Shweta_math8

November 9, 2024

1 Shweta Ajay Shinde 017548687

MSDA, SJSU , Data 220- Math Method for DA

1) Make a gradient descent code (Python) to find the minimum of the following function (4 pts).

   f(x)=2x^2-x+2 (Set up your own starting point, learning rate and epochs)

[17]:
```python
import numpy as np

# Define the function and its derivative
def f(x):
    return 2 * x**2 - x + 2

def f_prime(x):
    return 4 * x - 1

# Set up parameters for gradient descent
learning_rate = 0.1
epochs = 50
starting_point = 10

# Initialize the starting point
x = starting_point

# Perform gradient descent
for i in range(epochs):
    gradient = f_prime(x)
    x -= learning_rate * gradient

    # print the progress
    print(f"Epoch {i+1}: x = {x}, f(x) = {f(x)}")

# Final output
print("\nApproximate minimum point:", x)
print("Function value at minimum point:", f(x))
```

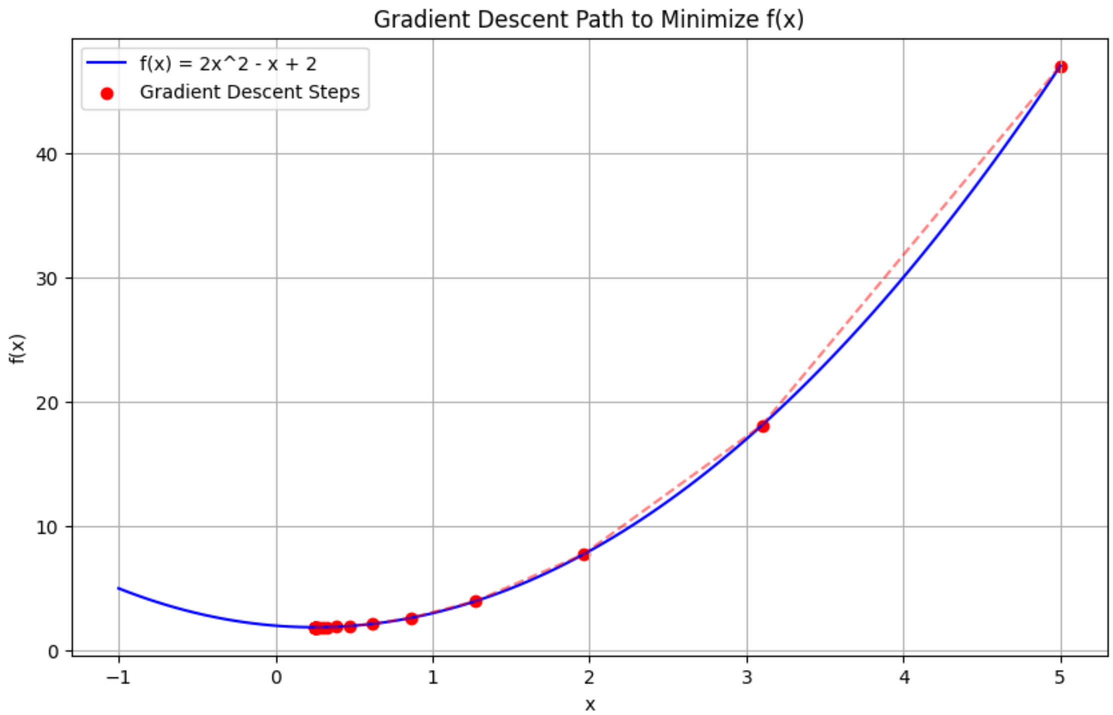
```
Epoch 1: x = 6.1, f(x) = 70.32
Epoch 2: x = 3.76, f(x) = 26.5152
```

```
Epoch 3: x = 2.356, f(x) = 10.745472
Epoch 4: x = 1.5135999999999998, f(x) = 5.0683699199999985
Epoch 5: x = 1.0081599999999997, f(x) = 3.024613171199999
Epoch 6: x = 0.7048959999999997, f(x) = 2.2888607416319995
Epoch 7: x = 0.5229375999999999, f(x) = 2.02398986698752
Epoch 8: x = 0.41376255999999995, f(x) = 1.9286363521155072
Epoch 9: x = 0.348257536, f(x) = 1.8943090867615826
Epoch 10: x = 0.3089545216, f(x) = 1.8819512712341697
Epoch 11: x = 0.28537271296, f(x) = 1.8775024576443011
Epoch 12: x = 0.271223627776, f(x) = 1.8759008847519485
Epoch 13: x = 0.2627341766656, f(x) = 1.8753243185107014
Epoch 14: x = 0.25764050599936, f(x) = 1.8751167546638525
Epoch 15: x = 0.254584303599616, f(x) = 1.8750420316789869
Epoch 16: x = 0.2527505821597696, f(x) = 1.8750151314044352
Epoch 17: x = 0.2516503492958618, f(x) = 1.8750054473055968
Epoch 18: x = 0.25099020957751705, f(x) = 1.8750019610300148
Epoch 19: x = 0.25059412574651024, f(x) = 1.8750007059708054
Epoch 20: x = 0.2503564754479061, f(x) = 1.87500025414949
Epoch 21: x = 0.25021388526874366, f(x) = 1.8750000914938163
Epoch 22: x = 0.2501283311612462, f(x) = 1.8750000329377738
Epoch 23: x = 0.2500769986967477, f(x) = 1.8750000118575987
Epoch 24: x = 0.2500461992180486, f(x) = 1.8750000042687356
Epoch 25: x = 0.25002771953082914, f(x) = 1.8750000015367447
Epoch 26: x = 0.2500166317184975, f(x) = 1.8750000005532281
Epoch 27: x = 0.2500099790310985, f(x) = 1.875000000199162
Epoch 28: x = 0.2500059874186591, f(x) = 1.8750000000716984
Epoch 29: x = 0.25000359245119547, f(x) = 1.8750000000258114
Epoch 30: x = 0.2500021554707173, f(x) = 1.8750000000092921
Epoch 31: x = 0.25000129328243037, f(x) = 1.875000000003345
Epoch 32: x = 0.25000077596945824, f(x) = 1.8750000000012044
Epoch 33: x = 0.25000046558167494, f(x) = 1.8750000000004334
Epoch 34: x = 0.250000279349005, f(x) = 1.875000000000156
Epoch 35: x = 0.25000016760940297, f(x) = 1.8750000000000562
Epoch 36: x = 0.25000010056564176, f(x) = 1.8750000000000202
Epoch 37: x = 0.25000006033938504, f(x) = 1.8750000000000073
Epoch 38: x = 0.25000003620363104, f(x) = 1.8750000000000027
Epoch 39: x = 0.2500000217221786, f(x) = 1.8750000000000009
Epoch 40: x = 0.25000001303330716, f(x) = 1.8750000000000004
Epoch 41: x = 0.25000000781998427, f(x) = 1.875
Epoch 42: x = 0.2500000046919906, f(x) = 1.875
Epoch 43: x = 0.25000000281519436, f(x) = 1.875
Epoch 44: x = 0.250000016891166, f(x) = 1.875
Epoch 45: x = 0.25000000101347, f(x) = 1.875
Epoch 46: x = 0.250000000608082, f(x) = 1.875
Epoch 47: x = 0.2500000003648492, f(x) = 1.875
Epoch 48: x = 0.2500000002189095, f(x) = 1.875
Epoch 49: x = 0.2500000001313457, f(x) = 1.875
Epoch 50: x = 0.2500000000788074, f(x) = 1.875
```

```
Approximate minimum point: 0.2500000000788074
Function value at minimum point: 1.875
```

