EEE 591 Machine Learning with deployment to FPGA

HOMEWORK 4

ASU ID: 1225713099

Praveen Paidi

1.

Least square cost function:

g(W) = $\frac{1}{P}\sum_{p=1}^{P}\left(b + X_p^T\, W - Y_p\right)^2$

first derivative is:

d g(W) /dw = $\frac{d}{dw}\frac{1}{P}\sum_{p=1}^{P}\left(b + X_p^T\, W - Y_p\right)^2$

= $2.\frac{1}{P}\sum_{p=1}^{P}\left(X_p^T\, W - Y_p\right)X_p^T.$

Second derivative is given by:

= $\frac{d}{dw}.\,2.\frac{1}{P}\sum_{p=1}^{P} X_p\left(X_p^T\, W - Y_p\right).$

= $2.\frac{1}{P}\, X_p(X_p^T) + 0$

= $2.\frac{1}{P}\,(X_p.\, X_p^T)$

2.

Sigmoid function:

S(x) = $\frac{1}{1+e^{-x}}$

Differentiate the equation:

dS/dx =$\frac{d}{dx}\frac{1}{1+e^{-x}}$

= (- 1. $\frac{d}{dx}$ (1 + e^{-x}) + 0) / (1+ e^{-x})^2

= -1 (0 +(-1). e^{-x}) / (1+ e^{-x})^2

= e^{-x} / (1+ e^{-x})^2

= $\frac{e^{-x}}{(1+\, e^{-x})^2}$

Rationalizing:

$= \frac{1}{(1+e^{-x})} \cdot \frac{e^{-x}}{(1+e^{-x})}$

$= \frac{1}{(1+e^{-x})} \cdot 1 - \frac{1}{(1+e^{-x})}$

$V\sigma(t) = \sigma(t)(1 - \sigma(t))$

3.

Cross Entropy Cost function:

g(W) = $\frac{1}{P} . \sum_{p=1}^{p} -y_p \log\left(\sigma\left(X_p^T w\right)\right) - \left(1 - y_p\right)\log\left(1 - \sigma\left(X_p^T w\right)\right)$

dg/ d(W) = $\frac{1}{P} . \sum_{p=1}^{p} -0. \log\left(\sigma\left(X_p^T w\right)\right) - y_p . \frac{1}{\sigma(X_p^T w)} . \sigma\left(X_p^T\right)$ - $\frac{1}{1-\sigma(X_p^T w)} . \sigma\left(X_p^T\right)$ - y_p $\frac{1}{1-\sigma(X_p^T w)} . \sigma\left(X_p^T\right)$

= $\frac{1}{P} . \sum_{p=1}^{p} -y_p . \frac{1}{\sigma(X_p^T w)} . \sigma\left(X_p^T\right)$ - $\frac{1}{1-\sigma(X_p^T w)} . \sigma\left(X_p^T\right)$ + y_p $\frac{1}{1-\sigma(X_p^T w)} . \sigma\left(X_p^T\right)$

= $\frac{1}{P} . \sum_{p=1}^{p} -y_p . \frac{1}{\sigma(X_p^T w)} . \sigma\left(X_p^T\right)$ + y_p $\frac{1}{1-\sigma(X_p^T w)} . \sigma\left(X_p^T\right)$ - $\frac{1}{1-\sigma(X_p^T w)} . \sigma\left(X_p^T\right)$

= $\frac{1}{P} . \sum_{p=1}^{p} \left(X_p^T \left(y_p - \sigma\left(X_p^T w\right)\right)\right) . \frac{1-\sigma(X_p^T w)}{1-\sigma(X_p^T w)}$

= $\frac{1}{P} . \sum_{p=1}^{p} \left(X_p^T \left(y_p - \sigma\left(X_p^T w\right)\right)\right)$

Question 4.

```
Question 4
Multiclass
Learning rate is 0.1
Train_errors are  1.0
Test_errors are  3.0
Testing accuracy = 0.9333333333333333
Training accuracy = 0.9904761904761905

Most Highly Correlated
        FirstVariable     SecondVariable  Correlation
0   petal length (cm)   petal width (cm)     0.962865
1   sepal length (cm)  petal length (cm)     0.871754
2   sepal length (cm)   petal width (cm)     0.817941
3    sepal width (cm)  petal length (cm)    -0.428440
4    sepal width (cm)   petal width (cm)    -0.366126
5   sepal length (cm)   sepal width (cm)    -0.117570
6   sepal length (cm)  sepal length (cm)     0.000000
7    sepal width (cm)  sepal length (cm)    -0.000000
8    sepal width (cm)   sepal width (cm)     0.000000
9   petal length (cm)  sepal length (cm)     0.000000
10  petal length (cm)   sepal width (cm)    -0.000000
11  petal length (cm)  petal length (cm)     0.000000
12   petal width (cm)  sepal length (cm)     0.000000
13   petal width (cm)   sepal width (cm)    -0.000000
14   petal width (cm)  petal length (cm)     0.000000
15   petal width (cm)   petal width (cm)     0.000000

    sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2

 Question 5
Canned algorithm Logistic Regression
Training Accuracy is  0.9809523809523809
Testing Accuracy is  0.9777777777777777
```

5.


```
 Question 5
Canned algorithm Logistic Regression
Training Accuracy is  0.9809523809523809
Testing Accuracy is  0.9777777777777777
```


Canned Algorithm is better than the normal one with more accuracy , can be varied by changing the hyper parameters.

In [1]:
```
from sklearn import datasets
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from matplotlib import pyplot as plt

iris = datasets.load_iris()
X = iris.data[:,0:4]
Y = iris.target

# Compact Notation
m = np.size(Y)
Ones = np.ones(m,dtype=int)
X_compact = np.column_stack((Ones,X))

X_train,X_test,Y_train,Y_test = train_test_split(X_compact,Y,test_size=0.3,random_state=0)

# One hot Encoding
targets = np.array(Y_train).reshape(-1)
Y_train_one= np.eye(3)[targets]

targets = np.array(Y_test).reshape(-1)
Y_test_one= np.eye(3)[targets]

converged=False
W = np.ones((5,3))
alpha=0.1
iterations=0
Training_error=[]
Testing_error=[]

def sigmoid(X,W):
    return 1/(1 + np.exp(-(np.dot(X.T,W))))

while not converged:
    U = - 1/m * np.dot(X_train.T,(Y_train_one - sigmoid(X_train.T,W)))
    W_new = W - alpha * U
    epsilon=np.abs(W_new-W)
    W = np.copy(W_new)

    converged=(np.abs(epsilon) < 0.0001).all()
    if iterations>4000:
        converged = True

    Y_train_pred=np.dot(X_train,W)
    Y_train_pred1=np.argmax(Y_train_pred,axis=1)
    Train_error = np.sum((Y_train != Y_train_pred1)/np.size(Y_train))

    Y_test_pred=np.dot(X_test,W)
    Y_test_pred1=np.argmax(Y_test_pred,axis=1)
    Test_error = np.sum((Y_test != Y_test_pred1)/np.size(Y_test))

    Training_error.append(Train_error)
    Testing_error.append(Test_error)

    iterations+=1
print('Question 4')
print('Multiclass')
print('Learning rate is 0.1')
print('Train_errors are ', Train_error*np.size(Y_train))
print('Test_errors are ', Test_error*np.size(Y_test))
print("Testing accuracy =",(np.size(Y_test) - Test_error*np.size(Y_test))/np.size(Y_test))
print("Training accuracy =",(np.size(Y_train) - Train_error*np.size(Y_train))/np.size(Y_train))

def mosthighlycorrelated(mydataframe, numtoreport):
    cormatrix = mydataframe.corr()
    cormatrix *= np.tri(*cormatrix.values.shape, k=-1).T
    cormatrix = cormatrix.stack()
    cormatrix = cormatrix.reindex(cormatrix.abs().sort_values(ascending=False).index).reset_index()
    cormatrix.columns = ["FirstVariable", "SecondVariable", "Correlation"]
    return cormatrix.head(numtoreport)

irisp = pd.DataFrame(iris.data,columns=iris.feature_names)

print("\nMost Highly Correlated")
print(mosthighlycorrelated(irisp,50))
print('\n',irisp.head())

print('\n Question 5')
from sklearn.linear_model import LogisticRegression
clf = LogisticRegression(random_state=0,solver='lbfgs', max_iter=2000).fit(X_train,Y_train)
print("Canned algorithm Logistic Regression")
print("Training Accuracy is ",clf.score(X_train, Y_train))
print("Testing Accuracy is ",clf.score(X_test, Y_test))
```

```
Question 4
Multiclass
Learning rate is 0.1
Train_errors are  1.0
Test_errors are  3.0
Testing accuracy = 0.9333333333333333
Training accuracy = 0.9904761904761905

Most Highly Correlated
        FirstVariable     SecondVariable  Correlation
0   petal length (cm)   petal width (cm)     0.962865
1   sepal length (cm)  petal length (cm)     0.871754
2   sepal length (cm)   petal width (cm)     0.817941
3    sepal width (cm)  petal length (cm)    -0.428440
4    sepal width (cm)   petal width (cm)    -0.366126
5   sepal length (cm)   sepal width (cm)    -0.117570
6   sepal length (cm)  sepal length (cm)     0.000000
7    sepal width (cm)  sepal length (cm)    -0.000000
8    sepal width (cm)   sepal width (cm)     0.000000
9   petal length (cm)  sepal length (cm)     0.000000
10  petal length (cm)   sepal width (cm)    -0.000000
11  petal length (cm)  petal length (cm)     0.000000
12   petal width (cm)  sepal length (cm)     0.000000
13   petal width (cm)   sepal width (cm)    -0.000000
14   petal width (cm)  petal length (cm)     0.000000
15   petal width (cm)   petal width (cm)     0.000000

    sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2

 Question 5
Canned algorithm Logistic Regression
Training Accuracy is  0.9809523809523809
Testing Accuracy is  0.9777777777777777
```

In [ ]: