The GeoPPI Documentation

May 1st, 2021

1. An overview of GeoPPI

GeoPPI is a deep learning based framework that uses deep geometric representations of protein complexes to model the effects of mutations on the binding affinity. To achieve both the powerful expressive capacity for geometric structures and the robustness of prediction, GeoPPI sequentially employs two components, namely a geometric encoder (excelling in extracting graphical features) and a gradient-boosting tree (GBT, excelling in avoiding overfitting). The geometric encoder is a graph neural network that performs neural message passing on the neighboring atoms for updating representations of the center atom. It is trained via a novel self-supervised learning scheme to produce deep geometric representations for protein structures. Based on these learned representations of both a complex and its mutant, the GBT learns from the mutation data to predict the corresponding binding affinity change.

Thanks to the above design, GeoPPI enjoys accurate predictive power, strong generalizability, and high inference speed for the estimation of the mutation impact.

2. Environment setup

This section describes how to create the required environment for GeoPPI under Ubuntu 20.04 and Python 3. The environment setup on other operating systems such as Windows is similar. Here, we provide three environment setup strategies (quick setup, step-by-step setup and setup with CUDA support) for users with different demands. Users can choose one of them to build the required environment for GeoPPI.

2.1 Quick setup

Users need to accomplish the following three steps to complete the installation.

Step 1: Clone the GitHub repository

The GeoPPI repository can be obtained at the GitHub website. Run the following commands to download the repository:

```
$ git clone https://github.com/Liuxg16/GeoPPI.git
$ cd GeoPPI
```

Step 2: Build the required dependencies

Users need to execute the following command to satisfy the minimum set of requirements.

```
$ sh install.sh [flag]
```

If your system has already installed Anaconda software, please set [flag] to 1, otherwise set [flag] to 0.

Step 3: Download FoldX

The FoldX Suite is available through academic and commercial licenses. Please apply for a license and download FoldX v4.0 binary file from: http://foldxsuite.crg.eu/

Once you download the FoldX file, please unzip the file and put the FoldX binary file in this main directory (i.e., GeoPPI/foldx). For example, suppose the file name is "foldxLinux64.tar.gz", run the following commands (ubuntu environment):

```
$ cp foldxLinux64.tar.gz GeoPPI/
$ tar -zxvf  ./foldxLinux64.tar.gz
$ chmod  a+x ./foldx
```

Congratulations! The environment is ready to run GeoPPI. Please see Section 4 for running examples.

2.2 Step-by-step setup

This section will introduce the required dependencies (step 2) in details. If you want to understand more about the required environment of GeoPPI or meet some problems during the quick setup, you can use the following step-by-step instructions.

2.2.1 Install Anaconda

Conda is a cross-platform software package manager. Installing Anaconda requires two commands similar to the following:

```
$ wget https://repo.anaconda.com/archive/Anaconda3-2020.11-Linux-
x86_64.sh
$ sh Anaconda3-2020.11-Linux-x86_64.sh -b
```

More installation information can be found at:
https://conda.io/projects/conda/en/latest/user-guide/install/index.html

2.2.2 Create a virtual environment

A virtual environment is a named, independent working **environment** of Python so that

users can work with specific versions of libraries without affecting other Python projects. Creating and activating a virtual environment requires two commands similar to the following:

```
$ conda create -n ppi python==3.8.5 -y
$ conda activate ppi
```

2.2.3 Install PyTorch

PyTorch is an open-source machine learning library that features in tensor calculation. Installing PyTorch requires a single command similar to the following:

```
$ pip install --no-cache-dir torch==1.7.0+cpu -f
https://download.pytorch.org/whl/torch_stable.html
```

More installation information can be found at: https://pytorch.org/

2.2.4 Install torch-geometric

Torch-geometric is a library for deep learning on irregular input data such as graphs, point clouds, and manifolds. It requires other python dependencies such as torch-cluster, torch-scatter and torch-sparse. In particular, the installation of torch-geometric is sensitive to the versions of these python dependencies. Here, we provide a compatible combination of the versions of these dependencies as follows:

```
$ pip install --no-cache-dir torch-cluster==1.5.8 -f
https://pytorch-geometric.com/whl/torch-1.7.0+cpu.html
$ pip install --no-cache-dir torch-scatter==2.0.5 -f
https://pytorch-geometric.com/whl/torch-1.7.0+cpu.html
$ pip install --no-cache-dir torch-sparse==0.6.8 -f
https://pytorch-geometric.com/whl/torch-1.7.0+cpu.html
$ pip install --no-cache-dir torch-geometric==1.4.1
```

