Gradient_decent

June 4, 2023

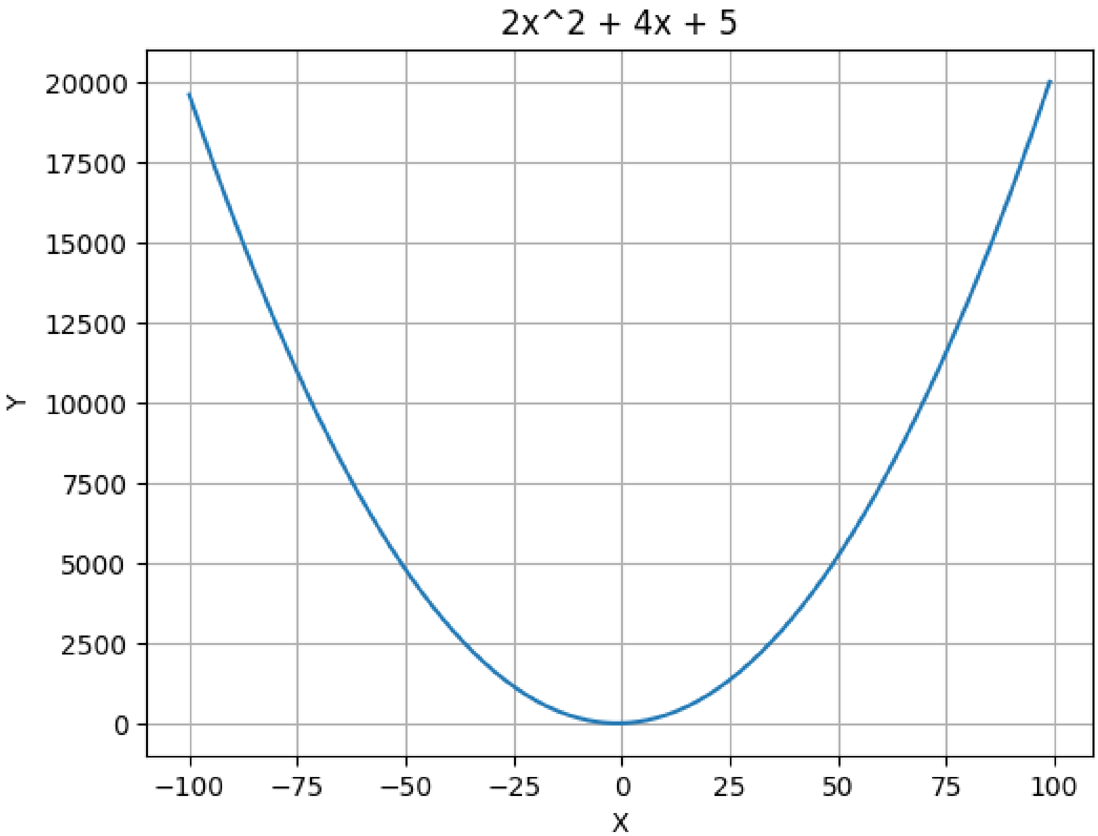
[1]:
```
# Import necessary libraries
import numpy as np
import matplotlib.pyplot as plt
```

0.1 Define the function

0.1.1 f(x) = 2x^2 + 4x + 5

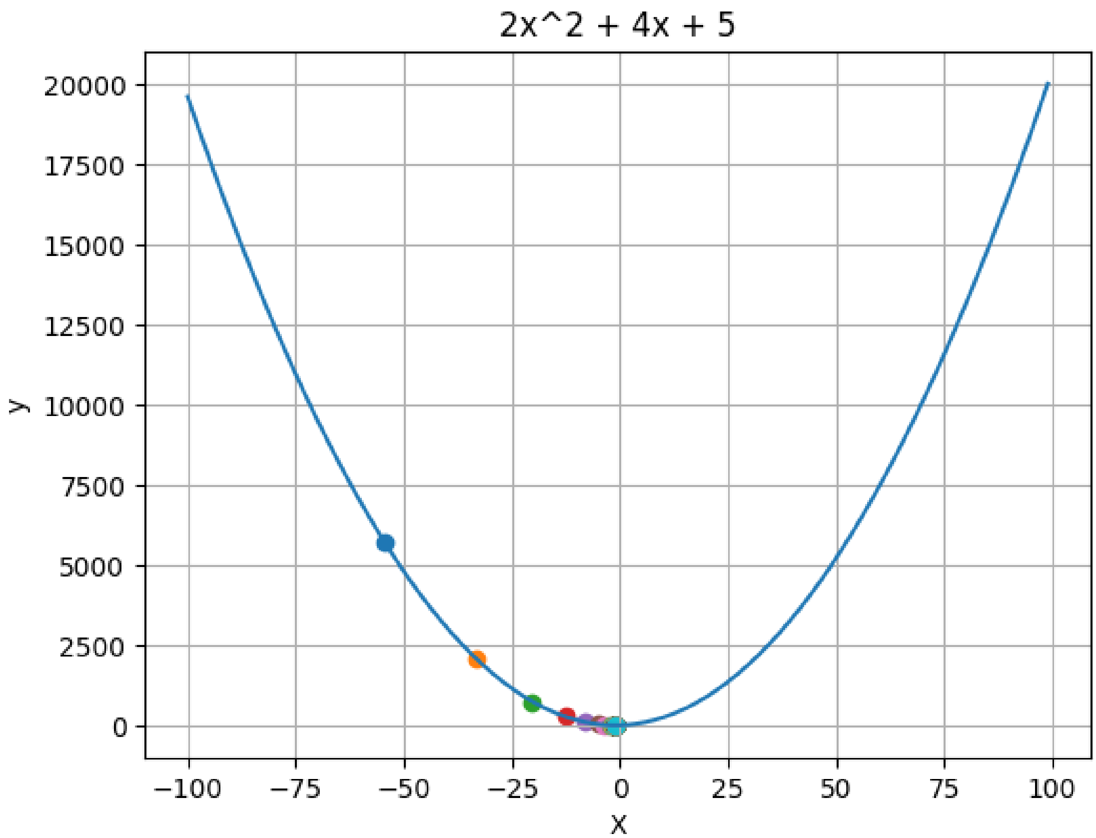
[27]:
```
# Genrate -100 to 100 number corresponding y value ie, y = f(x)
X = np.arange(-100,100)
Y = (2*(X**2)) + (4*X) + 5
```

[28]:
```
# Plot the function
plt.title("2x^2 + 4x + 5")
plt.xlabel("X")
plt.ylabel("Y")
plt.plot(X,Y);
plt.grid()
```

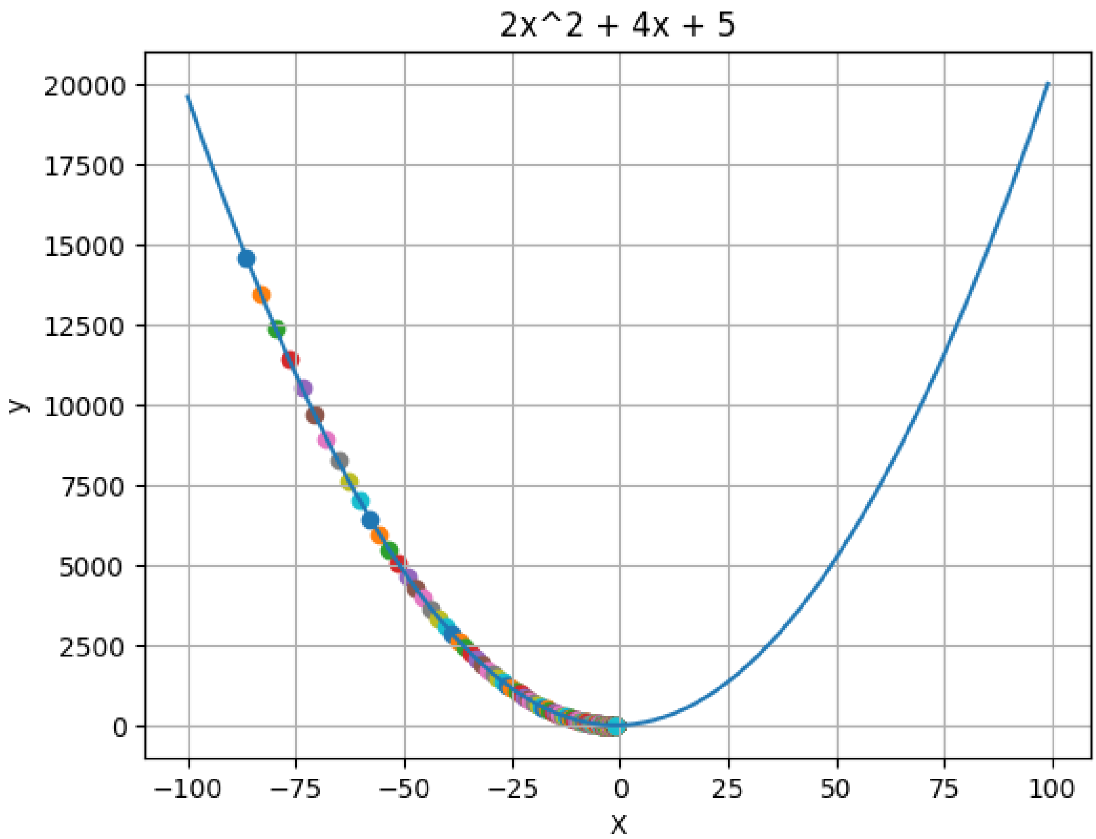


0.1.2 Gradient decesent algorithm

[37]:
```
plt.title("2x^2 + 4x + 5")
plt.xlabel("X")
plt.ylabel("y")
# Intialize the random value to x
#np.random.seed(60)
x = -90
# Intialize the learning rate
lr = 0.1
plt.plot(X,Y)
for i in range(1000):
    gradient = (4*x) + 4
    x = x - (lr*gradient)
    y = (2*(x**2)) + (4*x) + 5
    plt.scatter(x,y)
plt.grid()
plt.show()
```



[38]:
```
plt.title("2x^2 + 4x + 5")
plt.xlabel("X")
plt.ylabel("y")
# Intialize the random value to x
#np.random.seed(60)
x = -90
# Intialize the learning rate
lr = 0.01
plt.plot(X,Y)
for i in range(1000):
    gradient = (4*x) + 4
    x = x - (lr*gradient)
    y = (2*(x**2)) + (4*x) + 5
    plt.scatter(x,y)
plt.grid()
plt.show()
```



2x^2 + 4x + 5
y
20000
17500
15000
12500
10000
7500
5000
2500
0
−100
−75
−50
−25
0
25
50
75
100
X


```
[ ]:
```