SGD_For_Linear_Reg

June 3, 2020

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import load_boston
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import pandas as pd
import sklearn.metrics as metrics
import matplotlib.pyplot as plt
from numpy import random
import math
from statistics import mean
from scipy.signal import argrelextrema
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19:
FutureWarning: pandas.util.testing is deprecated. Use the functions in the
public API at pandas.testing instead.
  import pandas.util.testing as tm
```

0.1 Loading boston dataset to use in SkLearn Linear regression and custom Linear regression

[0]:
```
boston=load_boston()
bos = pd.DataFrame(boston.data)

bos['PRICE'] = load_boston().target

X = bos.drop('PRICE', axis = 1)
Y = bos['PRICE']
Y=Y.values.reshape(-1,1)

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.33,␣
↪random_state = 5)
```

0.2 Defining MSE loss function

[0]:
```
def sq_error(ys_orig, ys_line):
  return ((np.sum(np.subtract(ys_line,ys_orig)**2)/len(ys_orig)))
```

0.3 Using SKlearn Linear regression and getting MSE for it

[4]:
```
lm = LinearRegression()
lm.fit(X_train, Y_train)

Y_pred = lm.predict(X_test)

print('error with skleran algorithm {}'.format(sq_error(Y_test,Y_pred)))
```

```
error with skleran algorithm 28.530458765974597
```

0.4 Formulating Custom Linear regression model

[0]:
```
''' Mathematical formulation of
linear regression'''


def add_1_to_start(mat):
  r,c=mat.shape
  mat=np.c_[ mat, np.ones(r) ]
  mat[:,[0,-1]]=mat[:,[-1,0]]
  return mat



def best_fit_parameter(xs,ys):
  xs=add_1_to_start(xs)

  beta= np.linalg.inv(xs.T.dot(xs)).dot(xs.T.dot(ys))


  return beta
```

0.5 Getting MSE for Custom Linear Regression model

[6]:
```
weights= best_fit_parameter(X_train,Y_train)


reg_lr =add_1_to_start(X_test).dot(weights)

print('error with devised algorithm {}'.format(sq_error(Y_test,reg_lr)))
```

```
error with devised algorithm 28.53045876597665
```

0.6 Loading Anew the boston dataset to work with SGD

[0]:
```
boston=load_boston()
bos = pd.DataFrame(boston.data)

bos['PRICE'] = load_boston().target

X = bos.drop('PRICE', axis = 1)
Y_B= bos['PRICE']
Y_B=Y_B.values.reshape(-1,1)
```

0.7 Finding the importance of each column

[8]:
```
from sklearn.ensemble import ExtraTreesRegressor

feature= ExtraTreesRegressor()
fit= feature.fit(X,Y)

series= pd.DataFrame(data={'feature':X.columns,'score':fit.
 ↪feature_importances_})
series=series.nlargest(13,'score')
series
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:4:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples,), for example using
ravel().
  after removing the cwd from sys.path.
```

[8]:

	feature	score
5	5	0.349578
12	12	0.319401
10	10	0.059426
9	9	0.046548
4	4	0.043025
2	2	0.036391
7	7	0.033221
0	0	0.027991
8	8	0.022631
6	6	0.021163
3	3	0.018433
11	11	0.017371
1	1	0.004821

0.8 Making a new dataframe with columns in decreasing order of importance

[0]:
```
df_feature = pd.DataFrame()
count=0
for i,feat in enumerate(series['feature']):
  df_feature[i]= X[feat]
```

[10]:
```
df_feature.head(2)
```

[10]:
```
      0     1     2      3      4     5   ...       7    8     9    10     11
12
0  6.575  4.98  15.3  296.0  0.538  2.31  ...  0.00632  1.0  65.2  0.0  396.9
18.0
1  6.421  9.14  17.8  242.0  0.469  7.07  ...  0.02731  2.0  78.9  0.0  396.9
0.0

[2 rows x 13 columns]
```

0.9 Using only top 8 columns of importance

[0]:
```
X_New=df_feature.iloc[:,:8]
Y_New=Y_B
```

[12]:
```
print(Y_New.shape)
print(X_New.shape)
```

```
(506, 1)
(506, 8)
```

0.10 Defining A function to determine Learning rate with iterations

[0]:
```
def step_decay(it):
   initial_lrate = 0.00000000001
   drop = 0.5
   it_drop = 1000.0
   lrate = initial_lrate * math.pow(drop,
           math.floor((1+it)/it_drop))
   return lrate
```

[0]:
```
def step_decay2(it):
   initial_lrate = 0.0000000000000001
   drop = 0.5
   it_drop = 1000.0
   lrate = initial_lrate * math.pow(drop,
           math.floor((1+it)/it_drop))
   return lrate
```

0.11 Defining a function which will run a batch of data through it and return the weight update which has the least train error (trying to mimic checkpoints in keras)

[0]:
```
def update(x_data, X, y_data, Y, beta,lr_func):

  it = 0

  x=x_data
  y=y_data
  b_deriv = np.full((len(x[0]),1),0)

  learning_rate = lr_func(it)
  error_tr=[]
  b0= beta[:x.shape[1]]
  b=[]
  while True:

    error_tr.append(sq_error(y,x.dot(b0)))
    b.append(b0)

   #adding l2 regularizer derivative too
    lam=b0.T.dot(b0)

    b_deriv = b_deriv+(-2*(x.T.dot(y-x.dot(b0)))) + lam*2*b0

    b1 = b0 - (b_deriv * learning_rate)

    #running itereations for  20k times
    if it==20000:
      if b0.all()==b1.all():
        return b0
        break

      else:
        return b[np.argmin(error_tr)]
        break

    else:
      it += 1
      b0 = b1
      learning_rate= lr_func(it)
```

0.12 Defining SGD to pass batchs of size 'n' to update function resulting in an epoch

[0]:
```
def SGD(X1, Y1,beta,lr_func):

  error_tr = []
  error_ts = []

  global X, Y


  xtr=X1


  ytr = Y1

  N=len(xtr)
  minn=np.inf
  k=0
  n=100
  for i in range((N//n)+1):
    if N>k:
      k+=n

      beta=update(xtr[k-n:k],X,ytr[k-n:k],Y,beta,lr_func)
      # error_tr.extend(tr)
      # error_ts.extend(ts)
      tr=sq_error(ytr,xtr.dot(beta))
      ts=sq_error(Y,X.dot(beta))
      if minn>tr:
        minn=tr
        min2=ts
        beta_min=beta

