Nipun garg 2017CH10224

Q1

a) Apply Batch LMS, Stochastic LMS and Least Square closed form solution and compare the results. Plot the graphs of the obtained results and training data. Use the learning rate of 0.1. Analyze the results. (Convergence time, accuracy etc.)(Don't use in-built packages.)

1. Batch LMS

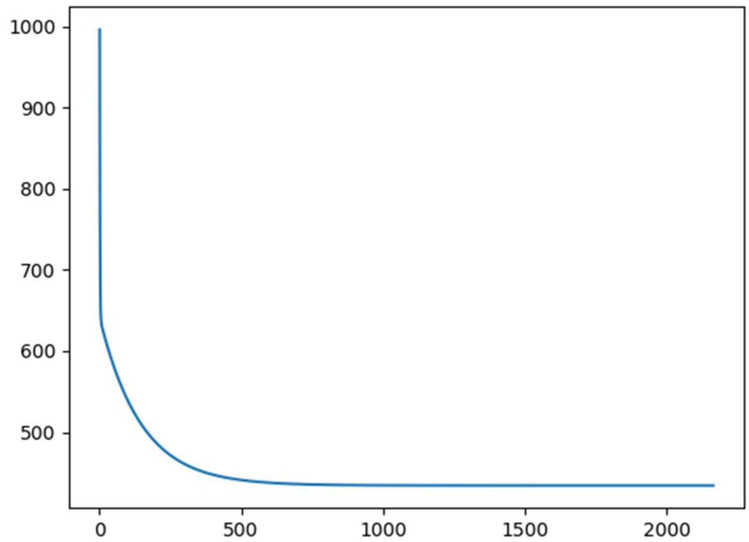
   Learning rate used = 0.0001

```
while(abs(t0-t0old)>pow(10,-8)):
    t0old=t0
    t1old=t1
    val0 = 0.0
    val1 = 0.0
    cost = 0.0
    for i in range(1, sh.nrows):
        val0 += sh.cell_value(i,1)-t0old-t1old*sh.cell_value(i,0)
        val1 +=(sh.cell_value(i,1)-t0old-t1old*sh.cell_value(i,0))*sh.cell_value(i,0)
        cost += pow((sh.cell_value(i,1)-t0old-t1old*sh.cell_value(i,0)),2)
    cost=0.5*cost
    k+=1
    n+=1
    t0=t0old+2*alpha*val0
    hello.append(cost)
    t1=t1old+2*alpha*val1
```

Results

Theta0 =-3.9121814841758518 Theta1 = 1.1927443182285786

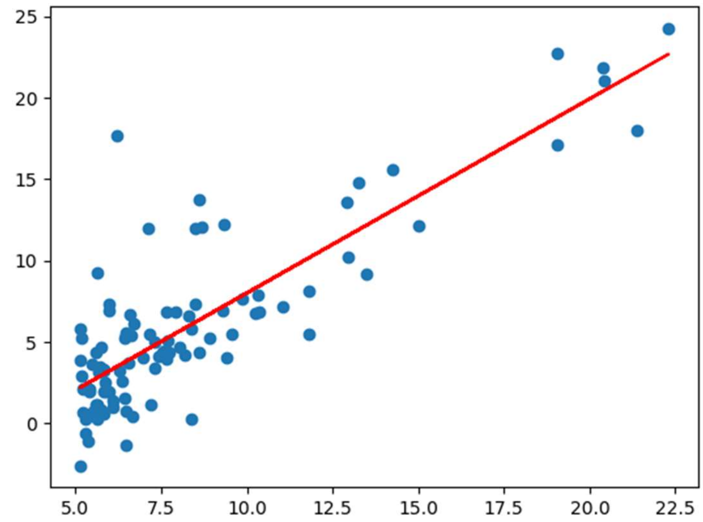
Convergence criteria subsequent theta values < 10power-5



