DMfold Software Guide

DMfold mainly consists of three parts: Data\_Preprocessing, PU, and CU. Data\_Preprocessing is mainly to transform the original clean data into matrix data which can be used in deep learning model. The code of this part stores in Data\_Prepeocessing folder. PU takes the matrix data as input and label. Mainly responsible for predicting RNA sequences as dot-bracket sequences. The code of this part stores in PU\_Part folder. CU is mainly responsible to correct the prediction results of PU and output final prediction results. The code of this part stores in CU\_Part folder. The following is the specific workflow of each part.

1. Data\_Preprocessing：

The folder of All\_Clearn\_Data stores all original clean data (CT).

1. The file of Classification\_Train\_Test.py is used to divide the clean data into TestData and Train\_Validation\_Data. The data of TestData accounts for 10% of the total data, and the data of Train\_Validation\_Data accounts for 90% of the total data. TestData, which is the testing set of the model, is used to test the performance of the model. Train\_Validation\_Data is used to perform 10-fold cross-validation.
2. The file of Validation\_Data\_Classification.py is used to randomly divide Train\_Validation\_Data into ten equal parts and stores them in the folder of Ten\_Fold\_Validation. Those ten parts of data are the Validation data.
3. The file of Create\_TrainSet.py is used to randomly select nine parts of Validation data as the TrainData. Hence, it is can generate ten training sets. Those training sets store in the folder of TrainData.
4. In the final, the file of Create\_Train\_Test\_Matrix.py is used to make training sets and testing set into matrixes and store them as h5 files. Each of those sets (Containing ten training sets and one test set.) generates an h5 file. Those h5 files are stored in the folder of Deep\_Model\_Matrix\_Data.
5. PU:

The matrix data of the training set and testing set are stored in the folder of Deep\_Model\_Matrix\_Data.

1. The file of SecondaryDataRead.py is used to read the data of the training set and testing set.
2. The file of SecondaryModel.py is used to call SecondaryDataRead.py to get the train and test data. After that, using the data to training and testing the deep learning model. Record the model in the folder of Model, when the testing accuracy is greater 85%.
3. The file of DataSave.py is used to call SecondaryDataRead.py to get the train and test (may be a single-family test data) data. After that, using the model which record in the folder of Model to prediction the training set and testing set. In the final, those prediction results are stored in the folder of Saver\_Result in the format of h5 files.
4. CU

The prediction results of PU are stored in the folder of Saver\_Result. The real structure data is stored in the folder Real\_Data. The real structure is used to help to calculate those parameters (PPV, SEN and F-score) of those prediction results.

1. The file of DataReadClass.py is used to read the prediction structure and real structure.
2. The file of MainFunctionModel.py is used to call DataReadClass.py to get the real and prediction structure. After that, obtain the correct prediction structures and calculate the accuracy of those prediction structures. Those accuracy parameters are output on the console.
3. The file of MainFunctionModel.py is used to call DataReadClass.py to get the real and prediction structure. After that, obtain the correct prediction structures and store them in two formats. The first formate is CT, which stored in the folder of CT\_Result. The secondary formate is fasta, which stored in the folder of Fasta\_Result.

Due to the storage space limitations of the GitHub, we only put clean data in the program. Other intermediate results and final results require the reader to run the program to generate. All programs are running in the environment of python3.5 and TensorFlow 1.5. The hardware of our computer is: CPU is i7, memory is 16G and GPU is 1080Ti.