      # error_ts.append(ts)
      # error_tr.append(tr)

    else:



      beta=update(xtr[k-n:N],X,ytr[k-n:N],Y,beta,lr_func)
      # error_tr.extend(tr)
      # error_ts.extend(ts)
      tr=sq_error(ytr,xtr.dot(beta))
      ts=sq_error(Y,X.dot(beta))
      if minn>tr:
        minn=tr
```

```
        min2=ts
        beta_min=beta

      # error_ts.append(ts)
      # error_tr.append(tr)

  return beta_min
```

0.13 Data preprocessing

[0]:
```
#preprocessing data
xtr, X, ytr, Y = train_test_split(X_New,Y_New, test_size = 0.33)

ss= StandardScaler(with_mean= False)
xtr=ss.fit_transform(xtr)
X=ss.transform(X)

xtr=add_1_to_start(xtr)
X=add_1_to_start(X)
```

[0]:
```
#initializing stuff
tr=[]
ts=[]

lr_func=step_decay
epoch=0
flag_change = False
```

0.14 Running SGD on randomly initialized weights

[78]:
```
c=SGD(xtr,ytr, np.random.randint(0,100, size=(14,1)),lr_func)
ts.append(sq_error(Y,X.dot(c)))
tr.append(sq_error(ytr,xtr.dot(c)))

epoch+=1
print('for epoch {} the train score is {} and test score is {}'.
 ↪format(epoch,tr[-1],ts[-1]))



while True:

  if epoch >= 2 and not(flag_change):
    if float("{:.2f}".format(tr[-1]))==float("{:.2f}".format(tr[-2])) or␣
 ↪float("{:.5f}".format(tr[-1]))>float("{:.5f}".format(tr[-2])):
      lr_func= step_decay2
```

```
      flag_change= True
      print('learnig fuction changed to step_decay2')



  c=SGD(xtr,ytr,c,lr_func)
  ts.append(sq_error(Y,X.dot(c)))
  tr.append(sq_error(ytr,xtr.dot(c)))
  epoch+=1
  print('for epoch {} the train score is {} and test score is {}'.
↪format(epoch,tr[-1],ts[-1]))




  if ts[-1]==ts[-2] or ts[-1]< 24:
    zeros_sgd= X.dot(c)
    pass


  else:

    c=SGD(xtr,ytr,c,lr_func)
    ts.append(sq_error(Y,X.dot(c)))
    tr.append(sq_error(ytr,xtr.dot(c)))
    epoch+=1
    print('for epoch {} the train score is {} and test score is {}'.
↪format(epoch,tr[-1],ts[-1]))
```

```
for epoch 1 the train score is 1162.4983495834683 and test score is
1189.4192008053915
for epoch 2 the train score is 745.3830561680218 and test score is
767.6103366750067
for epoch 3 the train score is 545.0978496345091 and test score is
567.4903129862008
for epoch 4 the train score is 426.28218914504896 and test score is
448.1959498009902
for epoch 5 the train score is 347.48921603671437 and test score is
368.60461865693065
for epoch 6 the train score is 291.54891139064455 and test score is
311.7114311532534
for epoch 7 the train score is 249.96202585310667 and test score is
269.1042618635233
for epoch 8 the train score is 218.0018205163483 and test score is
236.10491893707874
```

```
for epoch 9 the train score is 192.81382579845672 and test score is
209.88703787308785
for epoch 10 the train score is 172.56524593533382 and test score is
188.6343888934054
for epoch 11 the train score is 156.02235682973395 and test score is
171.12295537511915
for epoch 12 the train score is 142.32308527825066 and test score is
156.4961279228402
for epoch 13 the train score is 130.75709373021303 and test score is
148.04962284513206
for epoch 14 the train score is 122.35287004971308 and test score is
138.62618830379833
for epoch 15 the train score is 115.02257122512636 and test score is
130.33100243462565
for epoch 16 the train score is 108.59347010956114 and test score is
122.99559945473138
for epoch 17 the train score is 102.92478936244602 and test score is
116.47539414876323
for epoch 18 the train score is 97.90153795324551 and test score is
110.65178727084398
for epoch 19 the train score is 93.42911149093193 and test score is
105.4267795811548
for epoch 20 the train score is 89.4291713042585 and test score is
100.71884564260974
for epoch 21 the train score is 85.83646384459792 and test score is
96.45974619558316
for epoch 22 the train score is 82.5963401930994 and test score is
92.59203950731931
for epoch 23 the train score is 79.66280340807002 and test score is
89.06711985264887
for epoch 24 the train score is 76.99695838852672 and test score is
85.84365811130132
for epoch 25 the train score is 74.56577187012581 and test score is
82.88635232808943
for epoch 26 the train score is 72.3410736252682 and test score is
80.16491947976772
for epoch 27 the train score is 70.29874686753418 and test score is
77.6532765771343
for epoch 28 the train score is 68.41806823118753 and test score is
75.32887156818985
for epoch 29 the train score is 66.68116684125077 and test score is
73.17213362703377
for epoch 30 the train score is 65.07257882236776 and test score is
71.1660192255958
for epoch 31 the train score is 63.578878750676665 and test score is
69.29563552531137
for epoch 32 the train score is 62.18837347955824 and test score is
67.54792653986557
```

```
for epoch 33 the train score is 60.89084678609971 and test score is
65.91141052615706
for epoch 34 the train score is 59.67734562001791 and test score is
64.37595938775351
for epoch 35 the train score is 58.54000055768741 and test score is
62.9326126899732
for epoch 36 the train score is 57.471874493712825 and test score is
61.57342031090227
for epoch 37 the train score is 56.466834732263365 and test score is
60.29130887897515
for epoch 38 the train score is 55.51944453837237 and test score is
59.0799680432072
for epoch 39 the train score is 54.624870927079584 and test score is
57.93375333808117
for epoch 40 the train score is 53.77880604476612 and test score is
56.84760298041905
for epoch 41 the train score is 52.97739996229549 and test score is
55.81696640016207
for epoch 42 the train score is 52.21720307674967 and test score is
54.8377426839031
for epoch 43 the train score is 51.49511662558284 and test score is
53.90622741707572
for epoch 44 the train score is 50.808350067910936 and test score is
53.01906666186671
for epoch 45 the train score is 50.1543842934332 and test score is
52.173217014160436
for epoch 46 the train score is 49.53093978885207 and test score is
51.36591085275124
for epoch 47 the train score is 48.935949031463444 and test score is
50.59462603456495
for epoch 48 the train score is 48.36753249539092 and test score is
49.85705940614667
for epoch 49 the train score is 47.82397775212011 and test score is
49.15110359864489
for epoch 50 the train score is 47.30372122708008 and test score is
48.47482665439835
for epoch 51 the train score is 46.80533224090001 and test score is
47.82645410091596
for epoch 52 the train score is 46.327499019964584 and test score is
47.20435314482036
for epoch 53 the train score is 45.86901640785895 and test score is
46.60701870606497
for epoch 54 the train score is 45.428775048815446 and test score is
46.033061053009
for epoch 55 the train score is 45.00575184756473 and test score is
45.481194832931195
for epoch 56 the train score is 44.59900153813334 and test score is
44.95022932140286
```

```
for epoch 57 the train score is 44.20764921793012 and test score is
44.439059738380415
for epoch 58 the train score is 43.83088372366152 and test score is
43.94665949969969
for epoch 59 the train score is 43.46795174277143 and test score is
43.47207329038454
for epoch 60 the train score is 43.11815256870774 and test score is
43.014410861336806
for epoch 61 the train score is 42.78083342077995 and test score is
42.57284146394618
for epoch 62 the train score is 42.45538526001236 and test score is
42.1465888482707
for epoch 63 the train score is 42.141239041516975 and test score is
41.734926760008776
for epoch 64 the train score is 41.837862351713625 and test score is
41.3371748796906
for epoch 65 the train score is 41.5447563854437 and test score is
40.95269515462521
for epoch 66 the train score is 41.26145322379513 and test score is
40.580888480265266
for epoch 67 the train score is 40.98751337842308 and test score is
40.221191692934845
for epoch 68 the train score is 40.72252357246218 and test score is
39.873074840499996
for epoch 69 the train score is 40.46609473181801 and test score is
39.536038701517576
for epoch 70 the train score is 40.21786016385684 and test score is
39.20961252689327
for epoch 71 the train score is 39.97747390329261 and test score is
38.89335198110008
for epoch 72 the train score is 39.744609207489944 and test score is
38.58683726264819
for epoch 73 the train score is 39.51895718551629 and test score is
38.289671385807544
for epoch 74 the train score is 39.3002255471033 and test score is
38.001478607605726
for epoch 75 the train score is 39.088137459285306 and test score is
37.72190298590063
for epoch 76 the train score is 38.882430499877046 and test score is
37.4506070558743
for epoch 77 the train score is 38.682855698180944 and test score is
37.18727061367485
for epoch 78 the train score is 38.489176654390555 and test score is
36.931589597140466
for epoch 79 the train score is 38.30116873008829 and test score is
36.68327505458784
for epoch 80 the train score is 38.11861830307714 and test score is
36.44205219361252
```

```
for epoch 81 the train score is 37.941322080506566 and test score is
36.20765950266343
for epoch 82 the train score is 37.76908646489708 and test score is
35.97984793889643
for epoch 83 the train score is 37.60172696823768 and test score is
35.75838017647077
for epoch 84 the train score is 37.439067669828894 and test score is
35.54302991002622
for epoch 85 the train score is 37.28094071398992 and test score is
35.33358120860308
for epoch 86 the train score is 37.127185844141806 and test score is
35.12982791572688
for epoch 87 the train score is 36.977649970127864 and test score is
34.9315730917886
for epoch 88 the train score is 36.832186765941984 and test score is
34.73862849522204
for epoch 89 the train score is 36.690656295313964 and test score is
34.55081409930677
for epoch 90 the train score is 36.55292466284464 and test score is
34.3679576417213
for epoch 91 the train score is 36.418863688605825 and test score is
34.189894204234115
for epoch 92 the train score is 36.28835060431777 and test score is
34.01646582016074
for epoch 93 the train score is 36.16126776938478 and test score is
33.84752110741956
for epoch 94 the train score is 36.037502405239415 and test score is
33.68291492522413
for epoch 95 the train score is 35.916946346574925 and test score is
33.5225080526102
for epoch 96 the train score is 35.799495808182186 and test score is
33.36616688716095
for epoch 97 the train score is 35.685051166214095 and test score is
33.21376316243033
for epoch 98 the train score is 35.5735167528056 and test score is
33.0651736826884
for epoch 99 the train score is 35.464800663075145 and test score is
32.92028007373902
for epoch 100 the train score is 35.358814573607276 and test score is
32.77896854865492
for epoch 101 the train score is 35.25547357159837 and test score is
32.64112968736823
for epoch 102 the train score is 35.154695993915816 and test score is
32.50665822915391
for epoch 103 the train score is 35.05640327537873 and test score is
32.375452877106156
for epoch 104 the train score is 34.96051980562893 and test score is
32.247416113789626
```