Cost wrt iterations

Nipun garg 2017CH10224

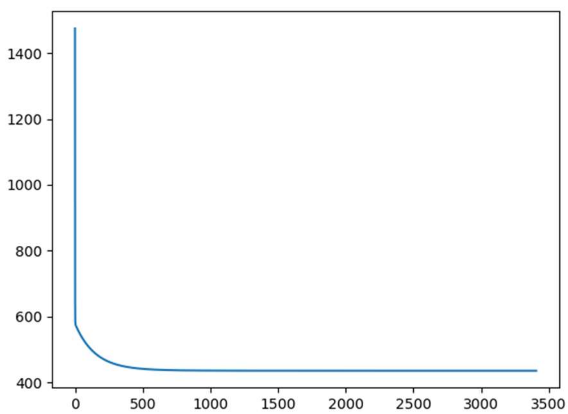
Scatter plot



Stochastic -3.822544830773419 1.1143435302224423
Alpha = 0.0001
Convergence = 10 power -5

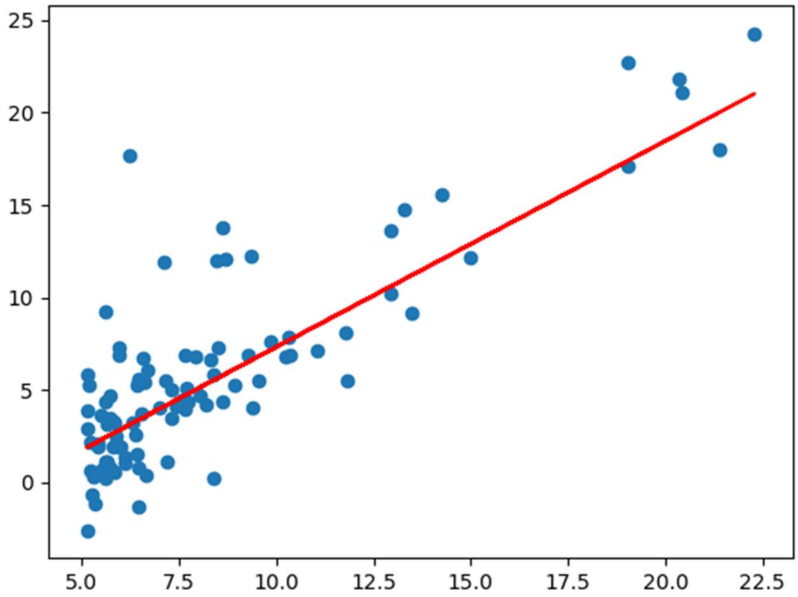
```
chastic.py > ...
for i in range(1, sh.nrows):
    y.append(sh.cell_value(i,1))
    x.append(sh.cell_value(i,0))
start=time.time()
while(abs(delta)>pow(10,-8)):
    cost=0.0
    for j in range(1, sh.nrows):
        t0old=t0
        t1old=t1
        #print(t0old, t1old)

        val0 = sh.cell_value(j,1)-t0old-t1old*sh.cell_value(j,0)
        val1 =(sh.cell_value(j,1)-t0old-t1old*sh.cell_value(j,0))*sh.cell_value(j,0)
        cost += pow((sh.cell_value(j,1)-t0old-t1old*sh.cell_value(j,0)),2)
        t0=t0old+2*alpha*val0
        t1=t1old+2*alpha*val1
    delta=t1-temp
    hi.append(cost/2)
    temp=t1
```



Cost vs iterations

Nipun garg 2017CH10224



Scatter plot

Closed form approach
Results [-3.91508424 1.19303364]

```
least_square.py > ...
sh = book.sheet_by_index(0)
y=[]
x=[]
for i in range(1, sh.nrows):
    y.append(sh.cell_value(i,1))
    x.append(sh.cell_value(i,0))
A=np.ones(sh.nrows-1)
X=np.asarray(x)
X = np.vstack([A, X])
Y=np.asarray(y)
transpose= X.transpose()
multi=np.dot(X,transpose)
print(multi)
multi2=np.dot(X,Y)
print(multi2)
multi3=np.linalg.inv(multi)
multi4=np.dot(multi3,multi2)
print(multi4)

#theta=np.dot(np.linalg.inv(np.dot(X,X.transpose())),X.transpose(),Y)
```

As we can observe from the above plots
Convergence time Batch>Stochastic(based on no of iterations)
Accuracy Batch>Stochastic

Nipun garg 2017CH10224

b) Manually perform the locally weighted least linear regression using the first four data points given in excel sheet. Query point is 7.576 and bandwidth parameter is 0.5. Perform four iterations by using stochastic LMS.

Question 1(b)

Manually perform locally weighted regression

Population in 10000's	Profit in Lakhs (L)
6.2101	17.692
5.6277	9.23
8.6186	13.762
7.1032	11.954

Query pt. = 7.576

Band width Para = (τ) = 0.5

Fitting θ to minimise

$$\sum_{i=1} w^{(i)}\left(y^{(i)} - \theta^T x^{(i)}\right)^2$$

So, $w^{(i)} = \exp\left(-\frac{(x^{(i)} - x)^2}{2\tau^2}\right)$ $x \to$ query point.

$$w^{(1)} = \exp\left(-\frac{(6.201 - 7.576)^2}{2\times(0.5)^2}\right) = 0.0239$$

llly $w^{(2)} = 5.61 \times 10^{-4}$

$w^{(3)} = 0.1137$

$w^{(4)} = 0.639$

$$J(\theta) = \frac{1}{2}\sum_i w^{(i)}\left(\theta^T x^i - y^{(i)}\right)^2$$

& For all gradient discent.

Nipun garg 2017CH10224

Let θ $[\theta_0, \theta_1] = [0, 0]$

let $\alpha = 0.1$

1st iteration.

$$\theta_0 = \theta_0 + \alpha w^{(1)}(y^{(1)} - \theta_0 - \theta_1 x^{(1)}) = 0.042$$

$$\theta_1 = \theta_1 + \alpha w^{(1)}(y^{1} - \theta_0 - \theta_1 x^{1}).x^{(1)} = 0.262$$

2^{nd} iteration :->

$$\theta_0 = \theta_0 + \alpha w^{(2)}(y^{(2)} - \theta_0 - \theta_1 x^{(2)}) = 0.042$$

lly $\theta_1 = 0.26344$

lly 3^{rd} iteration :->

$\theta_0 = 0.172$

$\theta_1 = 1.38$

4^{th} :-

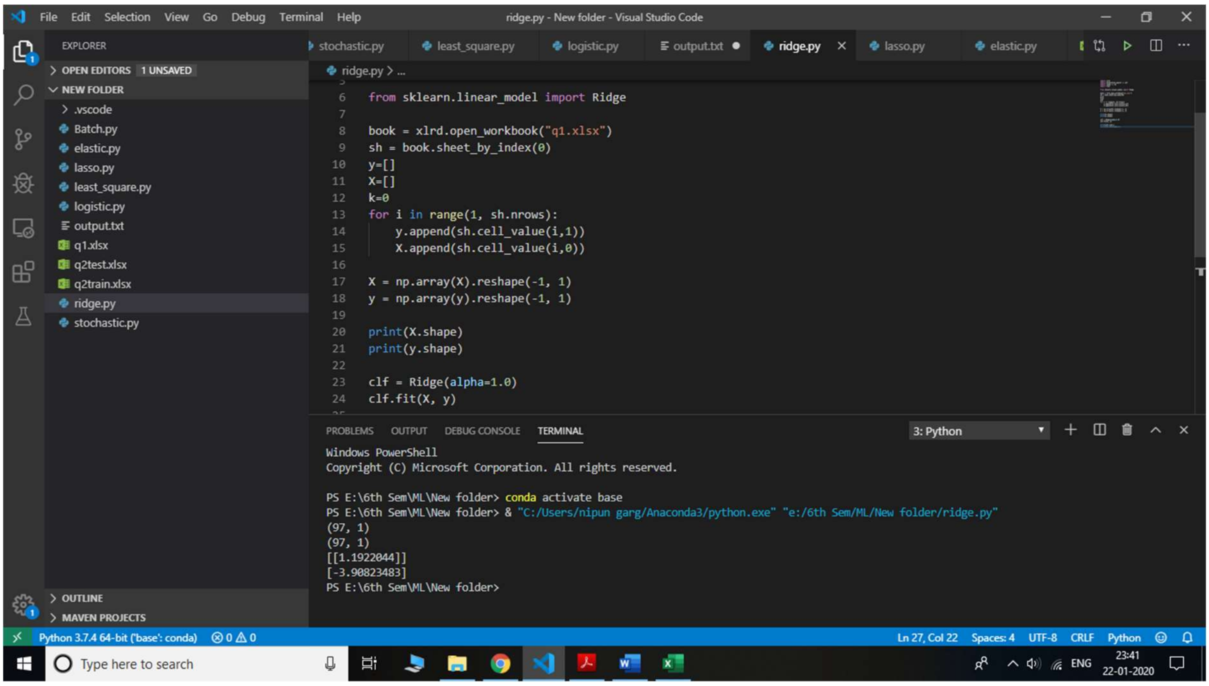
$\theta_0 = 0.343$

$\theta_1 = 2.278$

$$\boxed{\text{After 4 iterations.} \quad \theta_0 = 0.343 \qquad \theta_1 = 2.278}$$

Nipun garg 2017CH10224

c) Compare the results of Elastic net, Lasso and Ridge regression. (Use in-built packages)



Using ridge lasso and elactic net

Ridge alpha =1 result [[1.1922044]] [-3.90823483] highly accurate and fast

Lasso alpha = 0.1 [1.18628674] [-3.85935614] Inaccurate   for large alpha slowly converges but is very stable accuracy increases with decreasing alpha

Elastic net independent of alpha value results = [1.18566042] [-3.85418289] less accurate but is stable

Nipun garg 2017CH10224

Q2

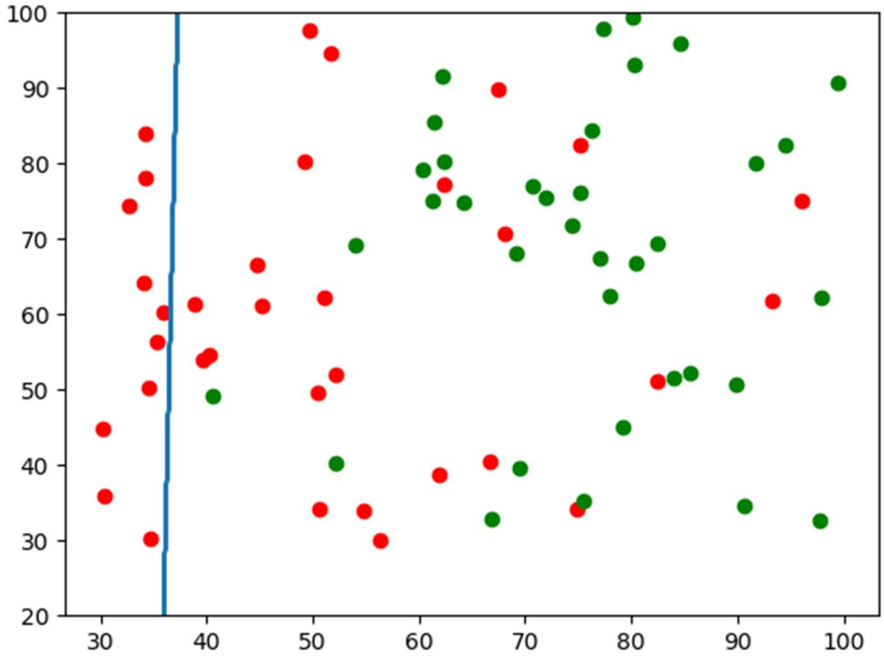
Apply logistic regression on training data with the first 2 columns as input data and the third column as output. Use any suitable learning rate. Now predict admission results on test data (q2test.csv) and print the result in output1.txt with every line of the text file containing either 0 or 1. Plot the results. (Don’t use in-built packages.)

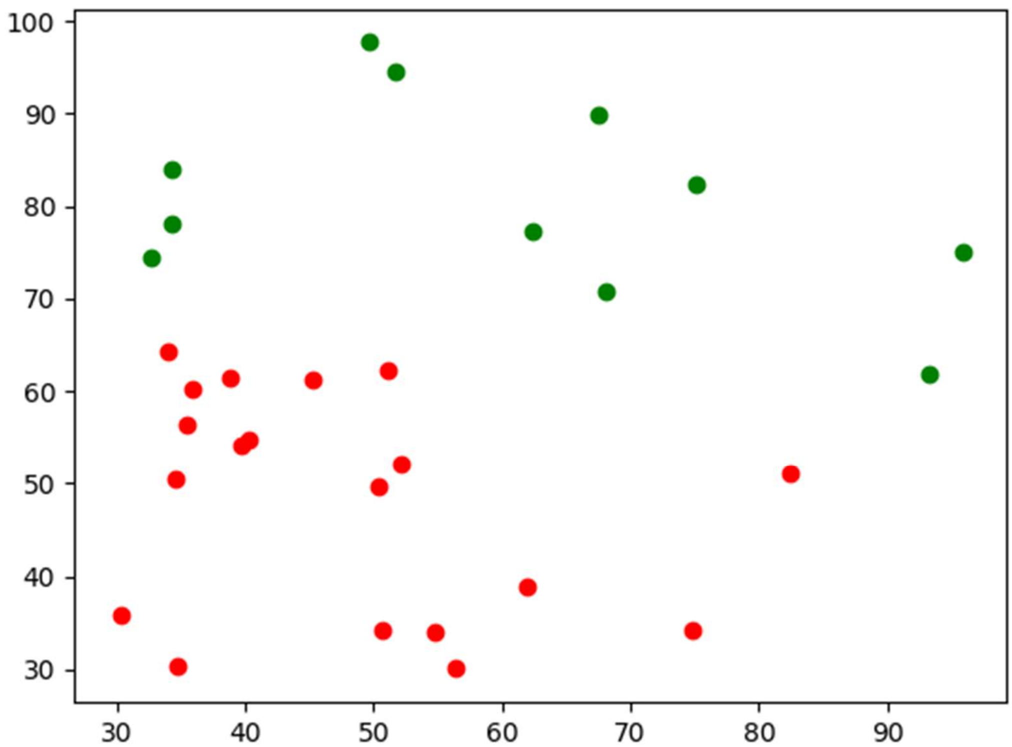
Code using stochastic

```
logistic.py > ...
temp=1.0
while(abs(delta)>pow(10,-5)):
#for j in range(1,10):
    cost=0
    for i in range(0,sh.nrows-1):
        val0=y[i]-sigmoid(theta0+theta1*aptitude[i]+theta2*verbal[i])
        val1=(y[i]-sigmoid(theta0+theta1*aptitude[i]+theta2*verbal[i]))*aptitude[i]
        val2=(y[i]-sigmoid(theta0+theta1*aptitude[i]+theta2*verbal[i]))*verbal[i]
        cost+=-(y[i]*(math.log(sigmoid(theta0+theta1*aptitude[i]+theta2*verbal[i])))+(1-y[i])*math.log(1-ma
        theta0=theta0+alpha*val0
        theta1=theta1+alpha*val1
        theta2=theta2+alpha*val2
        #print(val0,val1,val2)
    #print(theta0,theta1,theta2)
    cost=cost/100
    hello.append(cost)
    #print(hello)
    delta=cost-temp
    temp=cost
print(theta0,theta1,theta2)
```

Alpha =0.04
results
Theta0 = -215.29620595444084 Theta1 = 6.044947341981405 Theta2 = -0.10110343573777571

Nipun garg 2017CH10224





Test Data with theta values -6.3, 0.015, 0.08

Nipun garg 2017CH10224