EEE 498 ML with Deployment to FPGAs

All homework is handed in online. Don't hand in pictures of the code files, hand in the actual code files so I can see if they work. Hand in images of outputs of your code (displays, plots …) converted to pdf. If there are questions to be answered answer them in PowerPoint, Word, or on paper and convert to pdf (It doesn't actually matter how you get to the pdf).

Organize your work so that the answer to each problem can be easily identified. You will not get credit if your submission is poorly organized, or if the grader cannot find the answers.

You can use any language python, c, C++, MATLAB,…, to do coding though Python is likely the easiest and easiest to translate from lectures.

Be careful copying code from office products to Spyder or other editors, some characters are changed like single quotes. Sometimes invisible characters occur causing you to get an error on a line until you completely retype it.

Homework 7

In this homework we will explore the Pytorch method of making a 2 layer fully connected Neural Network and LightGBM and compare to other methods. You may use Research Computing's Agave cluster to do this assignment. See the reference in Canvas to find commands you need.

You may need to import these libraries

```
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
```

We want an arbitrarily large dataset for this problem so let's make the data using random numbers for the features and then a nonlinear calculation method for the target.

```
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
these are the features, we can put them together in dataset X using
X = np.c_[x1,x2,x3,x4]
I will actually use X = np.c_[x1,x2,x3,x4*10] to make it more challenging because of more noise
##
## random.RandomState.binomial(n, p, size=None)
##  n number of trials
##  N number of successes
## https://numpy.org/devdocs/reference/random/generated/numpy.random.RandomState.binomial.html
## P(N) = (n N)p^N (1-p)^(n-N)
##
p = expit(np.sin(3 * x1) - 4 * x2 + x3)
y = random_state.binomial(1, p, size=n_samples)
```

This is an ensemble method generally using decision trees as a weak learner. The advantage of this method is use of bagging and boosting.

We will work on this problem …

Standardize the data in the usual way.

Train and test using 30% of the dataset for testing. Report training and testing accuracy, and miss-classifications for each method used. The training dataset is X_train, the test is X_test.

Use this method to determine the time required by each method below.

```
import time
tbeg = time()
training steps
tend = time()
print("total training time ", tend-tbeg)
```

For these methods adjust parameters to get a good result in the test accuracy and missclassifications. Report training and testing accuracy, miss-classifications for each method used, and the time required to train the model.

For a computing resource use the Agave (Research Computing) system. See W9_Lecture_20_498ML_RC_F.pdf available in Canvas Modules Recources.

1) Classify using Logistic Regression using the Sklearn library.
   This is the way I analyzed the results, and you should do something similar for 2),3) and 4)

```
print('Number in train ',len(y_train))
y_pred = lr.predict(X_train)
mc_train = (y_train != y_pred).sum()
print('Misclassified samples: %d' % mc_train)
acc_train = accuracy_score(y_train, y_pred)
print('Accuracy: %.2f' % acc_train)
#
print('Number in test ',len(y_test))
y_pred = lr.predict(X_test)
mc_test = (y_test != y_pred).sum()
print('Misclassified samples: %d' % mc_test)
acc_test = accuracy_score(y_test, y_pred)
print('Accuracy: %.2f' % acc_test)
```

2) Classify using SVM linear kernel, done to be comparable as well.
3) LGBM using CPU
   LGBM GPU we won't do this one because RC doesn't have it working correctly
   This is the way I did it ---

```
import lightgbm as lgb
import warnings
warnings.filterwarnings("ignore")
tbeg = time.time()
lgb_train = lgb.Dataset(X_train, y_train)
lgb_test = lgb.Dataset(X_test, y_test)
## 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_train))

print('Number in train ',len(y_train))
y_train_pred = model.predict(X_train)
y_pred = np.where(y_train_pred<0.5,0,1)
tend = time.time()
```

4) Two layer Neural network using Pytorch (sequential)
   You will need to import these libraries for pytorch

```
import torch
from torch.autograd import Variable
import torch.nn as nn
```

your data will have to be in tensor form for Pytorch this is the way I did it taking advantage of numpy datasets which were already split and standardized

```
XX_train = torch.from_numpy(X_train)
## targets = y_train.astype(int)    ## cast to int
targets0 = np.eye(2)[y_train.astype(int)]   ## one hot code target
yy_train = torch.from_numpy(np.eye(2)[y_train.astype(int)]).type(torch.FloatTensor)
```

Take a look at code Pytorch_NN_4b.py in week 9

Turn in all your code, screen captures of the results of each, and a table showing training accuracy, test accuracy, training missclassifications, test missclassifications, and training time for each.

My results and times will be in the solution.

To run on the RC site, log on to agave by

ssh userid@agave.asu.edu

password

then use these commands

interactive -p gpu -q wildfire --gres=gpu:1 -t 00-00:10 ## this is for 10 minutes which is enough for a run, but just stay under 6 hours

module load anaconda/py3   ## loads anaconda with python 3

source activate lightgbm-3.1.1 ## this will allow you to everything including light gbm but not pytorch

source activate pytorch-gpu ## this will allow you to do pytorch, so it will have to be done separately

then you can run spyder or whatever IDE normally

Let me know of any other information you need and have fun.

Hand in all output as a screen grab for each problem converted to pdf. It is a good practice to paste well organized and labeled images into PowerPoint, answer all questions in the PowerPoint file, and then convert the PowerPoint file to pdf. Hand in all your code files. If code files are missing you will not get credit. It is typically easier to grade individual files rather than a zipped file, but either is accepted.