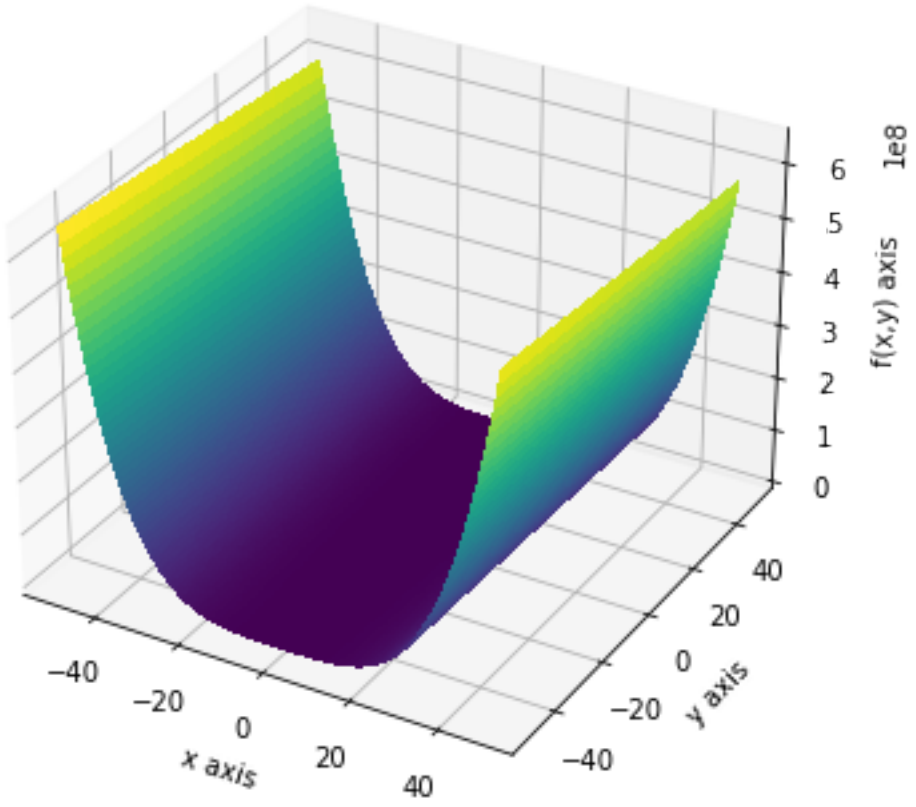
Lab 3 - ML Programming

November 26, 2021

1 EXERCISE 1

1.1 Gradient Descent on Rosenbrock Function

[1]:
```
## Implement a 3D plot to visualize the function f(x, y) = (1 - x)^2 + 100(y -
 ↪x^2)^2
## Code based on answer found at https://stackoverflow.com/questions/9170838/
 ↪surface-plots-in-matplotlib

import numpy as np
import matplotlib.pyplot as plt
from matplotlib import cm
from mpl_toolkits.mplot3d import Axes3D

def f(x,y):
    return ((1-x)**2) + (100*((y-(x**2))**2))

fig = plt.figure(figsize=(8,6))
ax = fig.add_subplot(projection='3d')
x = y = np.arange(-50.0, 50.0, 0.5)
X, Y = np.meshgrid(x, y)
zs = np.array(f(X, Y))
Z = zs.reshape(X.shape)

ax.plot_surface(X, Y, Z, cmap='viridis', rstride=1, cstride=1, linewidth=0,
 ↪antialiased=False)

ax.set_xlabel('x axis')
ax.set_ylabel('y axis')
ax.set_zlabel('f(x,y) axis')

plt.show()
```



