Optimization Techniques For Machine Learning

Dr. Venkataramana Veeramsetty
Center For AI and Deep Learning
Assistant Professor In Dept. of EEE
S R University

October 1, 2020

What is Optimization?

- Optimization is the process of finding the values of decision variable at which the objective function is either minimum or maximum
- Every optimization problem has been defined in terms of objective function, constraints and boundaries
- Optimization problems are classified into two categories
  - Linear optimization problem
  - Non linear optimization problem

Linear Programming Problem

- Both Objective function and constraints must be linear in decision variables

Mathematical Modeling:

$$Minimize \quad f(X) \tag{1}$$

Subjected to

$$InequalityConstraint \quad g_i(X) \leq 0 \quad i = 1, 2, ..., m \tag{2}$$

$$EqualityConstraint \quad h_j(X) = 0 \quad i = 1, 2, ..., n \tag{3}$$

$$Boundaries \quad X \geq 0 \tag{4}$$

Problem

Consider a chocolate manufacturing company that produces only two types of chocolate i.e. A and B. Both the chocolates require Milk and Choco only.

To manufacture each unit of A and B, the following quantities are required:

Each unit of A requires 1 unit of Milk and 3 units of Choco

Each unit of B requires 1 unit of Milk and 2 units of Choco

The company kitchen has a total of 5 units of Milk and 12 units of Choco. On each sale, the company makes a profit of Rs 6 per unit A sold and Rs 5 per unit B sold.

Now, the company wishes to maximize its profit. How many units of A and B should it produce respectively?

Mathematical Modeling

Objective function:

$$Maximize \quad 6X + 5Y \tag{5}$$

Subjected To:

$$Inequality \ \ Constraint: \quad X + Y \leq 5 \tag{6}$$

$$Inequality \ \ Constraint: \quad 3X + 2Y \leq 12 \tag{7}$$

$$Boundaries: \quad X, Y \geq 0 \tag{8}$$

Email : dr.vvr.research@gmail.com
(Dr. Venkataramana Veeramsetty)

Python Code

```
from scipy.optimize import linprog
obj=[-6,-5]
lhs_ieq=[[1,1],[3,2]]
rhs_ieq=[5,12]
bnd = [(0, float("inf")),(0, float("inf"))]
opt = linprog(c=obj, A_ub=lhs_ieq, b_ub=rhs_ieq,bounds=bnd,method="Simplex")
opt
```

```
     fun: -27.0
 message: 'Optimization terminated successfully.'
     nit: 2
   slack: array([0., 0.])
  status: 0
 success: True
       x: array([2., 3.])
```

De

solver: "interior-point"

Assignment-1

A farmer has recently acquired a 110 hectares piece of land. He has decided to grow Wheat and barley on that land. Due to the quality of the sun and the region's excellent climate, the entire production of Wheat and Barley can be sold. He wants to know how to plant each variety in the 110 hectares, given the costs, net profits and labor requirements according to the data shown below:

Crop	Cost (Rs/Hec)	Profit (Price/Hec)	Man-days/Hec
Wheat	7000	50	10
Barley	2000	120	30

The farmer has a budget of Rs. 7,00,000 and availability of 1,200 man-days during the planning horizon. Find the optimal solution and the optimal value.

Mathematical Modeling

Objective function:

$$\textit{Maximize} \quad 50X + 120Y \tag{9}$$

Subjected To:

$$\textit{Inequality Constraint}: \quad 10X + 30Y \leq 1200 \tag{10}$$

$$\textit{Inequality Constraint}: \quad 7000X + 2000Y \leq 700000 \tag{11}$$

$$\textit{Inequality Constraint}: \quad X + Y \leq 110 \tag{12}$$

$$\textit{Boundaries}: \quad X, Y \geq 0 \tag{13}$$

Email : dr.vvr.research@gmail.com
(Dr. Venkataramana Veeramsetty)

Non linear Optimization

- Objective function is non-linear and the constraints, if they exist at all, may be linear or non-linear
- Non-linear problems may have many local optimum solutions, which are optimum in a specific sub-region of the solution space.
- However, the optimum in the whole region for which the problem is defined is called the global optimum.

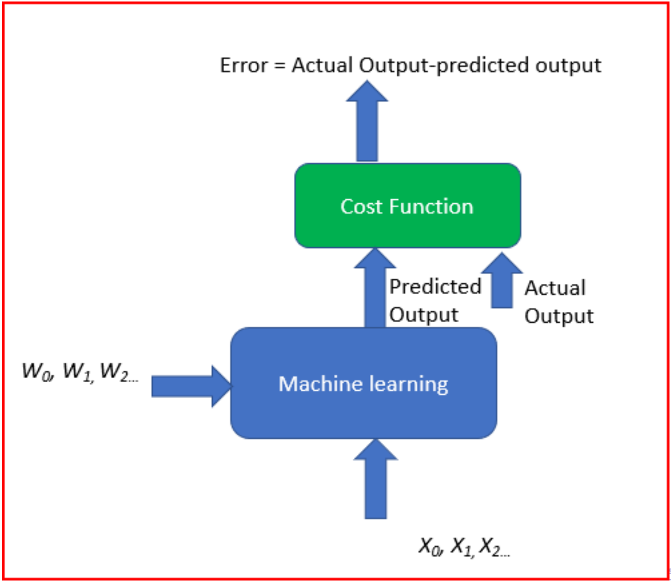
$$\textit{Objective Function} : \textit{Minimize} \quad f(x,y) = x^3 + Y^2 + x + 1 \tag{14}$$

$$x^2 + 2y \leq 16 \tag{15}$$

$$5x + 2y = 6 \tag{16}$$

$$x, y \geq 0 \tag{17}$$

Where OT lands on ML



Email : dr.vvr.research@gmail.com
(Dr. Venkataramana Veeramsetty)

Loss/Cost/Objective functions for ML

Let consider Y^{act} is actual value and Y^{pred} is a predicted value by ML model. Then the loss functions are shown below

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(Y^{act} - Y^{pred})^2 \tag{18}$$

$$binarycross - entropy = -ylog(p) - (1-y)log(1-p) \tag{19}$$

$$Categoricalcross - entropy = -\sum_{i=1}^{c} y_i log(p_i) \tag{20}$$

- During the training process, parameters of ML model will be updated such that the loss function will be minimized.
- As loss functions are non linear, we can see this optimization as non linear optimization problem.
- Non-linear optimization problems can be solved by using either Gradient Descent optimization techniques or Meta-Heuristic Techniques.

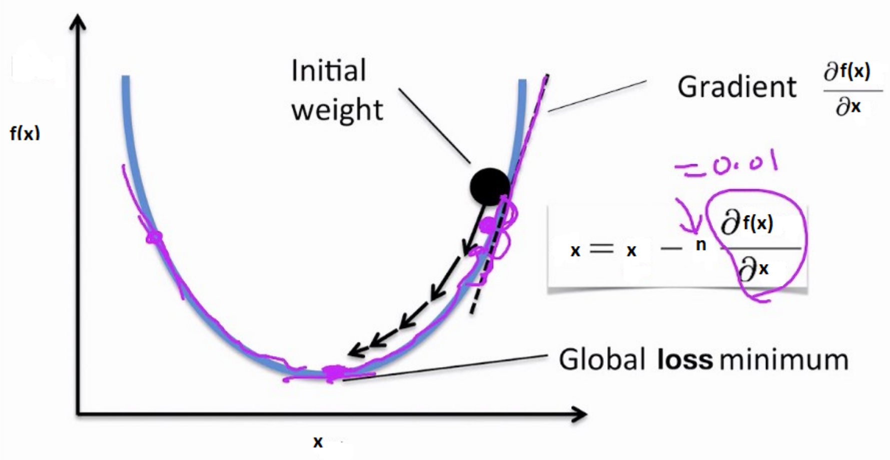
Non linear Optimization techniques

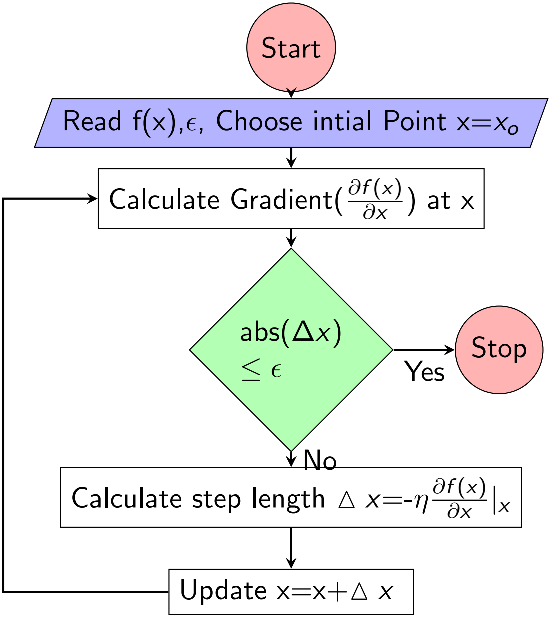
- Gradient Descent optimization techniques
  - Stochastic gradient descent
  - Batch gradient descent
  - Mini-batch gradient descent
  - Momentum
  - Nesterov accelerated gradient
  - Adagrad
  - Adadelta
  - RMSprop
  - Adam
  - AdaMax
  - Nadam
- Meta-Heuristic Algorithms
  - Genetic Algorithm, Differential Evolution algorithm, Particle Swarm Optimization etc.

Email : dr.vvr.research@gmail.com
(Dr. Venkataramana Veeramsetty)

Gradient Descent Optimization

Let consider minimization of f(x) is an objective function. Assume f(x) is pure convex function.





Email : dr.vvr.research@gmail.com
(Dr. Venkataramana Veeramsetty)

Example Problem

Find the minimum value of f(x)=x^2+5

Iteration 1:

- Choose initial value for x, let x=2 and η=0.01
- Find gradient at x=2 i.e $\frac{\partial f(x)}{\partial x}|_{x=2}$=2(2)=4
- As gradient not near to zero, calculate step length $\triangle$ x=-0.01*4=-0.04
- Updata x value as x=2-0.04=1.96

Iteration 2:

- Find gradient at x=1.96 i.e $\frac{\partial f(x)}{\partial x}|_{x=1.96}$=2(1.96)=3.92
- As gradient not near to zero, calculate step length $\triangle$ x=-0.01*3.92=-0.0392
- Update x value as x=1.96-0.0392=1.921

This procedure is repeating until gradient is near to zero

Email : dr.vvr.research@gmail.com
(Dr. Venkataramana Veeramsetty)

Python code

```
# Initilaization of parameters
x_o = 2 # The algorithm starts at x=3
eta = 0.9 # Learning rate
eps = 0.000001 #This tells us when to stop the algorithm
del_x = 1 #
max_iters = 10000 # maximum number of iterations
iters = 0 #iteration counter

def deriv(x):
    x_deriv = 2*(x)
    return x_deriv

while abs(del_x) > eps and iters < max_iters:
    prev_x = x_o #Store current x value in prev_x
    del_x= -eta * deriv(prev_x)
    x_o = x_o+del_x  #Grad descent
    iters = iters+1 #iteration count
    print("Iteration",iters,"\nX value is",x_o) #Print iterations

print("The local minimum occurs at", x_o)
```

Email : dr.vvr.research@gmail.com
(Dr. Venkataramana Veeramsetty)

Assignment 2

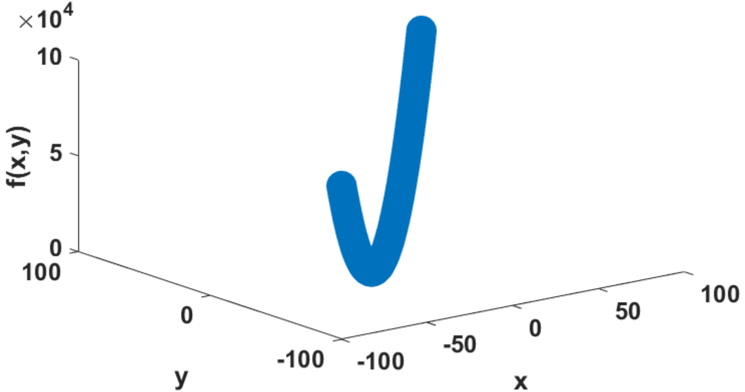
Find the global minimum point and value for the function f(x)=$x^4 + 3x^2 + 10$

- Do manual calculations for two iterations
- Find the optimal solution using python programming

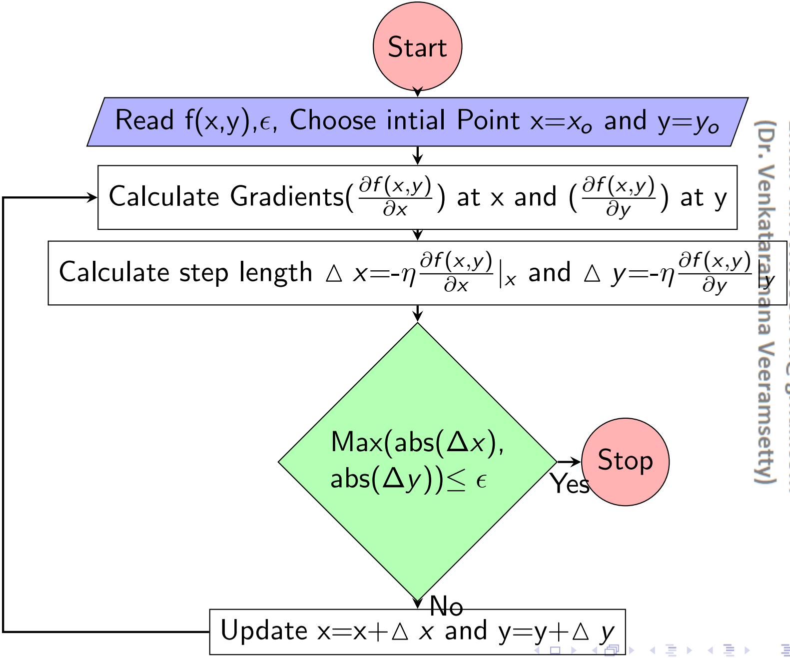
Email : dr.vvr.research@gmail.com
(Dr. Venkataramana Veeramsetty)

Gradient Descent Optimization: Multiple Variables

Let consider minimization of f(x,y) is an objective function and assume f(x,y) is a poor convex function.



Email : dr.vvr.research@gmail.com
(Dr. Venkataramana Veeramsetty)



Email : dr.vvr.research@gmail.com
(Dr. Venkataramana Veeramsetty)

Example Problem

Find the minimum value of f(x,y)==$3x^2+5y^2+10$

Iteration 1:

- Choose initial value for x, y and η. Let consider x=2, y=3 and η=0.01.
- Find gradient at x=2 i.e $\frac{\partial f(x,y)}{\partial x}|_{x=2}$=6(2)=12
- Find gradient at y=3 i.e $\frac{\partial f(x,y)}{\partial y}|_{y=3}$=10(3)=30
- As gradient not near to zero, calculate step length $\triangle\ x$=-0.01*12=-0.12 and $\triangle\ y$=-0.01*30=-0.3
- Updata x value as x=2-0.12=1.88 and y=3-0.3=2.7

This procedure is repeating until gradient is near to zero. For the next iteration x=1.88 and y=2.7

Email : drvvr.research@gmail.com
(Dr. Venkataramana Veeramsetty)

Python Code

```
# Initilaization of parameters
x_o = 2 # The algorithm starts at x=2
y_o = 3 # The algorithm starts at y=3
eta = 0.1 # Learning rate
eps = 0.000001 #This tells us when to stop the algorithm
del_x = 1#
del_y = 1#
max_iters = 10000 # maximum number of iterations
iters = 0 #iteration counter
def deriv(x,y):
    x_deriv = 6*(x)
    y_deriv = 10*(y)
    return x_deriv,y_deriv
while max(abs(del_x),abs(del_y)) > eps and iters < max_iters:
    prev_x = x_o #Store current x value in prev_x
    prev_y = y_o #Store current x value in prev_x
    del_x,del_y= deriv(prev_x,prev_y)
    del_x=-eta *del_x
    del_y=-eta *del_y
    x_o = x_o+del_x  #Grad descent
    y_o = y_o+del_y  #Grad descent
    iters = iters+1 #iteration count
    print("Iteration",iters,"\nX value is",x_o,"\nY value is",y_o) #Print iterations
print("The local minimum occurs at", x_o,y_o)
```

Assignment 3

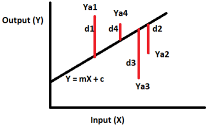
Find the global minimum point and value for the function f(x,y)$=3x^2 + 5e^{-y} + 10$

- Do manual calculations for two iterations
- Find the optimal solution using python programming

Email : dr.vvr.research@gmail.com
(Dr. Venkataramana Veeramsetty)

Simple Linear Regression Using GDA

Simple linear regression model is a linear equation which has minimum average distance from all the data points.



Email : dr.vvr.research@gmail.com
(Dr. Venkataramana Veeramsetty)

$$Y_i = mX_i^a + c \tag{21}$$

$$Min \quad MSE = \frac{1}{2n_s}\sum_{i=1}^{n_s}(Y_i^a - Y_i)^2 \tag{22}$$

- Optimization algorithm is required to find the values of m and c for the given inputs (X_i^a) and corresponding outputs (Y_i^a), such that the error shown in equation (22) is minimum.
- The objective function is i.e minimization of MSE is a convex function so we can go with GDA.

GDA classified into three categories

- Stochastic Gradient Descent Optimization (SGD)
  - Objective function: $\frac{1}{2}(Y_i^a - Y_i)^2$
- Mini Batch Gradient Descent Optimization
  - Objective function: $\frac{1}{2n}\sum_{i=1}^{n}(Y_i^a - Y_i)^2$ Where n<m
- Batch Gradient Descent Optimization
  - Objective function: $\frac{1}{2n_s}\sum_{i=1}^{n_s}(Y_i^a - Y_i)^2$

where n_s is total number of samples in dataset, and n is number of samples in mini batch

SLR with SGD

Step1 Read dataset $[x_i^a, y_i^a]$,Initialize η,ε,epochs, set m=m_o and c=c_o.

Step2 Set Iteration=1

Step3 Set sample i=1

Step4 Calculate Y using equation (23)

$$Y = mx_i^a + c \tag{23}$$

Step5 Calculate Error (Objective function) using equation (55)

$$E = \frac{1}{2}(Y_i^a - mx_i^a - c)^2 \tag{24}$$

Step6 Calculate gradients of error using equations (56) and (57)

$$\frac{\partial E}{\partial m} = -(Y_i^a - mx_i^a - c)x_i^a \tag{25}$$

$$\frac{\partial E}{\partial c} = -(Y_i^a - mx_i^a - c) \tag{26}$$

Step7 Calculate step lengths $\triangle\ m$ and $\triangle\ c$ using equations (27) and (28)

$$\triangle\ m = -\eta \frac{\partial E}{\partial m} = \eta(Y_i^a - m x_i^a - c) x_i^a \tag{27}$$

$$\triangle\ c = -\eta \frac{\partial E}{\partial c} = \eta(Y_i^a - m x_i^a - c) \tag{28}$$

Step8 Update m and c using equations (29) and (30)

$$m = m + \triangle\ m \tag{29}$$

$$c = c + \triangle\ c \tag{30}$$

Step9 Sample i=i+1, if i¿n_s go to next step else go to Step4

Step10 Iteration=Iteration+1, if Iteration>epochs go to next step else go to step 3

Step11 Calculate error (MSE=$\frac{1}{n_s}\sum_{i=1}^{n_s}(y_i^a - y_i)^2$), Print MSE,m and c

Step12 stop

Example

Let consider a sample dataset have one input (X_i^a) and one output (Y_i^a), and number of samples 4. Develop a simple linear regression model with one iteration and using stochastic gradient optimizer

Sample(i)	X_i^a	Y_i^a
1	0.2	3.4
2	0.4	3.8

Email:dr.vvr.research@gmail.com
(Dr.Venkataramana Veeramsetty)

Step1 Read dataset,η=0.1,epochs=1,m=1 and c=-1

Step2 Set iteration=1

Step3 Set sample i=1

Step4 Y=(1)(0.2)-1=-0.8

Step5 E=0.5*$(3.4+0.8)^2$=8.82

Step6 $\frac{\partial E}{\partial m}$=-(3.4+0.8)*0.2=-0.84 and $\frac{\partial E}{\partial c}$=-(3.4+0.8)=-4.2

Step7 Δm=-(0.1)(-0.84)=0.084 and Δm=-(0.1)(-4.2)=0.42

Step8 m=1+0.084=1.084 and c=-1+0.42=-0.58

Step9 Sample i=i+1=2 and 2$< n_s = 4$

Step10 **Y=(1.084)(0.4)-0.58=-0.1464**

Step11 **E=0.5*$(3.8+0.1464)^2$=7.79**

Step12 **$\frac{\partial E}{\partial m}$=-(3.8+0.1464)*0.4=-1.58 and $\frac{\partial E}{\partial c}$=-(3.8+0.1464)=-3.94**

Step13 **Δm=-(0.1)(-1.58)=0.158 and Δc=-(0.1)(-3.94)=0.394**

Step14 **m=1.084+0.158=1.242 and c=-0.58+0.394=-0.186**

Step15 **Sample i=i+1=2 and 2**$not < n_s = 2$

Step16 **iteration=iteration+1=2 and iteration**$not < epochs = 1$

Step17 **Stop**

Email : dr.vvr.research@gmail.com
(Dr. Venkataramana Veeramsetty)

$$\text{MSE}=\frac{1}{n_s}\sum_{i=1}^{n_s}(y_i^a-y_i)^2=\frac{1}{2}\sum_{i=1}^{2}(y_i^a-y_i)^2$$

m=1.242 and c=-0.186					
Sample(i)	X_i^a	Y_i^a	Y_i	SE	AE
1	0.2	3.4	0.0624	11.14	3.34
2	0.4	3.8	0.3108	12.17	3.49

$$\text{MSE}=\frac{11.14+12.17}{2}=11.65$$
$$\text{RMSE}=\sqrt{MSE}=\sqrt{11.65}=3.413$$
$$\text{MAE}=\frac{3.34+3.49}{2}=3.415$$

Email : dr.vvr.research@gmail.com
(Dr. Venkataramana Veeramsetty)

Python code

```python
# Initilaization of parameters
m_o = 1 # The algorithm starts at x=2
c_o = -1 # The algorithm starts at y=3
eta = 0.1 # Learning rate
del_m = 1#
del_c = 1#
max_iters = 10000 # maximum number of iterations
iters = 0 #iteration counter
def deriv(m,c,x,y):
    m_deriv = -1*(y-m*x-c)*x
    c_deriv = -1*(y-m*x-c)
    return m_deriv,c_deriv
while iters < max_iters:
    for i in range(len(data[:,0])):
        prev_m = m_o #Store current x value in prev_x
        prev_c = c_o #Store current x value in prev_x
        del_m,del_c= deriv(prev_m,prev_c,data[i,0],data[i,1])
        del_m=-eta *del_m
        del_c=-eta *del_c
        m_o = m_o+del_m  #Grad descent
        c_o = c_o+del_c  #Grad descent
    iters = iters+1 #iteration count
    print("Iteration",iters,"\nm value is",m_o,"\nc value is",c_o) #Print iterations
print("The local minimum occurs at", m_o,c_o)
```

Assignment-4

Estimate the bicarbonates of well water based on its pH value using simple regression model. Consider SGD optimizer.
Dataset: Union Carbide Technical Report

- Do the manual calculation for two iteration by taking only first two samples in the dataset
- Write the python code to build simple linear regression model using SGD optimizer
  - Do the data normalization
  - Split the data for train and test (90:10)
  - Train the simple linear regression model using SGD with training data
  - Compute MSE, RMSE and MAE with training data
  - Compute MSE, RMSE and MAE with testing data

SLR with BGD

Half Mean Square Error (31) will be considered as an objective function in case of Batch Gradient Descent Optimization. Now the main goal of BGD is minimization of Half Mean Square Error by updating the parameters m and c

$$E = \frac{1}{2n_s} \sum_{i=1}^{n_s} (Y_i^a - Y_i)^2 \tag{31}$$

Step1 Read dataset $[x_i^a, y_i^a]$,Initialize η,epochs, set m=m_o and c=c_o.

Step2 Set Iteration=1

Step3 Calculate gradients of error using equations (32) and (33)

$$\frac{\partial E}{\partial m} = -\frac{1}{n_s}\sum_{i=1}^{n_s}(Y_i^a - mx_i^a - c)x_i^a \tag{32}$$

$$\frac{\partial E}{\partial c} = -\frac{1}{n_s}\sum_{i=1}^{n_s}(Y_i^a - mx_i^a - c) \tag{33}$$

Step4 Calculate step lengths $\triangle\ m$ and $\triangle\ c$ using equations (34) and (35)

$$\triangle\ m = -\eta\frac{\partial E}{\partial m} = \frac{\eta}{n_s}\sum_{i=1}^{n_s}(Y_i^a - mx_i^a - c)x_i^a \tag{34}$$

$$\triangle\ c = -\eta\frac{\partial E}{\partial c} = \frac{\eta}{n_s}\sum_{i=1}^{n_s}(Y_i^a - mx_i^a - c) \tag{35}$$

Step5 Update m and c using equations (36) and (37)

$$m = m + \triangle m \tag{36}$$

$$c = c + \triangle c \tag{37}$$

Step6 Iteration=Iteration+1, if Iteration>epochs go to next step else go to step 3

Step7 Calculate error (MSE=$\frac{1}{n_s}\sum_{i=1}^{n_s}(y_i^a - y_i)^2$), Print MSE,m and c

Step8 stop

Example

Let consider a sample dataset have one input (X_i^a) and one output (Y_i^a), and number of samples 2. Develop a simple linear regression model with one iteration and using Batch gradient descent optimizer

Sample(i)	X_i^a	Y_i^a
1	0.2	3.4
2	0.4	3.8

Email : dr.vvr.research@gmail.com
(Dr. Venkataramana Veeramsetty)

Step1 Read dataset,η=0.1,epochs=1,m=1 and c=-1

Step2 Set iteration=1

Step3: $\frac{\partial E}{\partial m}$=-(0.5)[(3.4-1*0.2+1)*0.2+(3.8-1*0.4+1)*0.4]
=-0.5(0.84+1.76)=-1.3
$\frac{\partial E}{\partial c}$=-(0.5)[(3.4-1*0.2+1)+(3.8-1*0.4+1)]=-0.5(4.2+4.4)=-4.3

Step4: Step Length:Δm=-(0.1)(-1.3)=0.13 and Δc=-(0.1)(-4.3)=0.43

Step5: Update: m=1+0.13=1.13 and c=-1+0.43=-0.57

Step6 Iteration=Iteration+1, if Iteration>epochs go to next step else go to step 3

Step7 Calculate error (MSE=$\frac{1}{n_s}\sum_{i=1}^{n_s}(y_i^a - y_i)^2$), Print MSE,m and c

Step8 stop

$$\text{MSE}=\frac{1}{n_s}\sum_{i=1}^{n_s}(y_i^a - y_i)^2=\frac{1}{2}\sum_{i=1}^{2}(y_i^a - y_i)^2$$

m=1.13 and c=-0.57					
Sample(i)	X_i^a	Y_i^a	Y_i	SE	AE
1	0.2	3.4	-0.344	13.99	3.74
2	0.4	3.8	-0.118	15.37	3.92

$$\text{MSE}=\frac{13.99+15.37}{2}=14.68$$
$$\text{RMSE}=\sqrt{MSE}=\sqrt{14.68}=3.83$$
$$\text{MAE}=\frac{3.34+3.49}{2}=3.83$$

Email : dr.vvr.research@gmail.com
(Dr. Venkataramana Veeramsetty)

Python code

```
m_o = 1 # The algorithm starts at m=1
c_o = -1 # The algorithm starts at c=-1
eta = 0.1 # Learning rate
del_m = 1#
del_c = 1#
max_iters = 10000 # maximum number of iterations
iters = 0 #iteration counter
def deriv(m,c,dataX,dataY):
    m_deriv=0
    c_deriv=0
    for i in range(len(dataX)):
        x=dataX[i]
        y=dataY[i]
        m_deriv=m_deriv+(y-m*x-c)*x
        c_deriv=c_deriv+(y-m*x-c)
    m_deriv=-m_deriv/len(dataX)
    c_deriv=-c_deriv/len(dataX)
    return m_deriv,c_deriv
while iters < max_iters:
    prev_m = m_o #Store current x value in prev_x
    prev_c = c_o #Store current x value in prev_x
    del_m,del_c= deriv(prev_m,prev_c,data[:,0],data[:,1])
    del_m=-eta *del_m
    del_c=-eta *del_c
    m_o = m_o+del_m  #Grad descent
    c_o = c_o+del_c  #Grad descent
    iters = iters+1 #iteration count
    print("Iteration",iters,"\nm value is",m_o,"\nc value is",c_o) #Pri
print("The local minimum occurs at", m_o,c_o)
```

Email : dr.vvr.research@gmail.com
(Dr. Venkataramana Veeramsetty)

Assignment-5

Estimate the weight of liquid nitrogen based on its pressure using simple regression model. Consider BGD optimizer.
Dataset: Pressure and Weight in Cryogenic Flow Meters

- Do the manual calculation for two iteration by taking only first three samples in the dataset (No need of data normalization)
- Write the python code to build simple linear regression model using BGD optimizer
  - Do the data normalization
  - Split the data for train and test (90:10)
  - Train the simple linear regression model using SGD with training data
  - Compute MSE, RMSE and MAE with training data
  - Compute MSE, RMSE and MAE with testing data

SLR with MBGD

Step1 Read dataset $[x_i^a, y_i^a]$, Initialize η, n, epochs, set m=m_o and c=c_o.

Step2 Split data into 'n' batches

Step3 Set Iteration=1

Step4 Set batch i=1

Step5 Calculate gradients of error using equations (38) and (39)

$$\frac{\partial E}{\partial m} = -\frac{1}{n}\sum_{i=1}^{n}(Y_i^a - mx_i^a - c)x_i^a \tag{38}$$

$$\frac{\partial E}{\partial c} = -\frac{1}{n}\sum_{i=1}^{n}(Y_i^a - mx_i^a - c) \tag{39}$$

Step6 Calculate step lengths $\triangle\ m$ and $\triangle\ c$ using equations (40) and (41)

$$\triangle\ m = -\eta \frac{\partial E}{\partial m} = \frac{\eta}{n} \sum_{i=1}^{n} (Y_i^a - m x_i^a - c) x_i^a \tag{40}$$

$$\triangle\ c = -\eta \frac{\partial E}{\partial c} = \frac{\eta}{n} \sum_{i=1}^{n} (Y_i^a - m x_i^a - c) \tag{41}$$

Step7 Update m and c using equations (42) and (43)

$$m = m + \triangle\ m \tag{42}$$

$$c = c + \triangle\ c \tag{43}$$

Step8 batch i=i+1, if i>n go to next step else go to step 5

Step9 Iteration=Iteration+1, if Iteration>epochs go to next step else go to step 4

Step10 Calculate error (MSE=$\frac{1}{n_s}\sum_{i=1}^{n_s}(y_i^a - y_i)^2$), Print MSE,m and c

Step11 stop

Example

Let consider a sample dataset have one input (X_i^a) and one output (Y_i^a), and number of samples 4. Develop a simple linear regression model with one iteration and using Mini batch gradient descent optimizer

Sample(i)	X_i^a	Y_i^a
1	0.2	3.4
2	0.4	3.8
3	0.6	4.2
4	0.8	4.6

Batch-1			Batch-2		
Sample(i)	X_i^a	Y_i^a	Sample(i)	X_i^a	Y_i^a
1	0.2	3.4	1	0.6	4.2
2	0.4	3.8	2	0.8	4.6

Step1 Read dataset,η=0.1,epochs=1,m=1 and c=-1,n=2

Step2 Set iteration=1

Step3 Set batch i=1

Step4 $\frac{\partial E}{\partial m}$=-(0.5)[(3.4-1*0.2+1)*0.2+(3.8-1*0.4+1)*0.4]
=-0.5(0.84+1.76)=-1.3
$\frac{\partial E}{\partial c}$=-(0.5)[(3.4-1*0.2+1)+(3.8-1*0.4+1)]=-0.5(4.2+4.4)=-4.3

Step5 Step Length:Δm=-(0.1)(-1.3)=0.13 and Δc=-(0.1)(-4.3)=0.43

Step6 Update: m=1+0.13=1.13 and c=-1+0.43=-0.57

Step7 Set batch i=i+1=2 and i=2 not less than n

Step5 $\frac{\partial E}{\partial m}$=-(0.5)[(4.2-1.13*0.6+0.57)*0.6+(4.6-1.13*0.8+0.57)*0.8]
=-0.5(2.45+3.41)=-2.93
$\frac{\partial E}{\partial c}$=-(0.5)[(4.2-1.13*0.6+0.57)+(4.6-1.13*0.8+0.57)]=-0.5(4.09+4.27)=-4.18

Step6 Step Length:Δm=-(0.1)(-2.93)=0.293 and
Δc=-(0.1)(-4.18)=0.418

Step7 Update: m=1.13+0.293=1.42 and c=-0.57+0.418=-0.152

Step8 Set batch i=i+1=3 and i=3 grater than n

Step9 Iteration=Iteration+1=2, Iteration=2>epochs=1

$$MSE=\frac{1}{n_s}\sum_{i=1}^{n_s}(y_i^a-y_i)^2=\frac{1}{4}\sum_{i=1}^{4}(y_i^a-y_i)^2$$

m=1.42 and c=-0.152					
Sample(i)	X_i^a	Y_i^a	Y_i	SE	AE
1	0.2	3.4	0.132	10.69	3.27
2	0.4	3.8	0.416	11.42	3.38
3	0.6	4.2	0.7	12.25	3.5
4	0.8	4.6	0.984	13.1	3.62

$$MSE=\frac{10.69+11.42+12.25+13.1}{4}=11.86$$

$$RMSE=\sqrt{MSE}=\sqrt{11.86}=3.44$$

$$MAE=\frac{3.34+3.49}{2}=3.44$$

Email : dr.vvr.research@gmail.com
(Dr. Venkataramana Veeramsetty)

Python code: Initialization

```python
# Initilaization of parameters
import numpy as np
batch_size=2
data=np.array([[0.2,3.4],[0.4,3.8],[0.6,4.2],[0.8,4.6]])
n_minibatches = data.shape[0]//batch_size
m_o = 1 # The algorithm starts at m=1
c_o = -1 # The algorithm starts at c=-1
eta = 0.1 # Learning rate
del_m = 1#
del_c = 1#
max_iters = 10000 # maximum number of iterations
iters = 0 #iteration counter
```

Python code: Mini Batches

```
def create_mini_batches(data, batch_size):
    mini_batches = []
    n_minibatches = data.shape[0] // batch_size
    i = 0
    for i in range(n_minibatches + 1):
        mini_batch = data[i * batch_size:(i + 1)*batch_size, :]
        X_mini = mini_batch[:, :-1]
        Y_mini = mini_batch[:, -1].reshape((-1, 1))
        mini_batches.append((X_mini, Y_mini))
    if data.shape[0] % batch_size != 0:
        mini_batch = data[i * batch_size:data.shape[0]]
        X_mini = mini_batch[:, :-1]
        Y_mini = mini_batch[:, -1].reshape((-1, 1))
        mini_batches.append((X_mini, Y_mini))
    return mini_batches
mini_batches=create_mini_batches(data, batch_size)
```

Email : dr.vvr.research@gmail.com
(Dr. Venkataramana Veeramsetty)

Python code: Gradient Function

```
def deriv(m,c,dataX,dataY):
    m_deriv=0
    c_deriv=0
    for i in range(len(dataX)):
        x=dataX[i]
        y=dataY[i]
        m_deriv=m_deriv+(y-m*x-c)*x
        c_deriv=c_deriv+(y-m*x-c)
    m_deriv=-m_deriv/len(dataX)
    c_deriv=-c_deriv/len(dataX)
    return m_deriv,c_deriv
```

Email : dr.vvr.research@gmail.com
(Dr. Venkataramana Veeramsetty)

Python code: SLR with MBGD

```
while iters < max_iters:
    for i in range(n_minibatches):
        prev_m = m_o #Store current x value in prev_x
        prev_c = c_o #Store current x value in prev_x
        X,Y=mini_batches[i]
        del_m,del_c= deriv(prev_m,prev_c,X,Y)
        del_m=-eta *del_m
        del_c=-eta *del_c
        m_o = m_o+del_m  #Grad descent
        c_o = c_o+del_c  #Grad descent
    iters = iters+1 #iteration count
    print("Iteration",iters,"\nm value is",m_o,"\nc value is",c_o) #Print iterat
print("The local minimum occurs at", m_o,c_o)
```

Assignment-6

Estimate the housing prince based sq.ft. area using simple regression model. Consider MBGD optimizer.
Dataset: House Sales in King Country, USA

- Do the manual calculation for two iteration by taking only first three samples in the dataset (No need of data normalization)
- Write the python code to build simple linear regression model using BGD optimizer
    - Do the data normalization
    - Split the data for train and test (90:10)
    - Train the simple linear regression model using SGD with training data
    - Compute MSE, RMSE and MAE with training data
    - Compute MSE, RMSE and MAE with testing data

SGD Vs. BGD Vs. MBGD

- √ SGD
  - More frequently parameters updates
  - More computation time
  - More noisy update
- √ BGD
  - Less frequently parameters updates
  - Less computation time
  - Smooth update
- √ MBGD
  - Moderate frequently parameters updates
  - Computation time between SGD and BGD
  - Update much smooth or noisy
- √ Limitations
  - Choosing optimal learning rate (η)
  - Minimizing highly non-linear convex function
  - Same learning rate for all parameters update

Email : dr.vvr.research@gmail.com
(Dr. Venkataramana Veeramsetty)

- Let assume the training dataset has 100 samples. SGD is used to train the SLR model. If the maximum number of iterations is equal to 100 then how many times the parameter "m" will be updated.
- Let assume the training dataset has 100 samples. BGD optimization algorithm is used to train the SLR model. If the maximum number of iterations is equal to 100 then how many times the parameter "m" will be updated
- Let assume the training dataset has 100 samples. MBGD optimization algorithm is used to train the SLR model. If the maximum number of iterations is equal to 100 and the batch size is 10 then how many times the parameter "c" will be updated.

Momentum

- Momentum is a method that helps accelerate SGD in the relevant direction and dampens oscillations

$$v_t = \gamma v_{t-1} - \eta \frac{\partial F(x)}{\partial x} \tag{44}$$

$$x = x + v_t \tag{45}$$

γ varying between $0.5 --> 0.9$, it is called "Coefficient of momentum"

Email : dr.vvr.research@gmail.com
(Dr. Venkataramana Veeramsetty)

SLR with SGD-Momentum

Step1 Read dataset $[x_i^a, y_i^a]$, Initialize η, epochs, set m=m_o and c=c_o, set v_m^t=0, v_c^t=0

Step2 Set Iteration t=1

Step3 Set sample i=1

Step4 Calculate Y using equation (23)

Step5 Calculate Error (Objective function) using equation (55)

$$E = \frac{1}{2}(Y_i^a - mx_i^a - c)^2 \tag{46}$$

Step6 Calculate gradients of error using equations (56) and (57)

$$\frac{\partial E}{\partial m} = -(Y_i^a - mx_i^a - c)x_i^a \tag{47}$$

$$\frac{\partial E}{\partial c} = -(Y_i^a - mx_i^a - c) \tag{48}$$

Step7 Calculate velocity v_m and v_c using equations (49) and (50)

$$v_m^t = \gamma * v_m^{t-1} - \eta \frac{\partial E}{\partial m} = \gamma * v_m^{t-1} + \eta (Y_i^a - m x_i^a - c) x_i^a \tag{49}$$

$$v_c^t = \gamma * v_c^{t-1} - \eta \frac{\partial E}{\partial c} = \gamma * v_c^{t-1} + \eta (Y_i^a - m x_i^a - c) \tag{50}$$

Step8 Update m and c using equations (60) and (61)

$$m = m + v_m^t \tag{51}$$

$$c = c + v_c^t \tag{52}$$

Step9 Sample i=i+1, if i¿n_s go to next step else go to Step4

Step10 t=t+1, if t>epochs go to next step else go to step 3

Step11 Calculate error (MSE=$\frac{1}{n_s}\sum_{i=1}^{n_s}(y_i^a - y_i)^2$), Print MSE,m and c

Step12 stop

Example

Let consider a sample dataset have one input (X_i^a) and one output (Y_i^a), and number of samples 2. Develop a simple linear regression model with one iteration and using stochastic gradient optimizer with momentum

Sample(i)	X_i^a	Y_i^a
1	0.2	3.4
2	0.4	3.8

Step1 Read dataset,η=0.1,epochs=1,m=1,c=-1,γ=0.9,v_m=0 and v_c=0

Step2 Set iteration=1

Step3 Set sample i=1

Step4 Y=(1)(0.2)-1=-0.8

Step5 E=0.5*$(3.4+0.8)^2$=8.82

Step6 $\frac{\partial E}{\partial m}$=-(3.4+0.8)*0.2=-0.84 and $\frac{\partial E}{\partial c}$=-(3.4+0.8)=-4.2

Step7 v_m=0.9*0+0.1*(0.84)=0.084 and v_c=0.9*0+0.1*(4.2)=0.42

Step8 m=1+0.084=1.084 and c=-1+0.42=-0.58

Step9 Sample i=i+1=2 and 2< $n_s = 4$

Step10 **Y=(1.084)(0.4)-0.58=-0.1464**

Step11 **E=0.5*$(3.8+0.1464)^2$=7.79**

Step12 $\frac{\partial E}{\partial m}$**=-(3.8+0.1464)*0.4=-1.58 and** $\frac{\partial E}{\partial c}$**=-(3.8+0.1464)=-3.94**

Step13 v_m**=0.9*0.084+0.1*(1.58)=0.2336 and** v_c**=0.9*0.42+0.1*(3.94)=0.772**

Step14 **m=1.084+0.2336=1.3176 and c=-0.58+0.772=0.192**

Step15 **Sample i=i+1=2 and 2**$not < n_s = 2$

Step16 **iteration=iteration+1=2 and iteration**$not < epochs = 1$

Step17 **Stop**

$$MSE=\frac{1}{n_s}\sum_{i=1}^{n_s}(y_i^a - y_i)^2=\frac{1}{2}\sum_{i=1}^{2}(y_i^a - y_i)^2$$

m=1.3176 and c=0.192					
Sample(i)	X_i^a	Y_i^a	Y_i	SE	AE
1	0.2	3.4	0.4555	11.14	2.94
2	0.4	3.8	0.719	12.17	3.08

$$MSE=\frac{8.67+9.48}{2}=9.078$$
$$RMSE=\sqrt{MSE}=\sqrt{9.078}=3.01$$
$$MAE=\frac{2.94+3.08}{2}=3.01$$

Email : dr.vvr.research@gmail.com
(Dr. Venkataramana Veeramsetty)

Assignment-7

Estimate load at particular hour on 33/11KV substation based on load at same time but previous day. Consider momentum+SGD optimizer.

Dataset: Active power load dataset

- Do the manual calculation for two iteration by taking only first two samples in the dataset (No need of data normalization)
- Write the python code to build simple linear regression model using momentum+SGD optimizer
  - Do the data normalization
  - Split the data for train and test (90:10)
  - Train the simple linear regression model using momentum+SGD with training data
  - Compute MSE, RMSE and MAE with training data
  - Compute MSE, RMSE and MAE with testing data

Nesterov accelerated gradient

- Nesterov accelerated gradient effectively look ahead by calculating the gradient not w.r.t. to our current parameters but w.r.t. the approximate future position of our parameters

$$v_t = \gamma v_{t-1} - \eta \frac{\partial F(x + \gamma v_{t-1})}{\partial x} \tag{53}$$

$$x = x + v_t \tag{54}$$

γ varying between $0.5 - - > 0.9$

SLR with NAD

Step1 Read dataset $[x_i^a, y_i^a]$, Initialize η, epochs, set m=m_o and c=c_o, set v_m^t=0, v_c^t=0

Step2 Set Iteration t=1

Step3 Set sample i=1

Step4 Calculate Y using equation (23)

Step5 Calculate Error (Objective function) using equation (55)

$$E = \frac{1}{2}(Y_i^a - mx_i^a - c)^2 \tag{55}$$

Step6 Calculate gradients of error using equations (56) and (57)

$$\frac{\partial E}{\partial m} = -(Y_i^a - (m + \gamma * v_m^{t-1}) * x_i^a - c - \gamma * v_c^{t-1})x_i^a \tag{56}$$

$$\frac{\partial E}{\partial c} = -(Y_i^a - (m + \gamma * v_m^{t-1}) * x_i^a - c - \gamma * v_c^{t-1}) \tag{57}$$

Step7 Calculate velocity v_m and v_c using equations (58) and (59)

$$v_m^t = \gamma * v_m^{t-1} - \eta \frac{\partial E}{\partial m} \tag{58}$$

$$v_c^t = \gamma * v_c^{t-1} - \eta \frac{\partial E}{\partial c} \tag{59}$$

Step8 Update m and c using equations (60) and (61)

$$m = m + v_m^t \tag{60}$$

$$c = c + v_c^t \tag{61}$$

Step9 Sample i=i+1, if i¿n_s go to next step else go to Step4

Step10 t=t+1, if t>epochs go to next step else go to step 3

Step11 Calculate error (MSE=$\frac{1}{n_s}\sum_{i=1}^{n_s}(y_i^a - y_i)^2$), Print MSE,m and c

Step12 stop

Example

Let consider a sample dataset have one input (X_i^a) and one output (Y_i^a), and number of samples 2. Develop a simple linear regression model with one iteration and using Nesterov accelerated gradient

Sample(i)	X_i^a	Y_i^a
1	0.2	3.4
2	0.4	3.8

Step1 Read dataset,η=0.1,epochs=1,m=1,c=-1,γ=0.9,v_m=0 and v_c=0

Step2 Set iteration=1

Step3 Set sample i=1

Step4 Y=(1)(0.2)-1=-0.8

Step5 $E=0.5*(3.4+0.8)^2=8.82$

Step6 $\frac{\partial E}{\partial m}$=-(3.4-(1+0.9*0)*0.2+1-0.9*0)*0.2=-(3.4-0.2+1)*0.2=-0.84 and
$\frac{\partial E}{\partial c}$=-(3.4-(1+0.9*0)*0.2+1-0.9*0)=-4.2

Step7 v_m=0.9*0+0.1*(0.84)=0.084 and v_c=0.9*0+0.1*(4.2)=0.42

Step8 m=1+0.084=1.084 and c=-1+0.42=-0.58

Step9 Sample i=i+1=2 and 2< $n_s = 4$

Step10 **Y=(1.084)(0.4)-0.58=-0.1464**

Step11 **$E=0.5*(3.8+0.1464)^2=7.79$**

Step12 **$\frac{\partial E}{\partial m}$=-(3.8-(1.084+0.9*0.084)*0.4+0.58-0.9*0.42)*0.4=-(3.8-0.403+0.958)*0.4=-1.74 and**
$\frac{\partial E}{\partial c}$=-(3.8-(1.084+0.9*0.084)*0.4+0.58-0.9*0.42)=-(3.8-0.403+0.958)=-4.36

Step13 **v_m=0.9*0.084+0.1*(1.74)=0.25 and**
v_c=0.9*0.42+0.1*(4.36)=0.814

Step14 **m=1.084+0.25=1.334 and c=-0.58+0.814=0.234**

Step15 **Sample i=i+1=2 and 2**$not < n_s = 2$

Step16 **iteration=iteration+1=2 and iteration**$not < epochs = 1$

Step17 **Stop**

MSE=$\frac{1}{n_s}\sum_{i=1}^{n_s}(y_i^a - y_i)^2$=$\frac{1}{2}\sum_{i=1}^{2}(y_i^a - y_i)^2$

m=1.334 and c=0.234					
Sample(i)	X_i^a	Y_i^a	Y_i	SE	AE
1	0.2	3.4	0.5	8.41	2.9
2	0.4	3.8	0.77	9.18	3.03

MSE=$\frac{8.41+9.18}{2}$=8.795

RMSE=$\sqrt{MSE}$=$\sqrt{9.078}$=2.97

MAE=$\frac{2.9+3.03}{2}$=2.965

Email : dr.vvr.research@gmail.com
(Dr.Venkataramana Veeramsetty)

Assignment-8

Build a simple linear regression model using Nesterov Accelerated Gradient (NAG) + SGD optimizer to help 33/11KV substation electric utility to trade power effectively in an hourly ahead energy market by estimating load at a particular hour based on the load at the previous hour. Dataset: Active power load dataset

- Do the manual calculation for two iteration by taking only first two samples in the dataset (No need of data normalization)
- Write the python code to build simple linear regression model using NAG+SGD optimizer
  - Do the data normalization
  - Split the data for train and test (90:10)
  - Train the simple linear regression model using NAG+SGD with training data
  - Compute MSE, RMSE and MAE with training data
  - Compute MSE, RMSE and MAE with testing data

Adagrad Optimizer

- It adapts the learning rate (η) to the parameters, performing larger updates for infrequent and smaller updates for frequent parameters.
- It is well-suited for dealing with sparse data.
- Adagrad greatly improved the robustness of SGD
- Google, Youtube, Train GloVe word embedding

Objective function:

$$E(m,c) = \frac{1}{2}(y_i^a - mx_i^a - c)^2 \tag{62}$$

$$g_m = \frac{\partial E(m,c)}{\partial m} = -(y_i^a - mx_i^a - c)x_i^a \tag{63}$$

$$g_c = \frac{\partial E(m,c)}{\partial c} = -(y_i^a - mx_i^a - c) \tag{64}$$

SLR with Adagrad

Step1 Read dataset $[x_i^a, y_i^a]$, Initialize η, epochs, set m=m_o and c=c_o, set G_m^2=0, G_c^2=0

Step2 Set Iteration t=1

Step3 Set sample i=1

Step4 Calculate g_m and g_c using equations (148) and (149). and also calculate sum of gradient squares using equations (65) and (66)

$$G_m^2 = G_m^2 + [g_m]^2 \tag{65}$$

$$G_c^2 = G_c^2 + [g_c]^2 \tag{66}$$

Step4 Update m and c using equations (67) and (68) respectively

$$m = m - \frac{\eta}{\sqrt{G_m^2 + \epsilon}} * g_m \tag{67}$$

$$c = c - \frac{\eta}{\sqrt{G_c^2 + \epsilon}} * g_c \tag{68}$$

Step5 Set sample i=i+1, if i> n_s go to next step else go to Step4

Step6 Set iteration t=t+1, if t¿maxiter go tn next step else go to Step3

Step7 Compute error metrics

Step8 Stop

Example

Let consider a sample dataset have one input (X_i^a) and one output (Y_i^a), and number of samples 2. Develop a simple linear regression model with one iteration and using Adagrad optimizer

Sample(i)	X_i^a	Y_i^a
1	0.2	3.4
2	0.4	3.8

Step1 Read dataset,η=0.1,epochs=1,m=1,c=-1,$\epsilon=10^{-8}$,G_m^2=0 and G_c^2=0

Step2 Set iteration=1

Step3 Set sample i=1

Step4 Calculate G_m^2 and G_c^2

- $g_m = \frac{\partial E}{\partial m}$=-(3.4+0.8)*0.2=-0.84
- $\frac{\partial E}{\partial c}$=-(3.4+0.8)=-4.2
- $G_m^2 = 0 + (-0.84)^2 = 0.7056$
- $G_c^2 = 0 + (-4.2)^2 = 17.64$

Step5 Update m and c

- m=1-$\frac{0.1}{\sqrt{0.7056+10^{-}8}}$*-0.84=1.1
- c=-1-$\frac{0.1}{\sqrt{17.64+10^{-}8}}$*-4.2=-0.9

Step6 Sample i=i+1=1+1=2 not greater than number of samples

Step4 Calculate G_m^2 and G_c^2

- $g_m = \frac{\partial E}{\partial m}$=-(3.8+0.46)*0.4=-1.704
- $g_c = \frac{\partial E}{\partial c}$=-(3.8+0.46)=-4.26
- $G_m^2 = 0.7056 + (-1.704)^2 = 3.61$
- $G_c^2 = 17.64 + (-4.26)^2 = 35.78$

Email : dr.vvr.research@gmail.com
(Dr. Venkataramana Veeramsetty)

Step5 Update m and c

- m=1.1-$\frac{0.1}{\sqrt{3.61+10^-8}}$*-1.704=1.2
- c=-0.9-$\frac{0.1}{\sqrt{35.78+10^-8}}$*-4.26=-0.83

Step6 Sample i=i+1=2+1=3 is greater than number

Step7 Iteration t=t+1=1+1=2 is greater than number epochs

Step8 Compute errors

$$MSE=\frac{1}{n_s}\sum_{i=1}^{n_s}(y_i^a-y_i)^2=\frac{1}{2}\sum_{i=1}^{2}(y_i^a-y_i)^2$$

m=1.2 and c=-0.83					
Sample(i)	X_i^a	Y_i^a	Y_i	SE	AE
1	0.2	3.4	-0.59	15.92	3.99
2	0.4	3.8	-0.35	17.22	4.15

$$MSE=\frac{15.92+17.22}{2}=16.57$$
$$RMSE=\sqrt{MSE}=\sqrt{16.57}=4.07$$
$$MAE=\frac{3.99+4.15}{2}=4.07$$

Email : dr.vvr.research@gmail.com
(Dr. Venkataramana Veeramsetty)

Assignment-9

Build a simple linear regression model using AdaGrad + SGD optimizer to help 33/11KV substation electric utility to trade power effectively in an day ahead energy market by estimating load at a particular hour based on the load at the same day and hour but in previous week. Dataset: Active power load dataset

- Do the manual calculation for two iteration by taking only first two samples in the dataset (No need of data normalization)
- Write the python code to build simple linear regression model using Adagrad+SGD optimizer
  - Do the data normalization
  - Split the data for train and test (90:10)
  - Train the simple linear regression model using Adagrad+SGD with training data
  - Compute MSE, RMSE and MAE with training data
  - Compute MSE, RMSE and MAE with testing data

Adadelta

- Adadelta is an extension of Adagrad that seeks to reduce its aggressive, monotonically decreasing learning rate.

$$E(m,c) = \frac{1}{2}(y_i^a - mx_i^a - c)^2 \tag{69}$$

$$g_m = \frac{\partial E(m,c)}{\partial m} = -(y_i^a - mx_i^a - c)x_i^a \tag{70}$$

$$g_c = \frac{\partial E(m,c)}{\partial c} = -(y_i^a - mx_i^a - c) \tag{71}$$

Email : dr.vvr.research@gmail.com
(Dr. Venkataramana Veeramsetty)

- Idea 1: Accumulate Over Window

$$E^2_{g_x,t} = \gamma E^2_{g_x,t-1} + (1-\gamma)[g_x]^2 \tag{72}$$

$$RMS[g_x] = \sqrt{E^2_{g_x,t} + \epsilon} \tag{73}$$

$$\Delta x_t = -\frac{\eta}{RMS[g_x]}[g_x] \tag{74}$$

$$x_t = x_{t-1} + \Delta x_t \tag{75}$$

- Idea 2: Correct Units

$$E^2_{x,t-1} = \gamma E^2_{x,t-2} + (1-\gamma)[\Delta x_{t-1}]^2 \tag{76}$$

$$RMS[\Delta x_{t-1}] = \sqrt{E^2_{x,t-1} + \epsilon} \tag{77}$$

$$\Delta x_t = -\frac{\eta RMS[\Delta x_{t-1}]}{RMS[g_x]}[g_x] \tag{78}$$

$$x_t = x_{t-1} + \Delta x_t \tag{79}$$

Step1 Read dataset $[x_i^a, y_i^a]$,Initialize η,epochs, set m=m_o and c=c_o, set $E^2_{g_m,0}$=$E^2_{g_c,0}$=0, $E^2_{c,0}$=$E^2_{m,0}$=0,Δm_0=Δc_0=0

Step2 Set Iteration t=1

Step3 Set sample i=1

Step4 Calculate g_m and g_c using equations (148) and (149). and also exponential decaying average of gradient squares using equations (89) and (90)

$$E^2_{g_m,t} = \gamma E^2_{g_m,t-1} + (1-\gamma)[g_m]^2 \tag{80}$$

$$E^2_{g_c,t} = \gamma E^2_{g_c,t-1} + (1-\gamma)[g_c]^2 \tag{81}$$

Step5 Calculate exponential decaying average of step length using equations (82) and (83)

$$E^2_{m,t} = \gamma E^2_{m,t-1} + (1-\gamma)[\Delta m_{t-1}]^2 \tag{82}$$

$$E^2_{c,t} = \gamma E^2_{c,t-1} + (1-\gamma)[\Delta c_{t-1}]^2 \tag{83}$$

Step6 Update m and c using equations (91) and (92) respectively

$$m = m - \frac{\sqrt{E^2_{m,t} + \epsilon}}{\sqrt{E^2_{g_m,t} + \epsilon}} * g_m \tag{84}$$

$$c = c - \frac{\sqrt{E^2_{c,t} + \epsilon}}{\sqrt{E^2_{g_c,t} + \epsilon}} * g_c \tag{85}$$

Step5 Set sample i=i+1, if i> n_s go to next step else go to Step4

Step6 Set iteration t=t+1, if t¿maxiter go tn next step else go to Step3

Step7 Compute error metrics

Step8 Stop

Example

Let consider a sample dataset have one input (X_i^a) and one output (Y_i^a), and number of samples 4. Develop a simple linear regression model with one iteration and using AdaDelta optimizer

Sample(i)	X_i^a	Y_i^a
1	0.2	3.4
2	0.4	3.8
3	0.6	4.2
4	0.8	4.6

Step1 Read dataset,Set η=0.1,epochs=1,m-1,c=-1,$E^2_{g_m,0}=E^2_{g_c,0}=0$, $E^2_{c,0}=E^2_{m,0}=0,\Delta m_0=\Delta c_0=0$

Step2 Set iteration=1

Step3 Set sample i=1

Step4 Calculate g_m and g_c

- $g_m = \frac{\partial E}{\partial m}$=-(3.4+0.8)*0.2=-0.84
- $g_c = \frac{\partial E}{\partial c}$=-(3.4+0.8)=-4.2

- Calculate exponential decay averages $E^2_{g_m,t}$ and $E^2_{g_c,t}$
  - $E^2_{g_m,t}$=0.9*0+(1-0.9)*$(-0.84)^2$=0.071
  - $E^2_{g_c,t}$=0.9*0+(1-0.9)*$(-4.2)^2$=1.764

Step5 Calculate exponential decaying average of step length

- $E^2_{m,1}$=0.9*0+(1-0.9)*0=0
- $E^2_{c,1}$=0.9*0+(1-0.9)*0=0

Step5 Update m and c

- m=1-$\frac{\sqrt{0+10^{-8}}}{\sqrt{0.071+10^{-8}}}$(-0.84)=1.0003
- c=-1-$\frac{\sqrt{0+10^{-8}}}{\sqrt{1.764+10^{-8}}}$(-4.2)=-0.9997

Step5 Set sample i=i+1=2, if $2 not > n_s$ go to Step4

Email : dr.vvr.research@gmail.com
(Dr. Venkataramana Veeramsetty)

Step4 Calculate g_m and g_c

- $g_m = \frac{\partial E}{\partial m}$=-(3.8+0.6)*0.4=-1.76
- $g_c = \frac{\partial E}{\partial c}$=-(3.8+0.6)=-4.4

- Calculate exponential decay averages $E^2_{g_m,t}$ and $E^2_{g_c,t}$
  - $E^2_{g_m,t}$=0.9*0.071+(1-0.9)*$(-1.76)^2$=0.373
  - $E^2_{g_c,t}$=0.9*1.764+(1-0.9)*$(-4.4)^2$=3.523

Step5 Calculate exponential decaying average of step length

- $E^2_{m,1}$=0.9*0+(1-0.9)*0.00032=0.000032
- $E^2_{c,1}$=0.9*0+(1-0.9)*0.00032=0.000032

Step5 Update m and c

- m=1.0003-$\frac{\sqrt{0.000032+10^{-8}}}{\sqrt{0.373+10^{-8}}}$(-1.76)=1.0003-(-0.016)=1.017
- c=-0.9997-$\frac{\sqrt{0.000032+10^{-8}}}{\sqrt{3.523+10^{-8}}}$(-4.4)=-0.9997-(-0.0133)=-0.986

Step5 Set sample i=i+1=3, if 3$not >= n_s$ go to next step

Step6 iteration t=t+1=2>epochs

Email : dr.vvr.research@gmail.com
(Dr. Venkataramana Veeramsetty)

$$MSE=\frac{1}{n_s}\sum_{i=1}^{n_s}(y_i^a-y_i)^2=\frac{1}{2}\sum_{i=1}^{2}(y_i^a-y_i)^2$$

m=1.017 and c=-0.986					
Sample(i)	X_i^a	Y_i^a	Y_i	SE	AE
1	0.2	3.4	-0.783	17.64	4.2
2	0.4	3.8	-0.579	19.17	4.38

$$MSE=\frac{17.64+19.17}{2}=18.41$$
$$RMSE=\sqrt{MSE}=\sqrt{18.41}=4.29$$
$$MAE=\frac{4.2+4.38}{2}=4.29$$

Email : dr.vvr.research@gmail.com
(Dr. Venkataramana Veeramsetty)

Assignment-10

Build a simple linear regression model using AdaDelta + BGD optimizer to help 33/11KV substation electric utility to trade power effectively in an day ahead energy market by estimating load at a particular hour based on the load at same time but previous day .
Dataset: Active power load dataset

- Do the manual calculation for two iteration by taking only first two samples in the dataset (No need of data normalization)
- Write the python code to build simple linear regression model using AdaDelta + BGD optimizer
  - Do the data normalization
  - Split the data for train and test (90:10)
  - Train the simple linear regression model using AdaDelta + BGD with training data
  - Compute MSE, RMSE and MAE with training data
  - Compute MSE, RMSE and MAE with testing data

RMS Prop

- RMSprop and Adadelta have both been developed independently around the same time stemming from the need to resolve Adagrad's radically diminishing learning rates.

$$E(m,c) = \frac{1}{2}(y_i^a - mx_i^a - c)^2 \tag{86}$$

$$g_m = \frac{\partial E(m,c)}{\partial m} = -(y_i^a - mx_i^a - c)x_i^a \tag{87}$$

$$g_c = \frac{\partial E(m,c)}{\partial c} = -(y_i^a - mx_i^a - c) \tag{88}$$

Step1 Read dataset $[x_i^a, y_i^a]$,Initialize η,epochs, set m$=m_o$ and c$=c_o$, set $E^2_{g_m,0}=E^2_{g_c,0}=0$

Step2 Set Iteration t=1

Step3 Set sample i=1

Step4 Calculate g_m and g_c using equations (148) and (149). and also exponential decaying average of gradient squares using equations (89) and (90)

$$E^2_{g_m,t} = \gamma E^2_{g_m,t-1} + (1-\gamma)[g_m]^2 \quad (89)$$

$$E^2_{g_c,t} = \gamma E^2_{g_c,t-1} + (1-\gamma)[g_c]^2 \quad (90)$$

Step5 Update m and c using equations (91) and (92) respectively

$$m = m - \frac{\eta}{\sqrt{E^2_{g_m,t} + \epsilon}} * g_m \tag{91}$$

$$c = c - \frac{\eta}{\sqrt{E^2_{g_c,t} + \epsilon}} * g_c \tag{92}$$

Step6 Set sample i=i+1, if i> n_s go to next step else go to Step4

Step7 Set iteration t=t+1, if t¿maxiter go tn next step else go to Step3

Step8 Compute error metrics

Step9 Stop

Example

Let consider a sample dataset have one input (X_i^a) and one output (Y_i^a), and number of samples 2. Develop a simple linear regression model with one iteration and using RMSProp optimizer

Sample(i)	X_i^a	Y_i^a
1	0.2	3.4
2	0.4	3.8

Step1 Read dataset,Set η=0.1,epochs=1,m=1,c=-1,$E^2_{g_m,0}$=$E^2_{g_c,0}$=0, η=0

Step2 Set iteration=1

Step3 Set sample i=1

Step4 Calculate g_m and g_c

- $g_m = \frac{\partial E}{\partial m}$=-(3.4+0.8)*0.2=-0.84
- $g_c = \frac{\partial E}{\partial c}$=-(3.4+0.8)=-4.2

- Calculate exponential decay averages $E^2_{g_m,t}$ and $E^2_{g_c,t}$
  - $E^2_{g_m,t}$=0.9*0+(1-0.9)*$(-0.84)^2$=0.071
  - $E^2_{g_c,t}$=0.9*0+(1-0.9)*$(-4.2)^2$=1.764

Step5 Update m and c

- m=1-$\frac{0.1}{\sqrt{0.071+10^{-8}}}$*(-0.84)=1.315
- c=-1-$\frac{0.1}{\sqrt{1.764+10^{-8}}}$*(-4.2)=-0.683

Step5 Set sample i=i+1=2, if $2 not > n_s$ go to Step4

Email : dr.vvr.research@gmail.com
(Dr. Venkataramana Veeramsetty)

Step4 Calculate g_m and g_c

- $g_m = \frac{\partial E}{\partial m}$=-(3.8-0.157)*0.4=-1.46
- $g_c = \frac{\partial E}{\partial c}$=-(3.8-0.157)=-3.643

- Calculate exponential decay averages $E^2_{g_m,t}$ and $E^2_{g_c,t}$
  - $E^2_{g_m,t}$=0.9*0.071+(1-0.9)*$(-1.46)^2$=0.277
  - $E^2_{g_c,t}$=0.9*1.764+(1-0.9)*$(-3.64)^2$=2.92

Step5 Update m and c

- m=1.315-$\frac{0.1}{\sqrt{0.277+10^{-8}}}$(-1.46)=1.315-(-0.277)=1.592
- c=-0.683-$\frac{0.1}{\sqrt{2.92+10^{-8}}}$(-3.643)=-0.683-(-0.213)=-0.47

Step5 Set sample i=i+1=3, if $3 not >= n_s$ go to next step

Step6 iteration t=t+1=2>epochs

Email : dr.vvr.research@gmail.com
(Dr. Venkataramana Veeramsetty)

$$\text{MSE}=\frac{1}{n_s}\sum_{i=1}^{n_s}(y_i^a-y_i)^2=\frac{1}{2}\sum_{i=1}^{2}(y_i^a-y_i)^2$$

m=1.592 and c=-0.47					
Sample(i)	X_i^a	Y_i^a	Y_i	SE	AE
1	0.2	3.4	-0.152	12.6	3.55
2	0.4	3.8	0.1668	13.17	3.63

$$\text{MSE}=\frac{12.6+13.17}{2}=12.88$$
$$\text{RMSE}=\sqrt{MSE}=\sqrt{12.88}=3.59$$
$$\text{MAE}=\frac{3.55+3.63}{2}=3.59$$

Email : dr.vvr.research@gmail.com
(Dr. Venkataramana Veeramsetty)

Assignment-11

Build a simple linear regression model using RMSprop + SGD optimizer to help 33/11KV substation electric utility to trade power effectively in an day ahead energy market by estimating load at a particular hour based on the load at one hour and one day before.

Dataset: Active power load dataset

- Do the manual calculation for two iteration by taking only first two samples in the dataset (No need of data normalization)
- Write the python code to build simple linear regression model using RMSprop+SGD optimizer
  - Do the data normalization
  - Split the data for train and test (90:10)
  - Train the simple linear regression model using RMSprop+SGD with training data
  - Compute MSE, RMSE and MAE with training data
  - Compute MSE, RMSE and MAE with testing data

Assignment

Build a simple linear regression model using RMSprop + BGD optimizer to help the people who wants to take insurance. SLR model will estimate the insurance amount based on BMI value.

- Do the manual calculation for two iteration by taking only first two samples in the dataset (No need of data normalization)
- Write the python code to build simple linear regression model using RMSprop+BGD optimizer
  - Do the data normalization
  - Split the data for train and test (90:10)
  - Train the simple linear regression model using RMSprop+BGD with training data
  - Compute MSE, RMSE and MAE with training data
  - Compute MSE, RMSE and MAE with testing data

Adam

- Adaptive Moment Estimation (Adam) is another method that computes adaptive learning rates for each parameter
- In addition to storing an exponentially decaying average of past squared gradients like $E^2_{g_{m,t}}$ Adadelta and RMSprop, Adam also keeps an exponentially decaying average of past gradients g_m

Algorithm:

*Step*1 Read dataset $[x_i^a, y_i^a]$, Set η=0.1, epochs, β_1=0.9, β_2=0.999, $E^2_{g_{m,0}}$=$E^2_{g_{c,0}}$ =0, v_m=v_c=0

*Step*2 Set iteration t=1

*Step*3 Set Sample i=1

*Step*4 Read objective function $E_{m,c}$, Calculate g_m and g_c

$$E_{m,c} = \frac{1}{2}(y_i^a - mx_i^a - c)^2 \tag{93}$$

$$g_m = \frac{\partial E(m,c)}{\partial m} = -(y_i^a - mx_i^a - c)x_i^a \tag{94}$$

$$g_c = \frac{\partial E(m,c)}{\partial c} = -(y_i^a - mx_i^a - c) \tag{95}$$

*Step*5 Calculate exponential decaying average of squared gradients $(E_{m,t}^2)$,$E_{c,t}^2)$ and gradients $v_{m,t}$,$v_{c,t}$.

$$E_{m,t}^2 = \beta_2 E_{m,t-1}^2 + (1-\beta_2)[g_m]^2 \tag{96}$$

$$E_{c,t}^2 = \beta_2 E_{c,t-1}^2 + (1-\beta_2)[g_c]^2 \tag{97}$$

$$v_{c,t} = \beta_1 v_{c,t-1} + (1-\beta_1)g_c \tag{98}$$

$$v_{m,t} = \beta_1 v_{m,t-1} + (1-\beta_1)g_m \tag{99}$$

Step6 Update exponential decaying average of squared gradient and gradient

$$E^{\hat{2}}_{c,t} = \frac{E^2_{c,t}}{1-[\beta_2]^2} \tag{100}$$

$$E^{\hat{2}}_{m,t} = \frac{E^2_{m,t}}{1-[\beta_2]^2} \tag{101}$$

$$v_{\hat{c},t} = \frac{v_{c,t}}{1-[\beta_1]^2} \tag{102}$$

$$v_{\hat{m},t} = \frac{v_{m,t}}{1-[\beta_1]^2} \tag{103}$$

Step7 Update m and c

$$m = m - \frac{\eta}{\sqrt{E^{\hat{2}}_{m,t}+\epsilon}} v_{\hat{m},t} \tag{104}$$

$$c = c - \frac{\eta}{\sqrt{E^{\hat{2}}_{c,t}+\epsilon}} v_{\hat{c},t} \tag{105}$$

Step8 Sample i=i+1, if i> n_s go to next else go to step 4

Step9 Iteration t=t+1, if t>epochs go to next else go to step 3

Step10 Calculate errors and stop

Let consider a sample dataset have one input (X_i^a) and one output (Y_i^a), and number of samples 2. Develop a simple linear regression model with one iteration and using Adam optimizer

Sample(i)	X_i^a	Y_i^a
1	0.2	3.4
2	0.4	3.8

Email : dr.vvr.research@gmail.com
(Dr. Venkataramana Veeramsetty)

- Step1 Read dataset,η=0.1,epochs=1,m=1,c=-1,β_1=0.9,β_2=0.999,$v_{m,0}$=0,$v_{c,0}$=0,$E^2_{g_{m,0}}$=$E^2_{g_{c,0}}$ =0,
- Step2 Set iteration=1
- Step3 Set sample i=1

Step4 Calculate gradients

- $g_m = \frac{\partial E}{\partial m}$=-(3.4+0.8)*0.2=-0.84
- $g_c = \frac{\partial E}{\partial c}$=-(3.4+0.8)=-4.2

Step5 Calculate exponential decaying average of squared gradients and gradients

- $E^2_{m,t}$=0.999*0+(1-0.999)*$(-0.84)^2$=0.0007
- $E^2_{c,t}$=0.999*0+(1-0.999)*$(-4.2)^2$=0.01764
- v^t_c=0.9*0+(1-0.9)*(-4.2)=-0.42
- v^t_m=0.9*0+(1-0.9)*(-0.84)=-0.084

Step6 Update exponential decaying average of squared gradients and gradients

- $\hat{E^2}_{m,t} = \frac{0.0007}{1-0.999^1}$=0.7
- $\hat{E^2}_{c,t} = \frac{0.01764}{1-0.999^1}$=17.64
- $\hat{v_{c,t}} = \frac{-0.42}{1-0.9}$=-4.2
- $\hat{v_{m,t}} = \frac{-0.084}{1-0.9}$=-0.84

Step7 Update m and c

- m=1-$\frac{0.1}{\sqrt{0.7}+10^{-8}}$*(-0.84)=1+0.1=1.1
- c=-1-$\frac{0.1}{\sqrt{17.64}+10^{-8}}$*(-2.21)=-1+0.0526=-0.947

Step8 Sample i=i+1=2, if i is not greater than $n_s = 2$ so go to step 4

Step4 Calculate gradients

- $g_m = \frac{\partial E}{\partial m}$=-(3.8+0.51)*0.4=-1.724
- $g_c = \frac{\partial E}{\partial c}$=-(3.8+0.51)=-4.31

Step5 Calculate exponential decaying average of squared gradients and gradients

- $E^2_{m,t}$=0.999*0.0007+(1-0.999)*$(-1.724)^2$=0.0007+0.003=0.0037
- $E^2_{c,t}$=0.999*0.01764+(1-0.999)*$(-4.31)^2$=0.01762+0.01849=0.03611
- v_c^t=0.9*(-0.42)+(1-0.9)*(-4.31)=-0.378-0.43=-0.81
- v_m^t=0.9*(-0.084)+(1-0.9)*(-1.724)=-0.0756-0.172=-0.2476

Step6 Update exponential decaying average of squared gradients and gradients

- $\hat{E^2}_{m,t} = \frac{0.0037}{1-0.999^2}$=1.851
- $\hat{E^2}_{c,t} = \frac{0.03611}{1-0.999^2}$=18.06
- $\hat{v}_{c,t} = \frac{-0.81}{1-0.9^2}$=-4.26
- $\hat{v}_{m,t} = \frac{-0.2476}{1-0.9^2}$=-1.303

Step7 Update m and c

- m=1.1-$\frac{0.1}{\sqrt{1.851+10^{-8}}}$*(-1.303)=1.1+0.0957=1.1957
- c=-0.947-$\frac{0.1}{\sqrt{18.06+10^{-8}}}$*(-4.26)=-0.947+0.1=-0.847

Step8 Sample i=i+1=3, i is greater than $n_s = 2$ so go to next step

Step9 Iteration j=j+1=2, j is not greater than $epochs = 1$ so go to next step

Step10 Calculate errors and stop

Email : dr.vvr.research@gmail.com
(Dr. Venkataramana Veeramsetty)

$$\text{MSE}=\frac{1}{n_s}\sum_{i=1}^{n_s}(y_i^a-y_i)^2=\frac{1}{2}\sum_{i=1}^{2}(y_i^a-y_i)^2$$

m=1.1957 and c=-0.847					
Sample(i)	X_i^a	Y_i^a	Y_i	SE	AE
1	0.2	3.4	-0.61	16.08	4.01
2	0.4	3.8	-0.37	17.39	4.17

$$\text{MSE}=\frac{16.08+17.39}{2}=16.73$$

$$\text{RMSE}=\sqrt{MSE}=\sqrt{16.73}=4.09$$

$$\text{MAE}=\frac{4.01+4.17}{2}=4.09$$

Email : dr.vvr.research@gmail.com
(Dr. Venkataramana Veeramsetty)

Assignment-9

Estimate the housing prince based on sq.ft. area using simple regression model. Consider Adam optimizer.
Dataset: House Sales in King Country, USA

- Do the manual calculation for two iteration by taking only first two samples in the dataset
- Write the python code to build simple linear regression model using Adam optimizer

Email : dr.vvr.research@gmail.com
(Dr. Venkataramana Veeramsetty)

Assignment

Estimate load at particular hour on 33/11KV substation based on load at same time but previous day. Consider Adam+BGD optimizer. Dataset: Active power load dataset

- Do the manual calculation for two iteration by taking only first two samples in the dataset (No need of data normalization)
- Write the python code to build simple linear regression model using Adam+BGD optimizer
  - Do the data normalization
  - Split the data for train and test (90:10)
  - Train the simple linear regression model using Adam+BGD with training data
  - Compute MSE, RMSE and MAE with training data
  - Compute MSE, RMSE and MAE with testing data

AdaMax

*Step*1 Read dataset $[x_i^a, y_i^a]$, Set η=0.002, epochs, β_1=0.9, β_2=0.999, u_m=u_c =0, E_{gm}=E_{gc}=0

*Step*2 Set iteration t=1

*Step*3 Set Sample i=1

Step4 Read objective function $E_{m,c}$, Calculate g_m and g_c

$$E_{m,c} = \frac{1}{2}(y_i^a - mx_i^a - c)^2 \tag{106}$$

$$g_m = \frac{\partial E(m,c)}{\partial m} = -(y_i^a - mx_i^a - c)x_i^a \tag{107}$$

$$g_c = \frac{\partial E(m,c)}{\partial c} = -(y_i^a - mx_i^a - c) \tag{108}$$

Step5 Calculate exponential decaying average of gradients E_{gm}, E_{gc}.

$$E_{gc} = \beta_1 E_{gc} + (1 - \beta_1)g_c \tag{109}$$

$$E_{gm} = \beta_1 E_{gm} + (1 - \beta_1)g_m \tag{110}$$

Step6 Update exponential decaying average of gradient and exponential weighted average of infinity norm

$$\mu_{c,t} = max(\beta_2 * \mu_{c,t}, |g_{c,t}|) \tag{111}$$

$$\mu_{m,t} = max(\beta_2 * \mu_{m,t}, |g_{m,t}|) \tag{112}$$

$$\hat{E_{gc}} = \frac{E_{gc}}{1 - [\beta_1]^t} \tag{113}$$

$$\hat{E_{gm}} = \frac{E_{gm}}{1 - [\beta_1]^t} \tag{114}$$

Step7 Update m and c

$$m = m - \frac{\eta}{\mu_{m,t}} \hat{E_{gm}} \tag{115}$$

$$c = c - \frac{\eta}{\mu_{c,t}} \hat{E_{gc}} \tag{116}$$

Step8 Sample i=i+1, if i> n_s go to next else go to step 4

Step9 Iteration t=t+1, if t>epochs go to next else go to step 3

Step10 Calculate errors and stop

Example

Let consider a sample dataset have one input (X_i^a) and one output (Y_i^a), and number of samples 2. Develop a simple linear regression model with one iteration and using AdaMax optimizer

Sample(i)	X_i^a	Y_i^a
1	0.2	3.4
2	0.4	3.8

Step1 Read dataset,η=0.002,epochs=1,m=1,c=-1,β_1=0.9,β_2=0.999,$v_{m,0}$=0,$v_{c,0}$=0,$E^2_{g_{m,0}}=E^2_{g_{c,0}}$ =0,

Step2 Set iteration=1

Step3 Set sample i=1

Step4 Calculate gradients

- $g_m = \frac{\partial E}{\partial m}$=-(3.4+0.8)*0.2=-0.84
- $g_c = \frac{\partial E}{\partial c}$=-(3.4+0.8)=-4.2

Step5 Calculate exponential decaying average of squared gradients and gradients

- E_{gc}=0.9*0+(1-0.9)*(-4.2)=-0.42
- E_{gm}=0.9*0+(1-0.9)*(-0.84)=-0.084

Step6 Update exponential decaying average of gradients and exponential weighted average of infinity norm

- $\mu_{m,t}$=max(0.999*0,0.84)=0.84
- $\mu_{c,t}$=max(0.999*0,4.2)=4.2
- $\hat{E_{gc}} = \frac{-0.42}{1-0.9}$=-4.2
- $\hat{v_{gm}} = \frac{-0.084}{1-0.9}$=-0.84

Step7 Update m and c

- m=1-$\frac{0.002}{0.84}$*(-0.84)=1+0.002=1.002
- c=-1-$\frac{0.002}{4.2}$*(-4.2)=-1+0.002=-0.998

Step8 Sample i=i+1=2, if i is not greater than $n_s = 2$ so go to step 4

Email : dr.vvr.research@gmail.com
(Dr. Venkataramana Veeramsetty)

Step4 Calculate gradients

- $g_m = \frac{\partial E}{\partial m}$=-(3.8+0.6)*0.4=-1.76
- $g_c = \frac{\partial E}{\partial c}$=-(3.8+0.6)=-4.4

Step5 Calculate exponential decaying average of squared gradients and gradients

- E_{gc}=0.9*(-0.42)+(1-0.9)*(-4.4)=-0.378-0.44=-0.82
- E_{gm}=0.9*(-0.084)+(1-0.9)*(-1.76)=-0.0756-0.176=-0.252

Step6 Update exponential decaying average of gradients and infinity norm

- $\mu_{m,t}$=max(0.999*0.84,1.76)=1.76
- $\mu_{c,t}$=max(0.999*4.2,4.4)=4.4
- $\hat{E_{gc}} = \frac{-0.82}{1-0.9}$=-8.2
- $\hat{E_{gm}} = \frac{-0.252}{1-0.9}$=-2.52

Email :dr.vvr.research@gmail.com
(Dr. Venkataramana Veeramsetty)

Step7 Update m and c

- $m=1.002-\frac{0.002}{1.76}*(-2.523)=1.002+0.0957=1.005$
- $c=-0.998-\frac{0.002}{4.4}*(-8.2)=-0.998+0.1=-0.994$

Step8 Sample i=i+1=3, i is greater than $n_s = 2$ so go to next step

Step9 Iteration j=j+1=2, j is not greater than $epochs = 1$ so go to next step

Step10 Calculate errors and stop

$$\text{MSE}=\frac{1}{n_s}\sum_{i=1}^{n_s}(y_i^a - y_i)^2=\frac{1}{2}\sum_{i=1}^{2}(y_i^a - y_i)^2$$

m=1.005 and c=-0.994					
Sample(i)	X_i^a	Y_i^a	Y_i	SE	AE
1	0.2	3.4	-0.793	17.56	4.19
2	0.4	3.8	-0.592	19.27	4.39

$$\text{MSE}=\frac{17.56+19.27}{2}=18.42$$
$$\text{RMSE}=\sqrt{MSE}=\sqrt{16.73}=4.29$$
$$\text{MAE}=\frac{4.19+4.39}{2}=4.09$$

Email : dr.vvr.research@gmail.com
(Dr. Venkataramana Veeramsetty)

Assignment-10

Estimate the housing prince based on sq.ft. area using simple regression model. Consider AdaMax optimizer.
Dataset: House Sales in King Country, USA

- Do the manual calculation for two iteration by taking only first two samples in the dataset
- Write the python code to build simple linear regression model using AdaMax optimizer

Assignment

Estimate load at particular hour on 33/11KV substation based on load at same time but previous day. Consider AdaMax+BGD optimizer.

Dataset: Active power load dataset

- Do the manual calculation for two iteration by taking only first two samples in the dataset (No need of data normalization)
- Write the python code to build simple linear regression model using AdaMax+BGD optimizer
  - Do the data normalization
  - Split the data for train and test (90:10)
  - Train the simple linear regression model using AdaMax+BGD with training data
  - Compute MSE, RMSE and MAE with training data
  - Compute MSE, RMSE and MAE with testing data

Email : dr.vvr.research@gmail.com
(Dr. Venkataramana Veeramsetty)

NADAM (Nesterov-accelerated Adaptive Moment Estimation)

- It is a combination of Adam and Nesterov accelerated gradient (NAG)
- Momentum involves computing step in the direction of previous momentum vector and current gradient

$$g_m = \frac{\partial E(m,c)}{\partial m} \tag{117}$$

$$g_c = \frac{\partial E(m,c)}{\partial c} \tag{118}$$

$$v_{m,t} = \gamma v_{m,t-1} + \eta g_m \tag{119}$$

$$v_{c,t} = \gamma v_{c,t-1} + \eta g_c \tag{120}$$

$$m = m - v_{m,t} = m - \gamma v_{m,t-1} - \eta g_m \tag{121}$$

$$c = c - v_{c,t} = c - \gamma v_{c,t-1} - \eta g_c \tag{122}$$

- NAG allows to take more accurate step by computing the gradient based on previous momentum

$$g_m = \frac{\partial E(m - \gamma v_{m,t-1}, c)}{\partial m} \tag{123}$$

$$g_c = \frac{\partial E(m, c - v_{c,t-1})}{\partial c} \tag{124}$$

$$v_{m,t} = \gamma v_{m,t-1} + \eta g_m \tag{125}$$

$$v_{c,t} = \gamma v_{c,t-1} + \eta g_c \tag{126}$$

$$m = m - v_{m,t} = m - \gamma v_{m,t-1} - \eta g_m \tag{127}$$

$$c = c - v_{c,t} = c - \gamma v_{c,t-1} - \eta g_c \tag{128}$$

- **Dozet** modifies NAG as

$$g_m = \frac{\partial E(m,c)}{\partial m} \tag{129}$$

$$g_c = \frac{\partial E(m,c)}{\partial c} \tag{130}$$

$$v_{m,t} = \gamma v_{m,t-1} + \eta g_m \tag{131}$$

$$v_{c,t} = \gamma v_{c,t-1} + \eta g_c \tag{132}$$

$$m = m - v_{m,t} = m - \gamma v_{m,t} - \eta g_m \tag{133}$$

$$c = c - v_{c,t} = c - \gamma v_{c,t} - \eta g_c \tag{134}$$

- Now combine NAG with ADAM as shown below

$$v_{m,t} = \beta_1 v_{m,t-1} + (1-\beta_1) g_m \tag{135}$$

$$v_{c,t} = \beta_1 v_{c,t-1} + (1-\beta_1) g_c \tag{136}$$

$$\hat{v_{m,t}} = \frac{v_{m,t}}{1-\beta_1^t} \tag{137}$$

$$\hat{v_{c,t}} = \frac{v_{c,t}}{1-\beta_1^t} \tag{138}$$

$$m_t = m_t - \frac{\eta}{\sqrt{\hat{E^2_{m,t}} + \epsilon}} \hat{v_{m,t}} \tag{139}$$

$$c_t = c_t - \frac{\eta}{\sqrt{\hat{E^2_{c,t}} + \epsilon}} \hat{v_{c,t}} \tag{140}$$

$$m_t = m_t - \frac{\eta}{\sqrt{E_{m,t}^{\hat{2}}} + \epsilon}\left(\frac{\beta_1 v_{m,t-1}}{1-\beta_1^t} + \frac{(1-\beta_1)g_{m,t}}{1-\beta_1^t}\right) \quad (141)$$

$$c_t = c_t - \frac{\eta}{\sqrt{E_{m,t}^{\hat{2}}} + \epsilon}\left(\frac{\beta_1 v_{c,t-1}}{1-\beta_1^t} + \frac{(1-\beta_1)g_{c,t}}{1-\beta_1^t}\right) \quad (142)$$

$$m_t = m_t - \frac{\eta}{\sqrt{E_{m,t}^{\hat{2}}} + \epsilon}\left(\beta_1 v_{m,\hat{t}-1} + \frac{(1-\beta_1)g_{m,t}}{1-\beta_1^t}\right) \quad (143)$$

$$c_t = c_t - \frac{\eta}{\sqrt{E_{m,t}^{\hat{2}}} + \epsilon}\left(\beta_1 v_{m,\hat{t}-1} + \frac{(1-\beta_1)g_{c,t}}{1-\beta_1^t}\right) \quad (144)$$

$$m_t = m_t - \frac{\eta}{\sqrt{E_{m,t}^{\hat{2}}} + \epsilon}\left(\beta_1 v_{\hat{m},t} + \frac{(1-\beta_1)g_{m,t}}{1-\beta_1^t}\right) \quad (145)$$

$$c_t = c_t - \frac{\eta}{\sqrt{E_{m,t}^{\hat{2}}} + \epsilon}\left(\beta_1 v_{\hat{m},t} + \frac{(1-\beta_1)g_{c,t}}{1-\beta_1^t}\right) \quad (146)$$

NADAM-Algorithm

*Step*1 Read dataset $[x_i^a, y_i^a]$, Set η=0.1, epochs, β_1=0.9, β_2=0.999, $E^2_{g_{m,0}} = E^2_{g_{c,0}} = 0$, $v_m = v_c = 0$

*Step*2 Set iteration t=1

*Step*3 Set Sample i=1

*Step*4 Read objective function $E_{m,c}$, Calculate g_m and g_c

$$E_{m,c} = \frac{1}{2}(y_i^a - mx_i^a - c)^2 \tag{147}$$

$$g_m = \frac{\partial E(m,c)}{\partial m} = -(y_i^a - mx_i^a - c)x_i^a \tag{148}$$

$$g_c = \frac{\partial E(m,c)}{\partial c} = -(y_i^a - mx_i^a - c) \tag{149}$$

*Step*5 Calculate exponential decaying average of squared gradients $(E^2_{m,t}), E^2_{c,t})$ and gradients $v_{m,t}, v_{c,t}$.

$$E^2_{m,t} = \beta_2 E^2_{m,t-1} + (1-\beta_2)[g_m]^2 \tag{150}$$

$$E^2_{c,t} = \beta_2 E^2_{c,t-1} + (1-\beta_2)[g_c]^2 \tag{151}$$

$$v_{c,t} = \beta_1 v_{c,t-1} + (1-\beta_1) g_c \tag{152}$$

$$v_{m,t} = \beta_1 v_{m,t-1} + (1-\beta_1) g_m \tag{153}$$

Step6 Update exponential decaying average of squared gradient and gradient

$$E^{\hat{2}}_{c,t} = \frac{E^2_{c,t}}{1-[\beta_2]^2} \tag{154}$$

$$E^{\hat{2}}_{m,t} = \frac{E^2_{m,t}}{1-[\beta_2]^2} \tag{155}$$

$$\hat{v}_{c,t} = \frac{v_{c,t}}{1-[\beta_1]^2} \tag{156}$$

$$\hat{v}_{m,t} = \frac{v_{m,t}}{1-[\beta_1]^2} \tag{157}$$

Step7 Update m and c

$$m = m - \frac{\eta}{\sqrt{\hat{E^2_{m,t}}} + \epsilon}(\beta_1 \hat{v_{m,t}} + \frac{(1-\beta_1)g_{m,t}}{1-\beta_1^t}) \quad (158)$$

$$c = c - \frac{\eta}{\sqrt{\hat{E^2_{c,t}}} + \epsilon}(\beta_1 \hat{v_{c,t}} + \frac{(1-\beta_1)g_{c,t}}{1-\beta_1^t}) \quad (159)$$

Step8 Sample i=i+1, if i> n_s go to next else go to step 4

Step9 Iteration t=t+1, if t>epochs go to next else go to step 3

Step10 Calculate errors and stop

Email: dr.vvr.research@gmail.com
(Dr. Venkataramana Veeramsetty)

Example

Let consider a sample dataset have one input (X_i^a) and one output (Y_i^a), and number of samples 2. Develop a simple linear regression model with one iteration and using NADAM optimizer

Sample(i)	X_i^a	Y_i^a
1	0.2	3.4
2	0.4	3.8

Step1 Read dataset,η=0.1,epochs=1,m=1,c=-1,β_1=0.9,β_2=0.999,$v_{m,0}$=0,$v_{c,0}$=0,$E^2_{g_{m,0}}$=$E^2_{g_{c,0}}$ =0,

Step2 Set iteration=1

Step3 Set sample i=1

Step4 Calculate gradients

- $g_m = \frac{\partial E}{\partial m}$=-(3.4+0.8)*0.2=-0.84
- $g_c = \frac{\partial E}{\partial c}$=-(3.4+0.8)=-4.2

Step5 Calculate exponential decaying average of squared gradients and gradients

- $E^2_{m,t}$=0.999*0+(1-0.999)*$(-0.84)^2$=0.0007
- $E^2_{c,t}$=0.999*0+(1-0.999)*$(-4.2)^2$=0.01764
- v_c^t=0.9*0+(1-0.9)*(-4.2)=-0.42
- v_m^t=0.9*0+(1-0.9)*(-0.84)=-0.084

Step6 Update exponential decaying average of squared gradients and gradients

- $\hat{E^2}_{m,t} = \frac{0.0007}{1-0.999^1}$=0.7
- $\hat{E^2}_{c,t} = \frac{0.01764}{1-0.999^1}$=17.64
- $\hat{v}_{c,t} = \frac{-0.42}{1-0.9}$=-4.2
- $\hat{v}_{m,t} = \frac{-0.084}{1-0.9}$=-0.84

Step7 Update m and c

- m=1-$\frac{0.1}{\sqrt{0.7+10^{-8}}}$*(0.9*-0.84-0.84)=1-0.12(-1.6)=1+0.19=1.19
- c=-1-$\frac{0.1}{\sqrt{17.64+10^{-8}}}$*(0.9*-4.2-4.2)=-1-0.0238(-7.98)=-1+0.19=-0.81

Step8 Sample i=i+1=2, i is not greater than $n_s = 2$ so go to step 4

Step4 Calculate gradients

- $g_m = \frac{\partial E}{\partial m}$=-(3.8+0.334)*0.4=-1.65
- $g_c = \frac{\partial E}{\partial c}$=-(3.8+0.334)=-4.13

Step5 Calculate exponential decaying average of squared gradients and gradients

- $E^2_{m,t}$=0.999*0.0007+(1-0.999)*$(-1.65)^2$=0.0007+0.0027=0.0034
- $E^2_{c,t}$=0.999*0.01764+(1-0.999)*$(-4.13)^2$=0.01762+0.017=0.035
- v_c^t=0.9*(-0.42)+(1-0.9)*(-4.13)=-0.378-0.413=-0.791
- v_m^t=0.9*(-0.084)+(1-0.9)*(-1.65)=-0.0756-0.165=-0.241

Step6 Update exponential decaying average of squared gradients and gradients

- $\hat{E^2_{m,t}} = \frac{0.0034}{1-0.999^2}$=1.7
- $\hat{E^2_{c,t}} = \frac{0.035}{1-0.999^2}$=17.51
- $\hat{v_{c,t}} = \frac{-0.791}{1-0.9^2}$=-4.16
- $\hat{v_{m,t}} = \frac{-0.241}{1-0.9^2}$=-1.27

Step7 Update m and c

- m=1.19-$\frac{0.1}{\sqrt{1.7+10^{-8}}}$*(0.9*-1.27+$\frac{0.1*-1.65}{0.19}$)
=1.19-0.077(-1.143-0.868)=1.19+0.155=1.345
- c=-0.81-$\frac{0.1}{\sqrt{17.51+10^{-8}}}$*(0.9*-4.16+$\frac{0.1*-4.13}{0.19}$)=-0.81-0.024*(-3.74-2.17)=-0.81+0.142=-0.668

Step8 Sample i=i+1=3, i is greater than $n_s = 2$ so go to next step

Step9 Iteration j=j+1=2, j is not greater than $epochs = 1$ so go to next step

Step10 Calculate errors and stop

$$MSE = \frac{1}{n_s}\sum_{i=1}^{n_s}(y_i^a - y_i)^2 = \frac{1}{2}\sum_{i=1}^{2}(y_i^a - y_i)^2$$

m=1.345 and c=-0.668					
Sample(i)	X_i^a	Y_i^a	Y_i	SE	AE
1	0.2	3.4	-0.4	14.44	3.8
2	0.4	3.8	-0.13	15.44	3.93

$$MSE = \frac{14.44+15.44}{2} = 14.94$$
$$RMSE = \sqrt{MSE} = \sqrt{16.73} = 3.87$$
$$MAE = \frac{4.01+4.17}{2} = 3.885$$

Email : dr.vvr.research@gmail.com
(Dr. Venkataramana Veeramsetty)

Assignment-11

Estimate the housing prince based on sq.ft. area using simple regression model. Consider NADAM optimizer.
Dataset: House Sales in King Country, USA

- Do the manual calculation for two iteration by taking only first two samples in the dataset
- Write the python code to build simple linear regression model using NADAM optimizer

Conclusions

- If input data is sparse, then likely achieve the best results using one of the adaptive learning-rate methods like Adagrad, Adadelta, RMSprop, AdaMax and NADAM.
- An additional benefit is that you will not need to tune the learning rate
- RMSprop is an extension of Adagrad that deals with its radically diminishing learning rates.
- RMSprop is identical to Adadelta, except that Adadelta uses the RMS of parameter updates in the numerator update rule.
- RMSprop, Adadelta, and Adam are very similar algorithms that do well in similar circumstances.

- Adam slightly outperform RMSprop towards the end of optimization as gradients become sparser. Insofar, Adam might be the best overall choice.
- SGD usually achieves to find a minimum, but it might take significantly longer than with some of the optimizers
- SGD performance mostly depends on initial point and scheduling
- Fast convergence and train a deep or complex neural network, one of the adaptive learning rate methods.

References I

[1] T. Dozat. Incorporating nesterov momentum into adam. 2016.

[2] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[3] N. Qian. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151, 1999.

[4] S. Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[5] M. D. Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[6] S. Zhang, A. E. Choromanska, and Y. LeCun. Deep learning with elastic averaging sgd. In *Advances in neural information processing systems*, pages 685–693, 2015.

Email : dr.vvr.research@gmail.com
(Dr. Venkataramana Veeramsetty)