Chemprop

Journal of Chemical Information and Modeling > Vol 64/Issue 1 > Article

Open Access

Cite Share Jump to Expand

APPLICATION NOTE | December 26, 2023

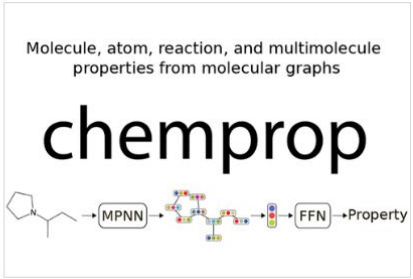
Chemprop: A Machine Learning Package for Chemical Property Prediction

Esther Heid, Kevin P. Greenman, Yunsie Chung, Shih-Cheng Li, David E. Graff, Florence H. Vermeire, Haoyang Wu, William H. Green, and Charles J. McGill*

Open PDF | Supporting Information (1)

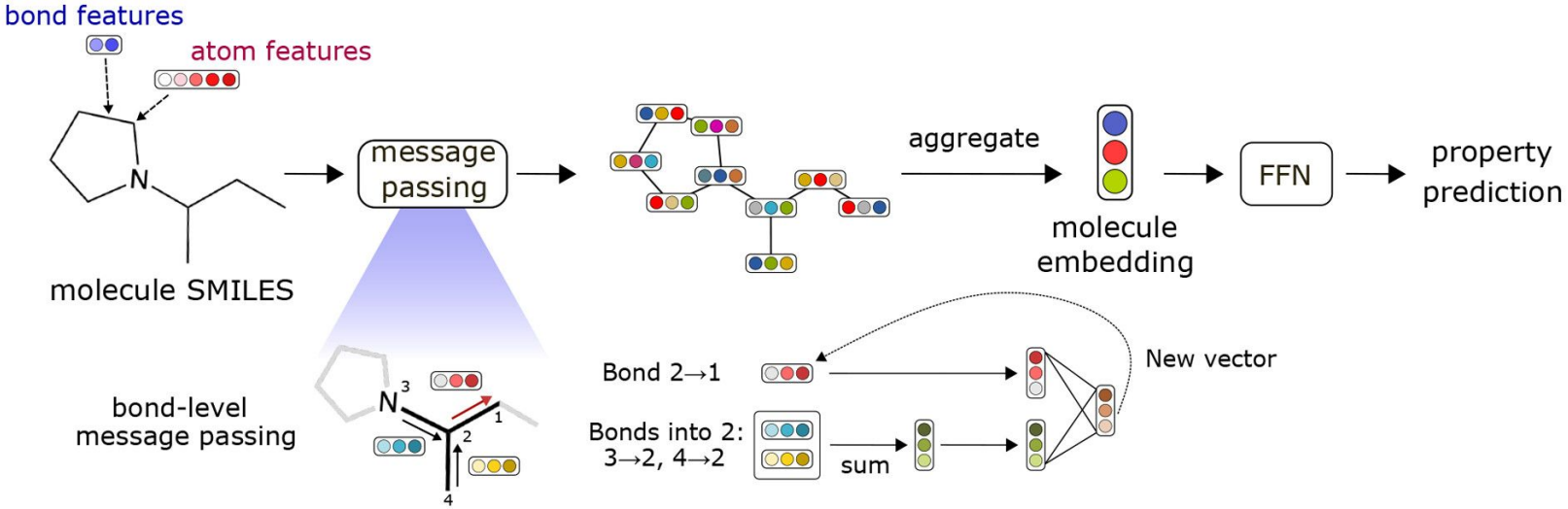
Abstract

Deep learning has become a powerful and frequently employed tool for the prediction of molecular properties, thus creating a need for open-source and versatile software solutions that can be operated by nonexperts. Among the current approaches, directed message-passing neural networks (D-MPNNs) have proven to perform well on a variety of property prediction tasks. The software package Chemprop implements the D-MPNN architecture and offers simple, easy, and fast access to machine-learned molecular properties. Compared to its initial version, we present a multitude of new Chemprop functionalities such as the support of multimolecule properties, reactions, atom/bond-level properties, and spectra. Further, we incorporate various uncertainty quantification and calibration methods along with related metrics as well as pretraining and transfer learning workflows, improved hyperparameter optimization, and other customization



- doi: 10.1021/acs.jcim.3c01250
- https://doi.org/10.1021/acs.jcim.3c01250

Chemprop's model architecture



Atom features: atomic number, molar mass, aromaticity, formal charge, …
Bond features: bond type, conjugation, stereo, …

ChemProp documentaion

https://chemprop.readthedocs.io/en/latest/index.html

Training

Training

To train a model, run:

```
chemprop train --data-path <input_path> --task-type <task> --output-dir <dir>
```

where `<input_path>` is the path to a CSV file containing a dataset, `<task>` is the type of modeling task, and `<dir>` is the directory where model checkpoints will be saved.

For example:

```
chemprop train --data-path tests/data/regression.csv \
    --task-type regression \
    --output-dir solubility_checkpoints
```

The following modeling tasks are supported:

- `regression`
- `regression-mve`
- `regression-evidential`
- `regression-quantile`
- `classification`
- `classification-dirichlet`
- `multiclass`
- `multiclass-dirichlet`
- `spectral`

Input Data

In order to train a model, you must provide training data containing molecules (as SMILES strings) and known target values. Targets can either be real numbers, if performing regression, or binary (i.e. 0s and 1s), if performing classification. Target values which are unknown can be left as blanks. A model can be trained as either single- or multi-task.

The data file must be be a **CSV file with a header row**. For example:

```
smiles,NR-AR,NR-AR-LBD,NR-AhR,NR-Aromatase,NR-ER,NR-ER-LBD,NR-PPAR-gamma,SR-ARE,SR-ATAD5,SR
CCOc1ccc2nc(S(N)(=O)=O)sc2c1,0,0,1,,,0,0,1,0,0,0,0
CCN1C(=O)NC(c2ccccc2)C1=O,0,0,0,0,0,0,0,,0,,0,0
...
```

Hyperparameter Optimization

Hyperparameter Optimization

Note

Chemprop relies on Ray Tune for hyperparameter optimization which is an optional install. To install the required dependencies, run `pip install -U ray[tune]` if installing with PyPI, or `pip install -e .[hpopt]` if installing from source.

Searching Hyperparameter Space

We include an automated hyperparameter optimization procedure through the Ray Tune package. Hyperparameter optimization can be run as follows:

```
chemprop hpopt --data-path <data_path> --task-type <task> --search-parameter-keywords <keywo
```

For example:

```
chemprop hpopt --data-path tests/data/regression.csv \
    --task-type regression \
    --search-parameter-keywords depth ffn_num_layers message_hidden_dim \
    --hpopt-save-dir results
```

The search parameters can be any combination of hyperparameters or a predefined set. Options include `basic` (default), which consists of:

- `depth` The number of message passing steps
- `ffn_num_layers` The number of layers in the FFN model
- `dropout` The probability (from 0.0 to 1.0) of dropout in the MPNN & FNN layers
- `message_hidden_dim` The hidden dimension in the message passing step
- `ffn_hidden_dim` The hidden dimension in the FFN model

Another option is `learning_rate` which includes:

- `max_lr` The maximum learning rate
- `init_lr` The initial learning rate. It is searched as a ratio relative to the max learning rate
- `final_lr` The initial learning rate. It is searched as a ratio relative to the max learning rate
- `warmup_epochs` Number of warmup epochs, during which the learning rate linearly increases from the initial to the maximum learning rate

Other individual search parameters include:

- `activation` The activation function used in the MPNN & FFN layers. Choices include `relu`, `leakyrelu`, `prelu`, `tanh`, and `elu` (`selu` is no longer supported since v2.2.0)
- `aggregation` Aggregation mode used during molecule-level predictor. Choices include `mean`, `sum`, `norm`
- `aggregation_norm` For `norm` aggregation, the normalization factor by which atomic features are divided
- `batch_size` Batch size for dataloader

Hyperparameter Optimization-2

You can use multiple GPUs for hyperparameter optimization :)

The following other common keywords may be used:

> - `--raytune-num-samples <num_samples>` The number of trials to perform
> - `--raytune-num-cpus <num_cpus>` The number of CPUs to use
> - `--raytune-num-gpus <num_gpus>` The number of GPUs to use
> - `--raytune-max-concurrent-trials <num_trials>` The maximum number of concurrent trials
> - `--raytune-search-algorithm <algorithm>` The choice of control search algorithm (either `random`, `hyperopt`, or `optuna`). If `hyperopt` is specified, then the arguments `--hyperopt-n-initial-points <num_points>` and `--hyperopt-random-state-seed <seed>` can be specified.

Prediction

Prediction

To load a trained model and make predictions, run:

```
chemprop predict --test-path <test_path> --model-paths <[model_paths]>
```

where `<test_path>` is the path to the data to test on, and `<[model_paths]>` is the location of checkpoint(s) or model file(s) to use for prediction. It can be a path to either a single pretrained model checkpoint (.ckpt) or single pretrained model file (.pt), a directory that contains these files, or a list of path(s) and directory(s). If a directory, will recursively search and predict on all found (.pt) models. By default, predictions will be saved to the same directory as the test path. If desired, a different directory can be specified by using `--preds-path <path>`. The predictions <path> can end with either .csv or .pkl, and the output will be saved to the corresponding file type.

For example:

```
chemprop predict --test-path tests/data/smis.csv \
    --model-path tests/data/example_model_v2_regression_mol.ckpt \
    --preds-path preds.csv
```

Specifying Data to Parse

By default, Chemprop will assume that the the 0th column in the data .csv will have the data. To use a separate column, specify:

> - `--smiles-columns` Text label of the column that includes the SMILES strings

Fingerprint

Fingerprint

To calculate the learned representations (encodings) of model inputs from a pretrained model, run

```
chemprop fingerprint --test-path <test_path> --model-path <model_path>
```

where `<test_path>` is the path to the CSV file containing SMILES strings, and `<model_path>` is the location of checkpoint(s) or model file(s) to use for prediction. It can be a path to either a single pretrained model checkpoint (.ckpt) or single pretrained model file (.pt), a directory that contains these files, or a list of path(s) and directory(s). If a directory, will recursively search and predict on all found (.pt) models. By default, predictions will be saved to the same directory as the test path. If desired, a different directory can be specified by using `--output <path>`. The output <path> can end with either .csv or .npz, and the output will be saved to the corresponding file type.

For example:

```
chemprop fingerprint --test-path tests/data/smis.csv \
    --model-path tests/data/example_model_v2_regression_mol.ckpt \
    --output fps.csv
```

Specifying FFN encoding layer

By default, the encodings are returned from the penultimate linear layer of the model's FFN. However, the exact layer to draw encodings from can be specified using `--ffn-block-index <index>`.

An index of 0 will simply return the post-aggregation representation without passing through the FFN. Here, an index of 1 will return the output of the first linear layer of the FFN, an index of 2 the second layer, and so on.

Appendix

If you are working with HPC, job combination of job scheduler might be useful.

link https://learning.rc.virginia.edu/notes/dl-drug-discovery/chemprop-slurm/