- A04
- ITAI 2376
- Jeffery Dirden
- June 24th, 2025

Imagine you're standing on top of a giant ice cream hill , and there's a big pile of candy at the bottom. Your goal? Slide all the way down to grab that candy!

But here's the twist… the hill is super foggy , so you can't see where the bottom is. You have to feel your way down.

That's where gradient descent comes in! It's like using your feet to feel which way is downhill and then taking small steps in that direction. Every step gets you closer to the candy. That's how computers learn too — they take little steps to get better at something, like recognizing pictures, voices, or playing games!

In this notebook, you'll see:

A purple line that shows how steep the hill is (that's called a loss function — it tells the computer how "wrong" its guess is).

A little red dot that moves down the hill one step at a time — that's like our computer learning and improving its guess.

So the first plot is showing us the shape of the hill. It's really high at the edges (when the guess is bad) and lowest in the middle (when the guess is good).

The second plot shows how our red dot — that's us! — keeps walking down, down, down, until we're almost at the best possible answer.

This is the same method used in AI when it learns to do cool things like:

Recommend videos

Translate languages

And even drive cars !

Let's follow along and watch how our guesses get better with every little step we take. "

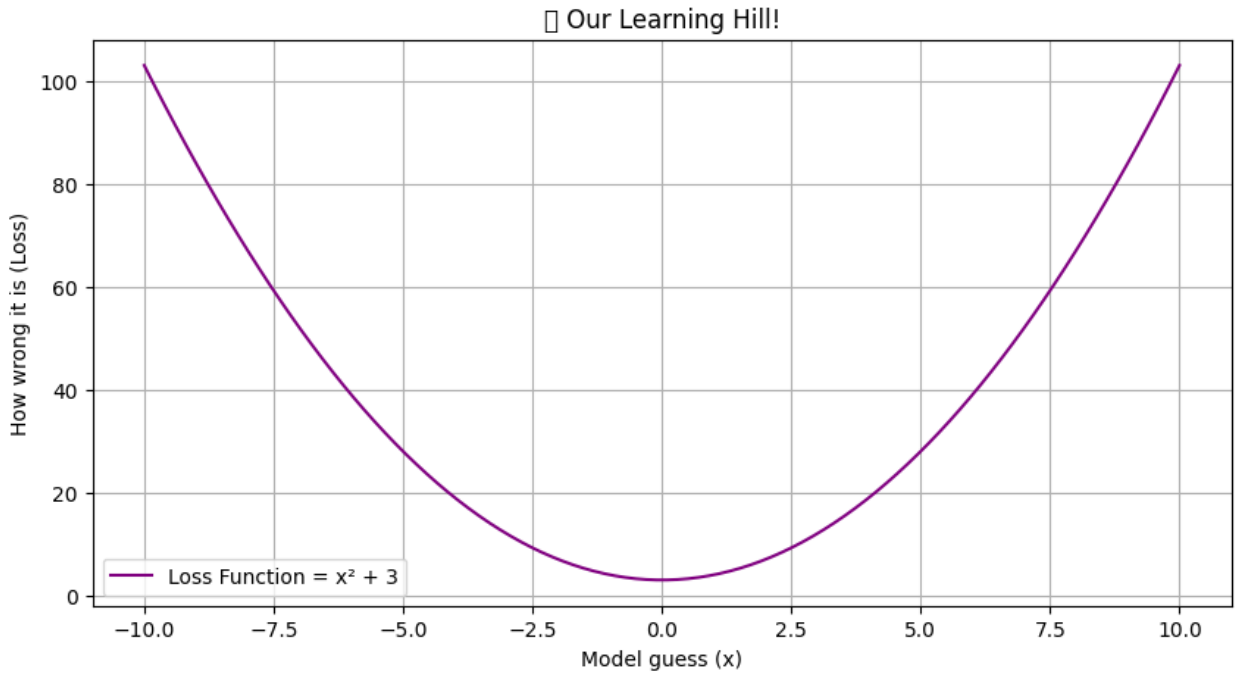
```
# A04_JefferyDirden_ITAI2376
# Gradient Descent Explained to an 11 Years Old 

# ========================
#  Welcome Message
# ========================

from IPython.display import display, Markdown

display(Markdown("""
#  Gradient Descent: Like Rolling Down a Hill
Imagine you're on top of a big hill  and you want to get to the
```

```
bottom where the candy is .
But it's really foggy, and you can't see far.

So what do you do?

You take small steps in the direction that feels like going downhill.
That's what gradient descent is! It's how computers learn the best
answer by taking small steps down a big hill of "wrongness."

Let's see it in action! 
"""))

# =========================
#  Simulate the Hill (Loss Function)
# =========================

import numpy as np
import matplotlib.pyplot as plt

# This is our "hill"
def loss_function(x):
    return x**2 + 3

# The gradient (slope) of the hill
def gradient(x):
    return 2 * x

# Show the hill
x = np.linspace(-10, 10, 100)
y = loss_function(x)

plt.figure(figsize=(10, 5))
plt.plot(x, y, label='Loss Function = x² + 3', color='purple')
plt.title(" Our Learning Hill!")
plt.xlabel("Model guess (x)")
plt.ylabel("How wrong it is (Loss)")
plt.grid(True)
plt.legend()
plt.show()

# =========================
#  Take Steps Down the Hill
# =========================

display(Markdown("##  Let's Take Steps Down the Hill!"))

# Start at the top
x_current = 9
learning_rate = 0.2
steps = [x_current]
```