```
for epoch 105 the train score is 34.86697279400636 and test score is
32.1224540273065
for epoch 106 the train score is 34.77569214189697 and test score is
32.000476147082594
for epoch 107 the train score is 34.68661032205544 and test score is
31.881395288728864
for epoch 108 the train score is 34.59966226444717 and test score is
31.765127407380376
for epoch 109 the train score is 34.51478524819144 and test score is
31.65159145896887
for epoch 110 the train score is 34.43191879921255 and test score is
31.540709268914053
for epoch 111 the train score is 34.351004593241875 and test score is
31.432405407766055
for epoch 112 the train score is 34.27198636383617 and test score is
31.326607073362272
for epoch 113 the train score is 34.19480981510352 and test score is
31.22324397909398
for epoch 114 the train score is 34.11942253885082 and test score is
31.12224824790513
for epoch 115 the train score is 34.04577393588492 and test score is
31.02355431167843
for epoch 116 the train score is 33.97381514122248 and test score is
30.927098815681266
for epoch 117 the train score is 33.90349895297564 and test score is
30.83282052776884
for epoch 118 the train score is 33.83477976470224 and test score is
30.740660252068484
for epoch 119 the train score is 33.76761350101923 and test score is
30.65056074687951
for epoch 120 the train score is 33.701957556292754 and test score is
30.562466646544994
for epoch 121 the train score is 33.63777073623331 and test score is
30.476324387070992
for epoch 122 the train score is 33.57501320223172 and test score is
30.392082135276254
for epoch 123 the train score is 33.51364641828607 and test score is
30.309689721280563
for epoch 124 the train score is 33.45363310037535 and test score is
30.22909857413624
for epoch 125 the train score is 33.39493716815212 and test score is
30.150261660443526
for epoch 126 the train score is 33.33752369882218 and test score is
30.073133425770504
for epoch 127 the train score is 33.28135888310337 and test score is
29.997669738742232
for epoch 128 the train score is 33.22640998314706 and test score is
29.92382783764272
```

```
for epoch 129 the train score is 33.17264529232435 and test score is
29.851566279405635
for epoch 130 the train score is 33.12003409677907 and test score is
29.780844890863722
for epoch 131 the train score is 33.06854663865571 and test score is
29.71162472213864
for epoch 132 the train score is 33.018154080920986 and test score is
29.64386800206539
for epoch 133 the train score is 32.96882847369416 and test score is
29.57753809554067
for epoch 134 the train score is 32.92054272201474 and test score is
29.5125994627029
for epoch 135 the train score is 32.87327055497354 and test score is
29.449017619847726
for epoch 136 the train score is 32.826986496142645 and test score is
29.386759101997054
for epoch 137 the train score is 32.781665835238826 and test score is
29.325791427033266
for epoch 138 the train score is 32.737284600963406 and test score is
29.26608306133044
for epoch 139 the train score is 32.693819534958756 and test score is
29.207603386799114
for epoch 140 the train score is 32.65124806683202 and test score is
29.150322669286876
for epoch 141 the train score is 32.60954829019263 and test score is
29.094212028262326
for epoch 142 the train score is 32.568698939658994 and test score is
29.03924340772516
for epoch 143 the train score is 32.52867936878789 and test score is
28.985389548285596
for epoch 144 the train score is 32.48946952888145 and test score is
28.93262396035046
for epoch 145 the train score is 32.451049948637504 and test score is
28.880920898377205
for epoch 146 the train score is 32.413401714599765 and test score is
28.83025533613725
for epoch 147 the train score is 32.3765064523737 and test score is
28.780602942943947
for epoch 148 the train score is 32.340346308573785 and test score is
28.731940060804877
for epoch 149 the train score is 32.304903933469205 and test score is
28.68424368245351
for epoch 150 the train score is 32.27016246429737 and test score is
28.637491430222017
for epoch 151 the train score is 32.23610550921581 and test score is
28.591661535718547
for epoch 152 the train score is 32.20271713186435 and test score is
28.54673282027243
```

```
for epoch 153 the train score is 32.169981836511354 and test score is
28.502684676113102
for epoch 154 the train score is 32.137884553759065 and test score is
28.45949704825374
for epoch 155 the train score is 32.106410626782356 and test score is
28.417150417044663
for epoch 156 the train score is 32.07554579808229 and test score is
28.375625781373344
for epoch 157 the train score is 32.04527619672693 and test score is
28.334904642477788
for epoch 158 the train score is 32.015588326064 and test score is
28.294968988349964
for epoch 159 the train score is 31.986469051882903 and test score is
28.255801278707224
for epoch 160 the train score is 31.957905591008327 and test score is
28.21738443050219
for epoch 161 the train score is 31.929885500306277 and test score is
28.179701803951858
for epoch 162 the train score is 31.90239666608787 and test score is
28.142737189065752
for epoch 163 the train score is 31.875427293894045 and test score is
28.106474792651547
for epoch 164 the train score is 31.84896589864137 and test score is
28.070899225774347
for epoch 165 the train score is 31.82300129512274 and test score is
28.035995491660298
for epoch 166 the train score is 31.797522588840327 and test score is
28.001748974018188
for epoch 167 the train score is 31.772519167163285 and test score is
27.96814542576772
for epoch 168 the train score is 31.747980690793025 and test score is
27.935170958153105
for epoch 169 the train score is 31.723897085527238 and test score is
27.902812030232546
for epoch 170 the train score is 31.700258534309935 and test score is
27.871055438725868
for epoch 171 the train score is 31.677055469554087 and test score is
27.839888308204557
for epoch 172 the train score is 31.654278565729296 and test score is
27.809298081613644
for epoch 173 the train score is 31.63191873220274 and test score is
27.779272511112193
for epoch 174 the train score is 31.609967106324053 and test score is
27.749799649219558
for epoch 175 the train score is 31.5884150467423 and test score is
27.720867840252996
for epoch 176 the train score is 31.567254126952065 and test score is
27.692465712053572
```

```
for epoch 177 the train score is 31.546476129052888 and test score is
27.664582167979155
for epoch 178 the train score is 31.526073037718653 and test score is
27.637206379161306
for epoch 179 the train score is 31.50603703436673 and test score is
27.61032777701377
for epoch 180 the train score is 31.48636049152207 and test score is
27.583936045985883
for epoch 181 the train score is 31.4670359673654 and test score is
27.55802111654629
for epoch 182 the train score is 31.44805620045922 and test score is
27.532573158392314
for epoch 183 the train score is 31.429414104649634 and test score is
27.507582573878498
for epoch 184 the train score is 31.411102764130483 and test score is
27.48303999165094
for epoch 185 the train score is 31.393115428670573 and test score is
27.458936260486603
for epoch 186 the train score is 31.375445508991014 and test score is
27.435262443321374
for epoch 187 the train score is 31.35808657229516 and test score is
27.412009811470778
for epoch 188 the train score is 31.341032337937936 and test score is
27.3891698390258
for epoch 189 the train score is 31.32427667323478 and test score is
27.366734197426208
for epoch 190 the train score is 31.30781358940292 and test score is
27.34469475019945
for epoch 191 the train score is 31.291637237631914 and test score is
27.323043547863417
for epoch 192 the train score is 31.275741905274874 and test score is
27.30177282298139
for epoch 193 the train score is 31.260122012161876 and test score is
27.280874985372535
for epoch 194 the train score is 31.24477210702712 and test score is
27.260342617464705
for epoch 195 the train score is 31.229686864045878 and test score is
27.24016846978663
for epoch 196 the train score is 31.214861079479757 and test score is
27.220345456598217
for epoch 197 the train score is 31.200289668425587 and test score is
27.200866651650877
for epoch 198 the train score is 31.185967661661678 and test score is
27.18172528407221
for epoch 199 the train score is 31.17189020259312 and test score is
27.162914734375434
for epoch 200 the train score is 31.158052544287635 and test score is
27.144428530585184
```

```
for epoch 201 the train score is 31.144450046602522 and test score is
27.126260344477597
for epoch 202 the train score is 31.131078173397682 and test score is
27.10840398793002
for epoch 203 the train score is 31.11793248983211 and test score is
27.09085340937813
for epoch 204 the train score is 31.10500865974183 and test score is
27.07360269037498
for epoch 205 the train score is 31.092302443094336 and test score is
27.05664604224889
for epoch 206 the train score is 31.079809693521405 and test score is
27.03997780286056
for epoch 207 the train score is 31.06752635592111 and test score is
27.02359243344816
for epoch 208 the train score is 31.05544846413383 and test score is
27.00748451556805
for epoch 209 the train score is 31.0435721386836 and test score is
26.991648748118276
for epoch 210 the train score is 31.031893584587475 and test score is
26.976079944448134
for epoch 211 the train score is 31.02040908922627 and test score is
26.960773029547813
for epoch 212 the train score is 31.009115020279598 and test score is
26.945723037319503
for epoch 213 the train score is 30.998007823718265 and test score is
26.93092510792263
for epoch 214 the train score is 30.987084021854937 and test score is
26.91637448519267
for epoch 215 the train score is 30.976340211451486 and test score is
26.90206651413394
for epoch 216 the train score is 30.965773061878647 and test score is
26.887996638479056
for epoch 217 the train score is 30.955379313329463 and test score is
26.8741603983178
for epoch 218 the train score is 30.9451557750818 and test score is
26.860553427788677
for epoch 219 the train score is 30.935099323811098 and test score is
26.847171452835255
for epoch 220 the train score is 30.92520690194912 and test score is
26.834010289022373
for epoch 221 the train score is 30.915475516090076 and test score is
26.82106583941324
for epoch 222 the train score is 30.90590223543945 and test score is
26.808334092500978
for epoch 223 the train score is 30.896484190306964 and test score is
26.795811120198234
for epoch 224 the train score is 30.88721857064129 and test score is
26.783493075880035
```

```
for epoch 225 the train score is 30.878102624604182 and test score is
26.771376192478385
for epoch 226 the train score is 30.869133657183657 and test score is
26.759456780625857
for epoch 227 the train score is 30.860309028846103 and test score is
26.747731226853208
for epoch 228 the train score is 30.851626154224736 and test score is
26.73619599182876
for epoch 229 the train score is 30.843082500842296 and test score is
26.724847608647988
for epoch 230 the train score is 30.834675587870407 and test score is
26.71368268116693
for epoch 231 the train score is 30.826402984919824 and test score is
26.70269788237705
learnig fuction changed to step_decay2
for epoch 232 the train score is 30.826395942030263 and test score is
26.702687889493525
for epoch 233 the train score is 30.826388899748633 and test score is
26.702677897252045
for epoch 234 the train score is 30.826381858074917 and test score is
26.702667905652614
for epoch 235 the train score is 30.826374817009125 and test score is
26.702657914695223
for epoch 236 the train score is 30.82636777655119 and test score is
26.702647924379818
for epoch 237 the train score is 30.826360736701155 and test score is
26.702637934706452
for epoch 238 the train score is 30.826353697458945 and test score is
26.70262794567503
for epoch 239 the train score is 30.826346658824562 and test score is
26.702617957285575
for epoch 240 the train score is 30.82633962079798 and test score is
26.70260796953802
for epoch 241 the train score is 30.826332583379177 and test score is
26.702597982432373
for epoch 242 the train score is 30.826325546568146 and test score is
26.702587995968603
for epoch 243 the train score is 30.82631851036486 and test score is
26.702578010146723
for epoch 244 the train score is 30.8263114747693 and test score is
26.7025680249667
for epoch 245 the train score is 30.826304439781467 and test score is
26.702558040428503
for epoch 246 the train score is 30.826297405401302 and test score is
26.702548056532116
for epoch 247 the train score is 30.826290371628836 and test score is
26.702538073277527
for epoch 248 the train score is 30.82628333846403 and test score is
```

```
26.702528090664725
for epoch 249 the train score is 30.82627630590683 and test score is
26.702518108693667
for epoch 250 the train score is 30.826269273957244 and test score is
26.702508127364332
for epoch 251 the train score is 30.826262242615215 and test score is
26.70249814667669
for epoch 252 the train score is 30.826255211880767 and test score is
26.70248816663072
for epoch 253 the train score is 30.82624818175393 and test score is
26.702478187226486
for epoch 254 the train score is 30.826241152234672 and test score is
26.702468208464012
for epoch 255 the train score is 30.826234123322852 and test score is
26.70245823034304
for epoch 256 the train score is 30.82622709501852 and test score is
26.70244825286365
for epoch 257 the train score is 30.826220067321646 and test score is
26.702438276025866
for epoch 258 the train score is 30.826213040232275 and test score is
26.702428299829716
for epoch 259 the train score is 30.82620601375036 and test score is
26.702418324275108
for epoch 260 the train score is 30.82619898787587 and test score is
26.702408349362045
for epoch 261 the train score is 30.826191962608732 and test score is
26.702398375090453
```

