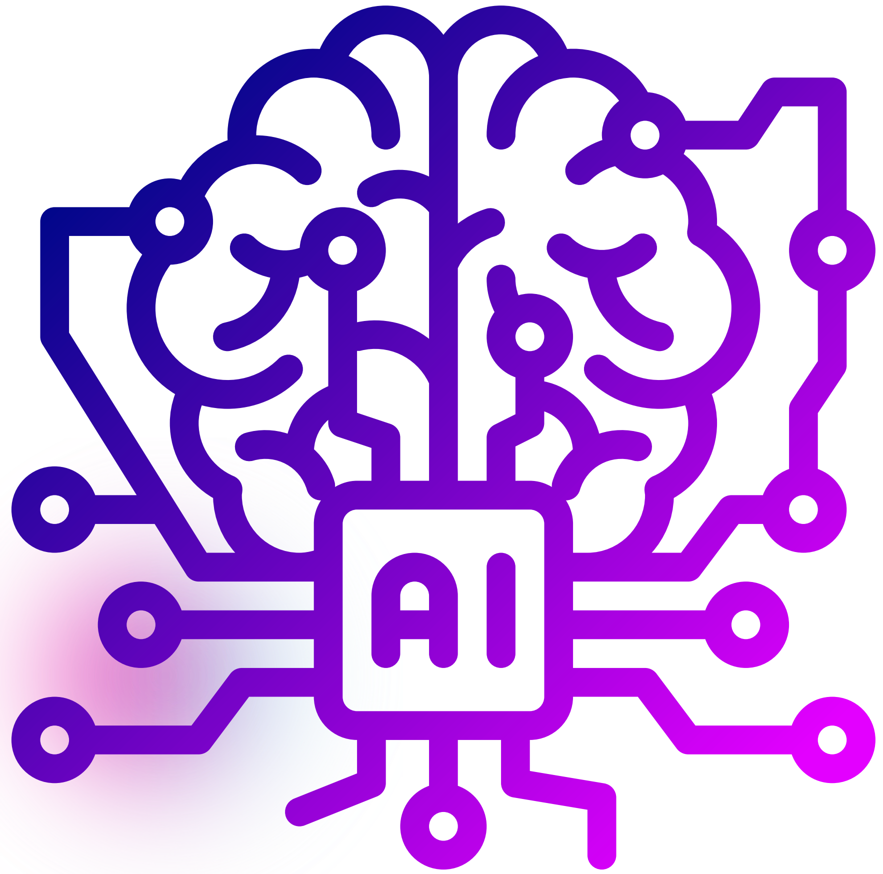
From Fundamentals to
Building Your Own Intelligent System

AI & MACHINE LEARNING BOOTCAMP 2025

Dr. Fazlul Hasan Siddiqui
Proferssor, Dept. of CSE, DUET
Chairman, ICT Cell; Director, IICT, DUET

