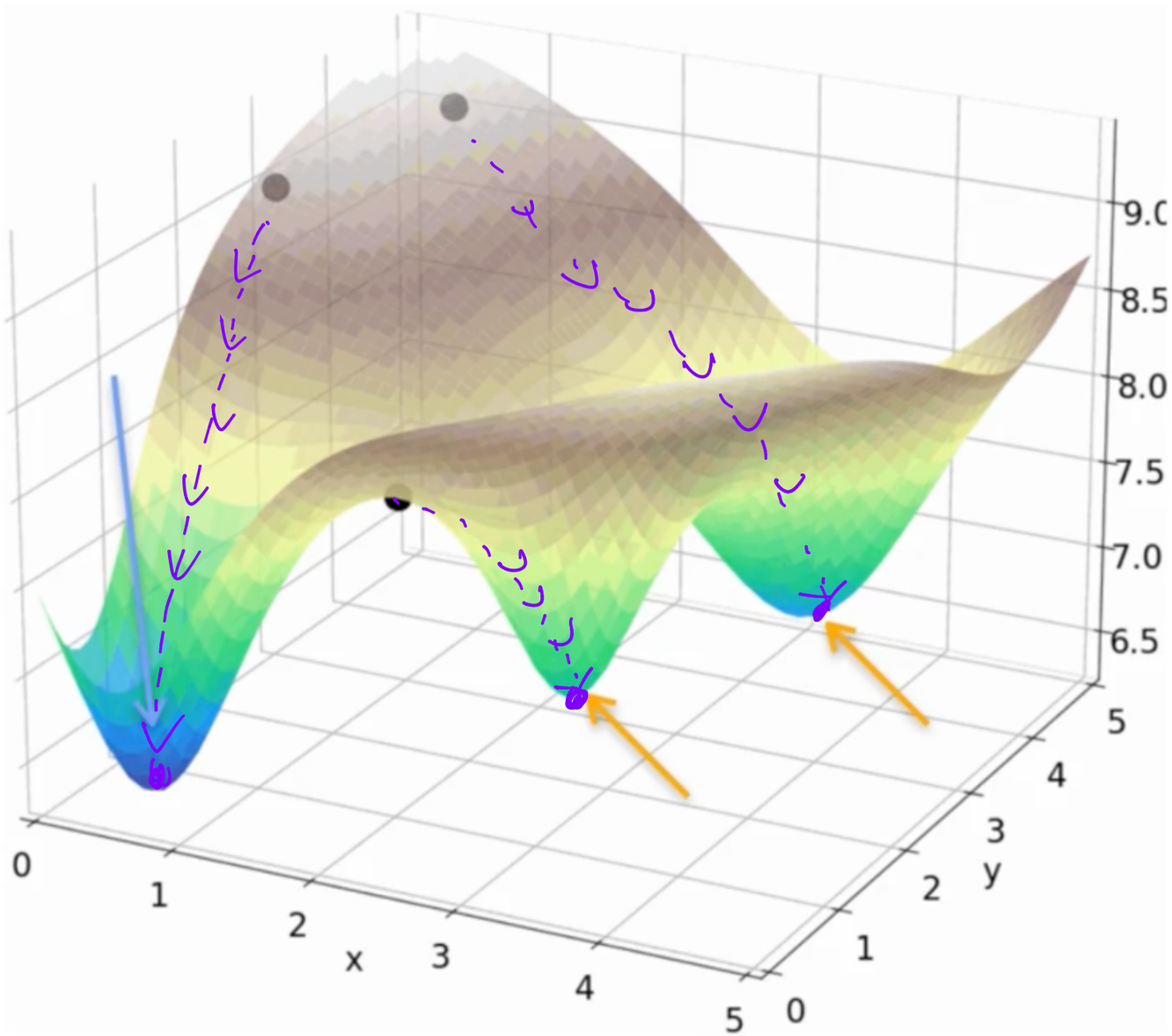
Repeat Step 2 until you are close enough to the true minimum (x^*, y^*)

DeepLearning.AI

repeat step two until you're

→ $\begin{bmatrix} x_k \\ y_k \end{bmatrix} = \begin{bmatrix} x_{k-1} \\ y_{k-1} \end{bmatrix} - \alpha \nabla f(x_{k-1}, y_{k-1})$ ↘ update Rule.

