CS410 Artificial Intelligence 2020 Fall
Project 1: Drug Molecular Toxicity Prediction
Due date: 23:59:59 (GMT +08:00), December 27 2020

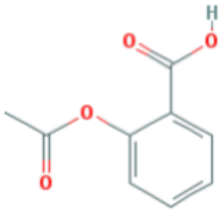
1. Project Specification

Almost all people are exposed to different chemicals during their lifetimes through different sources including food, household cleaning products and medicines. However, in some cases, these chemicals can be toxic and affect human health. As a matter of fact, over 30% of drugs have failed in human clinical trials because they are determined to be toxic despite promising pre-clinical studies in animal models. Consider real-world clinical trials for assessing drugs are extremely time-consuming, it is ideal if a computational drug molecular toxicity assessment method can be developed to quickly test whether certain chemicals have the potential to disrupt processes in the human body of great concern to human health.

Deep neural network has become a hot research topic in machine learning in recent years. Compared to other methods, deep learning has shown its advantages in handling large amount of data and achieving better performance. In this individual project, you will be given a dataset of drug molecules with their SMILES expressions (which will be explained later) and the binary labels indicating whether one drug molecule is toxic or not. You are going to develop a Deep Neural Network which can learn useful patterns from the data provided and predict the toxicity of a new list of molecules based on learned knowledge using TensorFlow or PyTorch package.

2. SMILES Expression

Simplified Molecular-Input Line-Entry System (SMILES) is a linear representation for molecular structure using 1D ASCII strings. For example, aspirin, a commonly used drug in daily life, its 2D structure is



and its SMILES is

CC(=O)OC1=CC=CC=C1C(=O)O

The one hot format of SMILES is a 2D {0,1} matrix, where each column represents a symbol in the SMILES notation of the current molecule, and each row is one ASCII character appeared in the dataset's SMILES dictionary. The size of the 2D matrices is the size of the dataset's SMILES dictionary * the length of the longest molecule SMILES, which means we have zeros padded after short molecule SMILES. For a SMILES notation, one at row i, col j means the j^{th} symbol of that SMILES is the i^{th} character in the dictionary. The one-hot example for aspirin is:

	C	C	(	=	O	)	O	C	1	=	C	C	=	C	C	=	C	1	C	(	=	O	)	O
C	■	■						■			■	■		■	■		■		■					
(			■																	■				
=				■						■			■			■					■			
O					■		■															■		■
)						■																	■	
1									■									■						

3. Dataset

The dataset provided is about the toxicity of some small molecules. We provide two folders for you, one is the training data under the folder "train" (8169 samples), and the other one is validation data under the folder "validation" (272 samples). There are three files in each folder:

File	Type	Description
names_smiles.txt	String	A txt file, each line contains a drug molecule's name and its SMILES expression, separated by comma (,)
names_labels.txt	String	A txt file, each line contains a drug molecule's name and its toxicity label, where 0 means non-toxic and 1 means toxic, separated by comma (,)
names_onehots.npy	Numeric	A npy file which can be loaded by numpy package, storing two ndarray; one is the names of the molecules, and the other is the one-hot representations of SMILES expressions of drug molecules

The source data are *names_smiles.txt* and *names_labels.txt* files. Your neural network is supposed to learn from (SMILES, label) records, and be able to predict label from SMILES in the end. The *names_onehots.npy* file is derived from *names_smiles.txt*, storing one-hot representations of SMILES expressions of drug molecules. Providing *names_onehots.npy* is to ease the burden of you on data preprocessing and focus on neural network construction. You are not constrained on how to use these data files as long as they are the only training data and validation data. You can build Convolutional Neural Networks or other kinds of neural networks as you like.

The molecules in "../train" and "../validation" do not overlap with each other. The data we use to mark your model is in a folder named "test" (610 samples), and you have no access to its labels, but the samples in *../test/names_smiles.txt* and *../test/names_onehots.npy* are for you to generate your own predictions of probability which should be stored in *../output_student_id.txt*. The format of these two files: *names_smiles.txt* and *names_onehots.npy* are the same with those for training and validation, but the molecules are new.

4. Assignment Requirement

1) This is an individual project.
2) Data
    a) train
    b) validation
    c) test (only samples are accessible)

    You can train your model on the data under the folder "train" and validate its performance on the data under the folder "validation", or you can train your model with both.
3) Features

    You can either use the one hot of SMILES as features for molecules, or directly use the SMILES notation as you wish.
4) Model

    You can use any deep neural network architecture in this assignment to achieve good performance, e.g. convolutional neural network (CNN), recurrent neural network (RNN), long short-term memory (LSTM), graph convolutional network (GCN), etc.
5) Prediction task

    You are going to predict toxicity of the molecules based on their structures. The output of your model is in [0, 1] indicating that the probability that current drugs are toxic
6) Output and Marking

Your model will be tested on the dataset under the folder "test". Your output file should follow the format and order provided in the *../test/output_sample.txt*. The first column should be the drug molecule's name, which is already given, and the second column should be the predictions of probability that current drugs are toxic.
Your submitted folder will be extracted and put alongside the "test" folder, and it means that you must use the relative path "../test/" in your submitted *test.py* file to access the marking data. In your *test.py* file, you need to first restore your model parameters and then test your model on the marking data ("../test/"), and you should output your predictions into a file named *output_student_id.txt* following the format of *output_sample.txt*, which should be in the same directory as *train.py* and *test.py*.

Your final score of this assignment will depend on the AUC（Area Under Roc） among your predictions and the true labels.

7) Deep learning library
You can choose one of the following three versions
   a) Numpy, Pandas and TensorFlow-gpu 1.15.0.
   b) NumPy, Pandas and TensorFlow-cpu 1.15.0.
   c) NumPy, Pandas and PyTorch 1.1.0.
   Please do not use other libraries. Otherwise, you will get zero mark for this assignment.
8) Programming language
   The only supported language for this assignment are Python 3.6 and Python 3.7.

5. Submission Requirement

You are required to submit both on Canvas and on Kaggle inClass competition. Join the Kaggle competition with URL for sharing.

1) Submission list (to Canvas)
   a) Source file for training. Name it as *train.py*
   b) Source file for recovering your network model. Name it as *test.py*.
   c) Your own predictions on the dataset under "../test". Name it as *output_ student_id.txt*, and strictly follow the format provided in *output_sample.txt*. The *output_ student_id.txt* you uploaded to Canvas, uploaded to Kaggle and the running output of the submitted model should be the same, or you violate Honor Code.
   d) TensorFlow and PyTorch generated files in "../weights", which store your trained model. With the trained model, *test.py* could be executed and output the result.
   e) Any other files that help your programs to work, such as preprocessing files, format-converting files.
   f) Report.

   The folder should look like this

```
├──518XXXXXXXXX.zip
│   │ ──output_518XXXXXXXXX.txt
│   │ ──test.py
│   │ ──train.py
│   │ ──Report.pdf
│   │ ──other files
│   └──weights (Take TensorFlow as an example)
│         │ ──checkpoint
│         │ ──model.data-00000-of-00001
│         │ ──model.index
│         │ ──model.meta
```

   Put everything in the submission list above into a folder, name the folder as your student id, zip or rar the folder WITHOUT encryption, also name the zip file as *student_id.zip or student_id.rar*. Submit the zip file or rar file to Canvas.

2) Submission list (to Kaggle)
   a) Your own predictions on the dataset under "../test". Name it as *output_ student_id.txt*, and strictly follow the format provided in *output_sample.txt.* The *output_ student_id.txt* you uploaded to Canvas, uploaded to Kaggle and the running output of the submitted model should be the same, or you violate Honor Code.
3) DO NOT add any of the data in "train" and "validation" folder in your zip file, because we will grade your model on the result of the test folder.

6. Important Points

To make this project fair and meaningful, there are some other points you MUST follows:

1) The time limits to run your *test.py* is 60s
2) The size of the whole zip file should be less than 200 MB
3) Plagiarism will be SERIOUSLY punished (ZERO mark plus reporting to department)
4) The *output_ student_id.txt* you uploaded to Canvas, uploaded to Kaggle and the running output of the submitted model should be the same, or you violate Honor Code.

7. Late Submission

No late submission

8. Bonus

Anyone who can solve the problem by using graph embedding of the smiles can have the ranking of their GCN output score increased by 8.