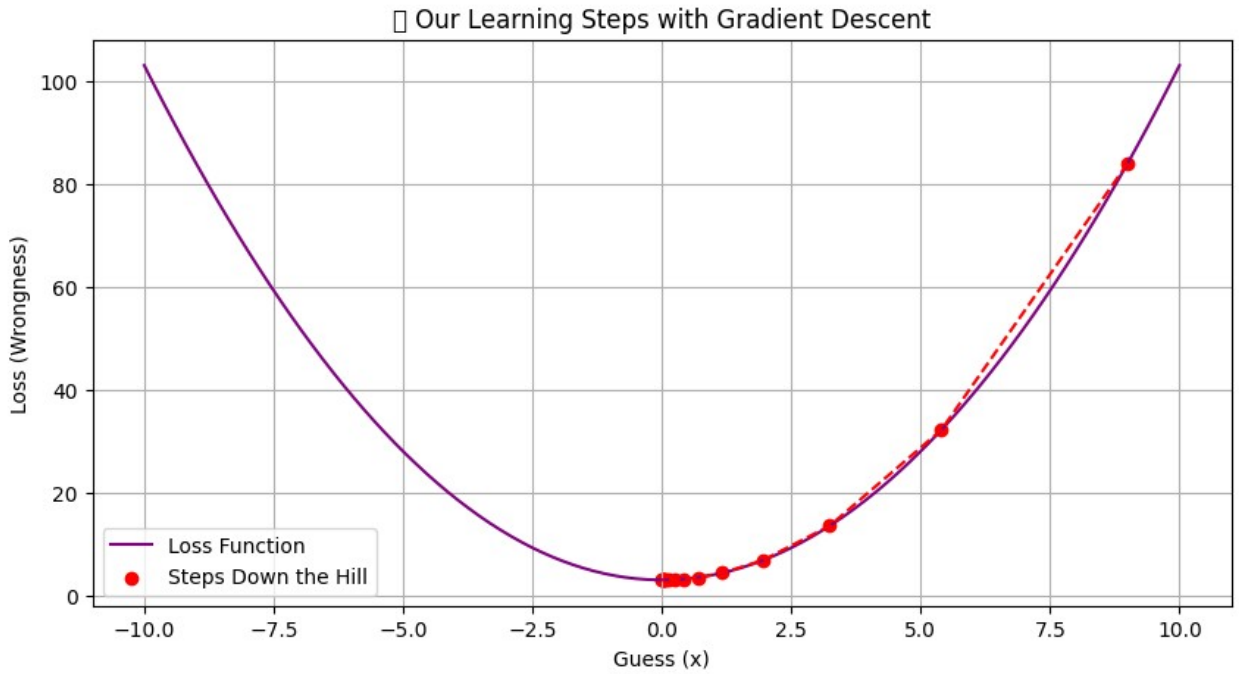
```
# Take 15 steps
for _ in range(15):
    grad = gradient(x_current)
    x_current = x_current - learning_rate * grad
    steps.append(x_current)

# Plot steps
y_steps = loss_function(np.array(steps))

plt.figure(figsize=(10, 5))
plt.plot(x, y, label='Loss Function', color='purple')
plt.scatter(steps, y_steps, color='red', label='Steps Down the Hill')
plt.plot(steps, y_steps, linestyle='--', color='red')
plt.title("□ Our Learning Steps with Gradient Descent")
plt.xlabel("Guess (x)")
plt.ylabel("Loss (Wrongness)")
plt.legend()
plt.grid(True)
plt.show()

# =========================
# □ Summary for Kids
# =========================

display(Markdown("""
## □ What Did We Learn?
- Gradient Descent is like **walking down a foggy hill** □.
- You take small steps in the **steepest downhill direction** □.
- Each step makes your guess a little **less wrong** ✔.
- After many steps, you get close to the **best answer** (the bottom
of the hill) □.

This is how computers learn! They don’t just guess once — they
practice over and over until they’re pretty good at stuff.

### □ Fun Fact:
This is how AI learns to do things like recognize your face, play
video games, or drive cars!

### □ Works Cited:
- Chollet, François. *Deep Learning with Python*. Manning
Publications, 2018.
- Brownlee, Jason. “What Is Gradient Descent?”
MachineLearningMastery.com
- Google Developers. “Gradient Descent.” YouTube, 2020.
"""))
```

```
<IPython.core.display.Markdown object>
```

```
/usr/local/lib/python3.11/dist-packages/IPython/core/
pylabtools.py:151: UserWarning: Glyph 127906 (\N{ROLLER COASTER})
missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
```




```
<IPython.core.display.Markdown object>

/usr/local/lib/python3.11/dist-packages/IPython/core/
pylabtools.py:151: UserWarning: Glyph 128095 (\N{ATHLETIC SHOE})
missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
```




```
<IPython.core.display.Markdown object>
```