Gradient Descent With Local Minima



The way to overcome this,

DeepLearning.AI

Optimization using Gradient Descent - Least Squares

Linear Regression: Gradient Descent

Goal: Minimize sum of squares cost

$$\nabla E = [28m + 12b - 42,\ 6b + 12m - 20]$$

$\nabla E = [28m + 12b - 42,\ 6b + 12m - 20]$

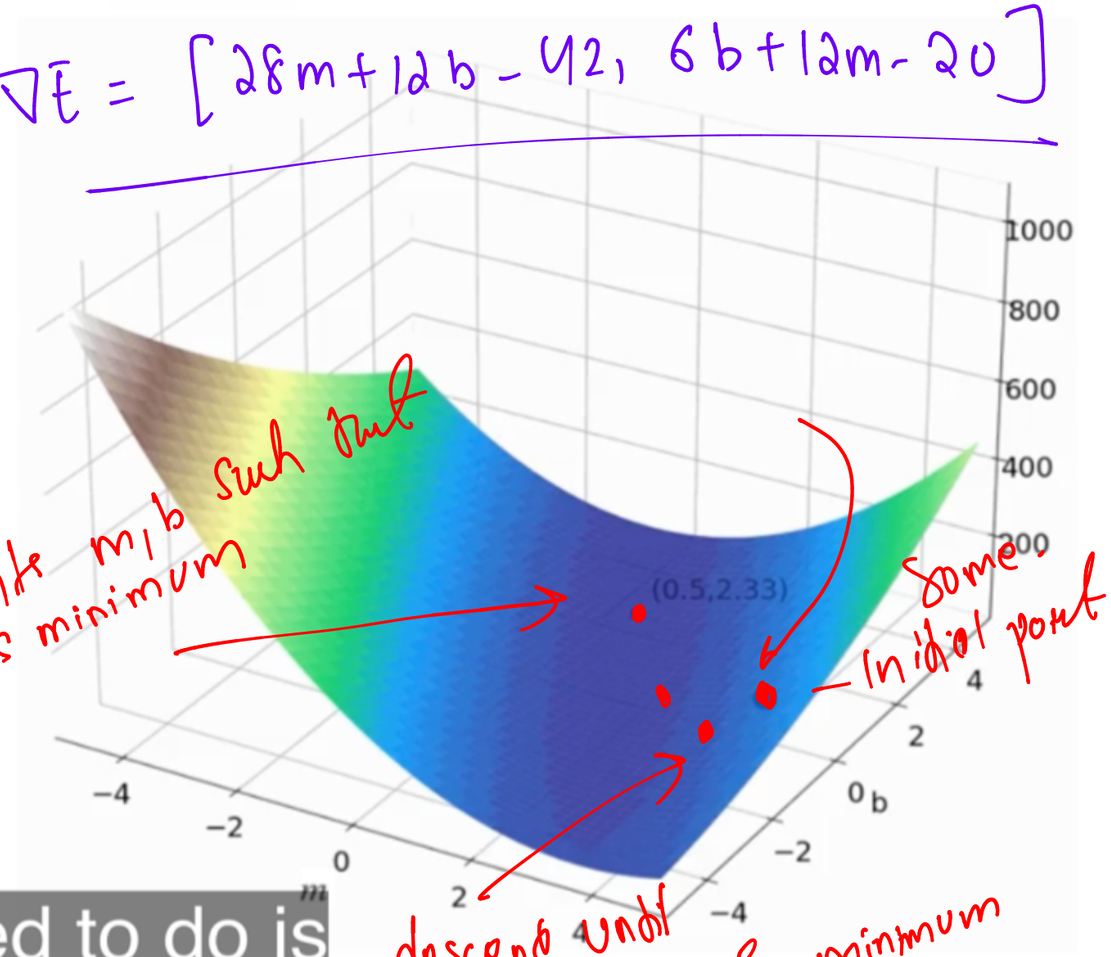
m = ?
b = ?

Steps:

Start with (m_0, b_0)

iterate:

$(m_{k+1}, b_{k+1}) = (m_k, b_k) - \alpha \nabla E(m_k, b_k)$



the point m, b such that the cost is minimum

Some initial point

descend until you find the minimum

So all you need to do is find the optimal M and B.

Linear Regression: Gradient Descent

Goal: Minimize sum of squares cost

$$\nabla E = \left[28m + 12b - 42,\ 6b + 12m - 20\right]$$

$$m = \quad b = \quad ?$$

The points m,b such that the cost is minimum

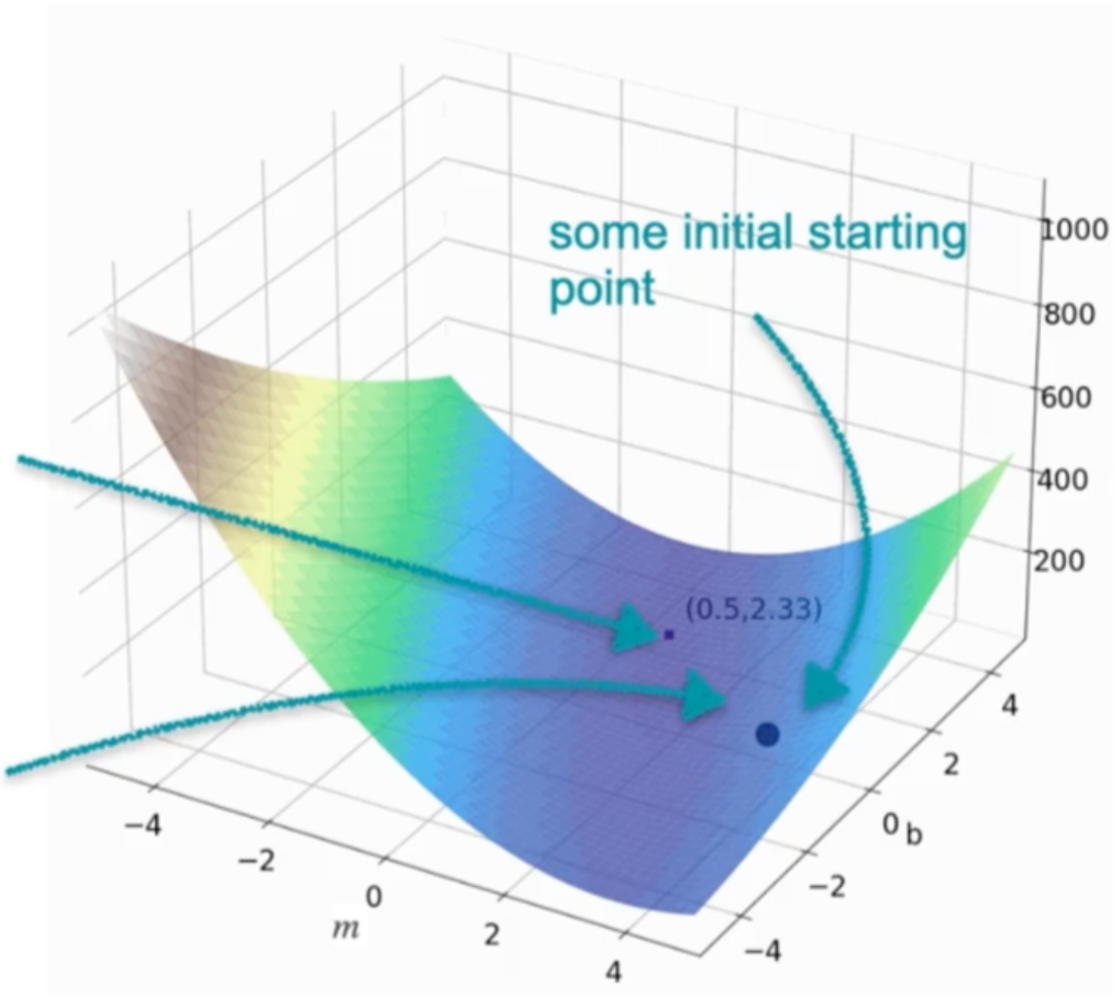
Steps:

Start with (m_0, b_0)

Iterate

$$(m_{k+1}, b_{k+1}) = (m_k, b_k) - \alpha \nabla E(m_k, b_k)$$

descend until you find the minimum

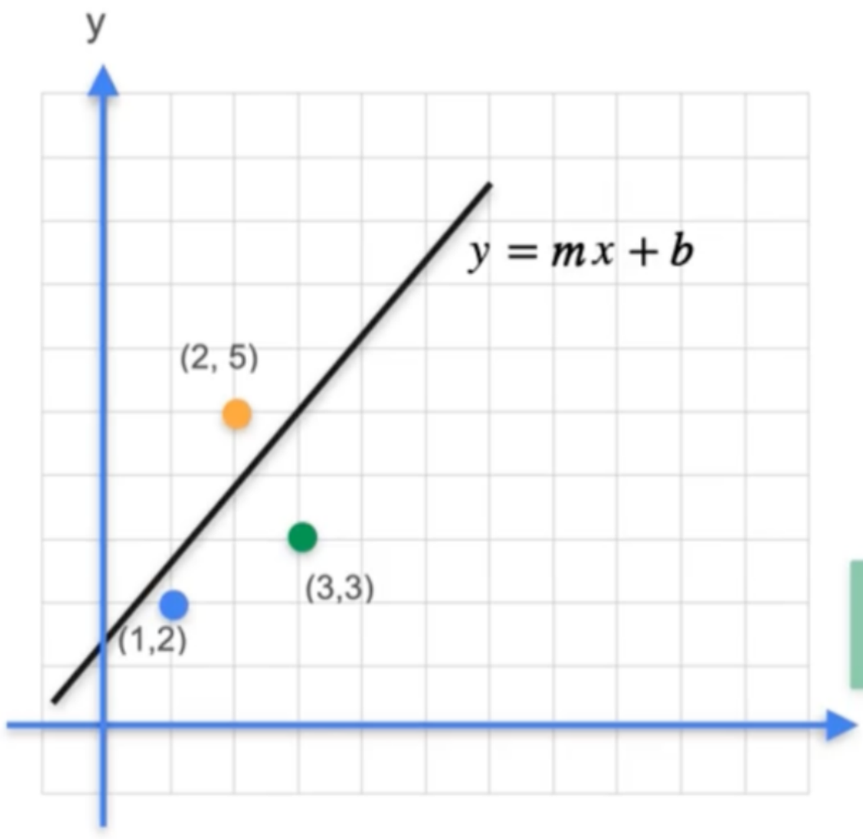


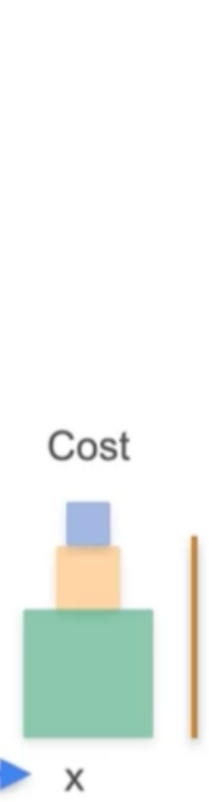
that gradient on top, evaluated at MK, BK.

