Design with ProteinMPNN

Bold text means that these files and/or this information is provided

Italicized text means that this material will NOT be conducted during the workshop

`Fixed with text means you should type the command into your terminal`

If you want to try making files that already exist (e.g., input files), write them to a different directory.

Introduction

This tutorial will give a short introduction in how to use the main applications of ProteinMPNN to redesign a protein sequence.

AlphaFold and Rosettafold are deep learning approaches which have shown to reliably predict protein structures from sequence. ProteinMPNN tries to solve the inverse problem, to find a sequence that matches a protein backbone. Based on the input structure it will predict a likely sequence based on its training data and provide a measure for the uncertainty of the predicted sequence. Therefore it can be used for different kinds of protein design challenges in which the backbone structure of the protein should stay fixed. Successful improvement in solubility and subsequently soluble protein yields have been already shown for monomers and homo-oligomers as well as improvement in yield and binding for de novo designed miniportein binders (references). In this tutorial we will redesign a small de novo designed miniprotein (pdb:1QYS) as well as redesign the protein interface between the SARS-CoV-2 spike protein and the miniprotein inhibitor LCB3 (pdb:7JZM). A straightforward and user-friendly platform to redesign a small number of sequences with ProteinMPNN is huggingface link. The focus of this tutorial will be how to design with ProteinMPNN on a local computer, which offers the opportunity to redesign a large number of structures via the command line and includes some additional design options compared to the browser application. Examples from this tutorial were partially adopted from https://github.com/dauparas/ProteinMPNN/blob/main/examples/.

Tutorial

Running ProteinMPNN via huggingface (link) or the available colab notebook (link) is an easy and user-friendly option to redesign single sequences without having to download the source code. To use a local version of ProteinMPNN the source code has to be downloaded from github. The code is dependent on multiple python packages so one of the most convenient ways to run it is by creating a conda environment with the necessary packages. Packages necessary to run this tutorial are listed in the **requirements.txt** file. Packages for ProteinMPNN and to execute this tutorial can be installed by running

These commands are for reference – the conda environment has already been activated on the workshop machines.

```
conda create --name pmpnn_tutorial
conda activate pmpnn_tutorial
conda install pytorch torchvision torchaudio cudatoolkit=11.3 -c pytorch
pip3 install -r requirements.txt
```

If you want to avoid mentioning the whole path to the ProteinMPNN directory explicitly in every run you may wish to set these environment variables:

```
export pmpnn="${HOME}/rosetta_workshop/ProteinMPNN/"
export PATH="${HOME}/rosetta_workshop/ProteinMPNN/:$PATH"
```

1. Basic usage

1.1 Activate the conda environment with the necessary packages to run ProteinMPNN.

```
conda activate pmpnn_tutorial
```

1.2 Display the help information of the main ProteinMPNN application by running

```
python ~/rosetta_workshop/ProteinMPNN/protein_mpnn_run.py -h
```

This command will display instructions on how to use the script and the available input arguments. It is technically the only script we need to initialize and run the model. Other available functions are used for preparing input data and creating option files.

1.3 First we do a simple redesign of the whole sequence of 1QYS without additional options. Change to the workshop directory and create a new directory for redesigning the 1QYS sequence.

```
cd ProteinMPNN_tutorial
mkdir 1qys_designs
python ~/rosetta_workshop/ProteinMPNN/protein_mpnn_run.py --pdb_path input/1qys/1qys.pdb \
  --out_folder ./1qys_designs/ --num_seq_per_target 10  --sampling_temp "0.1" \
  --seed 0 --batch_size 1 --model_name v_48_020
```

Sequence generation should only take a few seconds. The generated output sequences can be found in ./1qys_designs/seqs.