And if there is any confusion I created another scenario to help break it down even more.

- Jeffery Dirden
- ITAI 2376
- June 24th, 2025

"Okay, imagine this purple line is a giant candy hill , and our little gingerbread friend, Chip , is standing way up at the top. He wants to reach the bottom because that's where the candy treasure is hidden .

But here's the thing — it's super foggy! He can't see the whole hill, so he can't just jump to the bottom. Instead, he has to feel which way is downhill and take small steps in that direction.

The brown dots you see on the hill? Those are the steps Chip takes as he slides down. Each step makes him a little less wrong, a little closer to the bottom, and a little closer to the candy!

This is just like how a computer learns — by trying something, seeing how wrong it is, and then fixing it step by step."

```
# A04_SuperBrains_JefferyDirden_ITAI2376
#  Gradient Descent in Candyland - With Visuals!

from IPython.display import display, Markdown, HTML
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from matplotlib import rc
```

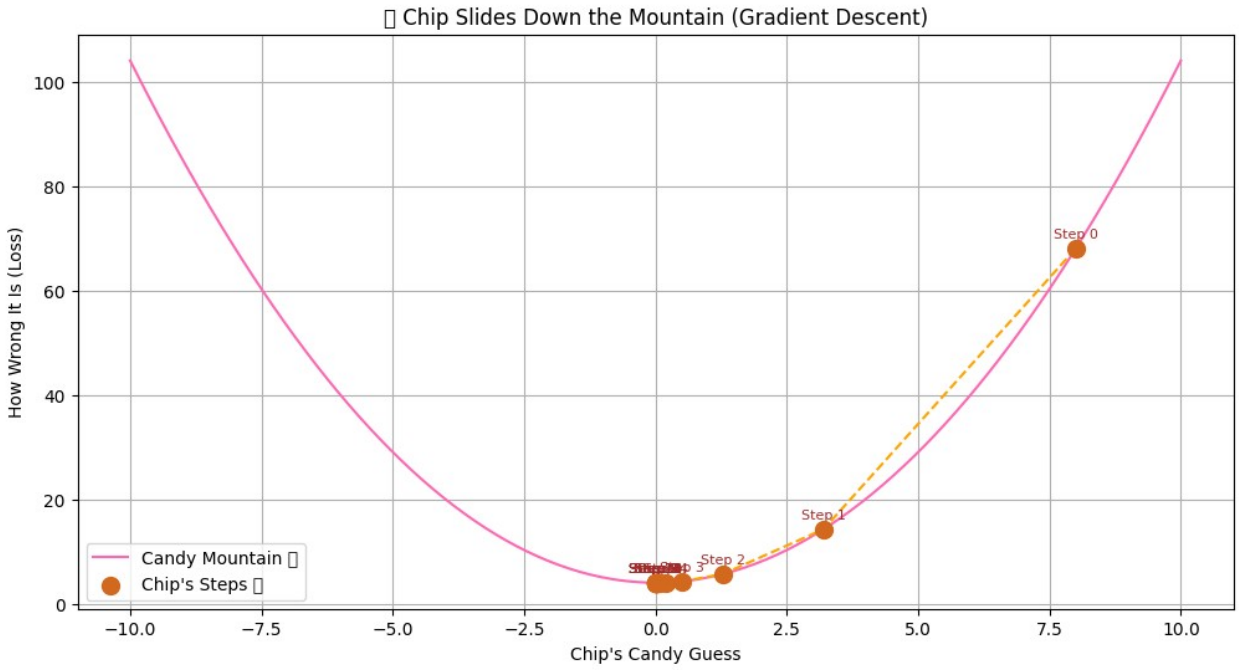
```
rc('animation', html='jshtml')

# ============================
# 🎬 Storybook Introduction
# ============================

display(Markdown("""
# 🍭 Gradient Descent in Candyland 🍭
### A Magical Journey by SuperBrains

Once upon a sugary time, in a land full of candy canes and jellybeans,
there lived a little Gingerbread AI named **Chip** 🤖.

Chip had a big dream: **find the Sweet Spot** at the bottom of Candy
Mountain 🍫, where the candy treasure lives!

But there was a problem... the whole mountain was covered in fog! 😨
Chip couldn't see the way down — he could only feel if he was going
**uphill or downhill**.

So he used a magical trick called **Gradient Descent** to help him
slide carefully toward the bottom — just like how real computers
learn!

Let's follow Chip's sweet adventure! 🚀
"""))

# ============================
# 🍫 Define the Candy Mountain
# ============================

def candy_mountain(x):
    return x**2 + 4  # A sweet, simple "hill"

def slope(x):
    return 2 * x  # The "feel" of the hill

x_vals = np.linspace(-10, 10, 300)
y_vals = candy_mountain(x_vals)

# ============================
# 🚶 Gradient Descent Steps
# ============================

chip = 8.0  # Starting at the top
learning_rate = 0.3
steps = [chip]

for _ in range(15):
    chip = chip - learning_rate * slope(chip)
    steps.append(chip)
```

```
step_y = [candy_mountain(x) for x in steps]

# ============================
#  Static Visual Example
# ============================

plt.figure(figsize=(12, 6))
plt.plot(x_vals, y_vals, color='hotpink', label="Candy Mountain ")
plt.scatter(steps, step_y, color='chocolate', s=100, zorder=5,
label="Chip's Steps ")
plt.plot(steps, step_y, linestyle='--', color='orange')

for i, (x, y) in enumerate(zip(steps, step_y)):
    plt.text(x, y+2, f"Step {i}", fontsize=8, ha='center',
color='brown')

plt.title(" Chip Slides Down the Mountain (Gradient Descent)")
plt.xlabel("Chip's Candy Guess")
plt.ylabel("How Wrong It Is (Loss)")
plt.grid(True)
plt.legend()
plt.show()

# ============================
#  Animated Descent
# ============================

fig, ax = plt.subplots(figsize=(12, 6))
ax.set_xlim(-10, 10)
ax.set_ylim(0, 110)
line, = ax.plot(x_vals, y_vals, color='deeppink', label="Candy
Mountain")
point, = ax.plot([], [], 'o', color='chocolate', markersize=12)
trail, = ax.plot([], [], '--', color='orange')
text = ax.text(0, 100, "", fontsize=14, ha='center')

# Background
ax.set_title(" Chip's Sweet Slide (Animated!)")
ax.set_xlabel("Chip's Guess")
ax.set_ylabel("Wrongness (Loss)")
ax.grid(True)
ax.legend()

def init():
    point.set_data([], [])
    trail.set_data([], [])
    text.set_text("")
    return point, trail, text
```

```
def animate(i):
    x = steps[:i+1]
    y = step_y[:i+1]
    point.set_data([x[-1]], [y[-1]]) # Pass lists to set_data
    trail.set_data(x, y)
    text.set_text(f"Step {i}")
    text.set_position((x[-1], y[-1] + 5))
    return point, trail, text

ani = animation.FuncAnimation(fig, animate, frames=len(steps),
init_func=init, blit=True, repeat=False)
ani
```

```
<IPython.core.display.Markdown object>
```

```
/usr/local/lib/python3.11/dist-packages/IPython/core/
pylabtools.py:151: UserWarning: Glyph 127906 (\N{ROLLER COASTER})
missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151
: UserWarning: Glyph 127846 (\N{SOFT ICE CREAM}) missing from font(s)
DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151
: UserWarning: Glyph 127850 (\N{COOKIE}) missing from font(s) DejaVu
Sans.
  fig.canvas.print_figure(bytes_io, **kw)
```



```
/tmp/ipython-input-3-2732500662.py:111: UserWarning: Glyph 127850 (\
N{COOKIE}) missing from font(s) DejaVu Sans.
  ani = animation.FuncAnimation(fig, animate, frames=len(steps),
init_func=init, blit=True, repeat=False)
/usr/local/lib/python3.11/dist-packages/IPython/core/formatters.py:345
: UserWarning: Glyph 127850 (\N{COOKIE}) missing from font(s) DejaVu
Sans.
  return method()
/usr/local/lib/python3.11/dist-packages/IPython/core/formatters.py:345
: UserWarning: Glyph 127850 (\N{COOKIE}) missing from font(s) DejaVu
Sans.
  return method()
/usr/local/lib/python3.11/dist-packages/IPython/core/formatters.py:345
: UserWarning: Glyph 127850 (\N{COOKIE}) missing from font(s) DejaVu
Sans.
  return method()

<matplotlib.animation.FuncAnimation at 0x78d9c07d3650>

/usr/local/lib/python3.11/dist-packages/IPython/core/events.py:89:
UserWarning: Glyph 127850 (\N{COOKIE}) missing from font(s) DejaVu
Sans.
  func(*args, **kwargs)
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151
: UserWarning: Glyph 127850 (\N{COOKIE}) missing from font(s) DejaVu
Sans.
  fig.canvas.print_figure(bytes_io, **kw)
```