University of New Orleans
Department of Computer Science

Spr 2019: CSCI 6990 Homework # 1

Machine Learning - I

Submitted to:
Professor Dr. Tamjidul Hoque

By
Name: **Astha Sharma**
Student ID: **2564173**

PART (A)

Rewrite the *CG* for minimizing MSE (mean square error) in terms of β. Replace, ∇f, *P*, *d and Q* from the CG algorithm with y_i, x_i ,β_j,X,Y,βetc. as needed. Show your derivation(s) (if any).

SOLUTION:

Here,

Mean Square Error (MSE) $= \frac{1}{N} RSS(\beta)$

$= \frac{1}{N} \sum_{i=1}^{N} (y_i - x_i\beta)^2$ ---------------------------------- (i)

In order to perform square operation in (i), we can write it as:

$= \frac{1}{N} (Y - X\beta)^T (Y - X\beta)$ (since $A^2 = A^T A$),

where, X is a N x p matrix including each input and Y is a N vector output

$= \frac{1}{N} (Y^T (X\beta)^T)(Y - X\beta)$

$= \frac{1}{N} (Y^T Y - Y^T X\beta - (X\beta)^T Y + (X\beta)^T X\beta$

$= \frac{1}{N} (Y^T Y - (X\beta)^T Y - (X\beta)^T Y + \beta^T X^T X\beta$

(since, $A^T B = B^T A$ and $(AB)^T = B^T A^T$)

MSE $= \frac{1}{N} (Y^T Y - 2\beta^T X^T Y + \beta^T X^T X\beta)$ ------------------ (ii)

$= \frac{Y^T Y}{N} - \frac{2\beta^T X^T Y}{N} + \frac{\beta^T X^T X\beta}{N}$

Now, CG can be derived from steepest descent method by converging it quadratically

We know, the quadratic equation is given by:

$f(X) = \frac{1}{2} X^T Q X - b^T X + C$ -- (iii)

equation (ii) can be re-written as:

MSE $= \frac{1}{2} * 2\, \beta^T \frac{X^T X}{N} \beta - \beta^T \frac{2X^T Y}{N} + \frac{Y^T Y}{N}$

MSE = $\frac{1}{2} * 2\, \beta^T \frac{X^T X}{N} \beta - (\frac{2X^T Y}{N})^T \beta + \frac{Y^T Y}{N}$ --------------------------------- (iv)

(since, $A^T B = B^T A$)

Comparing (iii) and (iv), we get:

Q = $\frac{2X^T X}{N}$

b = $\frac{2X^T Y}{N}$

C = $\frac{Y^T Y}{N}$

Differentiating equation (ii) with respect to β, we get

$$RSS'(\beta) = \frac{1}{N} * [\frac{\partial}{\partial \beta}(Y^T Y - 2\beta^T X^T Y + \beta^T X^T X \beta)]$$

$$= \frac{1}{N} * [\frac{\partial}{\partial \beta}(Y^T Y) - 2X^T Y \frac{\partial}{\partial \beta}(\beta^T) + \left(\beta^T X^T X \frac{\partial}{\partial \beta}(\beta) + X^T X \beta \frac{\partial}{\partial \beta}(\beta^T)\right)]$$

$$= \frac{1}{N} * [0 - 2X^T Y + \beta^T X^T X + X^T X \beta]$$

$$= \frac{1}{N} * [- 2X^T Y + 2X^T X \beta] \quad \text{(since } B^T A^T = (AB)^T \text{ and } (A)^T B = (B)^T A \text{)}$$

$$= \frac{1}{N} * [\, 2X^T X \beta - 2X^T Y \,]$$

Therefore, the given CG algorithm can be rewritten as follows:

BEGIN

STEP 1: Set i=0; select the initial point as β(0)

STEP 2: If ∇f(β(0)) = 0, STOP; else set d_0 = -∇f (β(0))

STEP 3: Compute: $\alpha_i = - \frac{\nabla f \, d_i}{d_i^T Q d_i}$

STEP 4: Compute: $\beta_{(i+1)} = \beta_{(i)} + \alpha_i d_i$

STEP 5: Compute: ∇f ($\beta_{(i+1)}$). If $\nabla f(\beta_{(i+1)})$ = 0, STOP

STEP 6: Compute: $\gamma_i = \frac{\nabla f(\beta_{i+1})^T Q \, d_i}{d_i^T Q d_i}$

STEP 7: Compute: $d_{i+1} = -\nabla f\,(\beta_{(i+1)}) + \gamma_i d_i$

STEP 8: Set i=i+1; go to STEP 3.

END

PART (B)

Draw the space of the equation (in MATLAB): $f(x_1, x_2) = x_1 - x_2 + 2x_1^{\,2} + 2x_1x_2 + x_2$, twice for the following two cases:

(*i*) on the space-figure, superimpose iteratively obtained all the points starting from

$X_0 = \begin{Bmatrix} 0 \\ \ \end{Bmatrix}$ to X_6 using steepest descent algorithms. Then sequentially connect those points: $\{X_0, \ldots, X_6\}$ using lines. Appropriately, label your figure.

<u>SOLUTION</u>

MATLAB Code:

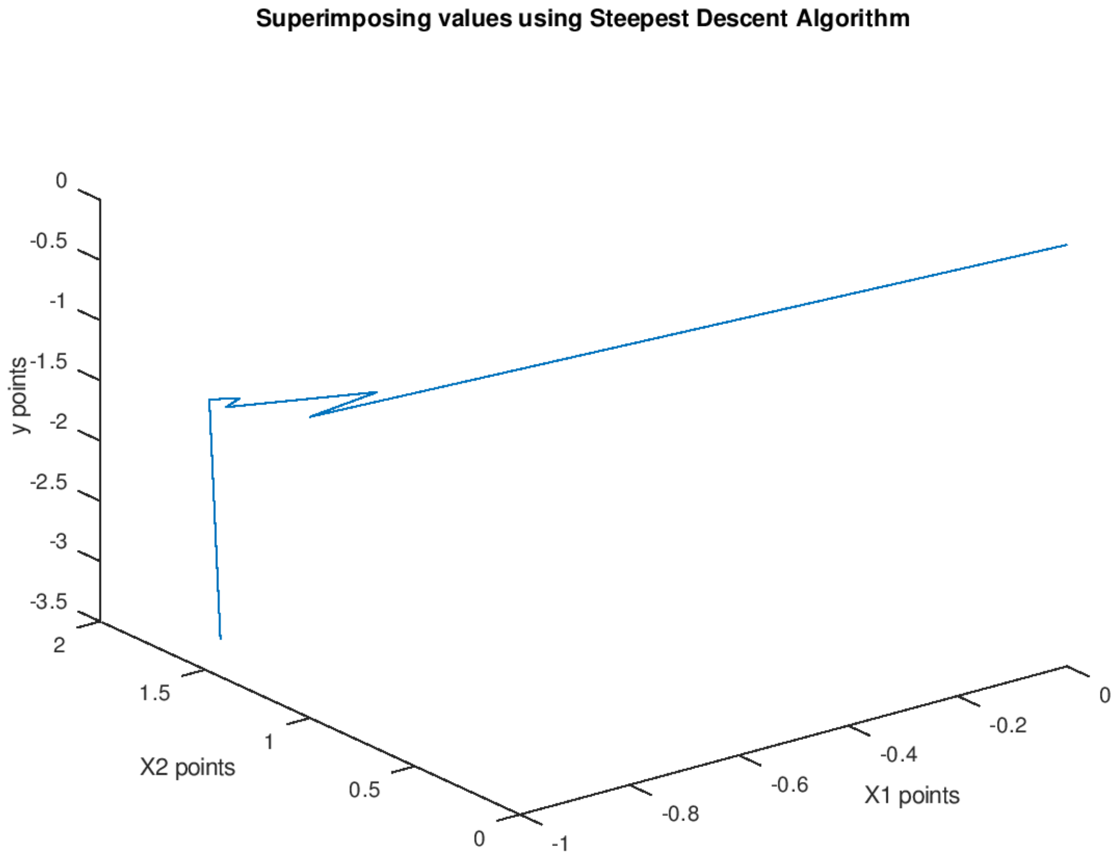
```
%initializing at origin
X = [0;0];
x1_plot = [0];
x2_plot = [0];
y_plot=[0];

i=0; %for counting iteration
while i<6

%the differentiation of given quadratic equation wrt to x1 is 1+4x1+2x2 and that wrt to x2 is -
1+2x1+2x2

%now substituting values of each x1 and x2 for new iteration, we have
val_x1 = 1+4*X(1)+2*X(2);
val_x2 = -1+2*X(1)+2*X(2);

%since direction is negative matrix of obtained differentiation value
direction = [-val_x1; -val_x2];
a=sym(a);
a_x = X(1)+(a*direction(1));
a_y = X(2)+(a*direction(2));

%substituting the x and y coordinates of alpha to the function, we get
func = (a_x)-(a_y)+(2*a_x*a_x)+(2*a_x*a_y)+(a_y*a_y);
diff_func =diff(func,a);
alpha_val=solve(diff_func,a);
```

```
X_update = X+(alpha_val.*direction);

%generating values of Y for corresponding values of x1 and x2
y = X_update(1) – X_update(2) + 2*X_update(1)*X_update(1) + 2*X_update(1)*X_update(2) +
X_update(2)*X_update(2);

%computing updated values of x1 and x2
x_plot = [x_plot X_update(1)];
y_plot = [y_plot X_update(2)];

X= X_update;
i= i+1;
end

%displaying of the final points

disp(x1_plot);
disp(x2_plot);
disp(y_plot);

%for calculated values of x1, x2, and y, the graph can be plotted as:
Plot3(x1_plot,x2_plot,y_plot);
title('Superimosing values using Steepest Descent Algorithm');
xlabel('X1 points');
ylabel('X2 points');
zlabel('Y points');
```

OUTPUT:

```
[ 0, -1, -0.8, -1, -0.96, -1, -0.992]
[ 0, 1, 1.2, 1.4, 1.44, 1.48, 1.448]
[0, -1, -1.2, -1.24, -1.248, -1.24, -3.216]
```

GRAPH:



(ii) Similarly, for the same equation-space (but in a separate figure), place and connect the points sequentially obtained by conjugate gradient approach. Appropriately, label your figure.

<u>SOLUTION</u>

MATLAB CODE:

```
%initialing Q from given equation and starting point as origin
X=[0;0];
Q=[4 2;2 2];
x1_plot=[0];
x2_plot=[0];
y_plot=[0];

i=0; %setting variable for iteration

%the differentiation of given quadratic equation wrt to x1 is 1+4x1+2x2 and that wrt to x2 is -
1+2x1+2x2
%now substituting values of each x1 and x2 for new iteration, we have
val_x1=1+(4*X(1))+(2*X(2));
val_x2=(2*X(1))+(2*X(2))-1;
```

```
func=[val_x1; val_x2];
dir=[-val_x1; -val_x2];

while i<2

%computing transpose function
func_t=func.';
dir_T=dir.';

%calculating the value of alpha
a=-dot(func_t,dir)/dot(dir_T,(Q*dir));

%compute X values
X_update=X.'+a*dir_T;
X_update=X_update.';

dir_x1=1+(4*X_update(1))+(2*X_update(2));
dir_x2=(2*X_update(1))+(2*X_update(2))-1;
func=[dir_x1; dir_x2];

%computing the value for beta
b=dot(func.',(Q*dir))/dot(dir_T,(Q*dir));
dir_update=-func+b.*dir;
dir=dir_update;

%generating values of Y for corresponding values of x1 and x2
y = X_update(1) – X_update(2) + 2*X_update(1)*X_update(1) + 2*X_update(1)*X_update(2) +
X_update(2)*X_update(2);


x_plot=[x_plot X_update(1)];
y_plot=[y_plot X_update(2)];

%updating the matrix X

X=X_update;
i=i+1;
end

%displaying of the final points
disp(x1_plot);
disp(x2_plot);
disp(y_plot);

%for calculated values of x1 and x2, the graph can be plotted as:
Plot3(x1_plot,x2_plot,y_plot);
title('Superimosing value using Conjugate Gradient Algorithm');
```

```
xlabel('X1 points');
ylabel('X2 points');
zlabel('Y points');
```

OUTPUT:

```
0  -1  -1
0.00000   1.00000   1.50000
0.00000  -1.00000  -1.25000
```

GRAPH: