Graphein

Release 0.0.1

Arian Jamasb

Jul 27, 2020

GETTING STARTED:

This package provides functionality for producing a number of types of graph-based representations of proteins. We provide compatibility with standard formats, as well as graph objects designed for ease of use with popular deep learning libraries.

Note: This is an early-stage project and a lot more documentation and functionality is planned to be included. If you are a structural biologist or machine learning researcher in computational biology, my inbox is always open for suggestions and assistance!

CHAPTER

ONE

INSTALLATION

Graphein depends on a number of other libraries for constructing protein graphs and meshes. These should be installed in advance.

Note: We recommend installing Graphein in a virtual environment. ..

Note: Some of these packages have more involved setup depending on your requirements (i.e. CUDA). Please refer to the original packages for more detailed information

```
conda create -n graphein python=3.7
```

1.1 Installing PyTorch Libraries

```
pip install torch
pip install dgl
pip install pytorch3d
```

1.2 Installing Pytorch Geometric

```
$ pip install torch-scatter==latest+${CUDA} -f https://pytorch-geometric.com/whl/
↪torch-1.5.0.html
$ pip install torch-sparse==latest+${CUDA} -f https://pytorch-geometric.com/whl/torch-
↪1.5.0.html
$ pip install torch-cluster==latest+${CUDA} -f https://pytorch-geometric.com/whl/
↪torch-1.5.0.html
$ pip install torch-spline-conv==latest+${CUDA} -f https://pytorch-geometric.com/whl/
↪torch-1.5.0.html
$ python setup.py install or pip install torch-geometric
```

Install all needed packages with ${CUDA} replaced by either cpu, cu92, cu100 or cu101 depending on your PyTorch installation:

1.3 GetContacts Requirements

```
# Install get_contact_ticc.py dependencies
$ conda install scipy numpy scikit-learn matplotlib pandas cython seaborn
$ pip install ticc==0.1.4

# Install vmd-python dependencies
$ conda install netcdf4 numpy pandas seaborn  expat tk=8.5  # Alternatively use pip
$ brew install netcdf pyqt # Assumes https://brew.sh/ is installed

# Set up vmd-python library
$ git clone https://github.com/Eigenstate/vmd-python.git
$ cd vmd-python
$ python setup.py build
$ python setup.py install
$ cd ..

# Set up getcontacts library
$ git clone https://github.com/getcontacts/getcontacts.git
$ echo "export PATH=`pwd`/getcontacts:\$PATH" >> ~/.bash_profile
$ source ~/.bash_profile

# Test installation
$ cd getcontacts/example/5xnd
$ get_dynamic_contacts.py --topology 5xnd_topology.pdb \
                          --trajectory 5xnd_trajectory.dcd \
                          --itypes hb \
                          --output 5xnd_hbonds.tsv
```

1.4 Install DSSP

We use DSSP to compute secondary structure features of proteins.

```
conda install -c salilab dssp
```

1.5 IPyMol

Install IPyMol from GitHub. The release on PyPI appears to behind the repository and some required functionality is unavailable. https://github.com/cxhernandez/ipymol

```
git clone https://github.com/cxhernandez/ipymol
cd ipymol
pip install .
```

1.6 Install Graphein

```
git clone https://github.com/a-r-j/grahein
cd graphein
pip install -e .
```

CHAPTER

TWO

GRAPHEIN

Graphein is a python library for constructing graph and surface-mesh representations of protein structures for computational analysis. The library interfaces with popular geometric deep learning libraries: DGL, PyTorch Geometric and PyTorch3D. As feature engineering is a vital step in a machine learning project, the library is designed to be highly flexible, allowing the user to parameterise the graph construction, scalable to facilitate working with large protein complexes,and containing useful pre-processing tools for preparing experimental structure files. Graphein is also designed to facilitate network-based and graph-theoretic analyses of protein structures in a high-throughput manner.

