EEE 591 Machine Learning with deployment to FPGA

HOMEWORK 7

ASU ID: 1225713099
Praveen Paidi

1.

```
Question 1
total training time 0.0
Number in train  1400
Misclassified samples: 360
Accuracy: 0.74
Number in test  600
Misclassified samples: 166
Accuracy: 0.72
```

2.

```
Question 2
total training time 0.0804741382598877
Number in train  1400
Misclassified samples: 349
Accuracy: 0.75
Number in test  600
Misclassified samples: 164
Accuracy: 0.73
```

3.

```
Question 3
[LightGBM] [Info] Total Bins 769
[LightGBM] [Info] Number of data points in the train set: 1400, number of used features: 4
[20]    training's l2: 0.146211 valid_1's l2: 0.186288
[40]    training's l2: 0.129856 valid_1's l2: 0.185633
[60]    training's l2: 0.120544 valid_1's l2: 0.188933
[80]    training's l2: 0.114389 valid_1's l2: 0.192395
[100]   training's l2: 0.108576 valid_1's l2: 0.195231
Number in train  1400

total training time 0.1600658893585205
Misclassified samples: 205
Accuracy: 0.85
Number in test  600
Misclassified samples: 173
Accuracy: 0.71
```

4.

```
Question 4
total training time 1.5606062412261963

 Number of samples in training: 1400
Accuracy: 0.75
Misclassified samples: 345

 Number of samples in test: 600
Accuracy: 0.73
Misclassified samples: 164
```

The time taken by each and every thing is different and can be changed by the GPU and hyper parameters chosen as well. The Pytorch normally took more time than the linreg, SVM for equal computation technology.