Note: To facilitate the installation process, we recommend the users install the python libraries with these specified versions. Other compatible combinations of the versions can be found at: https://pytorch-geometric.readthedocs.io/en/latest/notes/installation.html

2.2.5 Install scikit-learn

Scikit-learn is another Python library that contains more machine learning algorithms. Installing scikit-learn requires a single command similar to the following:

```
$ pip install --no-cache-dir scikit-learn==0.24.1
```

2.2.6 Install PyMOL

PyMOL is a powerful and comprehensive molecular visualization product for rendering and animating 3D molecular structures. Installing PyMOL via Anaconda requires a single command similar to the following:

```
$ conda install -c schrodinger pymol -y
```

2.3 Setup with CUDA support

CUDA (Compute Unified Device Architecture) is a parallel computing platform that works on Nvidia GPUs. Considering the versions of PyTorch and torch-geometric, we use CUDA v11.0. Please follow the instructions (https://developer.nvidia.com/cuda-11.0-download-archive) to install CUDA on your OS.

Based on the CUDA library, users can install python libraries by the following commands:

```
$ sh install-cuda.sh [flag]
```

Similarly, if your system has installed Anaconda software, please set [flag] to 1, otherwise set [flag] to 0.

3. How to use GeoPPI

Users can use GeoPPI to compute the binding affinity changes given the complex and the mutation information.

Before using GeoPPI, please activate the environment first.

```
$ conda activate ppi
```

Then, you can use the following command to obtain the results:

```
$ python run.py [pdb file] [Mutation] [partnerA_partnerB]
```

where [pdb file] is the complex structure of interest, [Mutation] denotes the mutation information and [partnerA_partnerB] describes the two interaction partners in the protein complex.

Format of [Mutation]: The mutation information includes WT residue, chain, residue index and mutant residue. such as “TI38F”, which stands for mutating the 38th acid amino at the I chain (i.e., phenylalanine) to threonine.

[partnerA_partnerB] are the chains of the two parts of the binding. For example, if the H

chain and the A chain of the complex belong to different proteins and interact with each other in the complex, [partnerA_partnerB] is "A_H". Similarly, "HL_WV" stands for the H and L chains interact with the W and V chains.

Program output: After several seconds of computing, the GeoPPI program will return the impact of the input mutation, i.e., $\Delta\Delta G$=$\Delta G_{wt} - \Delta G_{mutant}$. The positive value of $\Delta\Delta G$ stands for the higher binding affinity between two proteins, i.e., the stabilizing mutation.

For example, when we execute the command:

```
$ python run.py data/testExamples/1PPF.pdb  TI17F  E_I
```

The program output is similar to the following:

```
==========================Results=============================
The predicted binding affinity change (wildtype-mutant) is -1.76
kcal/mol (destabilizing mutation).
```

3.1 More examples

In the GeoPPI/data directory, there are several example complexes for users to test GeoPPI. Here, we also provide some example commands as follows.

```
python run.py data/testExamples/1PPF.pdb  TI17R  E_I
python run.py data/testExamples/1CZ8.pdb  KW84A  WV_HL
python run.py data/testExamples/1CSE.pdb  LI38I  E_I
python run.py data/testExamples/3SGB.pdb  KI7L  E_I
python run.py data/testExamples/3BT1.pdb  PU149A  U_A
```

Users can also use their own structures to analyze the mutation effects by putting the PDB files into the directory data/testExamples/.

4. Reproduce the split-by-structure cross validation experiment

In our work, we adopted a new cross-validation setting where the structures of complexes used in the training and testing are different. We used the ECOD (Evolutionary Classification of Domains) homology level to divide the data points to make different folds share no protein domain. We denote this new cross-validation experiment as the split-by-structure cross-validation (SSCV) to avoid confusion.

To reproduce the SSCV experiment on the S645 dataset, run the following command:

```
$ python sscv.py   data/S645/
```

After 30 minutes, you will get the prediction performance of GeoPPI (i.e., R_P=0.51).

5. Data included in the repository

Besides the example complexes, we provide the Extended Tables and the benchmark datasets in data/benchmarkDatasets. The data directory also contains the extracted features of the complexes on the S645 datasets for the reproduction of our results.

6. Author and contact support

If you encounter any problems during the setup of environment or the execution of GeoPPI, do not hesitate to contact liuxg16@mails.tsinghua.edu.cn or create an issue under the repository: https://github.com/Liuxg16/GeoPPI. Cheers!