ECS 189G-001

Deep Learning

Winter 2023

Course Project

Project Overview

There is a quarter-long course project of building deep learning models for various application tasks. The main purpose of this project is to help students get familiar with classic deep learning models and use them to solve concrete real-world problems. The project will account for 40% of your final grade, and it will have multiple stages:

- **Stage 1**: Group formation and programming environment setup. (0%)
- **Stage 2**: First trial with PyTorch and build up a Multi-Layer Perceptron model to classify the instances in a provided dataset. (10%)
- **Stage 3**: Object recognition from images with convolutional neural network model. (10%)
- **Stage 4**: Text classification and generation with recurrent neural network models. (10%)
- **Stage 5**: Network embedding and node classification with graph neural network models. (10%)

Stage 1: Group formation and Environment Setup

(Code Template, Report Template)

You need to find teammates to form a group with four students (more than four is not allowed). Create a cool **name for your group.** Submit a report to Canvas that includes your **team name**, **student names**, **student IDs**, and **emails**. Only one submission will be needed for each team, and the TA will help tie the group members up throughout the semester. You can either elect a team leader to do the submission, or you can take turns to take the responsibility for submissions for each stage. If one happens to leave/drop the class, the others should be on your own. So choose your groupmates carefully!

Note 1: If you prefer to work independently, then the solo mode should also be ok, but your solo project will be graded with the same criteria as others' team project.

Note 2: Choose your team members carefully. (Some teams may end up with some unhappy issues or conflicts…)

Note 3: If you have difficulties forming a team, here are the solutions: 1) actively contact your peers in the same class; (2) we will do random team assignments if you really cannot find a team at the end (we cannot guarantee who your teammates will be :( ).

Environment setup:

1-1: Make sure your computer has the latest python 3.x installed. If you don't have python, consider to download it at https://www.python.org/downloads/

1-2: Download and install the PyCharm: https://www.jetbrains.com/pycharm/ and verify it can work correctly by creating a toy "Hello world" project in Python 3.x. (Please use the educational license, and don't pay money to buy anything.)

1-3: You have two ways to install sklearn. (1) install "scikit-learn" into your Pycharm project as shown by https://www.youtube.com/watch?v=f4OvzM7x5jo&ab_channel=AamirAlamgir, or (2) Download and install sklearn toolkit into your computer by following the instruction provided at https://scikit-learn.org/stable/install.html

1-4: You have two ways to install pytorch. (1) install "torch" into your Pycharm project similar to the above, or (2) Download and install the pytorch toolkit into your computer by following the instructions provided at https://pytorch.org/get-started/locally/

1-5: Verify PyTorch and sklearn both can work correctly in your PyCharm by creating toy projects and call these two toolkits in your code.

1-6: Download the provided code template. Setup the interpreter and other packages, change your current working directory to the stage_1_script folder, try to run the four scripts for stage 1. If you have both PyTorch and sklearn installed and environment setup correctly, you should be able to run the code and see the generated results.

1-7: Check through the code template carefully, and figure out how the code is organized, all your future project code will be based on this template in this course.

1-8: (Optional) For the MLP method, change the loss function and optimizer, as well as their setups (e.g., learning rates or other hyper-parameters) to see their impacts on the model performance.

Stage 2: Data classification with MLP model based on PyTorch

(Dataset, Report Template)

This stage aims to help students get familiar with PyTorch, one of the most popular toolkit used in deep learning, and scikit-learn (sklearn), one of the most popular toolkit used in classic machine learning. Please write your first program with PyTorch to implement the MLP model introduced in class.

2-1: Download the dataset provided by the instructor, which covers both training set and testing set. Take a look at the dataset before writing your code.

2-2: Write your own code based on the template for stage 2 (First, copy the code from stage 1 folder into stage 2, and also change the import commands accordingly. You can reuse/make changes to the code provided in Stage 1, e.g., write a new Dataset_Loader for your new dataset, change the training/testing set partition code in the Setting (no train/test set split or cross validation will be needed for stage 2), change the Method_MLP architecture to get adapted to the new input/output, and include more evaluation metrics … , define more evaluation metrics, e.g., F1, Recall, Precision, we are doing multiclass classification here, so the binary F1/Recall/Prec cannot be used, you can use the "weighted"/"macro"/"micro" version of the metrics instead).

2-3: Train the MLP model with the training data, and apply the learned model to the testing set. Generate the learning curve plots, and report the final evaluation result.

2-4: Change the model architecture with more layers, with different loss functions, different optimizers and other settings to increase the performance score.

2-5: Write a report about your experimental process and results based on the provided report template (5 pages at the maximum, longer reports will be "penalized" :( ).

2-6: (Optional): If your computer GPU supports CUDA programming and you can also try to run your model with your GPU with examples like https://wandb.ai/wandb/common-ml-errors/reports/How-To-Use-GPU-with-PyTorch---VmlldzozMzAxMDk

Stage 3: Image classification for object recognition with CNN model (Dataset, Report Template)

This stage aims to help students get familiar with the convolutional neural network (CNN) model, and try to use the CNN model to classify image data for object recognition.

3-1: Download the three image datasets provided by the instructor, one about hand-written digit numbers, one about human faces and one about colored objects.

(For the human face dataset, it is converted to gray-scale image from colored images by assigning R/G/B with equal numbers. So its data also has three channels (R, G, B) but with equal pixel values for the R/G/B in these three channel matrices. You don't need to use three channel matrices, use one channel should be sufficient.)

3-2: Write your own CNN model in the provided code template.

3-3: Train a CNN model with the hand-written digit images, and apply it to recognize the digit images in the testing set. Generate the learning curves and report the evaluation results.

3-4: Train two other CNN models with the face images and colored objects, and apply them to recognize the faces and color objects in the testing set, respectively. Generate the learning curves and report the results.

3-5: Try to change the configurations of the provided CNN model (with different model depth, kernel size, padding, stride, pooling layer, hidden layer dimension, different loss function, etc.), and report the results to see the impacts of configuration on the model performance.

3-6: Write a report about your experimental process and results based on the provided report template (5 pages at the maximum).

3-7: (Optional): If you have GPUs supporting CUDA programming, you can also try to run the CNN model with your GPU instead.

Stage 4: Text classification and generation with RNN model

(Dataset, Report Template)

This stage aims to help students get familiar with the recurrent neural network (RNN) model, and try to use the model to classify and generate text data.

4-1: Download the two text datasets provided by the instructor, and take a look at the text content.

4-2: Write your own RNN models in the provided code template on text classification.

4-3: Train a RNN model with the text classification dataset, and apply it to classify the testing set. Generate the learning curves and report the evaluation results.

4-4: Train a RNN model with the text generation dataset, and use the trained model to generate a story starting with three starting words. Compare the generation result with the training data to evaluate its correctness, and report the results.

4-5: Try to change the RNN with LSTM and GRU units, respectively. You can also change other architecture settings that you think will improve the learning results. Re-do steps 4-2 to 4-4, and report the results.

4-6: Write a report about your experimental process and results based on the provided report template (5 pages at the maximum).

4-7: (Optional): If you have GPUs supporting CUDA programming, you can also try to run the RNN models with your GPU instead.

Stage 5: Graph Embedding and Node Classification with GNN model

(Dataset, Dataset_Loader, Report Template)

This stage aims to help students get familiar with the graph neural network (GNN) model, and try to use the model to embed graphs and classify nodes.

5-1: Download the three graph datasets provided by the instructor, and take a look at the graph data content first.

5-2: Write your own GCN model in the provided code template on graph embedding and node classification.

5-3: Train a GCN model with the Cora dataset, and apply it to classify the testing set. Generate the learning curves and report the evaluation results.

5-4: Re-do step 5-3 on Pubmed and Citeseer datasets as well, and report the results.

5-5: Write a report about your experimental process and results based on the provided report template (5 pages at the maximum).

5-6: (Optional): If you have GPUs supporting CUDA programming, you can also try to run the GCN models with your GPU instead.