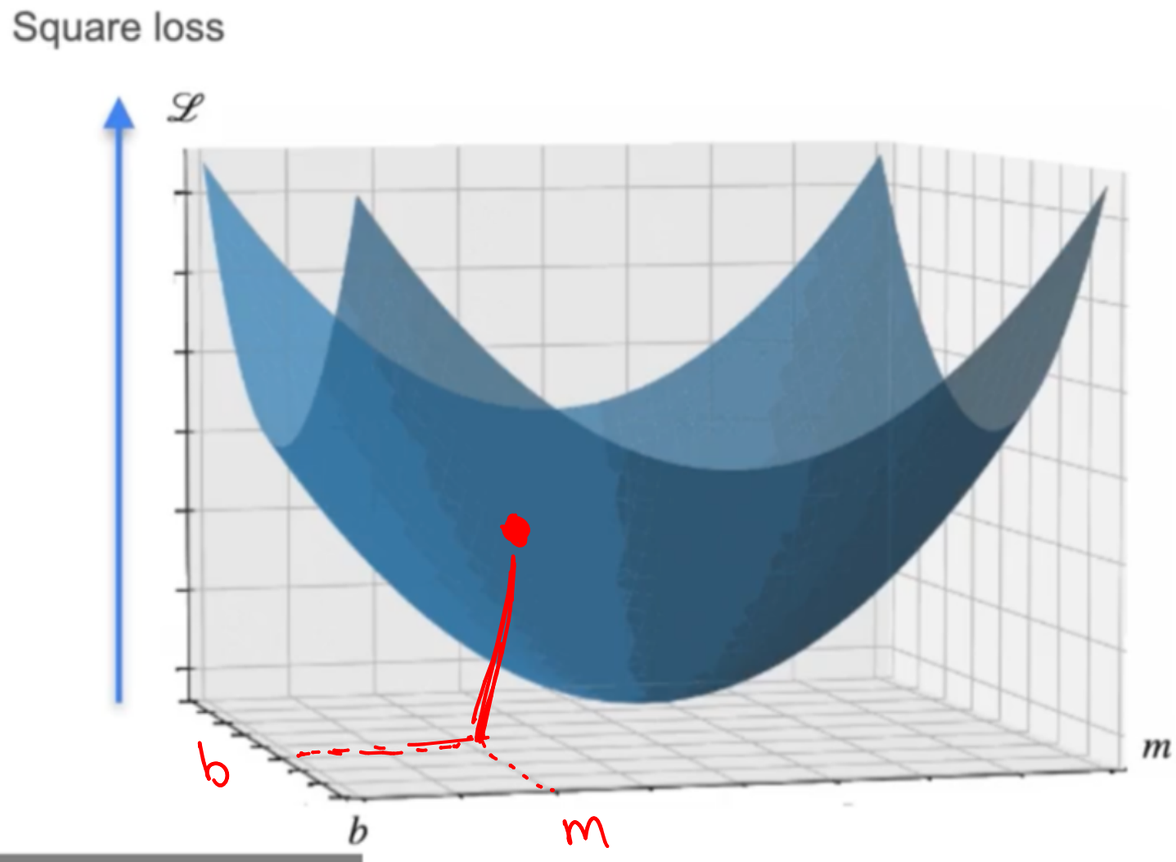
DeepLearning.AI

Optimization using Gradient Descent - Least Squares with Multiple Observations

Gradient Descent





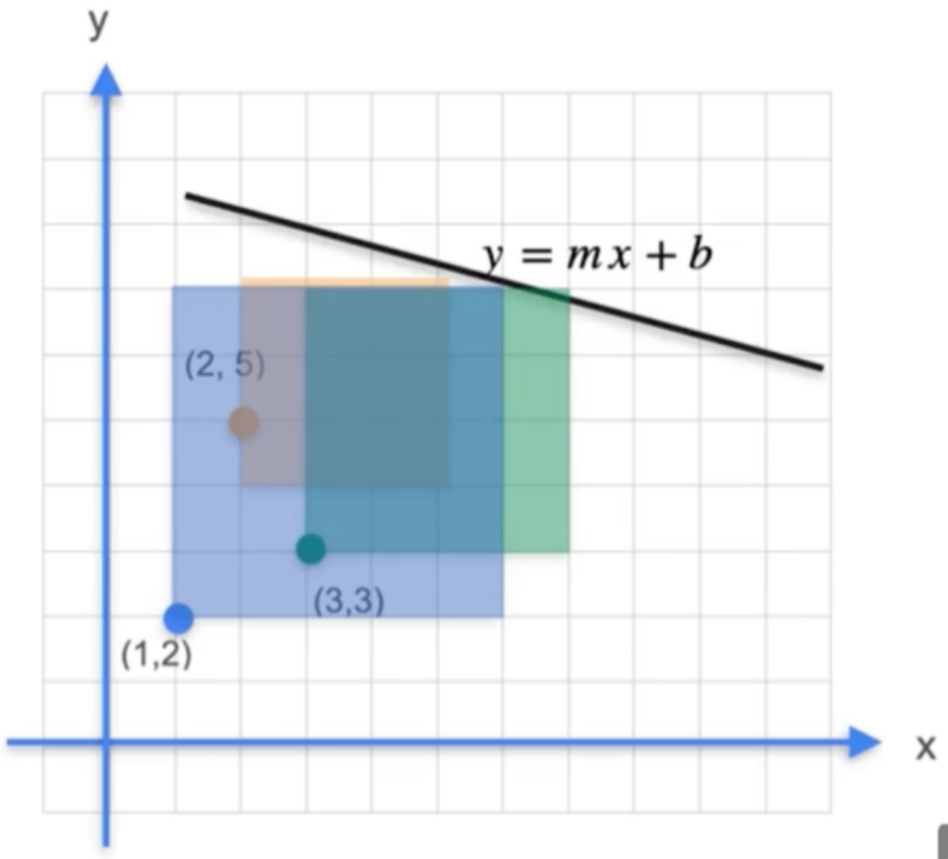


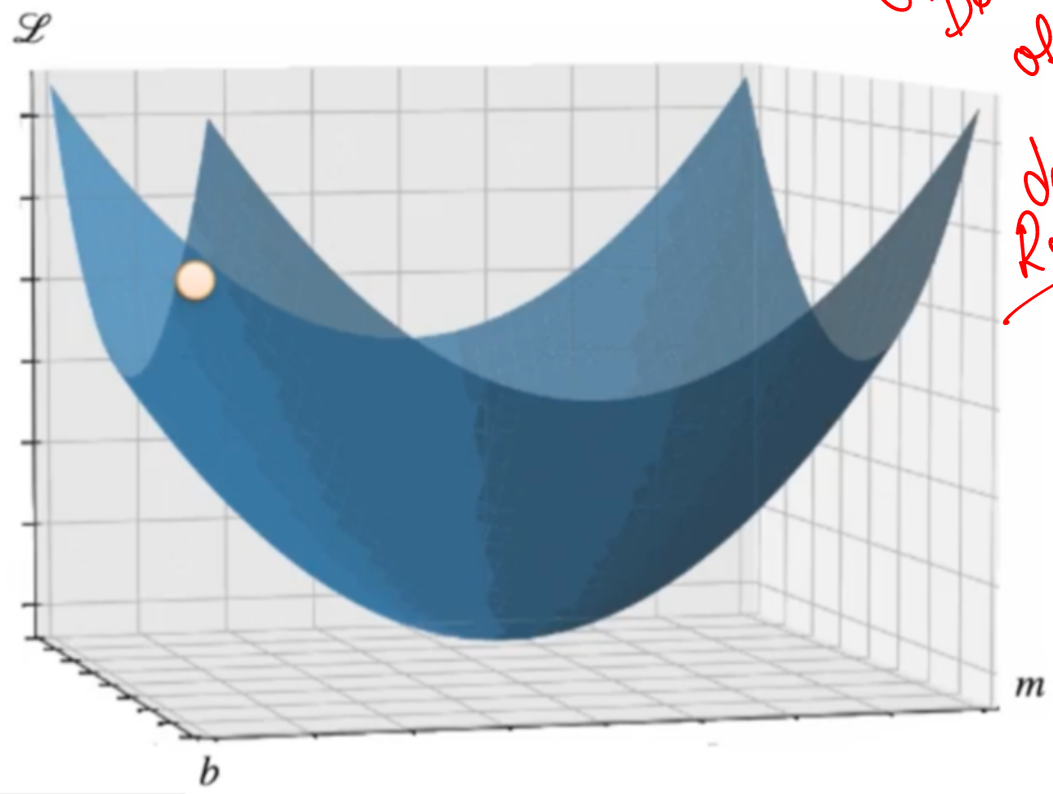
the slope and the
y-intercept m and b.