1.4 Take a look at the generated sequence file.

```
gedit ./1qys_designs/seqs/1qys.fa
```

1qys.fa contains the original sequence as well as all designs in fasta format. Each sequence is annotated with a score and a sequence recovery value. The score and global_score should be the same in this case. The global_score averages the residues score over all residues in the sequence whereas the score only represents an average over the designed residues. The sequence_recovery shows the percent identity of the designed residues.

2. Residue Bias

By installing ProteinMPNN locally we can make use of some of the applications published in the github repository. Scripts to generate input files to use some of the specialized options of ProteinMPNN (residue bias, omitting amino acids from the design, fixing positions) can be found in ProteinMPNN/helper_scripts/. One of these applications is to introduce a residue bias, for example to make hydrophobic residues more and cysteines less likely.

2.1 With the following command we redesign the 1qys sequence with a residue bias.

```
mkdir 1qys_designs_biased
python ~/rosetta_workshop/ProteinMPNN/helper_scripts/make_bias_AA.py --output_path 1qys_bias.jsonl \
  --AA_list "A F G I L M P V W Y C" \
  --bias_list "1.39 1.39 1.39 1.39 1.39 1.39 1.39 1.39 1.39 1.39 -1.1"

python ~/rosetta_workshop/ProteinMPNN/protein_mpnn_run.py --pdb_path input/1qys/1qys.pdb \
  --out_folder ./1qys_designs_biased/ --num_seq_per_target 10  --sampling_temp "0.1" \
  --seed 0 --batch_size 1 --model_name v_48_020 \
  --bias_AA_jsonl 1qys_bias.jsonl
```

2.2 By creating a sequence logo for both designs we can visualize if the amino acid bias had an effect on the designed sequence.

```
python scripts/create_logo.py --fasta 1qys_designs/seqs/1qys.fa --output 1qys_unbiased_logo.png
python scripts/create_logo.py --fasta 1qys_designs_biased/seqs/1qys.fa --output 1qys_biased_logo.png
```

Take a look at the two created sequence logos. You will see that biasing certain residues comes with risks and the type and intensity of the bias should be well thought out.

3. Fixing positions

Depending on the use case it might make sense to only redesign certain positions in the protein and keep other parts fixed. In the next case we want to redesign the interface between the SARS-CoV-2 spike receptor binding domain and the miniprotein inhibitor LCB3. The structure of the complex was crystallized and published under 7JZM and can be found in inputs/7JZM/7jzm.pdb.

3.1 We only want to design residues of LCB3 (chain A). To achieve this we first assign the chain that contains the positions we want to redesign.

```
python ~/rosetta_workshop/ProteinMPNN/helper_scripts/parse_multiple_chains.py --input_path input/7jzm/ \
  --output_path 7jzm.jsonl
python  ~/rosetta_workshop/ProteinMPNN/helper_scripts/assign_fixed_chains.py --input_path 7jzm.jsonl \
  --output_path 7jzm_assigned.jsonl --chain_list "A"
```

parse_multiple_chains.py takes a directory as input variable and can also be used to parse and afterwards design multiple pdb files in one directory at once. This can be especially useful if multiple structures, for example an ensemble of different docking conformations or a collection of hallucinated scaffolds, should be redesigned.

3.2 We roughly assume that the interface residues of LCB3 consist of all residues in 5 Angstrom distance from the spike protein. One way to inspect protein structures and select residues for design would be by using pymol. In this case we load the structure in pymol:

```
pymol input/7jzm/7jzm.pdb
```

and select and print out the residues of LCB3 that are part of the interface. For that we have to type into the pymol command line:

```
select interface, br. chain A within 5 of chain B
reslist =[]
iterate interface and name CA, reslist.append((resi))
print(*reslist)
```

3.3 You now have a list of positions that you can set as designable. ProteinMPNN is only able to accept a dictionary of positions that should be fixed. Use the following python scripts to create a jsonl file containing the fixed positions in a dictionary format.

```
python ~/rosetta_workshop/ProteinMPNN/helper_scripts/make_fixed_positions_dict.py --specify_non_fixed \
  --position_list "1 3 4 6 7 8 10 11 13 14 17 18 30 31 33 34 37 40" --chain_list "A" \
  --input_path input/7jzm.jsonl --output_path 7jzm_fixed_pos.jsonl
```

3.4 Now we can run ProteinMPNN with the selected residues. In addition, we want to output per residue scores this time, to get an impression which point mutations contribute the most to the sequence score.

```
mkdir 7jzm_designs
python ~/rosetta_workshop/ProteinMPNN/protein_mpnn_run.py --jsonl_path 7jzm.jsonl \
  --chain_id_jsonl 7jzm_assigned.jsonl --fixed_positions_jsonl 7jzm_fixed_pos.jsonl \
  --out_folder 7jzm_designs --num_seq_per_target 10 --sampling_temp "0.1" \
  --seed 0 --batch_size 1 --save_score 1 --save_probs 1
```

In the directory 7jzm_design three output directories will be created. In 7jzm_designs/seqs/ we will again find the designed sequences in fasta format.

```
gedit 7jzm_designs/seqs/7jzm.fasta
```

The sequence score and global_score should now differ, as we designed only part of the protein this time.

3.5. In a Jupyter notebook session we can interactively plot the amino acid probabilities. JupyterLab is an interactive interface for coding and allows us to easily change and run code. The notebook can be started with

```
jupyter lab scripts/plot_probs.ipynb
```

execute the two blocks of scripts by clicking into the respective fields and hitting the play/run button in the top bar or by pressing ctrl+Enter. The first block plots the outputted amino acid probabilities for all residues of the complex. In the second block we specify the redesigned positions. We can now investigate single scores for each of the positions we are interested in.

Depending on one's own prior knowledge ProteinMPNN can be completly run and results plotted in a python notebook. Code examples can be found under ProteinMPNN/colab_notebooks/ and can be customized according to need.

4. Using ProteinMPNN through the browser

If there is still time left try to reproduce the design steps on the ProteinMPNN huggingface page (link below). Running with residue bias will not be possible (status Nov, 2022) but it allows to select fixed residues with the vmd selection algebra. One additional useful tool is to model the output sequences with AlphaFold. *In general AlphaFold score can be used as an additional discriminator to select designs from ProteinMPNN. It definitely makes sense to check if the designs can be modeled in AlphaFold with an high confidence score.* On the server AlphaFold is run without creating a multiple sequence alignment of the input. In the case of the Covid spike protein complex it might lead to it not being able to portray the complex correctly.

Links

1. ProteinMPNN source code
2. ProteinMPNN huggingface
3. ProteinMPNN colab notebook

Literature

1. Dauparas J, Anishchenko I, Bennett N, Bai H, Ragotte RJ, Milles LF, Wicky BIM, Courbet A, de Haas RJ, Bethel N, Leung PJY, Huddy TF, Pellock S, Tischer D, Chan F, Koepnick B, Nguyen H, Kang A, Sankaran B, Bera AK, King NP, Baker D. Robust deep learning-based protein sequence design using ProteinMPNN. Science. 2022 Oct 7;378(6615):49-56. doi: 10.1126/science.add2187. Epub 2022 Sep 15. PMID: 36108050.
2. Wicky BIM, Milles LF, Courbet A, Ragotte RJ, Dauparas J, Kinfu E, Tipps S, Kibler RD, Baek M, DiMaio F, Li X, Carter L, Kang A, Nguyen H, Bera AK, Baker D. Hallucinating symmetric protein assemblies. Science. 2022 Oct 7;378(6615):56-61. doi: 10.1126/science.add1964. Epub 2022 Sep 15. PMID: 36108048.

3. Callaway E. Scientists are using AI to dream up revolutionary new proteins. Nature. 2022 Sep;609(7928):661-662. doi: 10.1038/d41586-022-02947-7. PMID: 36109683.