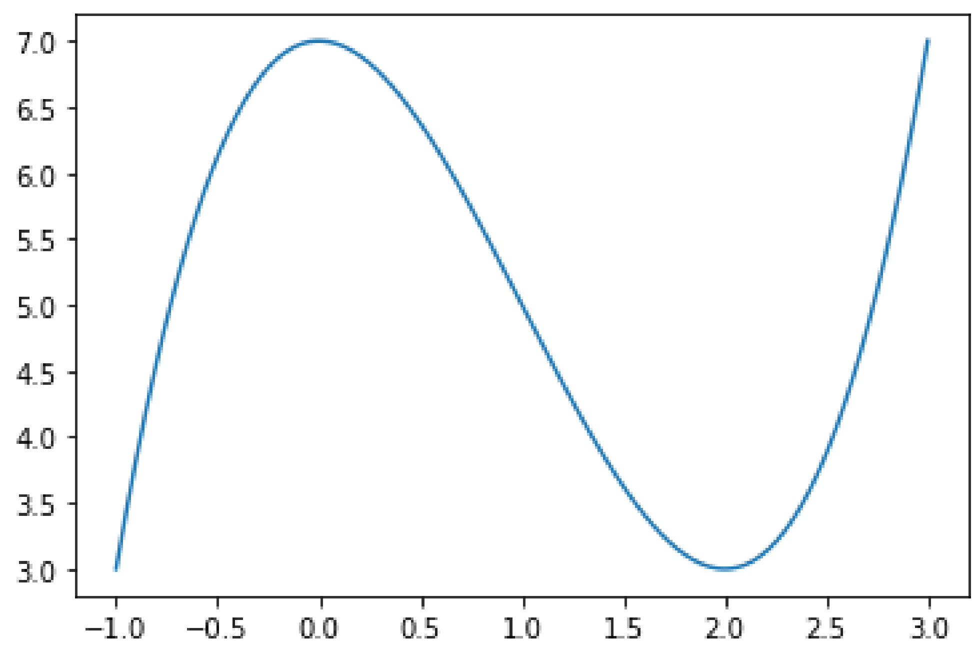
ng-fundamentals-in-depth-aravindh

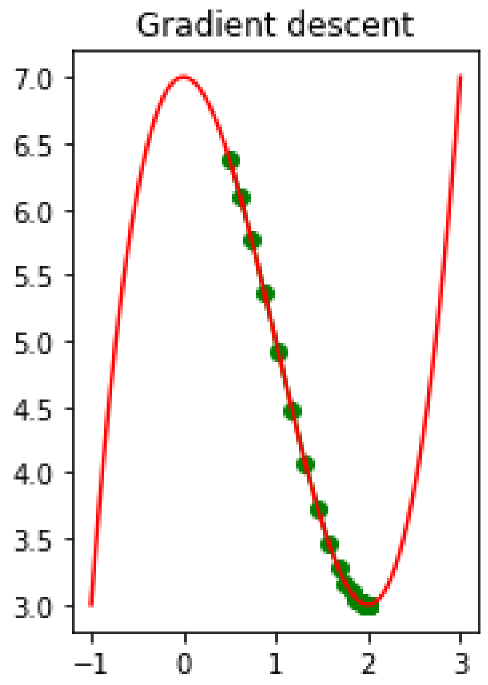
April 21, 2023

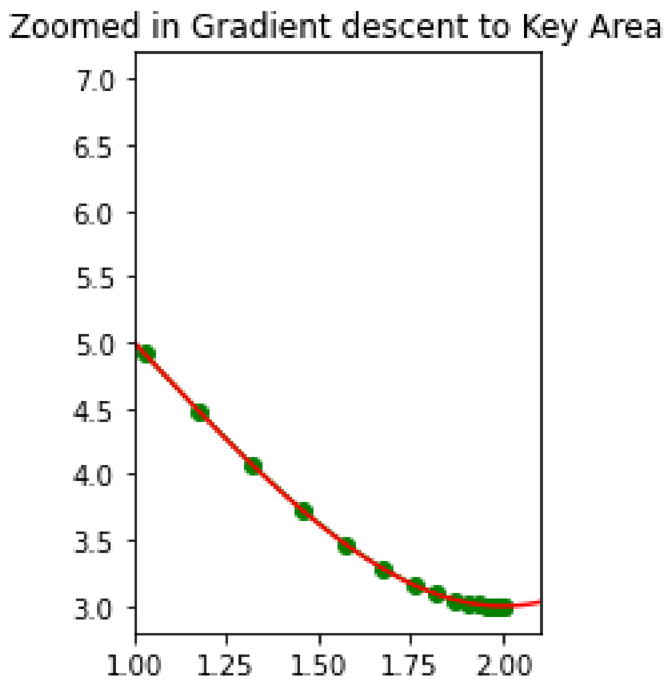
1.Perform Gradient Descent in Python with any loss function

[2]:
```
#import the basic modules
import numpy as np
import matplotlib.pyplot as plt
from scipy.constants.codata import precision

#creating the function and plotting it We will find the gradient decent of this␣
↪function - f(x) = x^3-3x^2+7
function = lambda x: (x ** 3)-(3 *(x ** 2))+7

#Get 1000 evenly spaced numbers between -1 and 3 (arbitratil chosen to ensure␣
↪steep curve)
x = np.linspace(-1,3,500)

#Plot the curve
plt.plot(x, function(x))
plt.show()

#This function takes in a value of x and returns its derivative based on the␣
↪initial function specified.
def deriv(x):
    x_deriv = 3* (x**2) - (6 * (x))
    return x_deriv

#This function takes in an initial or previous value for x, updates it based on␣
↪steps taken via the learning rate and outputs the most minimum value of x␣
↪that reaches the precision satisfaction.
def step(x_new, x_prev, precision, l_r):

    # create empty lists where the updated values of x and y wil be appended␣
↪during each iteration
    x_list, y_list = [x_new], [function(x_new)]

    # keep looping until your desired precision
    while abs(x_new - x_prev) > precision:

        # change the value of x
        x_prev = x_new
        # get the derivation of the old value of x
        d_x = - deriv(x_prev)
        # get your new value of x by adding the previous, the multiplication of␣
 ↪the derivative and the learning rate
        x_new = x_prev + (l_r * d_x)
        # append the new value of x to a list of all x-s for later␣
 ↪visualization of path
        x_list.append(x_new)
        # append the new value of y to a list of all y-s for later␣
 ↪visualization of path
        y_list.append(function(x_new))

    print ("Local minimum occurs at: "+ str(x_new))
    print ("Number of steps: " + str(len(x_list)))

    plt.subplot(1,2,2)
    plt.scatter(x_list,y_list,c="g")
    plt.plot(x_list,y_list,c="g")
    plt.plot(x,function(x), c="r")
    plt.title("Gradient descent")
    plt.show()

    plt.subplot(1,2,1)
    plt.scatter(x_list,y_list,c="g")
    plt.plot(x_list,y_list,c="g")
    plt.plot(x,function(x), c="r")
    plt.xlim([1.0,2.1])
    plt.title("Zoomed in Gradient descent to Key Area")
    plt.show()
    #Implement gradient descent (all the arguments are arbitrarily chosen)
step(0.5, 0, 0.001, 0.05)
```



```
Local minimum occurs at: 1.9980265135950486
Number of steps: 25
```





2.Difference between L1 & L2 Gradient descent method

Mean Absolute Error, L1-(MAE) is another loss function used for regression models. MAE is the sum of absolute differences between our target and predicted variables. So it measures the average magnitude of errors in a set of predictions, without considering their directions. The range is also 0 to ∞.

MAE=(1n)n i=1 yi−ypi

Mean Squared Error, L2-(MSE) is the most commonly used regression loss function. MSE is the sum of squared distances between our target variable and predicted values.

MSE=(1n)n i=1(yi−ypi)2

Since MSE squares the error (y — y_predicted = e), the value of error (e) increases a lot if e > 1. If we have an outlier in our data, the value of e will be high and e^2 will be >> |e|. This will make the model with MSE loss give more weight to outliers than a model with MAE loss.

3.What are the different loss functions for regression

word files attached mam/sir..for this questions..

4.What is the importance of learning rate

Learning rate ( ) is one such hyper-parameter that defines the adjustment in the weights of our network with respect to the loss gradient descent. It determines how fast or slow we will move towards the optimal weights.

The Gradient Descent Algorithm estimates the weights of the model in many iterations by minimizing a cost function at every step.

Learning rate is a scalar, a value that tells the machine how fast or how slow to arrive at some conclusion. The speed at which a model learns is important and it varies with different applications. A super-fast learning algorithm can miss a few data points or correlations which can give better insights into the data. Missing this will eventually lead to wrong classifications.

And, to find the next step or the adjacent data point, the gradient descent algorithms multiply the gradient by learning rate (also called step size).

For example, if the gradient magnitude is 1.5 and the learning rate is 0.01, then the gradient descent algorithm will pick the next point 0.015 away from the previous point.

If a learning rate is too small, learning will take too long:

And if a learning rate is too large, the next point will perpetually bounce haphazardly across the bottom of the valley:

5.How to evaluate linear regression

There are a number of metrics used in evaluating the performance of a linear regression model. They include:

`R-Squared: seldom used for evaluating model fit, itis also known as the coefficient of determi`

sklearn module : sklearn.metrics.r2_score

mathematical formula: Image for post

`MSE (Mean Squared Error): used for evaluating model fit, it measures the average of the square`

sklearn module: sklearn.metrics.mean_squared_error

mathematical formula: Image for post

`RMSE (Root Mean Squared Error): always used for evaluating model fit.is the measure of the dis`

using sklearn and math module to perform RMSE

from sklearn.metrics import mean_squared_error

from math import sqrt

rmse = sqrt(mean_squared_error(y_actual, y_predicted))

print(rmse)

mathematical formula Image for post

6.What is the difference between multiple and adjusted coefficient of determination

R^2 measures the proportion of the total variation in Y explained by the regression model.

Here are some key points about R^2:

It is a non-negative quantity with a range 0 R^2 1
R^2 = 0 implies that the regression line does not fit the data at all.
R^2 = 1 implies that the regression line is a perfect fit.

Adjusted R^2 is a modified version of R^2 adjusted with the number of predictors. It penalizes for adding unnecessary features and allows a comparison of regression models with a different number of predictors. The adjusted R-squared increases only if the new term improves the model more than would be expected by chance. It decreases when a predictor improves the model by less than expected by chance.

Obtaining a negative value for Adjusted R^2 can indicate few or all of the following:

```
The linear model is a poor fit for the data
The number of predictors is large
The number of samples is small
```

[ ]:

```

```