DeepLearning.AI

Gradient Descent

Change 'm' and 'b' to fit the best-fit line that how we solve the gradient Descent of Linear Regression

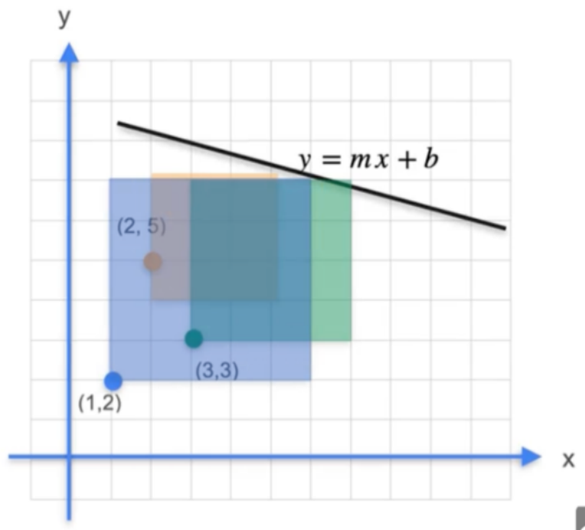


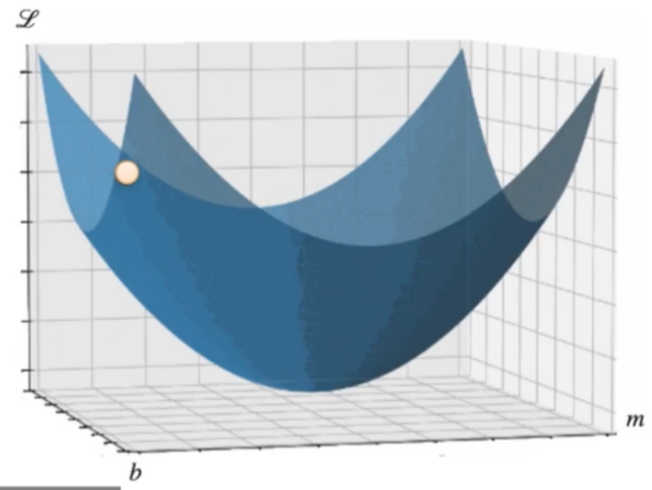


over here in the error function plot.

DeepLearning.AI

Gradient Descent





over here in the
error function plot

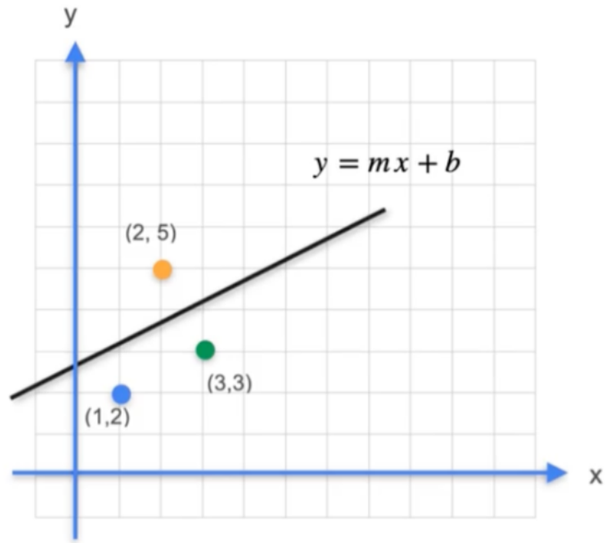
DeepLearning.AI

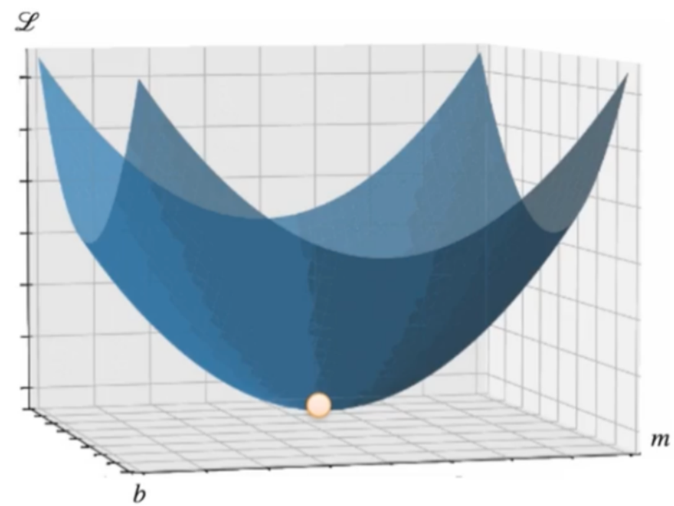
Solve dR using Gradient Descent.

Example: Predict Sales in terms of TV budget.

$y = mx + b$. → use multiple observations

Gradient Descent





That's how you solve

DeepLearning.AI

Another Example

TV budget	Sales
230.1	22.1
44.5	10.4
17.2	9.3

Multiple observations

Goal: Predict sales in terms of TV budget

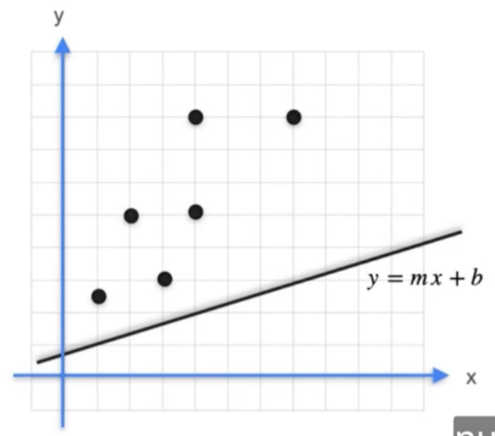
Tool: Linear regression

$$y = mx + b$$

gradient descent with a lot of observations

DeepLearning.AI

Gradient Descent



number of sales in

for one variable -

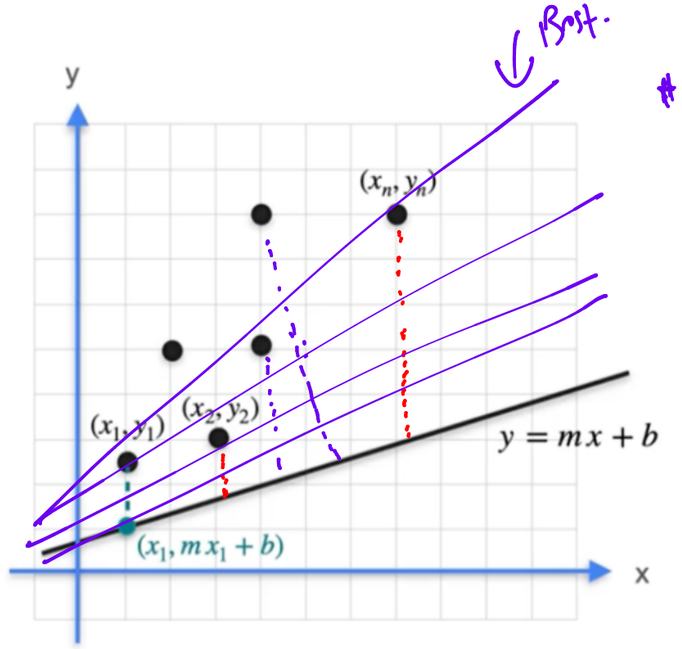
loss → $mx_1 + b - y_1 = 0$

$(mx_1 + b - y_1)^2$

then,

if we generalize -

Gradient Descent

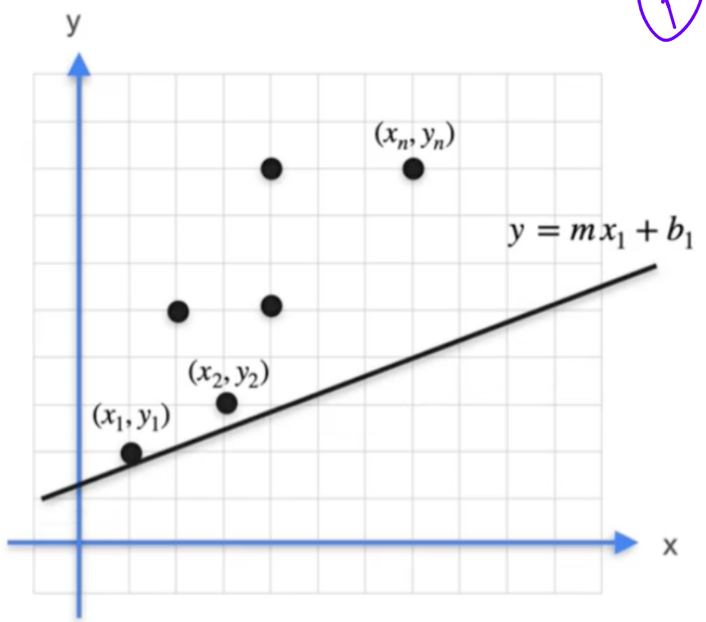


$$L(m, b) = \frac{1}{2n}\left[(mx_1 + b - y_1)^2 + \cdots + (mx_n + b - y_n)^2\right]$$

$$\begin{bmatrix} m_0 \\ b_0 \end{bmatrix} \Rightarrow \begin{bmatrix} m_1 \\ b_1 \end{bmatrix} \Rightarrow \begin{bmatrix} m_0 \\ b_0 \end{bmatrix} - \alpha \nabla d_L(m_0, b_0)$$

Well, the point on the line

Gradient Descent



(1)

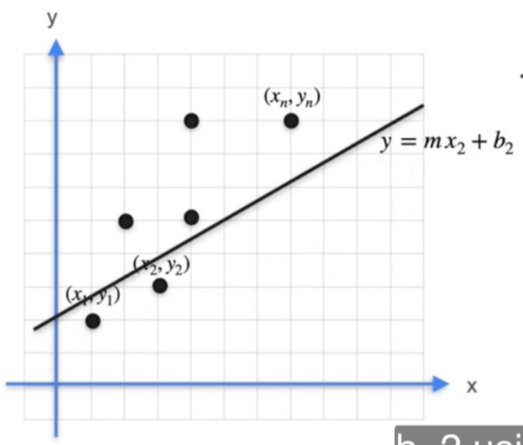
$$\mathscr{L}(m,b) = \frac{1}{2n}\left[(mx_1 + b - y_1)^2 + \cdots + (mx_n + b - y_n)^2\right]$$

$$\begin{bmatrix} m_0 \\ b_0 \end{bmatrix} \rightarrow \begin{bmatrix} m_1 \\ b_1 \end{bmatrix} = \begin{bmatrix} m_0 \\ b_0 \end{bmatrix} - \alpha \nabla \mathscr{L}_1(m_0, b_0)$$

→ Iterate N times until the $\mathscr{L}(m,b)$ is the minimum most minimized form.

Gradient Descent

(2)



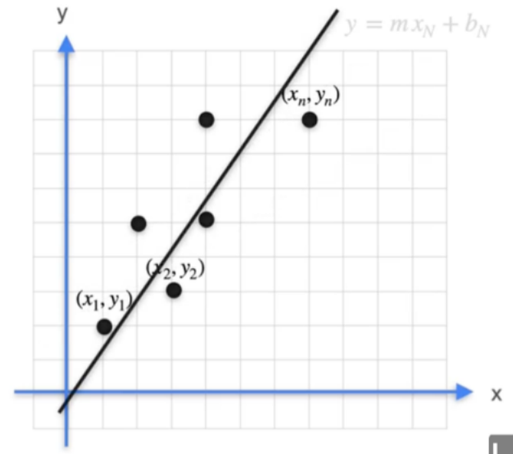
$$\mathscr{L}(m,b) = \frac{1}{2n}\left[(mx_1 + b - y_1)^2 + \cdots + (mx_n + b - y_n)^2\right]$$

$$\begin{bmatrix} m_1 \\ b_1 \end{bmatrix} \rightarrow \begin{bmatrix} m_2 \\ b_2 \end{bmatrix} = \begin{bmatrix} m_1 \\ b_1 \end{bmatrix} - \alpha \nabla \mathscr{L}_1(m_1, b_1)$$

b_2 using the same thing except plugging m_1 b_1,

Gradient Descent

(3)