CHAPTER

THREE

LICENSE

CHAPTER

FOUR

GRAPHEIN.CONSTRUCT_GRAPHS

4.1 graphein.construct_graphs.ProteinGraph

class graphein.construct_graphs.**ProteinGraph**(*granularity*, *keep_hets*, *insertions*, *node_featuriser*, *get_contacts_path*, *pdb_dir*, *contacts_dir*, *exclude_waters=True*, *covalent_bonds=True*, *include_ss=True*, *include_ligand=False*, *intramolecular_interactions=None*, *graph_constructor=None*, *edge_featuriser=None*, *edge_distance_cutoff=None*, *verbose=True*, *deprotonate=False*, *remove_string_labels=False*, *long_interaction_threshold=None*)

__init__(*granularity*, *keep_hets*, *insertions*, *node_featuriser*, *get_contacts_path*, *pdb_dir*, *contacts_dir*, *exclude_waters=True*, *covalent_bonds=True*, *include_ss=True*, *include_ligand=False*, *intramolecular_interactions=None*, *graph_constructor=None*, *edge_featuriser=None*, *edge_distance_cutoff=None*, *verbose=True*, *deprotonate=False*, *remove_string_labels=False*, *long_interaction_threshold=None*)

Initialise ProteinGraph Generator Class

Parameters

- **granularity** (*str*) – Specifies granularity of the graph construction. {'atom', 'CA', 'CB'}. CA = Alpha Carbon, CB = Beta Carbon
- **keep_hets** (*bool*) – Keep heteroatoms present in the PDB file. Typically, these correspond to metal ions or modified residues (e.g. MSE)
- **insertions** (*bool*) – Keep atoms/residues with multiple insertion positions. Multiple insertions exist when the electron density is too vague to define a single insertion
- **node_featuriser** (*DGL Node Featuriser*) – DGL Node featuriser for atom-level graphs. Canonical Featurises recommended.
- **pdb_dir** (*str*) – Directory to PDB files. We will download .PDB files to this folder if you don't have an existing local copy of the requisite structure
- **contacts_dir** (*str*) – Directory to GetContacts files
- **exclude_waters** (*bool*) – Specifies inclusion of water molecules. Not yet fully operational.

- **`covalent_bonds`** (*`bool`*) – Specifies inclusion of covalent backbone. E.g. joins adjacent residues in the sequence
- **`include_ss`** (*`bool`*) – Specifies inclusion of secondary structure features computed by DSSP. Future warning: this will be changed in a subsequent update for managing feature selection.
- **`include_ligand`** (*`bool`*) – Not yet implemented. Will specify option to include bound ligand(s) in the graph.
- **`intramolecular_interactions`** (*`list`*) – List of allowable intramolecular interactions to include from GetContacts. [‘sb’, ‘pc’, ‘ps’, ‘ts’, ‘vdw’, ‘hb’, ‘hbb’, ‘hbsb’, ‘hbbb’, ‘hbss’, ‘wb’, ‘wb2’, ‘hblb’, ‘hbls’, ‘lwb’, ‘lwb2’, ‘hp’]. See https://getcontacts.github.io/interactions.html for details.
- **`edge_distance_cutoff`** (*`float`*) – Distance in angstroms specifying cutoff distance for constructing an edge when using distance construction
- **`long_interaction_threshold`** (*`int`*) – Specifies minimum distance in sequence for two nodes to be connected

`dgl_graph_from_pdb_code`(*pdb_code=None*, *file_path=None*, *chain_selection='all'*, *contact_file=None*, *edge_construction=['contacts']*, *encoding=False*, *k_nn=None*, *custom_edges=None*)

Produces a DGL graph from a PDB code and a selection of polypeptide chains

Parameters

- **`file_path`** (*`str`*) –
- **`custom_edges`** (*`Pandas DataFrame, optional`*) – Pass user-defined custom edges to use in edge construction, defaults to None
- **`edge_construction`** (*`list`*) – Specifies edge construction methods. {‘contact’, ‘distance’, ‘custom’}, defaults to [‘contacts’]
- **`k_nn`** (*`int`*) – Specifies number of nearest neighbours to make K_NN edges with
- **`encoding`** (*`bool`*) – Indicates whether or not node names and labels should be encoded
- **`contact_file`** (*`str`*) – Path to local GetContacts output file, defaults to None
- **`pdb_code`** (*`str`*) – 4 character PDB accession code
- **`chain_selection`** (*`list`*) – string indicating which chains to select {‘A’, ‘B’, ‘AB’, . . . , ‘all’}, defaults to ‘all’