Dr. Sabah Binte Noor
Associate Proferssor,
Dept. of CSE, DUET



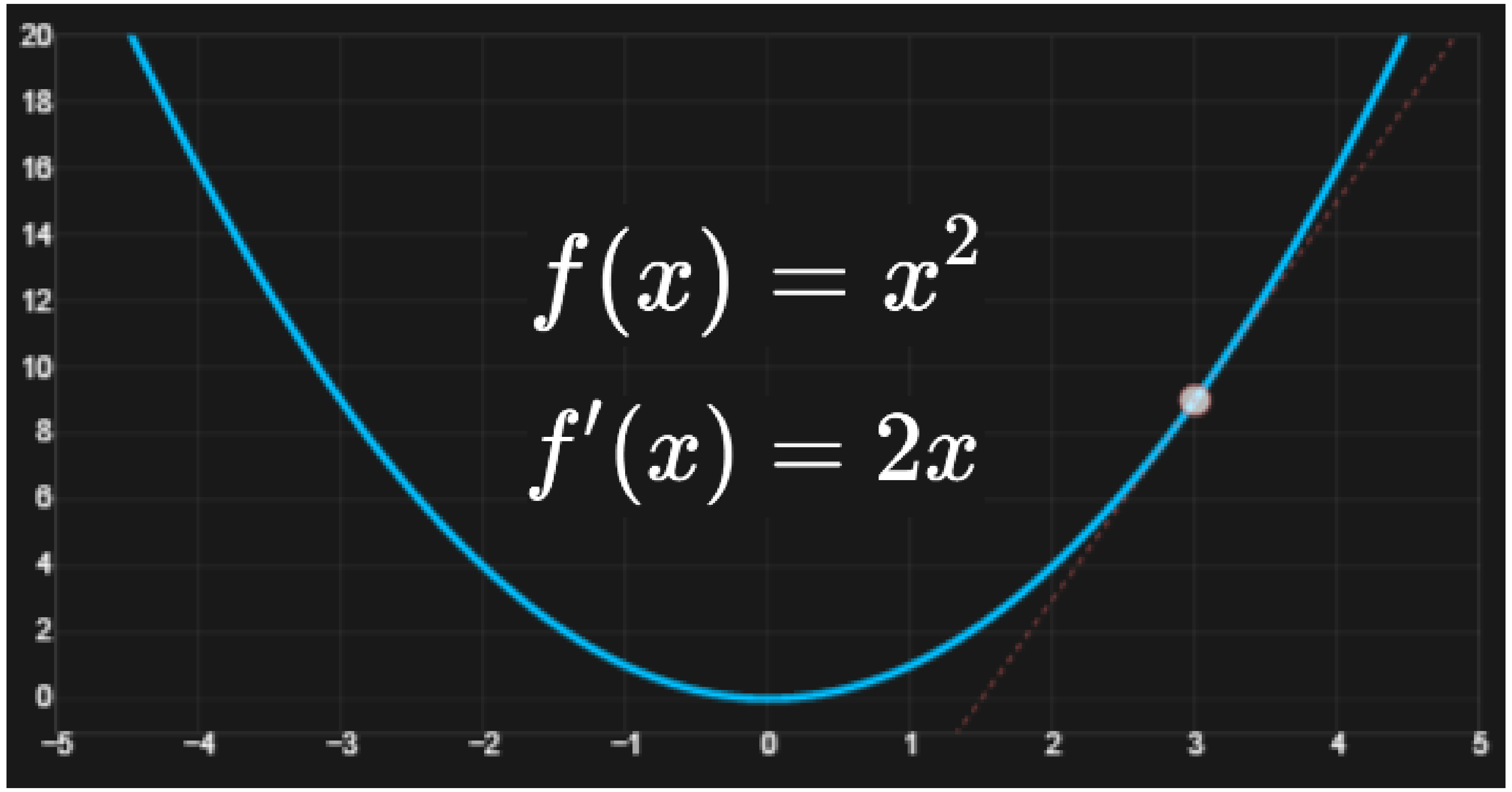
MODEL LEARNING

GRADIENT DESCENT

Credit: Zachary Huang

Find the minimum of a valley

- You're lost, you need to get to the bottom of the valley but you can only see your own two feet.
- You don't need to see the whole map. You just do gradient descent:
  - 1. Feel the slope right where you're standing.
  - 2. Take a small step in the steepest downhill direction.
  - 3. Repeat until you reach the bottom where the ground is flat.



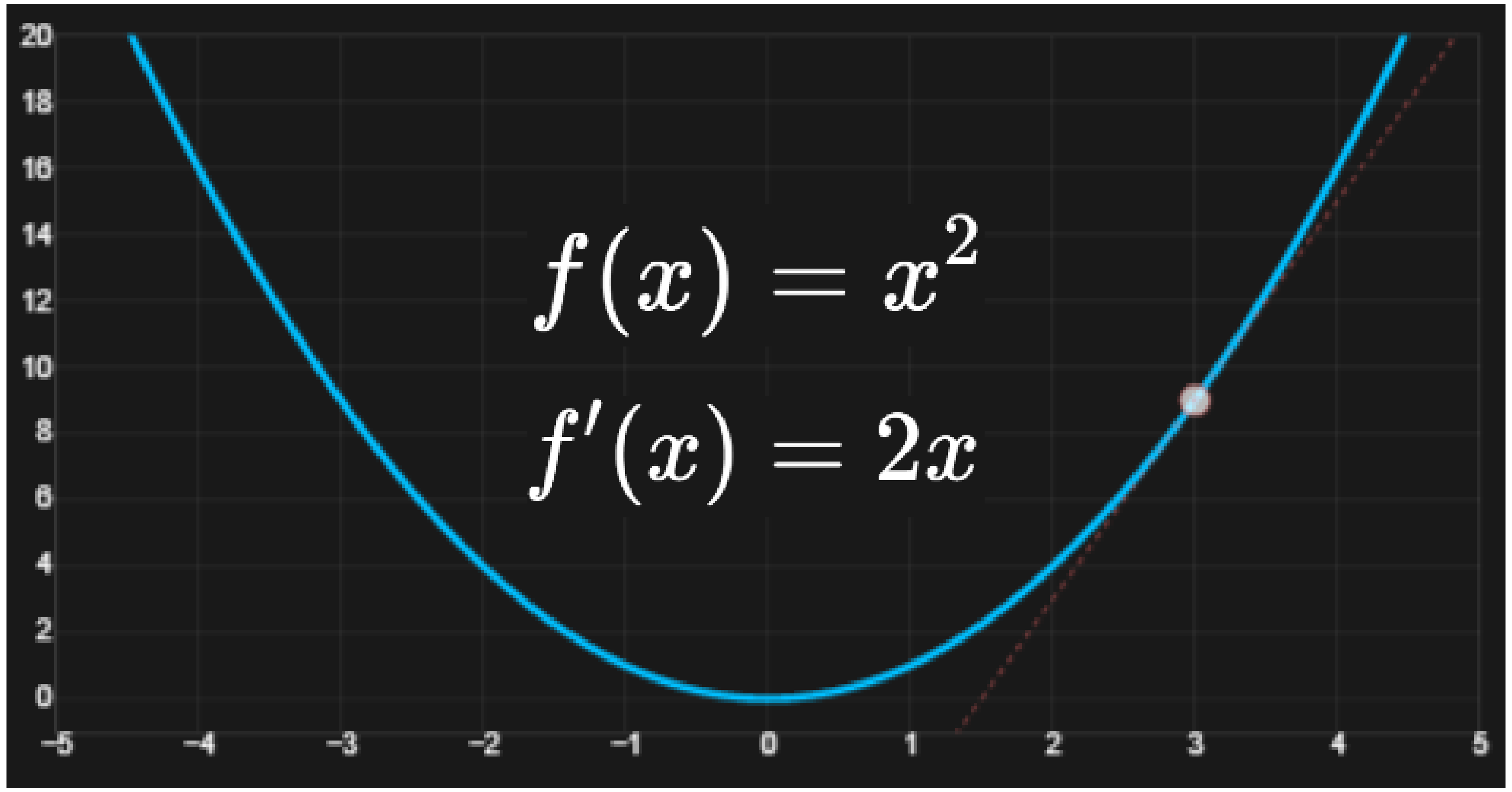
- At $x = 3$, slope is $f'(3) = 2 \times 3 = 6$.
- Positive slope means "downhill" is to the left.