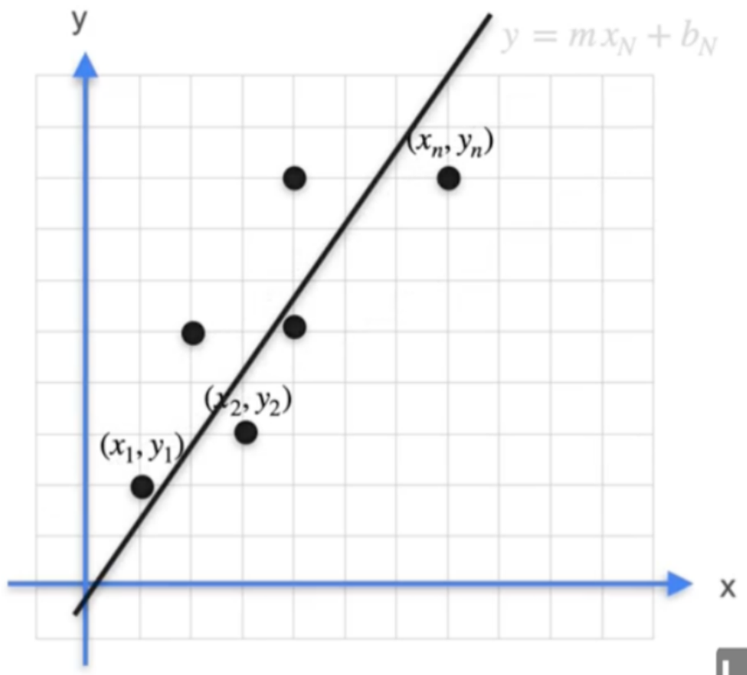


$$\mathscr{L}(m,b) = \frac{1}{2n}\left[(mx_1 + b - y_1)^2 + \cdots + (mx_n + b - y_n)^2\right]$$

$$\begin{bmatrix} m_N \\ b_N \end{bmatrix} \rightarrow \begin{bmatrix} m_N \\ b_N \end{bmatrix} = \begin{bmatrix} m_{N-1} \\ b_{N-1} \end{bmatrix} - \alpha \nabla \mathscr{L}_1(m_{N-1}, b_{N-1})$$

Let's say we do it capital N times

Gradient Descent



$$\mathscr{L}(m,b) = \frac{1}{2n}\left[(mx_1 + b - y_1)^2 + \cdots + (mx_n + b - y_n)^2\right]$$

$$\begin{bmatrix} m_N \\ b_N \end{bmatrix} \rightarrow \begin{bmatrix} m_N \\ b_N \end{bmatrix} = \begin{bmatrix} m_{N-1} \\ b_{N-1} \end{bmatrix} - \alpha \nabla \mathscr{L}_1(m_{N-1}, b_{N-1})$$

Let's say we do it capital N times

DeepLearning.AI

$\mathscr{L}(m,b) = \frac{1}{2n}\left[(mx_1 + b - y_1)^2 + \cdots + (mx_n + b - y_n)^2\right]$

$\rightarrow \begin{bmatrix} m_N \\ b_N \end{bmatrix} \rightarrow \begin{bmatrix} m_N \\ b_N \end{bmatrix} =$

$\begin{bmatrix} m_{N-1} \\ b_{N-1} \end{bmatrix} - \alpha \nabla \mathscr{L}_1(m_{N-1}, b_{N-1})$

$\begin{bmatrix} m_N \\ b_N \end{bmatrix} \rightarrow \begin{bmatrix} m_N \\ b_N \end{bmatrix} = \begin{bmatrix} m_{N-1} \\ b_{N-1} \end{bmatrix} - \alpha \nabla \mathscr{L}_1(m_{N-1}, b_{N-1})$ #.

Q. $f(x,y) = x^2 + y^2 - 6x$ and $\nabla f(x,y) = \begin{bmatrix} 2x - 6 \\ 2y \end{bmatrix}$ at $x_0(0,1)$.

$$x_{k+1} = x_k - \alpha \nabla f(x_k)$$

$$x_0 = (0,1)$$

$$\nabla f(0,1) \Rightarrow \begin{bmatrix} 2(0) - 6, & 2(1) \end{bmatrix}$$

$$\Rightarrow [-6, 2]$$

Update $\Rightarrow$ $x_1 = x_0 - \alpha \nabla f(x_0)$

$$\Rightarrow (0,1) - 0.1 (-6,2)$$

$$\Rightarrow (+0.6, 1 - 0.2)$$

$$\Rightarrow (0.6, 0.8)$$

To find the minimum value of $f(x,y) = 2x^2 + 3y^2 - 2xy - 10x$

Q. find Critical points: $f(x,y) = 2x^3 + 3y^2 - 2xy - 10x$. ⟶ eqn (I)

$$\frac{\partial f}{\partial x} = 4x - 2y - 10 = 0 \quad \longrightarrow \text{eqn (II)}$$

$$\frac{\partial f}{\partial y} \Rightarrow 6y - 2x = 0 \quad \text{— eqn (III)}.$$

$\Rightarrow$ $6y = 2x$ and $4x = 10 + 2y$

$x = 3y$

$4(3y) = 10 + 2y$

$12y - 2y = 10$

$y = 1$

Second Partial Derivative Test

$f_{xx} = 4$ → eqn (II)

$f_{yy} \Rightarrow 6$ ← eqn (III)

$f_{xy} \Rightarrow -2$ → eqn (I)

Step 3: Minimum Value

$$f(3,1) = 2(3)^2 + 3(1)^2 - 2(3)(1) - 10(3)$$

$$\Rightarrow 18 + 3 - 2 - 30$$

$$\boxed{f(3,1) \Rightarrow -15}$$

Conclusion: Calculus for Data Science and Machine Learning offered by Coursera.