```
␣
↪---------------------------------------------------------------------------

       KeyboardInterrupt                         Traceback (most recent call␣
↪last)

       <ipython-input-78-731935d5a201> in <module>()
        18
        19
   ---> 20   c=SGD(xtr,ytr,c,lr_func)
        21   ts.append(sq_error(Y,X.dot(c)))
        22   tr.append(sq_error(ytr,xtr.dot(c)))


       <ipython-input-76-2b20a7ba1ee6> in SGD(X1, Y1, beta, lr_func)
        20       k+=n
        21
   ---> 22       beta=update(xtr[k-n:k],X,ytr[k-n:k],Y,beta,lr_func)
        23       # error_tr.extend(tr)
```

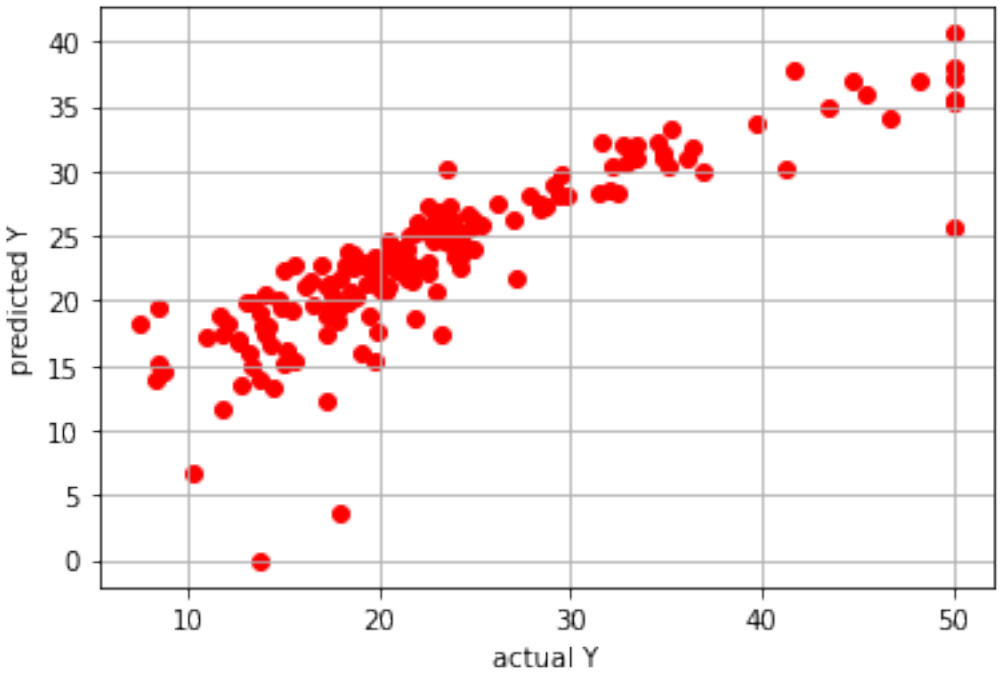
```
        24       # error_ts.extend(ts)


      <ipython-input-75-009f9d3445a9> in update(x_data, X, y_data, Y, beta,␣
↪lr_func)
        13   while True:
        14
   ---> 15     error_tr.append(sq_error(y,x.dot(b0)))
        16     b.append(b0)
        17


      KeyboardInterrupt:
```

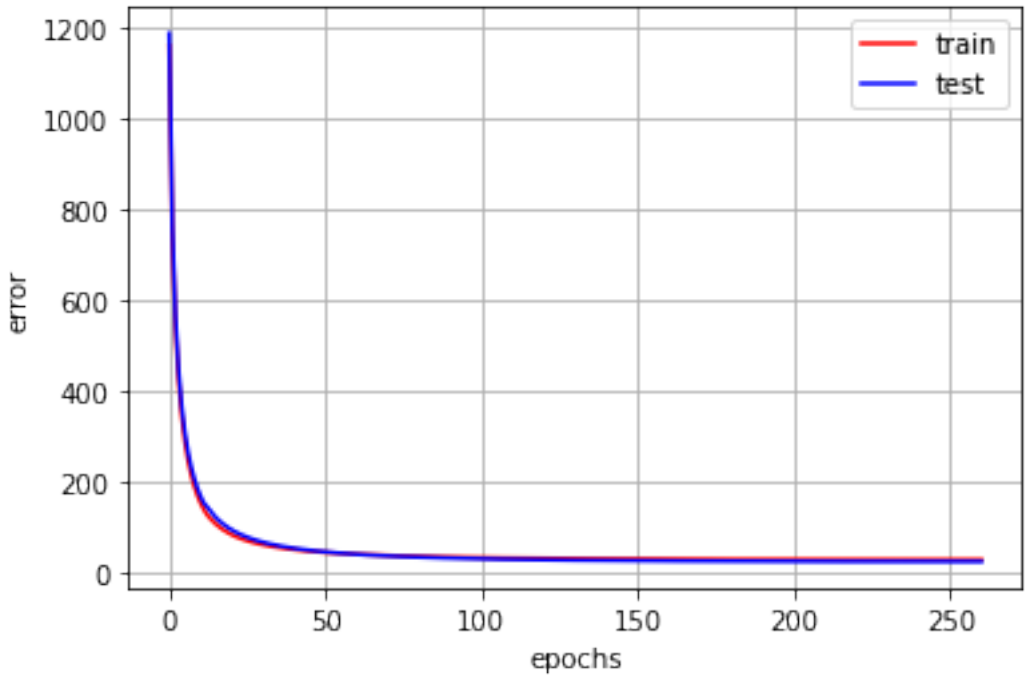
0.15 Forcefully interrupted the loop with keyboard interrupt as no progress can be seen

[98]:
```
plt.scatter(Y,X.dot(c), color='r')
plt.xlabel('actual Y')
plt.ylabel('predicted Y')
plt.grid()
```



[104]:
```
plt.plot(tr, color='r',label='train')
plt.plot(ts, color='b', label= 'test')
```

```
plt.ylabel('error')
plt.xlabel('epochs')
plt.legend()
plt.grid()
```



[86]:
```
'''we see TRAIN ERROR HIGHER THAN TEST ERROR as model generalizes well enough
but Test data size is too small hence these reults'''

print('Train error {}'.format(sq_error(ytr,xtr.dot(c))))

print('Test error {}'.format(sq_error(Y,X.dot(c))))
```

```
Train error 30.826191962608732
Test error 26.702398375090453
```

[0]:
```
from prettytable import PrettyTable
x = PrettyTable()
x.field_names = ["S.NO.", "MODEL",  "Error value"]
x.add_row(["1", "Skleran Algorithm", sq_error(Y_test,Y_pred)])
x.add_row(["2", "Formulated Algorithm", sq_error(Y_test,reg_lr)])
x.add_row(["3", "SGD with random weights initialization",sq_error(Y,X.dot(c))])
```

[106]:
```
print(x)
```

```
+-------+---------------------------------------+--------------------+
| S.NO. |                 MODEL                 |    Error value     |
+-------+---------------------------------------+--------------------+
|   1   |           Skleran Algorithm           | 28.530458765974597 |
|   2   |          Formulated Algorithm         | 28.53045876597665  |
|   3   | SGD with random weights initialization | 26.702398375090453 |
+-------+---------------------------------------+--------------------+
```

[107]:
```
!sudo apt-get install pandoc texlive-xetex
!jupyter nbconvert --to pdf SGD_for_Linear_reg.ipynb
```

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
pandoc is already the newest version (1.19.2.4~dfsg-1build4).
pandoc set to manually installed.
The following additional packages will be installed:
  fonts-droid-fallback fonts-lato fonts-lmodern fonts-noto-mono fonts-texgyre
  javascript-common libcupsfilters1 libcupsimage2 libgs9 libgs9-common
  libijs-0.35 libjbig2dec0 libjs-jquery libkpathsea6 libpotrace0 libptexenc1
  libruby2.5 libsynctex1 libtexlua52 libtexluajit2 libzzip-0-13 lmodern
  poppler-data preview-latex-style rake ruby ruby-did-you-mean ruby-minitest
  ruby-net-telnet ruby-power-assert ruby-test-unit ruby2.5
  rubygems-integration t1utils tex-common tex-gyre texlive-base
  texlive-binaries texlive-fonts-recommended texlive-latex-base
  texlive-latex-extra texlive-latex-recommended texlive-pictures
  texlive-plain-generic tipa
Suggested packages:
  fonts-noto apache2 | lighttpd | httpd poppler-utils ghostscript
  fonts-japanese-mincho | fonts-ipafont-mincho fonts-japanese-gothic
  | fonts-ipafont-gothic fonts-arphic-ukai fonts-arphic-uming fonts-nanum ri
  ruby-dev bundler debhelper gv | postscript-viewer perl-tk xpdf-reader
  | pdf-viewer texlive-fonts-recommended-doc texlive-latex-base-doc
  python-pygments icc-profiles libfile-which-perl
  libspreadsheet-parseexcel-perl texlive-latex-extra-doc
  texlive-latex-recommended-doc texlive-pstricks dot2tex prerex ruby-tcltk
  | libtcltk-ruby texlive-pictures-doc vprerex
The following NEW packages will be installed:
  fonts-droid-fallback fonts-lato fonts-lmodern fonts-noto-mono fonts-texgyre
  javascript-common libcupsfilters1 libcupsimage2 libgs9 libgs9-common
  libijs-0.35 libjbig2dec0 libjs-jquery libkpathsea6 libpotrace0 libptexenc1
  libruby2.5 libsynctex1 libtexlua52 libtexluajit2 libzzip-0-13 lmodern
  poppler-data preview-latex-style rake ruby ruby-did-you-mean ruby-minitest
  ruby-net-telnet ruby-power-assert ruby-test-unit ruby2.5
  rubygems-integration t1utils tex-common tex-gyre texlive-base
  texlive-binaries texlive-fonts-recommended texlive-latex-base
  texlive-latex-extra texlive-latex-recommended texlive-pictures
```

```
  texlive-plain-generic texlive-xetex tipa
0 upgraded, 46 newly installed, 0 to remove and 32 not upgraded.
Need to get 146 MB of archives.
After this operation, 460 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu bionic/main amd64 fonts-droid-fallback
all 1:6.0.1r16-1.1 [1,805 kB]
Get:2 http://archive.ubuntu.com/ubuntu bionic/main amd64 fonts-lato all 2.0-2
[2,698 kB]
Get:3 http://archive.ubuntu.com/ubuntu bionic/main amd64 poppler-data all
0.4.8-2 [1,479 kB]
Get:4 http://archive.ubuntu.com/ubuntu bionic/main amd64 tex-common all 6.09
[33.0 kB]
Get:5 http://archive.ubuntu.com/ubuntu bionic/main amd64 fonts-lmodern all
2.004.5-3 [4,551 kB]
Get:6 http://archive.ubuntu.com/ubuntu bionic/main amd64 fonts-noto-mono all
20171026-2 [75.5 kB]
Get:7 http://archive.ubuntu.com/ubuntu bionic/universe amd64 fonts-texgyre all
20160520-1 [8,761 kB]
Get:8 http://archive.ubuntu.com/ubuntu bionic/main amd64 javascript-common all
11 [6,066 B]
Get:9 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libcupsfilters1
amd64 1.20.2-0ubuntu3.1 [108 kB]
Get:10 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libcupsimage2
amd64 2.2.7-1ubuntu2.8 [18.6 kB]
Get:11 http://archive.ubuntu.com/ubuntu bionic/main amd64 libijs-0.35 amd64
0.35-13 [15.5 kB]
Get:12 http://archive.ubuntu.com/ubuntu bionic/main amd64 libjbig2dec0 amd64
0.13-6 [55.9 kB]
Get:13 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libgs9-common
all 9.26~dfsg+0-0ubuntu0.18.04.12 [5,092 kB]
Get:14 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libgs9 amd64
9.26~dfsg+0-0ubuntu0.18.04.12 [2,264 kB]
Get:15 http://archive.ubuntu.com/ubuntu bionic/main amd64 libjs-jquery all
3.2.1-1 [152 kB]
Get:16 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libkpathsea6
amd64 2017.20170613.44572-8ubuntu0.1 [54.9 kB]
Get:17 http://archive.ubuntu.com/ubuntu bionic/main amd64 libpotrace0 amd64
1.14-2 [17.4 kB]
Get:18 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libptexenc1
amd64 2017.20170613.44572-8ubuntu0.1 [34.5 kB]
Get:19 http://archive.ubuntu.com/ubuntu bionic/main amd64 rubygems-integration
all 1.11 [4,994 B]
Get:20 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 ruby2.5 amd64
2.5.1-1ubuntu1.6 [48.6 kB]
Get:21 http://archive.ubuntu.com/ubuntu bionic/main amd64 ruby amd64 1:2.5.1
[5,712 B]
Get:22 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 rake all
12.3.1-1ubuntu0.1 [44.9 kB]
```

```
24% [22 rake 42.0 kB/44.9 kB 93%]^C
[NbConvertApp] WARNING | pattern u'SGD_for_Linear_reg.ipynb' matched no files
This application is used to convert notebook files (*.ipynb) to various other
formats.

WARNING: THE COMMANDLINE INTERFACE MAY CHANGE IN FUTURE RELEASES.

Options
-------

Arguments that take values are actually convenience aliases to full
Configurables, whose aliases are listed on the help line. For more information
on full configurables, see '--help-all'.

--execute
    Execute the notebook prior to export.
--allow-errors
    Continue notebook execution even if one of the cells throws an error and
include the error message in the cell output (the default behaviour is to abort
conversion). This flag is only relevant if '--execute' was specified, too.
--no-input
    Exclude input cells and output prompts from converted document.
    This mode is ideal for generating code-free reports.
--stdout
    Write notebook output to stdout instead of files.
--stdin
    read a single notebook file from stdin. Write the resulting notebook with
default basename 'notebook.*'
--inplace
    Run nbconvert in place, overwriting the existing notebook (only
    relevant when converting to notebook format)
-y
    Answer yes to any questions instead of prompting.
--clear-output
    Clear output of current file and save in place,
    overwriting the existing notebook.
--debug
    set log level to logging.DEBUG (maximize logging output)
--no-prompt
    Exclude input and output prompts from converted document.
--generate-config
    generate default config file
--nbformat=<Enum> (NotebookExporter.nbformat_version)
    Default: 4
    Choices: [1, 2, 3, 4]
    The nbformat version to write. Use this to downgrade notebooks.
--output-dir=<Unicode> (FilesWriter.build_directory)
    Default: ''
```

```
    Directory to write output(s) to. Defaults to output to the directory of each
    notebook. To recover previous default behaviour (outputting to the current
    working directory) use . as the flag value.
--writer=<DottedObjectName> (NbConvertApp.writer_class)
    Default: 'FilesWriter'
    Writer class used to write the  results of the conversion
--log-level=<Enum> (Application.log_level)
    Default: 30
    Choices: (0, 10, 20, 30, 40, 50, 'DEBUG', 'INFO', 'WARN', 'ERROR',
'CRITICAL')
    Set the log level by value or name.
--reveal-prefix=<Unicode> (SlidesExporter.reveal_url_prefix)
    Default: u''
    The URL prefix for reveal.js (version 3.x). This defaults to the reveal CDN,
    but can be any url pointing to a copy  of reveal.js.
    For speaker notes to work, this must be a relative path to a local  copy of
    reveal.js: e.g., "reveal.js".
    If a relative path is given, it must be a subdirectory of the current
    directory (from which the server is run).
    See the usage documentation
    (https://nbconvert.readthedocs.io/en/latest/usage.html#reveal-js-html-
    slideshow) for more details.
--to=<Unicode> (NbConvertApp.export_format)
    Default: 'html'
    The export format to be used, either one of the built-in formats
    ['asciidoc', 'custom', 'html', 'latex', 'markdown', 'notebook', 'pdf',
    'python', 'rst', 'script', 'slides'] or a dotted object name that represents
    the import path for an `Exporter` class
--template=<Unicode> (TemplateExporter.template_file)
    Default: u''
    Name of the template file to use
--output=<Unicode> (NbConvertApp.output_base)
    Default: ''
    overwrite base name use for output files. can only be used when converting
    one notebook at a time.
--post=<DottedOrNone> (NbConvertApp.postprocessor_class)
    Default: u''
    PostProcessor class used to write the results of the conversion
--config=<Unicode> (JupyterApp.config_file)
    Default: u''
    Full path of a config file.

To see all available configurables, use `--help-all`

Examples
--------

    The simplest way to use nbconvert is
```

```
    > jupyter nbconvert mynotebook.ipynb

    which will convert mynotebook.ipynb to the default format (probably HTML).

    You can specify the export format with `--to`.
    Options include ['asciidoc', 'custom', 'html', 'latex', 'markdown',
'notebook', 'pdf', 'python', 'rst', 'script', 'slides'].

    > jupyter nbconvert --to latex mynotebook.ipynb

    Both HTML and LaTeX support multiple output templates. LaTeX includes
    'base', 'article' and 'report'.  HTML includes 'basic' and 'full'. You
    can specify the flavor of the format used.

    > jupyter nbconvert --to html --template basic mynotebook.ipynb

    You can also pipe the output to stdout, rather than a file

    > jupyter nbconvert mynotebook.ipynb --stdout

    PDF is generated via latex

    > jupyter nbconvert mynotebook.ipynb --to pdf

    You can get (and serve) a Reveal.js-powered slideshow

    > jupyter nbconvert myslides.ipynb --to slides --post serve

    Multiple notebooks can be given at the command line in a couple of
    different ways:

    > jupyter nbconvert notebook*.ipynb
    > jupyter nbconvert notebook1.ipynb notebook2.ipynb

    or you can specify the notebooks list in a config file, containing::

        c.NbConvertApp.notebooks = ["my_notebook.ipynb"]

    > jupyter nbconvert --config mycfg.py
```

[0]: