12.11.24

Mathematical Analysis
Homework 5
Tapordei Maia

Q: Ex 1

Use gradient descent to explore the convergence of a function $f: R \to R$
Gradient descent updated rule:
$x_{n+1} = x_n - \eta f'(x_n)$, $\eta \to$ the learning rate

a, b, c) We use a convex function $f(x) = x^2$, which has a global minimum at $x = 0$. $f'(x) = 2x$. We explore different values of η to see how they affect convergence.
Conclusions from results:
1) $\eta = 0.1 \Rightarrow$ Converges slowly to 0, showing that a small learning rate takes more iterations
2) $\eta = 0.5 \Rightarrow$ Converges faster, requiring fewer iterations to reach near zero
3) $\eta = 1.2 \Rightarrow$ Diverges, oscillating and increasing in magnitude due to a learning rate that is too large.

d) We use a non-convex function $f(x) = x^4 - x^2$, which has a local minimum at $x = \pm 1/\sqrt{2}$. $f'(x) = 4x^3 - 2x$, saddle point is 0.
Conclusions from results.
1) Starting from $x = 0.1$, $x = -0.1 \Rightarrow$ Gradient descent converges to the saddle point at 0, which is neither a local minimum or a local maximum.
2) Starting at $x = 0.5$, $x = -0.5 \Rightarrow$ Algorithm converges to $x = \pm 1/\sqrt{2}$ which are local minimum points. This illustrates how the gradient descent method can end up in local minimum depending on the starting position in non-convex functions.

convex.py × non-convex.py

```python
import numpy as np
import matplotlib.pyplot as plt


# Define a convex function for parts (a)-(c)
def function_convex(x):
    return x ** 2


# Its derivative
def derivative_function_convex(x):
    return 2 * x

```

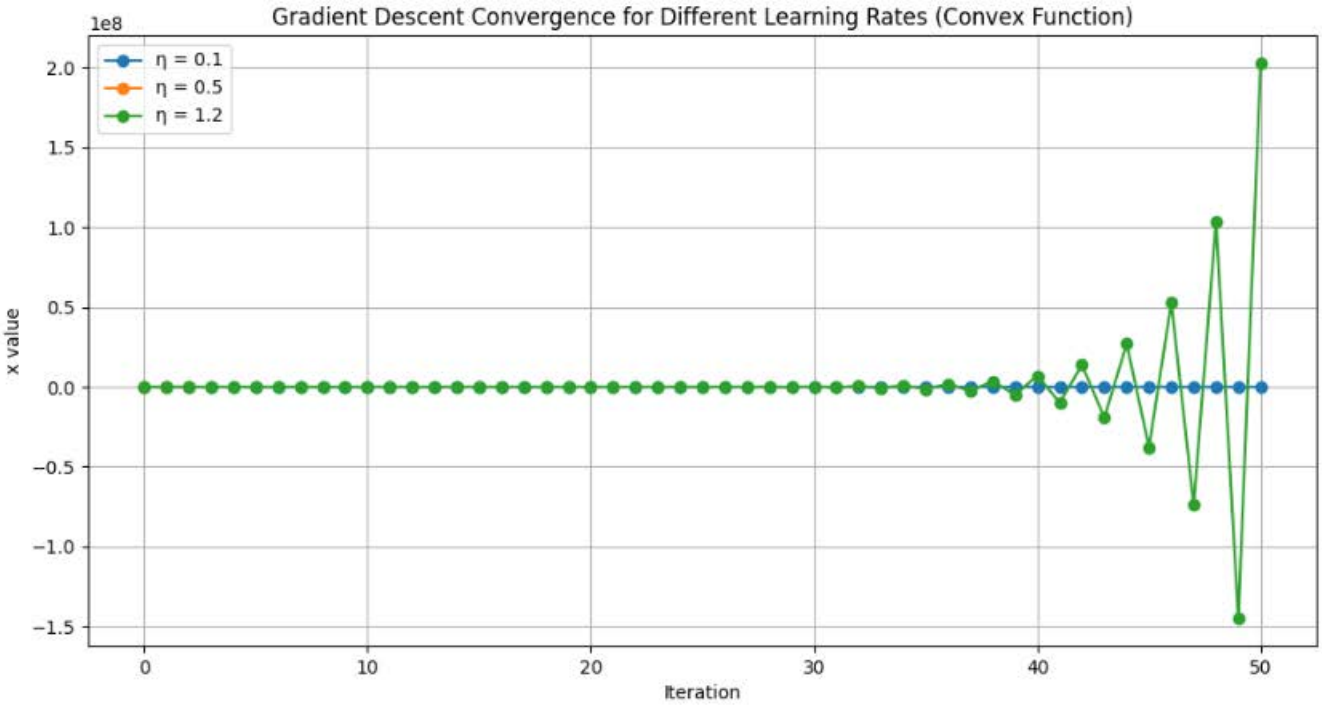
convex.py | non-convex.py

```python
def gradient_descent(f_prime, x_init, eta, max_iter=50, tol=1e-6):
    x = x_init
    history = [x]
    for _ in range(max_iter):
        x_new = x - eta * f_prime(x)
        history.append(x_new)
        # Check for convergence
        if abs(x_new - x) < tol:
            break
        x = x_new
    return np.array(history)


# Set initial values and learning rates for experiments
starting_point = 10  # Starting point, far from the minimum (0)

# Different learning rates to test
learning_rates = [0.1, 0.5, 1.2]  # Small, moderate, and too large
```

convex.py | non-convex.py

Reader M

```
34
35  # Run gradient descent with each learning rate
36  results = {}
37  for eta in learning_rates:
38      history = gradient_descent(derivative_function_convex, starting_point, eta)
39      results[eta] = history
40
41  # Plot results for parts (a), (b), and (c)
42  plt.figure(figsize=(12, 6))
43  for eta, history in results.items():
44      plt.plot( *args: history, label=f'n = {eta}', marker='o')
45
46  plt.xlabel("Iteration")
47  plt.ylabel("x value")
48  plt.title("Gradient Descent Convergence for Different Learning Rates (Convex Function)")
49  plt.legend()
50  plt.grid(True)
51  plt.show()
52
```


Gradient Descent Convergence for Different Learning Rates (Convex Function)
1e8
η = 0.1
η = 0.5
η = 1.2
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
x value
0
10
20
30
40
50
Iteration

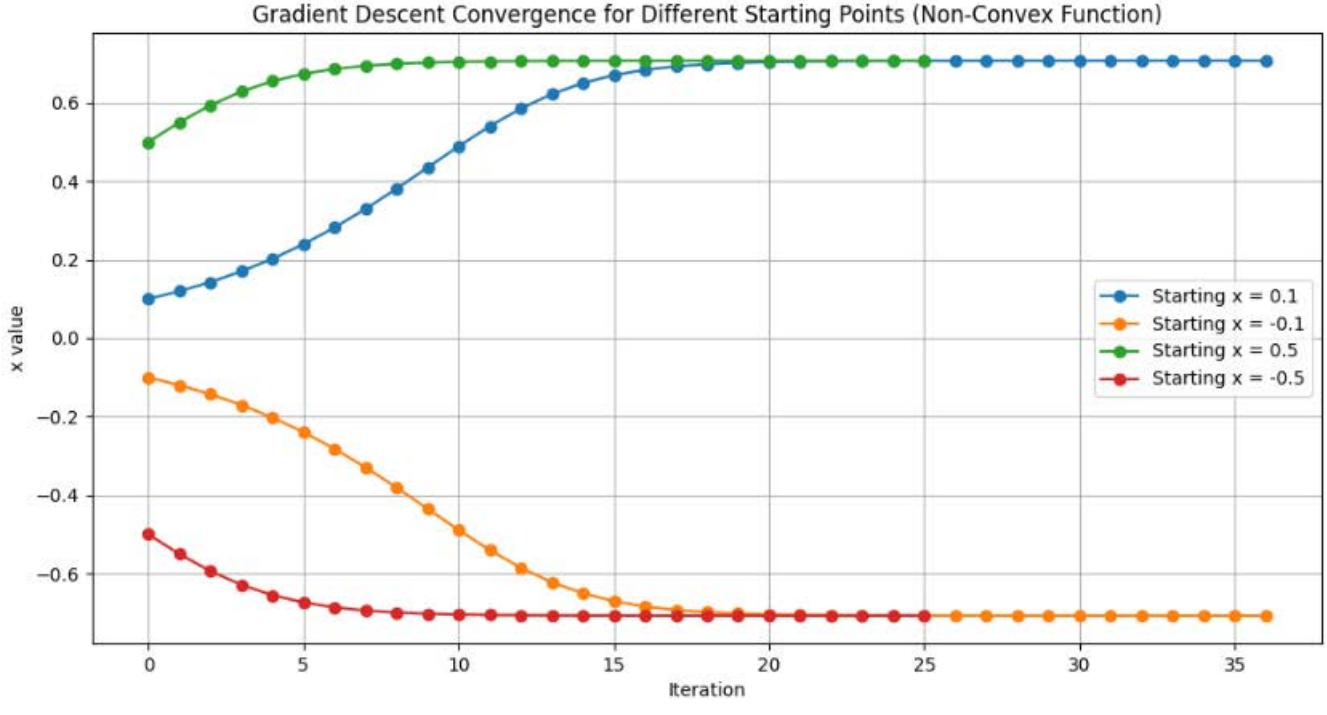
convex.py non-convex.py

```python
import numpy as np
import matplotlib.pyplot as plt

# Define a non-convex function for part (d)
def function_nonconvex(x):
    return x ** 4 - x ** 2


# Its derivative
def derivative_function_nonconvex(x):
    return 4 * x ** 3 - 2 * x

```

```
13  def gradient_descent(f_prime, x_init, eta, max_iter=50, tol=1e-6):
14      x = x_init
15      history = [x]
16      for _ in range(max_iter):
17          x_new = x - eta * f_prime(x)
18          history.append(x_new)
19          # Check for convergence
20          if abs(x_new - x) < tol:
21              break
22          x = x_new
23      return np.array(history)
24
25  # Different starting points for non-convex function
26  starting_points = [0.1, -0.1, 0.5, -0.5]  # Starting points close to local minima and zero
27
28  # Learning rate
29  learning_rate = 0.1  # Moderate learning rate for convergence
30
```

```python
# Run gradient descent with the non-convex function from different starting points
results_nonconvex = {}
for x_init in starting_points:
    history = gradient_descent(derivative_function_nonconvex, x_init, learning_rate, max_iter=100
    results_nonconvex[x_init] = history

# Plot results for part (d)
plt.figure(figsize=(12, 6))
for x_init, history in results_nonconvex.items():
    plt.plot( *args: history, label=f'Starting x = {x_init}', marker='o')

plt.xlabel("Iteration")
plt.ylabel("x value")
plt.title("Gradient Descent Convergence for Different Starting Points (Non-Convex Function)")
plt.legend()
plt.grid(True)
plt.show()
```

Reader Mode


Gradient Descent Convergence for Different Starting Points (Non-Convex Function)
x value
Iteration
Starting x = 0.1
Starting x = -0.1
Starting x = 0.5
Starting x = -0.5
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
5
10
15
20
25
30
35