Niranjan Rao - Assignment 8

November 6, 2024

[1]:

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import minimize
```

1 1. Gradient Descent Implementation and Results

[18]:

```python
def f(x):
    return 2 * x**2 - x + 2

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

def gradient_descent(starting_point, learning_rate, epochs):
    x = starting_point
    history = [x]
    for _ in range(epochs):
        gradient = df(x)
        x = x - learning_rate * gradient
        history.append(x)
    return x, history

starting_point = 5
learning_rate = 0.1
epochs = 50

minimum, history = gradient_descent(starting_point, learning_rate, epochs)

print(f"Minimum value found at x = {minimum}")
```

```
Minimum value found at x = 0.25000000003839334
```

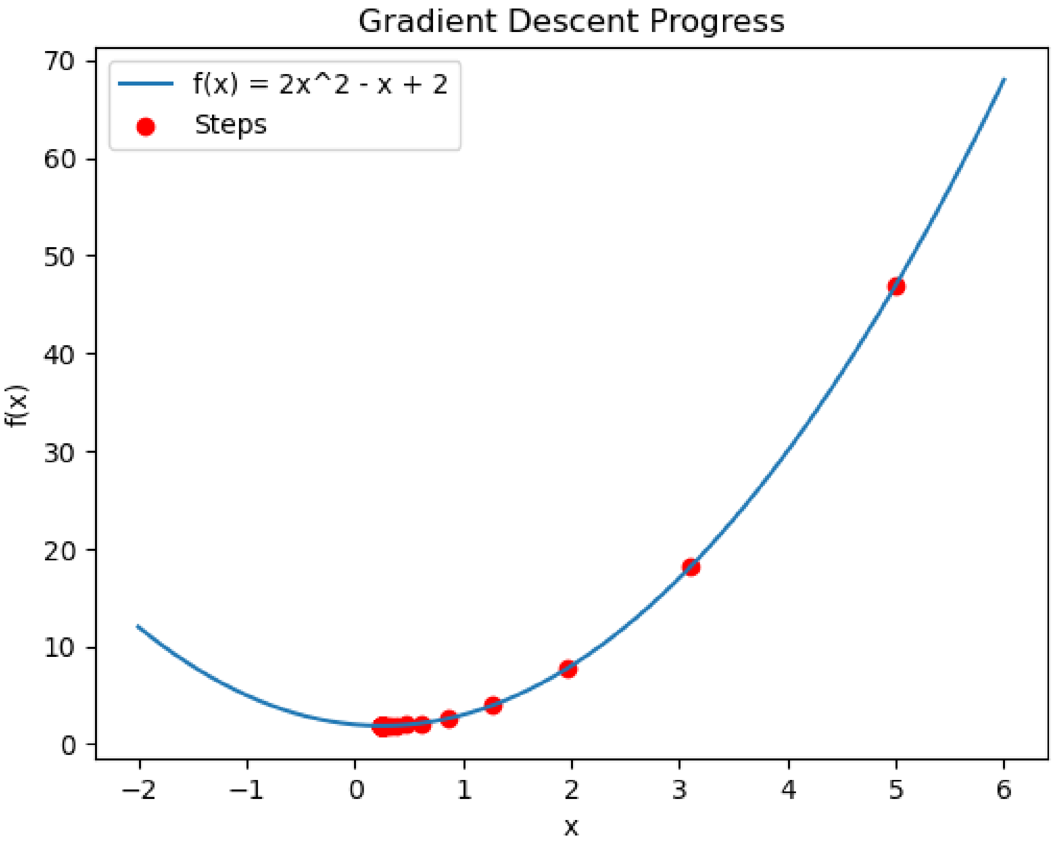
2 2. Visualization of Gradient Descent Steps

2.0.1 The plot below illustrates how the gradient descent algorithm updates the value of x at each iteration.

2.0.2 The red points indicate the progression of x values as they converge to the minimum of the function.

[24]:

```
x_vals = np.linspace(-2, 6, 100)
plt.plot(x_vals, f(x_vals), label='f(x) = 2x^2 - x + 2')
plt.scatter(history, [f(x) for x in history], color='red', label='Steps')
plt.title('Gradient Descent Progress')
plt.xlabel('x')
plt.ylabel('f(x)')
plt.legend()
plt.show()
```



3 3. Using Scipy to Find the Minimum

3.0.1 The scipy.optimize module provides optimization algorithms to find the minimum of a function.

3.0.2 Here, we use the minimize function to find the minimum and compare the result with our gradient descent implementation.

[27]:

```
result = minimize(f, x0=starting_point, method='BFGS')
print(f"Minimum found using scipy.optimize: x = {result.x[0]} with f(x) =␣
 ↪{result.fun}")
```

```
Minimum found using scipy.optimize: x = 0.24999999374054904 with f(x) = 1.875
```