- At $x = -2$, slope is $f'(-2) = 2 \times (-2) = -4$.
- Negative slope means "downhill" is to the right.

- At $x = 0$, slope is $f'(0) = 0$.
- The ground is flat. You've arrived!

Find the minimum of a valley

- You're lost, you need to get to the bottom of the valley but you can only see your own two feet.
- You don't need to see the whole map. You just do gradient descent:
  - 1. Feel the slope right where you're standing.
  - 2. Take a small step in the steepest downhill direction.
  - 3. Repeat until you reach the bottom where the ground is flat.

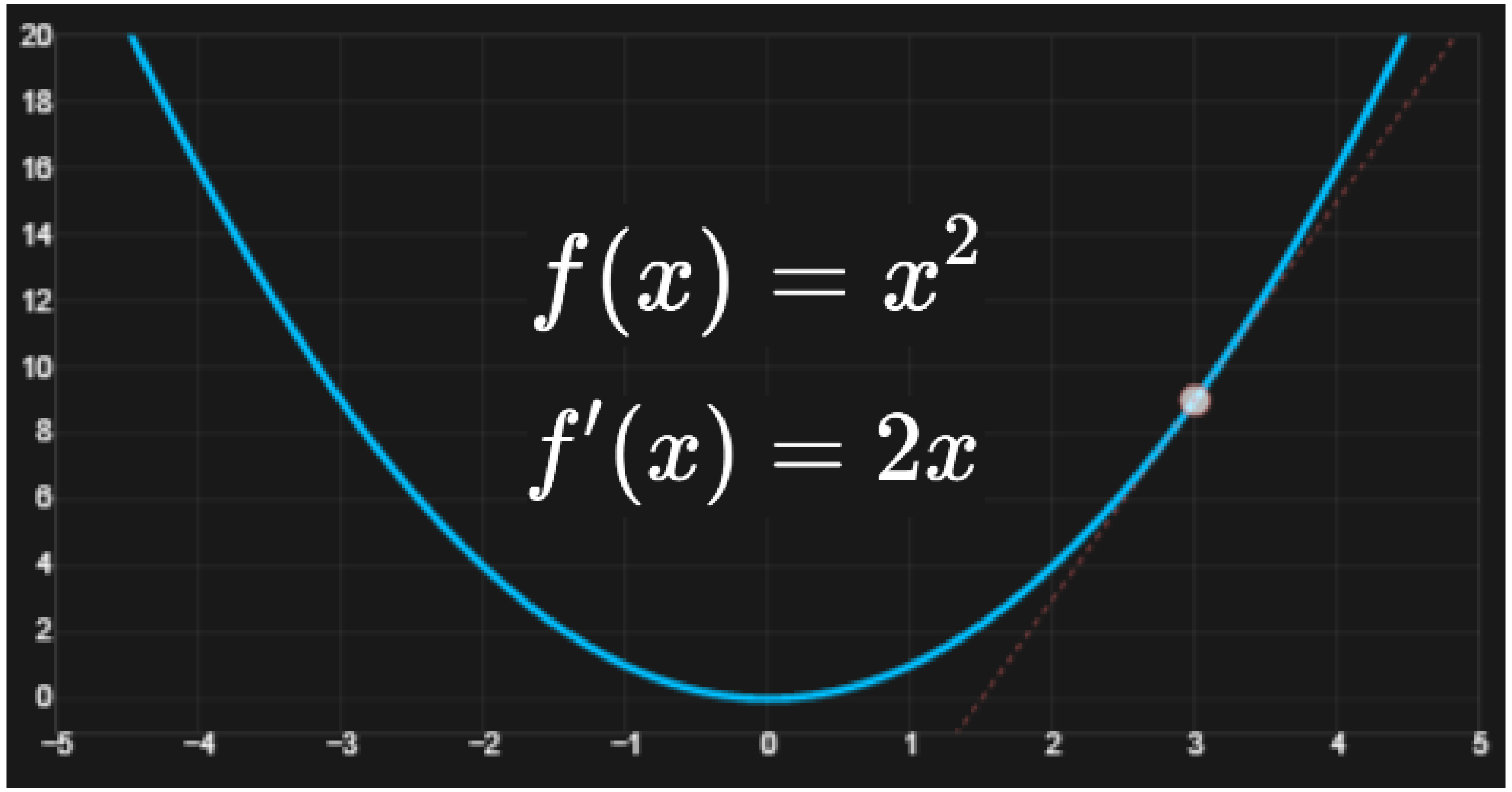


INPUT: function f(x)
OUTPUT: argmin_x f(x)

FOR 100 iterations:
  gradient = f'(x)
  x = x − η × gradient
RETURN x

Find the minimum of a valley

- You're lost, you need to get to the bottom of the valley but you can only see your own two feet.
- You don't need to see the whole map. You just do gradient descent:
  - 1. Feel the slope right where you're standing.
  - 2. Take a small step in the steepest downhill direction.
  - 3. Repeat until you reach the bottom where the ground is flat.

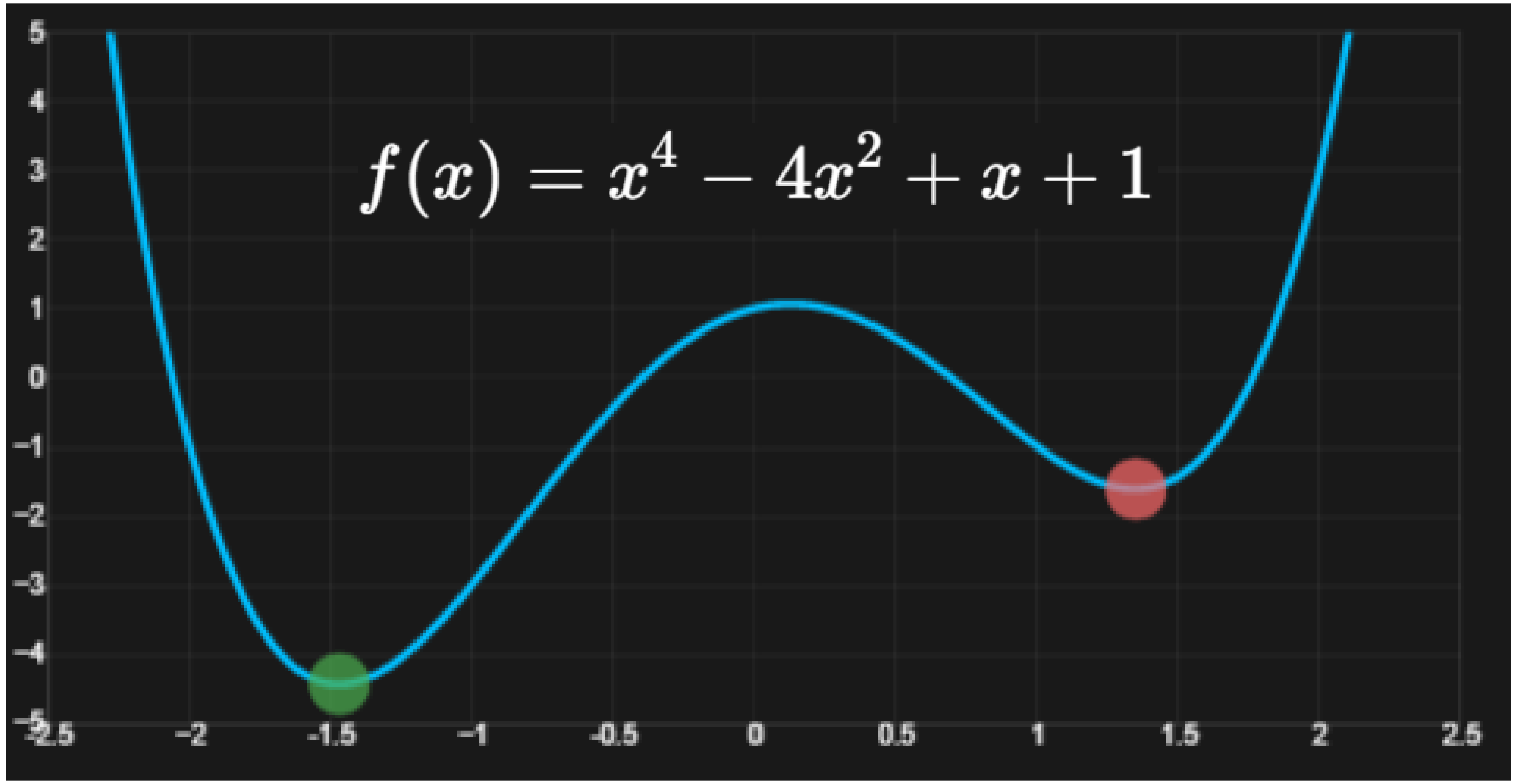


Start: $x = 3$ | Learning Rate: $\eta = 0.1$,
Update Rule: $x - 0.1 \times (2x)$

Iteration	Current x	$f(x)=x^2$	Gradient $f'(x)=2x$	New $x \leftarrow x - 0.1 \times (2x)$
0	3.000	9.000	6.000	3 - 0.1×6 = 2.400
1	2.400	5.760	4.800	2.4 - 0.1×4.8 = 1.920
2	1.920	3.686	3.840	1.92 - 0.1×3.84 = 1.536
3	1.536	2.359	3.072	1.536 - 0.1×3.072 = 1.229
...	...	...	...	...
10	0.322	0.104	0.644	0.322 - 0.1×0.644 = 0.258

Find the minimum of a valley - not simple

- You're lost, you need to get to the bottom of the valley but you can only see your own two feet.
- You don't need to see the whole map. You just do gradient descent:
    - 1. Feel the slope right where you're standing.
    - 2. Take a small step in the steepest downhill direction.
    - 3. Repeat until you reach the bottom where the ground is flat.



- A local minimum
- The true global minimum
- Same algorithm, different starting point, wildly different results
- NN with millions of parameters have an error landscape with billions of traps – GPT
- Most "local minima" are pretty good solutions
- Truly bad traps are incredibly rare

Find the steepest downhill path - Multi Var - Partial Derivative

The fundamental technique for training a neural network. **We treat every single weight as its own "knob"**. We calculate its partial derivative (its individual contribution to the total error), and then we nudge it slightly in the right direction

Function: $f(x_1, x_2) = x_1^2 + 2x_2^2$

Start at random point: (x1, x2) = (3, 2)

Learning rate: η = 0.1

Initial Error: $f(3,2) = 3^2 + 2(2^2) = \mathbf{17}$

In the first step, the gradient tells it to move

- 0.6 in the `x1` direction and
- 0.8 in the `x2` direction
- slashing the error in half

```
INPUT: function f(x1,x2)
FOR 100 iterations:
  grad_x1 = ∂f/∂x1
  grad_x2 = ∂f/∂x2

  x1 = x1 - η × grad_x1
  x2 = x2 - η × grad_x2
RETURN (x1,x2)
```

Iter	x_1	x_2	$f(x_1,x_2)$	$\partial f/\partial x_1=2x_1$	$\partial f/\partial x_2=4x_2$	New $(x_1,x_2) \leftarrow (x_1-0.1\times 2x_1,\ x_2-0.1\times 4x_2)$
0	3.000	2.000	17.000	6.000	8.000	(2.40, 1.20) ← (3-0.1×6, 2-0.1×8)
1	2.400	1.200	8.640	4.800	4.800	(1.92, 0.72) ← (2.4-0.1×4.8, 1.2-0.1×4.8)
2	1.920	0.720	4.722	3.840	2.880	(1.54, 0.43) ← (1.92-0.1×3.84, 0.72-0.1×2.88)
3	1.536	0.432	2.734	3.072	1.728	(1.23, 0.26) ← (1.54-0.1×3.07, 0.43-0.1×1.73)
...	...	...	...	...	...	...
10	0.403	0.028	0.164	0.806	0.112	(0.32, 0.017) ← (0.40-0.1×0.81, 0.028-0.1×0.11)

The Chain Rule

- In a deep neural network – A weight in the first layer doesn't directly touch the final error.
- Its influence travels through a long, complex chain.
- How do you calculate the "blame" for a single knob when its effect is buried 20 layers deep? – The Chain Rule
- To find out why, you trace the problem backward
    a. The presentation was bad...
    b. ...because the slides were confusing. (50% blame)
    c. ...because the data analysis was flawed. (80% blame)
    d. ...because the data collection was sloppy. (90% blame)
        - To find out how much the initial data collector is responsible for the final failed presentation...
        - just multiply the blame at each step – 90% × 80% × 50% = 36%

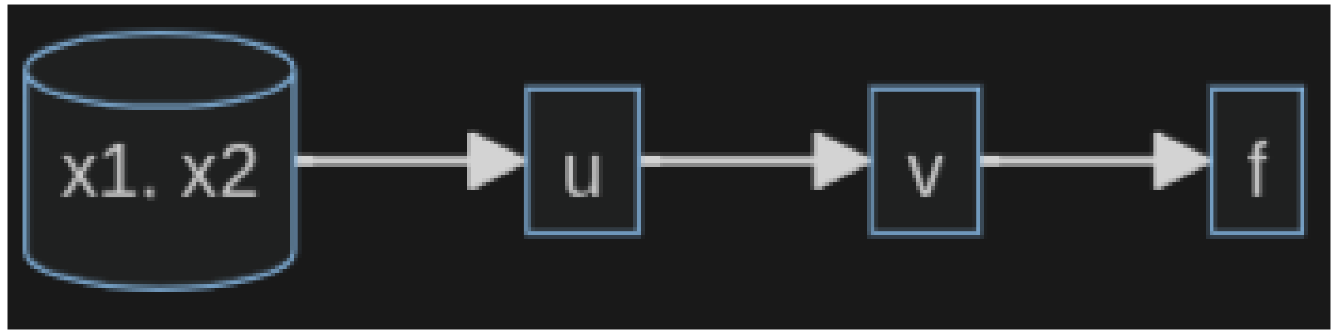
$f(x_1, x_2) = ((2x_1 + x_2)^2 + 3x_2^2)^3$, where $u = 2x_1 + x_2, v = u^2 + 3x_2^2, f = v^3$

Step	Question	Function	Derivative
1	How much does f blame v?	$f = v^3$	$3v^2$
2	How much does v blame u?	$v = u^2 + 3x_2^2$	$2u$
3	How much does u blame x_1?	$u = 2x_1 + x_2$	2

Total Blame: $\frac{\partial f}{\partial x_1} = \frac{\partial f}{\partial v} \times \frac{\partial v}{\partial u} \times \frac{\partial u}{\partial x_1} = 3v^2 \times 2u \times 2 = \mathbf{12uv^2}$

$$f(x_1, x_2) = ((2x_1 + x_2)^2 + 3x_2^2)^3$$

- First, we calculate $u = 2x_1 + x_2$
- Then, $v = u^2 + 3x_2^2$
- Finally, $f = v^3$



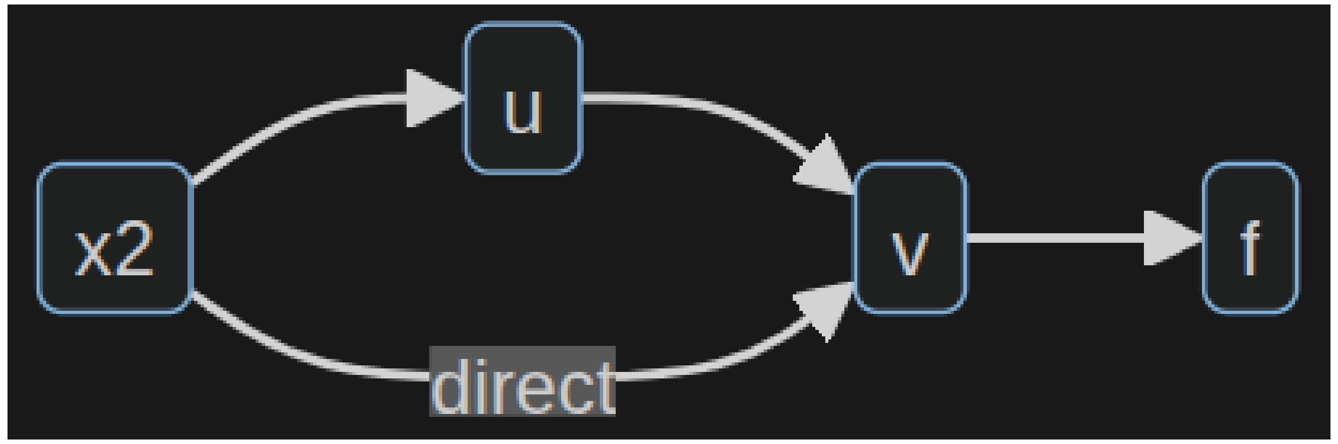
The Chain Rule

$f(x_1, x_2) = ((2x_1 + x_2)^2 + 3x_2^2)^3$, where $u = 2x_1 + x_2, v = u^2 + 3x_2^2, f = v^3$

$\frac{\partial f}{\partial x_2} = \frac{\partial f}{\partial v} \times \frac{\partial v}{\partial x_2}$ we know $\frac{\partial f}{\partial v} = 3v^2$. We need $\frac{\partial v}{\partial x_2}$ from both paths.

Path	Calculation	Result
Indirect: $x_2 \to u \to v$	$\frac{\partial v}{\partial u} \times \frac{\partial u}{\partial x_2} = 2u \times 1$	$2u$
Direct: $x_2 \to v$	$\frac{\partial}{\partial x_2}(3x_2^2)$	$6x_2$
Total $\frac{\partial v}{\partial x_2}$	Sum both paths	$2u + 6x_2$

$$\frac{\partial f}{\partial x_2} = 3v^2 \times (2u + 6x_2)$$



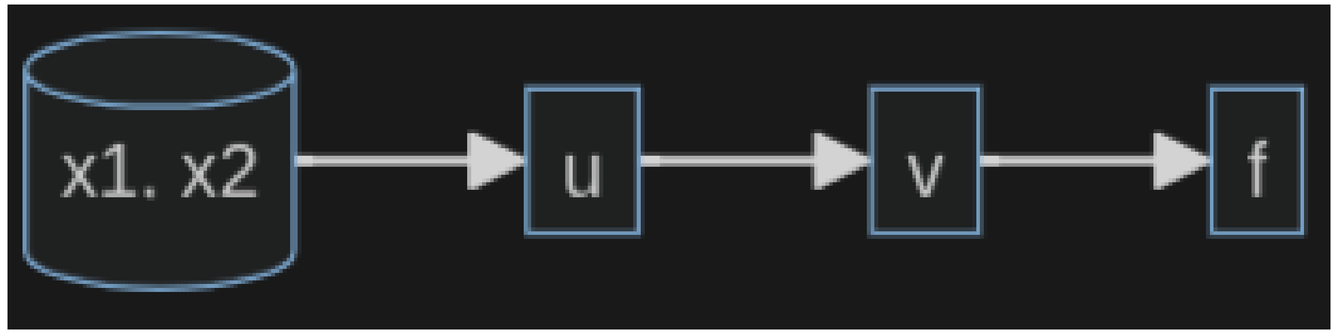
$f(x_1, x_2) = ((2x_1 + x_2)^2 + 3x_2^2)^3$, where $u = 2x_1 + x_2, v = u^2 + 3x_2^2, f = v^3$

Step	Question	Function	Derivative
1	How much does f blame v?	$f = v^3$	$3v^2$
2	How much does v blame u?	$v = u^2 + 3x_2^2$	$2u$
3	How much does u blame x_1?	$u = 2x_1 + x_2$	2

Total Blame: $\frac{\partial f}{\partial x_1} = \frac{\partial f}{\partial v} \times \frac{\partial v}{\partial u} \times \frac{\partial u}{\partial x_1} = 3v^2 \times 2u \times 2 = \mathbf{12uv^2}$

$$f(x_1, x_2) = ((2x_1 + x_2)^2 + 3x_2^2)^3$$

- First, we calculate $u = 2x_1 + x_2$
- Then, $v = u^2 + 3x_2^2$
- Finally, $f = v^3$



Summary

$f(x_1, x_2) = ((2x_1 + x_2)^2 + 3x_2^2)^3$, where $u = 2x_1 + x_2, v = u^2 + 3x_2^2, f = v^3$

$\frac{\partial f}{\partial x_2} = \frac{\partial f}{\partial v} \times \frac{\partial v}{\partial x_2}$ we know $\frac{\partial f}{\partial v} = 3v^2$. We need $\frac{\partial v}{\partial x_2}$ from both paths.