Returns DGLGraph object, nodes populated by residues or atoms as specified in class initialisation

`dgl_graph_from_pdb_file`(*file_path*, *chain_selection*, *contact_file*, *edges=None*)

Produces a DGL graph from a PDB file and a selection of polypeptide chains

Parameters

- **`edges`** (*`Pandas DataFram, optional`*) – User-defined custom edges, defaults to None
- **`contact_file`** (*`str`*) – Path to local GetContacts output file
- **`file_path`** (*`str`*) – 4 character PDB accession code
- **`chain_selection`** (*`str`*) – Polypeptide chains in structure to select {‘A’, ‘B’, ‘AB’, . . . , ‘all}

Returns DGLGraph object, nodes populated by residues or atoms as specified in class initialisation

Return type DGLGraph

`nx_graph_from_pdb_code`(*pdb_code*, *chain_selection='all'*, *contact_file=None*, *edge_construction=['contacts']*, *encoding=False*, *k_nn=None*, *custom_edges=None*)

Produces a NetworkX Graph Object

Parameters

- **`encoding`** –
- **`edges`** (`Pandas DataFrame, optional`) – User-supplied edges, defaults to None
- **`pdb_code`** (`str`) – 4 character PDB accession code
- **`chain_selection`** (`str`) – string indicating chain selection {'A', 'B', 'AB', . . . , 'all'}, defaults to 'all'
- **`contact_file`** (`str, optional`) – Path to GetContacts output file.

Returns NetworkX graph object of protein

Return type NetworkX graph

`nx_graph_from_pdb_file`(*pdb_code*, *chain_selection='all'*, *contact_file=None*)

Produces a NetworkX Graph Object

Parameters

- **`pdb_code`** (`str`) – 4 character PDB accession code
- **`chain_selection`** (`str`) – string indicating chain selection {'A', 'B', 'AB', . . . , 'all'}
- **`contact_file`** (`str, optional`) – Path to GetContacts output file.

Returns NetworkX graph object of protein

`torch_geometric_graph_from_pdb_code`(*pdb_code*, *chain_selection='all'*, *edge_construction=['contacts']*, *contact_file=None*, *encoding=False*, *k_nn=None*, *custom_edges=None*)

Produces a PyToch Geometric Data object from a protein structure

Parameters

- **`k_nn`** (`int, optional`) – Specifies K nearest neighbours to use in KNN edge construction, defaults to None
- **`custom_edges`** (`Pandas DataFrame, optional`) – User-supplied edges to use, defaults to None
- **`encoding`** (`bool`) –
- **`edge_construction`** (`list`) – List containing edge construction to be used. ['contacts', 'distance', 'delaunay'], defaults to ['contacts']
- **`pdb_code`** (`str`) – 4-character PDB accession code
- **`chain_selection`** (`str`) – Specifies polypeptide chains to include. e.g. one of {'A', 'B' ,'AB', 'BC'}, defaults to 'all'
- **`contact_file`** (`str`) – Path to contact file if using local file.

Returns Pytorch Geometric Graph of protein structure.

Return type PyTorch Geometric Data object

CHAPTER

FIVE

GRAPHEIN.CONSTRUCT_MESHES

5.1 graphein.construct_meshes.ProteinMesh

class graphein.construct_meshes.**ProteinMesh**

__init__()

Initialise ProteinGraph Generator Class

create_mesh(*pdb_code=None*, *pdb_file=None*, *out_dir=None*)

Creates a PyTorch3D Mesh from an .Obj file. pdb_code and pdb_file are optional arguments. Use one as suits your purposes

Parameters

- **pdb_code** (*str*) – 4-character PDB accession code
- **pdb_file** (*str*) – Path to local .PDB file
- **out_dir** (*str*) – Path to output directory

Returns verts, faces, aux

get_obj_file(*pdb_file=None*, *pdb_code=None*, *out_dir=None*)

Produces .Obj file from PDB structure through IPyMol. pdb_code and pdb_file are optional arguments. Use one as suits your purposes

Parameters

- **pdb_file** (*str*) – Path to local .PDB file
- **pdb_code** (*str*) – 4 character PDB accession code
- **out_dir** (*str*) – Path to output directory

Returns

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search

INDEX