CS 577 s21 – Assignment 1

Due by 2/16/2021

- In this assignment you need to implement several Neural Networks on GPU using a singularity container to solve classification and regression problems. This assignment will use the Xsede GPU cluster. The implementation should generally be done in Keras but you may use TensorFlow or PyTorch If you are already familiar with them and would like to use them. Note that we may not be able to offer help with frameworks other than Keras.
- Start by creating an account on Xsede[1]. When creating the account make sure to use your IIT email address. After creating the account fill the following form[2] to provide your Xsede user name so we can allocate GPU time for you.
- Make sure you are logged in your IIT gmail account before attempting to access the form above. If you need to ask for permission to access the form it means you are not logged in your IIT gmail account. In such a case log out all other gmail accounts then login to your IIT email account and try again.
- You must run the programs you write using GPU resources such as your own GPU card, Google Colab notebook, or on a singularity container in the Xsede cluster.
- Follow the submission instructions of the first assignment.

1 Questions

1. Explain the difference between training, validation, and testing data sets. Explain the need for such datasets.
2. Explain the 4 steps used in writing a network program using Keras (data, model, learning process, fitting).
3. Explain the basic parameters used to define a dense layer (number of units, and activation).
4. Explain the network configuration aspects when compiling the model (optimizer, loss, metrics). Explain the difference between loss and metric.

[1] https://portal.xsede.org/
[2] https://goo.gl/forms/1uXzUGw4oZBNJ31H3

5. Explain the 5 basic arguments provided to fit (input, output, batch size, epochs, validation data).
6. Explain the steps used to convert a variable length text string into a binary feature vector.
7. Explain possible conclusions when observing training and validation loss graphs over epochs (underfitting and overfitting).
8. Explain possible hyper-parameters that can be tuned (layers, units per layer, activation functions, loss).
9. Explain how a vector of predictions from a binary classifier with a logistic function in the output layer can be converted to class decisions.
10. Explain how one-hot-encoding is used to encode class labels of a multi-class classification problem.
11. Explain the meaning of the output layer when using softmax as an activation function in it.
12. Explain the difference between sparse-categorical-crossentropy and categorical-crossentropy.
13. Assuming a dataset with 5 classes where each class is represented equally, what will be the accuracy of a random classifier?
14. Explain how to normalize feature vectors to have equal mean and standard deviation. Explain the purpose of such normalization.
15. Explain the difference between the MSE and MAE metrics. Which is easier to interpret?
16. Explain how to perform k-fold cross validation. When is k-fold cross-validation needed?
17. Explain when performing K-fold cross-validation how to report the validation error and how to train the final model.

2 Programming

1. Go to the tensorflow playground webpage[3] and train a classifier for each of the 4 datasets there. Your goal is to classify all examples correctly while attempting to have a simple classifier. Set the hyper parameters (e.g. learning rate, activation, regularization, regularization rate) and select the features to use. Press the play button to start training. For each dataset report the following: hyper parameters selected, number of training epochs, network architecture, training loss, and test loss.
2. Load the CIFAR10 dataset and select a subset of three classes. Split the training set into a training and validation subsets. Vectorize the images and encode the known class labels using categorical encoding. Design a fully connected neural network to perform multi-class classification of this data. Justify your network design decisions (number of hidden layers, number of units per layer, loss function, and evaluation metric). Build and compile the network you designed. Plot training and validation loss as a function of epochs, then plot training and validation accuracy. Tune model

[3] http://playground.tensorflow.org/

hyper parameters to improve performance. Retrain the final model and test performance on the test collection. Report the performance you obtain. Make sure that your program can save and load the weights of the trained network.

3. Load the spam email data from the UCI repository "spambase"[4]. Prepare the data you loaded as a tensor suitable for a neural network. Normalize features as needed. Explain the steps you perform in preparing the data and justify them. Write a function "load_spam_data" to load the data and split it into a training and testing subsets. Repeat the steps you performed in the CIFAR10 question.

4. Load the crime data from the UCI repository "Communities and crime"[5]. Prepare the data you load as a tensor suitable for a neural network. Normalize features as needed. Explain the steps you perform in preparing the data and justify them. Write a function "load_crime_data" to load the data and split it into a training and testing subsets. Repeat the steps you performed in the CI?FAR 10 question except that in this case you are required to perform k-fold cross validation. To report the cross validation results, average the validation error of all the folds and plot it as a function of epochs. Retrain the final model on all training data (i.e. all folds) and test final performance on test set.

Submission Instructions

1. Create a folder AS1 in your bitbucket repository and create inside it the following sub-folders: src, doc, and data. Organize the submission materials inside the sub-folders as follows:

   - doc: Report prepared as a PDF file. The report should contain answers to questions, a summary of program design issues, description of specific problems you faced and the way in which you solved them, and sample input/output results (text/graphic). The report needs to be sufficiently detailed. It is very important that you evaluate the algorithms you implemented for correctness and for performance. Try using different parameters and observe (and report) the effect of the parameters.
   - src: All program files (stand alone or as a python notebook).
   - data:
     - A file "data.txt" containing links to where data files can be downloaded from. If the data is publicly available provide public links or if the data is not publicly available provide a link to a google drive or some other cloud storage.
     - A file "models.txt" containing links to saved model files in google drive or some other cloud storage. Your program must be able to load and run using the saved models you provide.
     - Do not upload any data or model files directly to bitbucket.

2. Note that we must be able to view your report and execute your program in order to grade it.

3. For programming questions use Python with Keras (TensorFlow or Pytorch are also allowed).

[4] https://archive.ics.uci.edu/ml/datasets/spambase
[5] https://archive.ics.uci.edu/ml/datasets/Communities+and+Crime

4. On or before the due date upload your submission to your bitbucket repository. If you are late, upload the submission when you are ready. To compute "late days will" we will use the last update date of your repository. Do not make any changes to the folder after submitting it so that it does not cause a change to your submission date.

5. Do not submit a paper copy of your report. You will be contacted by email if some material is missing or if you will need to meet with the TA.