In [1]:
```
import numpy as np
import random
## import torch
## from torch.autograd import Variable
## import torch.nn as nn
from scipy.special import expit, logit
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
##import matplotlib.pyplot as plt

n_samples = 2000
random_state = np.random.RandomState(13)
x1 = random_state.uniform(size=n_samples)
x2 = random_state.uniform(size=n_samples)
x3 = random_state.randint(0, 4, size=n_samples)
x4 = random_state.uniform(size=n_samples) ## note that this is noise
X = np.c_[x1,x2,x3,x4]

p = expit(np.sin(3 * x1) - 4 * x2 + x3)
y = random_state.binomial(1, p, size=n_samples)

X_tr, X_te, Y_tr,Y_te = train_test_split(X,y,test_size=0.3,random_state=0)


import time
from sklearn.linear_model import LogisticRegression as lr
tbeg = time.time()
linreg = lr(solver = 'liblinear').fit(X_tr,Y_tr)

tend = time.time()
print('Question 1')
print('total training time', tend-tbeg)

print('Number in train ',len(Y_tr))
y_pred = linreg.predict(X_tr)
mc_train = (Y_tr != y_pred).sum()
print('Misclassified samples: %d' % mc_train)
acc_train = accuracy_score(Y_tr, y_pred)
print('Accuracy: %.2f' % acc_train)
#
print('Number in test ',len(Y_te))
y_pred = linreg.predict(X_te)
mc_test = (Y_te != y_pred).sum()
print('Misclassified samples: %d' % mc_test)
acc_test = accuracy_score(Y_te, y_pred)
print('Accuracy: %.2f' % acc_test)


from sklearn.svm import SVC

tbeg = time.time()
SVM_lin = SVC(kernel='linear').fit(X_tr,Y_tr)
tend = time.time()

print('\nQuestion 2')
print('total training time', tend-tbeg)
print('Number in train ',len(Y_tr))
y_pred = SVM_lin.predict(X_tr)
mc_train = (Y_tr != y_pred).sum()
print('Misclassified samples: %d' % mc_train)
acc_train = accuracy_score(Y_tr, y_pred)
print('Accuracy: %.2f' % acc_train)
###
print('Number in test ',len(Y_te))
y_pred = SVM_lin.predict(X_te)
mc_test = (Y_te != y_pred).sum()
print('Misclassified samples: %d' % mc_test)
acc_test = accuracy_score(Y_te, y_pred)
print('Accuracy: %.2f' % acc_test)

print('\nQuestion 3')
import lightgbm as lgb
import warnings
warnings.filterwarnings("ignore")
tbeg = time.time()
lgb_train = lgb.Dataset(X_tr, Y_tr,params={'verbose':-1},free_raw_data=False)
lgb_test = lgb.Dataset(X_te, Y_te,params={'verbose':-1},free_raw_data=False)
## defaults num_leaves = 31,
params = {'force_col_wise': True, 'boosting_type': 'gbdt', 'num_iterations': 100,
'n_estimators': 100,
 'max_depth': 5, 'num_leaves': 100, 'feature_fraction': 0.75,
 'bagging_fraction': 0.75, 'bagging_freq': 1, 'lambda': 0.5, 'random_state': 3}
model = lgb.train(params, lgb_train, valid_sets=[lgb_train, lgb_test], verbose_eval=20)
print('Number in train ',len(Y_tr))
y_train_pred = model.predict(X_tr)
y_pred = np.where(y_train_pred<0.5,0,1)
tend = time.time()

print('\ntotal training time', tend-tbeg)
# print('Number in train ',len(y_train))
# y_pred = SVM.predict(X_train)
mc_train = (Y_tr != y_pred).sum()
print('Misclassified samples: %d' % mc_train)
acc_train = accuracy_score(Y_tr, y_pred)
```

```
print('Accuracy: %.2f' % acc_train)
###
print('Number in test ',len(Y_te))
y_test_pred = model.predict(X_te)
y_pred = np.where(y_test_pred<0.5,0,1)
mc_test = (Y_te != y_pred).sum()
print('Misclassified samples: %d' % mc_test)
acc_test = accuracy_score(Y_te, y_pred)
print('Accuracy: %.2f' % acc_test)

print('\nQuestion 4')
def onehtar(y):
    for i in range(len(y[:,0])):
        maxval = torch.max(y[i,:])
        for j in range(len(y[0,:])):
            if (y[i,j] == maxval):
                y[i,j] = 1.0
            else:
                y[i,j] = 0.0
    return(y)

import torch
from torch.autograd import Variable
import torch.nn as nn
from sklearn.preprocessing import OneHotEncoder

XX_train = torch.from_numpy(X_tr).type(torch.FloatTensor)
targets0 = np.eye(2)[Y_tr.astype(int)] ## one hot code target
yy_pred = torch.from_numpy(np.eye(2)[Y_tr.astype(int)]).type(torch.FloatTensor)

H = 100
```

```
model = torch.nn.Sequential(
    torch.nn.Linear(4, H),
    torch.nn.ReLU(),
    torch.nn.Linear(H, 2),
)

loss_fn = torch.nn.MSELoss(reduction='sum')
learning_rate = 0.0001
last = 1.0
error = 0
t = 0
converge=False
tbeg = time.time()

while not converge:
    y_pred = model(XX_train)
    loss = loss_fn(y_pred, yy_pred)
    model.zero_grad()

    loss.backward()
    error = abs(last-loss.item())
    last = loss.item()


    with torch.no_grad():
        for param in model.parameters():
            param -= learning_rate * param.grad

    t += 1
    if t>1000 or error<0.00001:
        converge=True

tend = time.time()
```

```
y_pred=torch.tensor(onehtar(y_pred))
XX_test = torch.from_numpy(X_te).type(torch.FloatTensor)
yy_pred_test = torch.from_numpy(np.eye(2)[Y_te.astype(int)]).type(torch.FloatTensor)


print("total training time", tend-tbeg)

print("\n Number of samples in training:",len(y_pred))
acc_train = accuracy_score(yy_pred, y_pred)
print('Accuracy: %.2f' % acc_train)
mc_train = (yy_pred != y_pred).sum()/2
print('Misclassified samples: %d' % mc_train)


y_pred_test=torch.tensor(onehtar(model(XX_test)))
acc_train = accuracy_score(yy_pred_test, y_pred_test)
print("\n Number of samples in test:",len(y_pred_test))
print('Accuracy: %.2f' % acc_train)
mc_train = (yy_pred_test != y_pred_test).sum()/2
print('Misclassified samples: %d' % mc_train)
```

```
Question 1
total training time 0.0
Number in train  1400
Misclassified samples: 360
Accuracy: 0.74
Number in test  600
Misclassified samples: 166
Accuracy: 0.72

Question 2
total training time 0.0804741382598877
Number in train  1400
Misclassified samples: 349
Accuracy: 0.75
Number in test  600
Misclassified samples: 164
Accuracy: 0.73

Question 3
[LightGBM] [Info] Total Bins 769
[LightGBM] [Info] Number of data points in the train set: 1400, number of used features: 4
[20]    training's l2: 0.146211 valid_1's l2: 0.186288
[40]    training's l2: 0.129856 valid_1's l2: 0.185633
[60]    training's l2: 0.120544 valid_1's l2: 0.188933
[80]    training's l2: 0.114389 valid_1's l2: 0.192395
[100]   training's l2: 0.108576 valid_1's l2: 0.195231
Number in train  1400

total training time 0.1600658893585205
Misclassified samples: 205
Accuracy: 0.85
Number in test  600
Misclassified samples: 173
Accuracy: 0.71

Question 4
total training time 1.5606062412261963

 Number of samples in training: 1400
Accuracy: 0.75
Misclassified samples: 345

 Number of samples in test: 600
Accuracy: 0.73
Misclassified samples: 164
```

In [ ]: