EEE 591 Machine Learning with deployment to FPGA

HOMEWORK 1

ASU ID: 1225713099

Praveen Paidi

1. **What are the zeroth, first, and second conditions of optimality?**

1.1

Zeroth order condition of optimality: w* such that $g(w^*) \leq g(w)$ for all w which means that there is a w* which is an optimal solution.

1.2

First order condition of optimality: For a vector function f: $R^n \rightarrow R$ be differentiable, such that $\nabla f(X) = 0$ at each extremum.

1.3

Second order condition of optimality: The derivative of a function $\nabla f(X) = 0$ which is extremum and if the f ''(x) is Positive at this extremum it is a curve up or minimum.

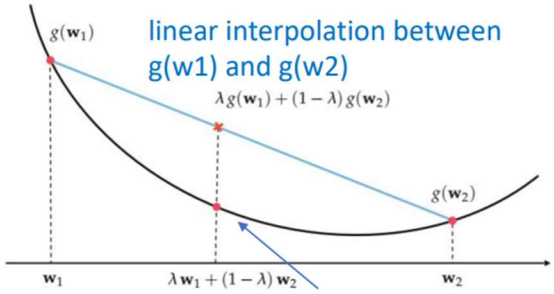
2) What are the zeroth, first conditions of convexity?

2.1

For any function, let λ between 0 and 1, w1 and w2 are any points, then it is said to be convex if it satisfies the following inequality:

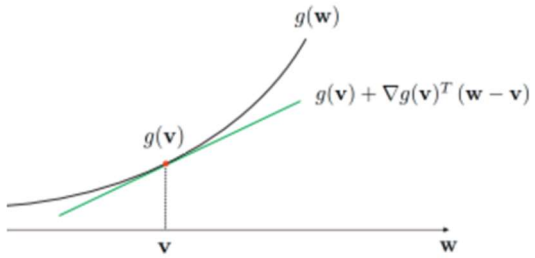
$g(\lambda w1 + (1 - \lambda)w2) \leq \lambda g(w1) + (1 - \lambda)g(w2)$

g(w1) and g(w2), evaluation at that point is below this point in a convex function



2.2

The differentiable function g is convex if and only if it's domain is convex and $g(w) \geq \nabla g(v)(w - v) + g(v)$ for all w, v ∈ domain(g)



3. What is the Least Squares Cost function? Use the compact notation. Show that it is convex.

Consider we model a system to be linear, the equation for which is given by:

$Y_p = x_p^T w + b$

$g(b, w) = \frac{1}{P}\sum_{P=1}^{P}\left(x_p^T w + b - y_p\right)^2$

where X_p is a single observation and Y_p is the corresponding target value; w is the vector of weights and b is the offset.

$Y_p = X_p^T W$

$Xp = \begin{bmatrix} 1 \\ X \end{bmatrix}$

$W = \begin{bmatrix} b \\ W \end{bmatrix}$

The compact notation results in :

$g(W) = \frac{1}{P}\sum_{P=1}^{P}\left(X_p^T W - y_p\right)^2$

Convex proof:

Function given by $g(W) = \frac{1}{P}\sum_{P=1}^{P}\left(X_p^T W - y_p\right)^2$

First Derivative given by: $Vg(W) = \frac{2}{P}\sum_{P=1}^{P} X_p\left(X_p^T W - y_p\right)$

Second Derivative function given by: $V^2 g(W) = \frac{2}{P}\sum_{P=1}^{P}\left(X_p^T X_p\right)$

Second derivative is a non-negative and defines cost function as convex function.

4. Code results:

```
Q4.a) Intercept :  0.07249080791598317
weights:  [ 2.04773925  3.16044263 -5.09550919 10.20993353]
Residual Squared :  0.9969209428506091

Q4.b) Intercept :  0.07249080791616605
weights:  [ 2.04773925  3.16044263 -5.09550919 10.20993353]
Residual Squared :  0.9969209428506091
Q4.c)
 Residual Squared from linalginv is :  0.9952691434895805
 Residual Squared  from linalgsolve is :  0.995269143489581

Q5) Intercept :  0.07550803385981701
weights:  [ 2.04767117  3.16032006 -5.09548412 10.20969749]
Residual Squared :  0.9952705595646534

Q6) intercept : 0.07249080791578422
weights: [ 2.04773925  3.16044263 -5.09550919 10.20993353]
Residual Squared :  0.9952691434895807
```

4. **What can you conclude about your trained model from a) and b)?**

Due to the fact that, np.linalg.solve does not find the inverse of the matrix, it is supposed
To be faster than np.linalg.inv. also, as it does not determine the inverse, no approximations or errors are introduced.
The R2 score of the linalg.inv is 0.9969209428506091.
The R2 score of the linalg.solve is 0.9969209428506091
The intercepts of inv and solve are 0.07249080791598317, 0.07249080791616605 respectively .
The small difference justifies the explanation.

5. **Compare with your solution from (4)**

The Gradient descent building from scratch has slightest differences in the solution compared to the above functions.
If we increase the iterations or the increase the accuracy value by decreasing the amount of error, the final output has slight changes in the solution.

6. **compare your resulting R2 and model weights to those in (4) and (5)**

The R2 score is almost equal to the 4^{th} and 5^{th} questions with same thing being happened in the case of weights.
The learning rate, accuracy level , number of iterations can be manipulated to change the output results what we observed.

In [3]:
```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression as lr
from sklearn.metrics import r2_score

## Read train CSV
df = pd.read_csv(r'D:\Spring 23\EEE 591\regressionprob1_train0.csv')
##read test data csv
dftest = pd.read_csv(r'D:\Spring 23\EEE 591\regressionprob1_test0.csv')

# DATA formualtion for train and test
x = df.iloc[:,0:4].values
y = df['F'].values
xtest = dftest.iloc[:,0:4].values
ytest = dftest['F'].values
Train_Size = len(y)
Test_Size = len(ytest)

# creating compact notation
Ones = np.ones(Train_Size,dtype=float).reshape(Train_Size,1)
X_train = np.hstack((Ones,x))
# weights
W = np.matmul(np.linalg.inv(np.dot(X_train.T,X_train)),np.dot(X_train.T,y))
# Prediciting the values for linalginv
Y_atrain= np.matmul(X_train,W)      # err=np.matmul(X,w)
R2a =r2_score(y,Y_atrain)     # r=Rsquared(y,err)


#solve using numpy.linalg.solve (b part)
W_ = np.linalg.solve(np.dot(X_train.T,X_train),np.dot(X_train.T,y))
# Prediciting the values for linalgsolve
Y_btrain= np.matmul(X_train,W_)
R2b = r2_score(y,Y_btrain)


# C part TESTING THE MODEL
Ones = np.ones(Test_Size,dtype=float).reshape(Test_Size,1)  ## make a column vector of ones
# creating compact notation
X_test = np.hstack((Ones,xtest))
#linalginv solution
Y_atest = np.matmul(X_test,W)
R2atest=r2_score(ytest,Y_atest)
#linalgsolve solution
Y_btest= np.matmul(X_test,W_)
R2btest=r2_score(ytest,Y_btest)

print('Q4.a) Intercept : ', W[0], '\nweights: ', W[1:], '\nResidual Squared : ', R2a)
print('\nQ4.b) Intercept : ', W_[0], '\nweights: ', W_[1:], '\nResidual Squared : ', R2b)
print('Q4.c) \n Residual Squared from linalginv is : ', R2atest, '\n Residual Squared  from linalgsolve is : ', R2btest)


# QUESTION 5 GRADIENT DESCENT
error = 0.0000000000001
alpha = 0.001
COUNT= 1
Converged=False
## Initiate Weights
W = np.ones(5)

while(not Converged):
    Wnew = W - alpha*(2/Train_Size)*np.dot((np.dot(X_train,W)-y),X_train)
    err = np.dot(Wnew - W,Wnew - W)
    COUNT += 1
    W = Wnew
    Converged=False

    if np.abs(err)<error :
        Converged=True
# Predicting Y
ypred= np.dot(X_test,W)
r2gradtest=r2_score(ytest,ypred)
print('\nQ5) Intercept : ', W[0], '\nweights: ', W[1:], '\nResidual Squared : ', r2gradtest)


#Question 6
##sklearn.linear_model.LinearRegression
slr = lr().fit(X_train, y)
ylrtest = slr.predict(X_test)
r2=slr.score(X_test,ytest)
print ('\nQ6) intercept :', slr.intercept_ ,'\nweights:', slr.coef_[1:], '\nResidual Squared : ', r2)
```

```
Q4.a) Intercept :  0.07249080791598317
weights:  [ 2.04773925  3.16044263 -5.09550919 10.20993353]
Residual Squared :  0.9969209428506091

Q4.b) Intercept :  0.07249080791616605
weights:  [ 2.04773925  3.16044263 -5.09550919 10.20993353]
Residual Squared :  0.9969209428506091
Q4.c)
 Residual Squared from linalginv is :  0.9952691434895805
 Residual Squared  from linalgsolve is :  0.995269143489581

Q5) Intercept :  0.07550803385981701
weights:  [ 2.04767117  3.16032006 -5.09548412 10.20969749]
Residual Squared :  0.9952705595646534

Q6) intercept : 0.07249080791578422
weights: [ 2.04773925  3.16044263 -5.09550919 10.20993353]
Residual Squared :  0.9952691434895807
```