Derive the partial gradients

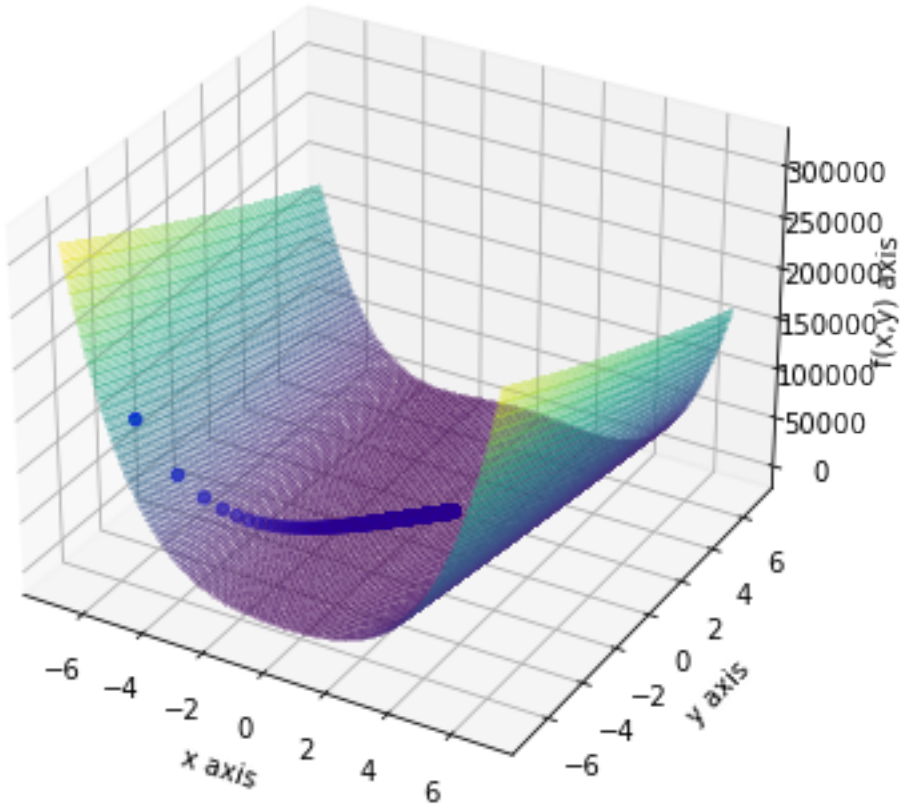
$$\nabla f(x,y) = \begin{cases} \dfrac{\partial f}{\partial x}(1-x)^2 + \dfrac{\partial f}{\partial x}100(y-x^2)^2 \\ \dfrac{\partial f}{\partial y}(1-x)^2 + \dfrac{\partial f}{\partial y}100(y-x^2)^2 \end{cases} = \begin{cases} -2(1-x) - 400x(y-x^2) \\ 200(y-x^2) \end{cases}$$

[2]:
```
## Convert the function and gradient of this function into equivalent code␣
 ↪representation
## Function already defined as f(x,y)

def derx(x,y):
    der_wrtx = -2*(1-x) - 400*x*(y-(x**2))
    return der_wrtx

def dery(x,y):
    der_wrty = 200*(y-(x**2))
    return der_wrty
```

[3]:
```
## Optimize the function with Gradient Descent
## Set the hyperparameters like initial valus of (x,y) and the steplength␣
↪through trial and error
## Code based on http://firsttimeprogrammer.blogspot.com/2014/09/
↪multivariable-gradient-descent.html

point_x = -10
point_y = -10
alpha = .00001
iterations = 0
precision = 0.0000001
printData = True
maxIterations = 1000000
xvalues = []
yvalues = []

while True:
    temppoint_x = point_x - alpha*derx(point_x,point_y)
    temppoint_y = point_y - alpha*derx(point_x,point_y)

    ## If number of iterations is too high, stop the loop. Theta(x or y) might␣
↪be diverging
    iterations += 1
    if iterations > maxIterations:
        print("Too many iterations. Adjust alpha and make sure that the␣
↪function is convex!")
        printData = False
        break

    ## If the value of theta changes is less then specified precision, minimum␣
↪has been found
    if abs(temppoint_x-point_x) < precision and abs(temppoint_y-point_y) <␣
↪precision:
        break

    ## Simultaneous update of variables
    point_x = temppoint_x
    point_y = temppoint_y

    ## Save x and y values generated for the plot
    xvalues.append(temppoint_x)
    yvalues.append(temppoint_y)

if printData:
    print("Number of iterations:",iterations,sep=" ")
    print(f'The local minimum occurs at ({point_x},{point_y})')
```

```
Number of iterations: 13432
The local minimum occurs at (0.9999751451502644,0.9999751451502644)
```

[4]:
```
## Visualize the trajectory on the same 3D plot.
## Should ideally lead to the function minimum, starting off with (x = 10, y = 10) for example

xvalues = np.array(xvalues)
yvalues = np.array(yvalues)
zvalues = []
zvalues = ((1-xvalues)**2) + (100*((yvalues-(xvalues**2))**2))
zvalues = np.array(zvalues)

fig = plt.figure(figsize=(8,6))
ax = fig.add_subplot(projection='3d')

x = y = np.arange(-7.0, 7.0, 0.1)
X, Y = np.meshgrid(x, y)
zs = np.array(f(X, Y))
Z = zs.reshape(X.shape)
ax.plot_surface(X, Y, Z, cmap='viridis', alpha=0.2, rstride=1, cstride=1, linewidth=0, antialiased=False)
ax.scatter(xvalues,yvalues,zvalues,c='blue')

ax.set_xlabel('x axis')
ax.set_ylabel('y axis')
ax.set_zlabel('f(x,y) axis')

plt.show()
```



2 EXERCISE 2

2.1 Part A: Datasets

[43]:
```
## Convert any non-numeric values to numeric values
## For example you can replace a country name w/ an integer value or more␣
 ↪appropriately use hot-one encoding
## If required drop out the rows with missing values or NA
## Are there any columns that can be dropped? if so, which ones are why

import pandas as pd
import csv
airfare = pd.read_csv('airq402.data', delim_whitespace=True, header=None)
airfare.columns =['City1','City2','Average Fare','Distance','Ave. weekly␣
 ↪passengers','market leading airline', \
                'market share1','Average Fare2','low price airline', 'market␣
 ↪share2','price']
airfare.dropna()
```

```
## Basic one hot encoding taken from https://stackoverflow.com/questions/
 ↪37292872/how-can-i-one-hot-encode-in-python
city1 = pd.get_dummies(airfare.City1)
airfare = pd.concat([airfare,city1],axis=1)
def encode_and_bind(original_dataframe, feature_to_encode):
    dummies = pd.get_dummies(original_dataframe[[feature_to_encode]])
    res = pd.concat([original_dataframe, dummies], axis=1)
    return(res)
airfare = encode_and_bind(airfare, 'City1')
airfare = encode_and_bind(airfare, 'City2')
airfare = encode_and_bind(airfare, 'market leading airline')
airfare = encode_and_bind(airfare, 'low price airline')


wine = pd.read_csv('winequality-red.csv', delimiter=';')

parkinsons = pd.read_csv('parkinsons_updrs.data', delimiter=',')

wine = wine.dropna()
parkinsons = parkinsons.dropna()
```

[6]:
```
## Split the dataset into 80% Train set and 20% Test set

wine_train = wine.sample(frac=0.8,random_state=200) ## random state is just a␣
 ↪seed value
wine_test = wine.drop(wine_train.index)

parkinsons_train = parkinsons.sample(frac=0.8,random_state=200)
parkinsons_test = parkinsons.drop(parkinsons_train.index)

airfare_train = airfare.sample(frac=0.8,random_state=200)
airfare_test = airfare.drop(airfare_train.index)
```

3 EXERCISE 2

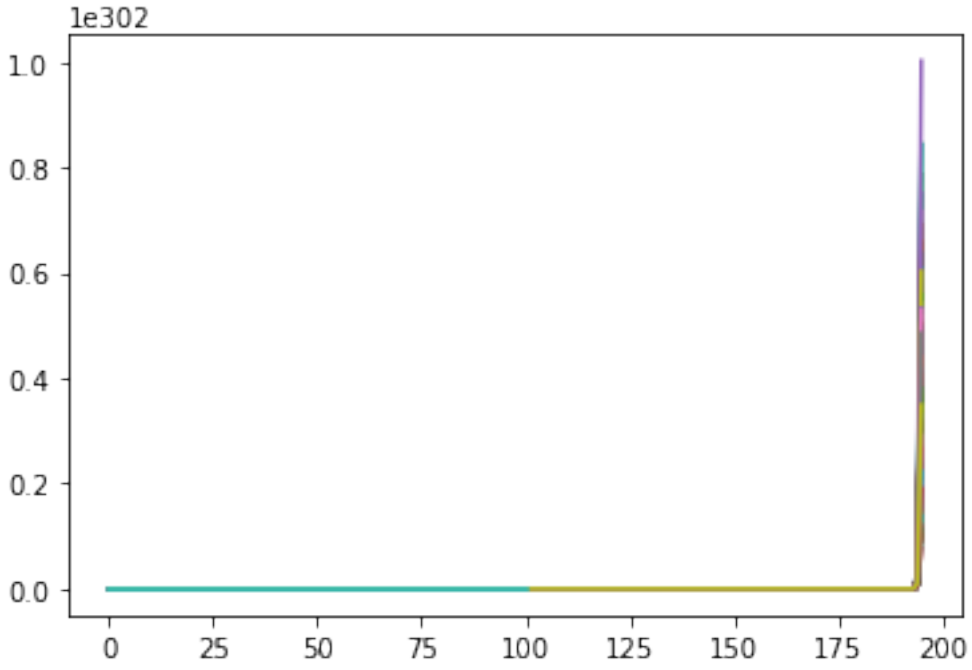
3.1 Part B: Linear Regression w/ Real World Data

[35]:
```
## Linear regression with gradient descent -- wine data

## First split training and testing data into features and targets
x_wine_train = wine_train.iloc[:,:-1].values
x_wine_train = np.array(x_wine_train, dtype=float)
y_wine_train = wine_train.iloc[:,-1].values
y_wine_train = np.array(y_wine_train, dtype=float)
x_wine_test = wine_test.iloc[:,:-1].values
x_wine_test = np.array(x_wine_test, dtype=float)
y_wine_test = wine_test.iloc[:,-1].values
```

```
y_wine_test = np.array(y_wine_test, dtype=float)

def linreg_withGD_abs(x,y,alpha):
    imax = 500 ## chose this because it is a middleground between too many and␣
 ↪too few iterations
    rows = x.shape[0]
    cols = x.shape[1]
    theta = [0]*cols
    abs_values = [0]*imax
    for i in range(imax):
        h = np.dot(x,theta)
        error = h-y
        gradient = x.T.dot(error)/rows
        theta = theta - (alpha*gradient)
        ypred = x*theta
        abso = abs((ypred[:,1]-y))
        abs_values[i] = abso
    return abs_values

abs_values = linreg_withGD_abs(x_wine_train,y_wine_train,0.01)
plt.plot(abs_values)

def linreg_withGD_rmse(x,y,alpha):
    imax = 500
    rows = x.shape[0]
    cols = x.shape[1]
    theta = [0]*cols
    rmse_values = [0]*imax
    for i in range(imax):
        h = np.dot(x,theta)
        error = h-y
        gradient = x.T.dot(error)/rows
        theta = theta - (alpha*gradient)
        ypred = x*theta
        MSE = np.square(np.subtract(ypred[:,1],y)).mean()
        RMSE = (MSE)**0.5
        rmse_values[i] = RMSE
    return rmse_values

rmse_values = linreg_withGD_rmse(x_wine_test,y_wine_test,0.01)
plt.plot(rmse_values)
plt.show()
```

```
<ipython-input-35-5937c0ced4ff>:23: RuntimeWarning: invalid value encountered in
subtract
  theta = theta - (alpha*gradient)
C:\Users\Nikita\anaconda3\lib\site-packages\numpy\core\_methods.py:178:
RuntimeWarning: overflow encountered in reduce
```

```
  ret = umr_sum(arr, axis, dtype, out, keepdims, where=where)
<ipython-input-35-5937c0ced4ff>:44: RuntimeWarning: overflow encountered in
square
  MSE = np.square(np.subtract(ypred[:,1],y)).mean()
<ipython-input-35-5937c0ced4ff>:42: RuntimeWarning: invalid value encountered in
subtract
  theta = theta - (alpha*gradient)
```



[42]:
```
## Linear regression with gradient descent -- parkinson data

## First split training and testing data into features and targets
x_parkinsons_train = parkinsons_train.drop("total_UPDRS", axis=1, inplace=True)
x_parkinsons_train = np.array(x_parkinsons_train, dtype=float)
y_parkinsons_train = parkinsons_train('total_UPDRS')
y_parkinsons_train = np.array(y_parkinsons_train, dtype=float)
x_parkinsons_test = parkinsons_test.drop("total_UPDRS", axis=1, inplace=True)
x_parkinsons_test = np.array(x_parkinsons_test, dtype=float)
y_parkinsons_test = parkinsons_test.total_UPDRS
y_parkinsons_test = np.array(y_parkinsons_test, dtype=float)

def linreg_withGD_abs(x,y,alpha):
    imax = 500 ## chose this because it is a middleground between too many and␣
 ↪too few iterations
    rows = x.shape[0]
    cols = x.shape[1]
```

```
    theta = [0]*cols
    abs_values = [0]*imax
    for i in range(imax):
        h = np.dot(x,theta)
        error = h-y
        gradient = x.T.dot(error)/rows
        theta = theta - (alpha*gradient)
        ypred = x*theta
        abso = abs((ypred[:,1]-y))
        abs_values[i] = abso
    return abs_values

abs_values = linreg_withGD_abs(x_parkinsons_train,y_parkinsons_train,0.01)
plt.plot(abs_values)

def linreg_withGD_rmse(x,y,alpha):
    imax = 500
    rows = x.shape[0]
    cols = x.shape[1]
    theta = [0]*cols
    rmse_values = [0]*imax
    for i in range(imax):
        h = np.dot(x,theta)
        error = h-y
        gradient = x.T.dot(error)/rows
        theta = theta - (alpha*gradient)
        ypred = x*theta
        MSE = np.square(np.subtract(ypred[:,1],y)).mean()
        RMSE = (MSE)**0.5
        rmse_values[i] = RMSE
    return rmse_values

rmse_values = linreg_withGD_rmse(x_parkinsons_test,y_parkinsons_test,0.01)
plt.plot(rmse_values)
plt.show()
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-42-bc6721ee1e41> in <module>
      2
      3 ## First split training and testing data into features and targets
----> 4 x_parkinsons_train = parkinsons_train.drop("total_UPDRS", axis=1,␣
 ↪inplace=True)
      5 x_parkinsons_train = np.array(x_parkinsons_train, dtype=float)
      6 y_parkinsons_train = parkinsons_train('total_UPDRS')
```

```
~\anaconda3\lib\site-packages\pandas\core\frame.py in drop(self, labels, axis,␣
↪index, columns, level, inplace, errors)
  4306                 weight  1.0     0.8
  4307         """
-> 4308         return super().drop(
  4309             labels=labels,
  4310             axis=axis,

~\anaconda3\lib\site-packages\pandas\core\generic.py in drop(self, labels, axis␣
↪index, columns, level, inplace, errors)
  4151         for axis, labels in axes.items():
  4152             if labels is not None:
-> 4153                 obj = obj._drop_axis(labels, axis, level=level,␣
↪errors=errors)
  4154
  4155         if inplace:

~\anaconda3\lib\site-packages\pandas\core\generic.py in _drop_axis(self, labels␣
↪axis, level, errors)
  4186                 new_axis = axis.drop(labels, level=level, errors=errors
  4187             else:
-> 4188                 new_axis = axis.drop(labels, errors=errors)
  4189             result = self.reindex(**{axis_name: new_axis})
  4190

~\anaconda3\lib\site-packages\pandas\core\indexes\base.py in drop(self, labels,␣
↪errors)
  5589         if mask.any():
  5590             if errors != "ignore":
-> 5591                 raise KeyError(f"{labels[mask]} not found in axis")
  5592             indexer = indexer[~mask]
  5593         return self.delete(indexer)

KeyError: "['total_UPDRS'] not found in axis"
```

[45]:
```
## Linear regression with gradient descent -- airfare data

## First split training and testing data into features and targets
x_airfare_train = airfare_train.drop("price", axis=1)
x_airfare_train = np.array(x_airfare_train, dtype=float)
y_airfare_train = airfare_train.price
y_airfare_train = np.array(y_airfare_train, dtype=float)
x_airfare_test = airfare_test.drop("price", axis=1, inplace=True)
x_airfare_test = np.array(x_airfare_test, dtype=float)
y_airfare_test = airfare_test.price
y_airfare_test = np.array(y_airfare_test, dtype=float)
```

```
def linreg_withGD_abs(x,y,alpha):
    imax = 500 ## chose this because it is a middleground between too many and␣
 ↪too few iterations
    rows = x.shape[0]
    cols = x.shape[1]
    theta = [0]*cols
    abs_values = [0]*imax
    for i in range(imax):
        h = np.dot(x,theta)
        error = h-y
        gradient = x.T.dot(error)/rows
        theta = theta - (alpha*gradient)
        ypred = x*theta
        abso = abs((ypred[:,1]-y))
        abs_values[i] = abso
    return abs_values

abs_values = linreg_withGD_abs(x_airfare_train,y_airfare_train,0.01)
plt.plot(abs_values)

def linreg_withGD_rmse(x,y,alpha):
    imax = 500
    rows = x.shape[0]
    cols = x.shape[1]
    theta = [0]*cols
    rmse_values = [0]*imax
    for i in range(imax):
        h = np.dot(x,theta)
        error = h-y
        gradient = x.T.dot(error)/rows
        theta = theta - (alpha*gradient)
        ypred = x*theta
        MSE = np.square(np.subtract(ypred[:,1],y)).mean()
        RMSE = (MSE)**0.5
        rmse_values[i] = RMSE
    return rmse_values

rmse_values = linreg_withGD_rmse(x_airfare_test,y_airfare_test,0.01)
plt.plot(rmse_values)
plt.show()
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-45-62dbe489c0ff> in <module>
      2
      3 ## First split training and testing data into features and targets
```

```
----> 4 x_airfare_train = airfare_train.drop("price", axis=1)
      5 x_airfare_train = np.array(x_airfare_train, dtype=float)
      6 y_airfare_train = airfare_train.price

~\anaconda3\lib\site-packages\pandas\core\frame.py in drop(self, labels, axis,␣
↪index, columns, level, inplace, errors)
   4306                 weight  1.0     0.8
   4307         """
-> 4308         return super().drop(
   4309             labels=labels,
   4310             axis=axis,

~\anaconda3\lib\site-packages\pandas\core\generic.py in drop(self, labels, axis ␣
↪index, columns, level, inplace, errors)
   4151         for axis, labels in axes.items():
   4152             if labels is not None:
-> 4153                 obj = obj._drop_axis(labels, axis, level=level,␣
↪errors=errors)
   4154
   4155         if inplace:

~\anaconda3\lib\site-packages\pandas\core\generic.py in _drop_axis(self, labels ␣
↪axis, level, errors)
   4186                 new_axis = axis.drop(labels, level=level, errors=errors
   4187             else:
-> 4188                 new_axis = axis.drop(labels, errors=errors)
   4189             result = self.reindex(**{axis_name: new_axis})
   4190

~\anaconda3\lib\site-packages\pandas\core\indexes\base.py in drop(self, labels,␣
↪errors)
   5589         if mask.any():
   5590             if errors != "ignore":
-> 5591                 raise KeyError(f"{labels[mask]} not found in axis")
   5592             indexer = indexer[~mask]
   5593         return self.delete(indexer)

KeyError: "['price'] not found in axis"
```

4 EXERCISE 3

4.1 Steplength Control for Gradient Descent

[ ]:
```
## steplength-backtracking
```

[ ]:
```
## steplength-bolddriver
```

[ ]:
```
## Look-ahead optimizer
```