ANN-CI User Manual

Koushik Seth, Sumanta Kr. Ghosh, Debashree Ghosh*

Abstract

The research group of Professor Debashree Ghosh focuses on the interdisciplinary subject of chemical physics. One area of focus for the group is the development of novel methodologies using cutting-edge tools like machine learning in the field of selected CI. These new techniques reduce the first principal-based method's enormous computational cost without sacrificing accuracy. This manual demonstrates how to use a handful of Prof. Ghosh's methodologies.

1 System Information and Login

A Linux-based system is required to run the codes. If execution is performed on a local machine, the following requirements are necessary for a smooth experience:

1. CPU - intel i5 (or higher)/ AMD Radeon 5 (or higher)
2. GPU - NVidia GPU (4GB/128-core or higher)
3. RAM - 8GB+
4. Storage - 512GB+ (preferably SSD)

For execution on any cluster or supercomputer, the user needs to log in and submit a calculation following the instructions of that particular cluster or supercomputer. Login details for the system for the MSCC workshop will be provided during the hands-on sessions.

2 System Information and Login

The following libraries are required to compile the codes –

1. Python3.6 +
2. PyTorch
3. Numba
4. f2py3
5. lapack
6. scikit-learn
7. Pandas
8. Matplotlib
9. Numpy

10. rdkit

3 Neural Network to Train Configuration Interaction (ANN-CI)

3.1 Description

This Artificial Neural Network (ANN) based supervised machine learning model differentiates the important and unimportant configurations without compromising the accuracy. The essence of the model is that it can predict accurately from the data generated from un-converged Monte Carlo.

3.2 Required Files

There are two files required to run this code –

1. Training data set (Data set for learning of ANN model).
2. Input file. (Contains all the information to execute the calculation).
3. Test data set (Unlabelled data set, which will be predicted by trained ANN model).

4 Active Learning Assisted Monte Carlo Configuration Inter action (AL-MCCI)

4.1 Description

Monte Carlo (MC) is a numerical technique where a problem is solved with the help of a random number. In Monte Carlo Configuration Interaction (MCCI), a system's electronic structure is solved using the CI-based method. Though MCCI helps study the electronic structure of a system, which is otherwise impossible to do, it suffers from slow convergence. We devised a protocol called active learning-assisted MCCI (AL-MCCI), where active learning makes the convergence many-fold faster and can also optimize Hilbert space for a particular target state.

Initially, the MCCI steps update the sub-Hilbert space and build the training data set. An artificial neural network (ANN) model learns from the data and, using that information predicts the relative importance of unlabeled configurations. The preliminary information about the configurations helps build a better Hilbert space, leading to faster convergence.

4.2 Required Files

There are two files required to run this code –

1. Bond Order file (Contains all the information about the system of interest).
2. Input file (Contains all the information to execute the calculation).

A set of sample required file are given in the drive link mentioned in the reference section. For more details follow the Github Page

5 How to Run

After getting the code from the drive-link/GitHub-page, the user needs to execute the following command –

```
f2py3 –L/path_of_lapack_library -llapack -c net_nstates.f -m net_nstates
```

This command generates a file net nstates.cpython-xxxxxx-gnu.so. Rename this file to net nstates.so using the following command

```
mv net_nstates.cpython-xxxxxx-gnu.so net_nstates.so
```

Finally, execute the following command to run the main program

```
python exe.py input_file.in &
```

Where input file.in is the name of the input file given by the user.

6 Expected Outputs

There is a total of 10 output files generated after successful calculations. The six main files are

1) input_file.in.out # Main output file, which contains information on subspace size
2) input_file.in.out.basis # Configurations of final sub-Hilbert space
3) input_file.in.out.ci # CI coeffcienet corrosponding to configurations
4) input_file.in.out.model.pth # Final optimized ANN model

5) input_file.in.out.error.dat # Train and test error at each AL iteration
6) input_file.in.out.TrainData_subSpace.csv # Train data set generated during calculation

These files are essential for system analysis. Apart from that, there are four more files for deeper analysis of machine learning performance..

7) input_file.in.out.predictData.csv
8) input_file.in.out.accVsPreTest.dat
9) input_file.in.out.accVsPreTrain.dat
10) input_file.in.out.enrich.csv

7 Important Links

7.1 Github Page

References

Sumanta K Ghosh, Madhumita Rano, and Debashree Ghosh. Configuration interaction trained by neu ral networks: Application to model polyaromatic hydrocarbons. The Journal of Chemical Physics, 154(9), 2021.

Koushik Seth and Debashree Ghosh. Active learning assisted mcci to target spin states. Journal of Chemical Theory and Computation, 19(2):524–531, 2023.