2) Visualize how gradient descent updates the values of to reach the minimum value of the function (3 pts).

[18]:
```python
import numpy as np
import matplotlib.pyplot as plt

# Define the function and its derivative
def f(x):
    return 2 * x**2 - x + 2

def f_prime(x):
    return 4 * x - 1

# Set up parameters for gradient descent
learning_rate = 0.1
epochs = 20
starting_point = 5

# Initialize the starting point and store x values during gradient descent
x = starting_point
x_values = [x]
f_values = [f(x)]

# Perform gradient descent
for i in range(epochs):
    gradient = f_prime(x)
    x -= learning_rate * gradient  # Update the current point
    x_values.append(x)             # Store new x
    f_values.append(f(x))          # Store new f(x)

# Plotting the function and gradient descent steps
x_range = np.linspace(-1, 5, 100)
y_range = f(x_range)

plt.figure(figsize=(10, 6))
plt.plot(x_range, y_range, label="f(x) = 2x^2 - x + 2", color="blue")
plt.scatter(x_values, f_values, color="red", label="Gradient Descent Steps")
plt.plot(x_values, f_values, color="red", linestyle="--", alpha=0.5)
plt.xlabel("x")
plt.ylabel("f(x)")
plt.title("Gradient Descent Path to Minimize f(x)")
plt.legend()
plt.grid(True)
plt.show()
```



3) Make use of “optimize” from “scipy” to get the similar result (3 pts).

   (Choose a right optimizer method)

[19]:
```python
from scipy import optimize

# Define the function to minimize
def f(x):
    return 2 * x**2 - x + 2

# Initial guess
initial_guess = 5

# Using scipy.optimize.minimize with the BFGS method
result = optimize.minimize(f, initial_guess, method='BFGS')

# Output the results
print("Approximate minimum point:", result.x[0])
print("Function value at minimum point:", f(result.x[0]))
```

```
Approximate minimum point: 0.24999999374054904
Function value at minimum point: 1.875
```