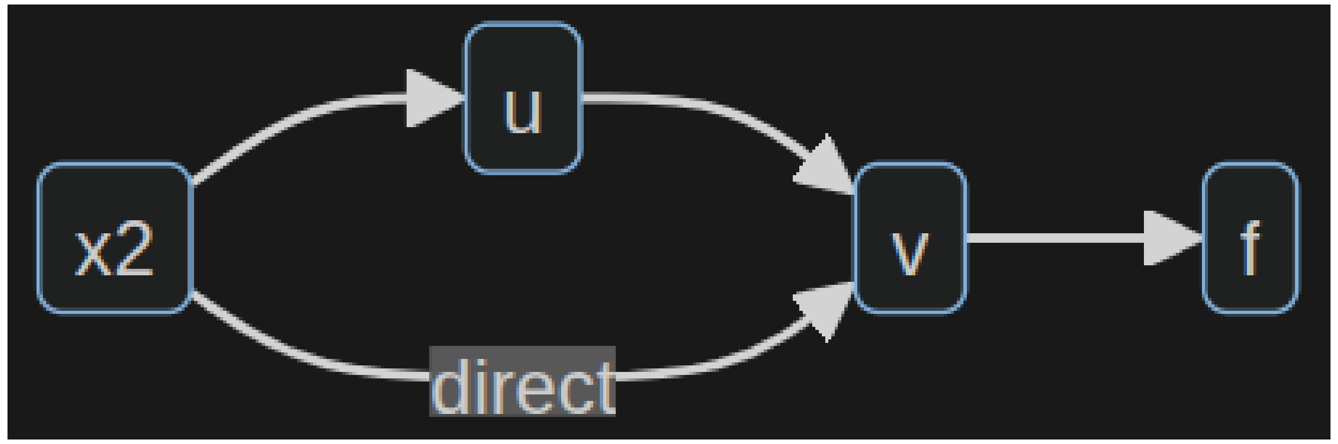
Path	Calculation	Result
Indirect: $x_2 \to u \to v$	$\frac{\partial v}{\partial u} \times \frac{\partial u}{\partial x_2} = 2u \times 1$	$2u$
Direct: $x_2 \to v$	$\frac{\partial}{\partial x_2}(3x_2^2)$	$6x_2$
Total $\frac{\partial v}{\partial x_2}$	Sum both paths	$2u + 6x_2$

$f(x_1, x_2) = ((2x_1 + x_2)^2 + 3x_2^2)^3$, where $u = 2x_1 + x_2, v = u^2 + 3x_2^2, f = v^3$

Step	Question	Function	Derivative
1	How much does f blame v?	$f = v^3$	$3v^2$
2	How much does v blame u?	$v = u^2 + 3x_2^2$	$2u$
3	How much does u blame x_1?	$u = 2x_1 + x_2$	2

Total Blame: $\frac{\partial f}{\partial x_1} = \frac{\partial f}{\partial v} \times \frac{\partial v}{\partial u} \times \frac{\partial u}{\partial x_1} = 3v^2 \times 2u \times 2 = \mathbf{12uv^2}$

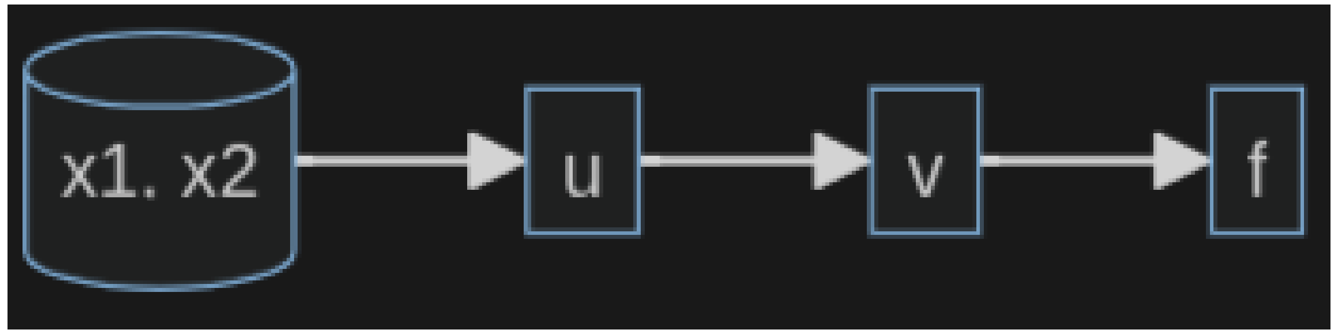
$$\frac{\partial f}{\partial x_2} = 3v^2 \times (2u + 6x_2)$$




```
FOR 100 iterations:
  # Calculate current values
  u = 2*x1 + 3*x2
  v = x1 + x2**2

  # Calculate the blame for each variable
  grad_x1 = 12 * u * v**2
  grad_x2 = (3 * v**2) * (2*u + 6*x2)

  # Nudge each variable in the right direction
  x1 = x1 - η * grad_x1
  x2 = x2 - η * grad_x2
RETURN (x1,x2)
```



Summary

1. We know how to go downhill (**Gradient Descent**).
2. We know how to find the slope for each knob (**Partial Derivatives**).
3. And now, we can trace blame through a long chain (**The Chain Rule**).