

COMP 396: Milestone 2

Representation learning :theoretical understanding & tentative implementation

Summary

1. Graph representational learning : Theoretical understanding
2. Data exploration for graph representation
3. Smiles + LSTM
4. Issues
5. Next steps

The second objective of the research project is to reproduce the baseline model Message passing neural networks paper <https://arxiv.org/pdf/1704.01212.pdf> (i.e Gated graph sequence neural network) and/ or the main message passing neural network that were implemented in the paper with the use of Pytorch Geometric.

MPNN papers : The following are the models listed in the paper as examples that use the neural message passing framework. We will be mainly focusing on the gated graph neural network and the final MPNN that was implemented.

Convolutional Networks for Learning Molecular Fingerprints, Duvenaud et al. (2015)
<https://arxiv.org/pdf/1509.09292.pdf>

Gated Graph Neural Networks (GG-NN), Li et al. (2016)
<https://arxiv.org/pdf/1511.05493.pdf>

Interaction Networks, Battaglia et al. (2016)
<https://arxiv.org/pdf/1612.00222.pdf>

Molecular Graph Convolutions: Moving beyond Fingerprints, Kearnes et al. (2016)
<https://arxiv.org/pdf/1603.00856.pdf>

Deep Tensor Neural Networks, Schutt et al. (2017)
<https://arxiv.org/pdf/1609.08259.pdf>

Laplacian Based Methods, Bruna et al. (2013); Defferrard et al. (2016); Kipf & Welling (2016) ->

Spectral Networks and Deep Locally Connected Networks on Graphs <https://arxiv.org/pdf/1312.6203.pdf>

Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering <https://arxiv.org/pdf/1606.09375.pdf>

Graph representation learning: theoretical understanding

In order to get a better understanding of how MPNN's work we start by looking at alternate representations of the QM9 datasets from the following sources:

-Pytorch geometric ->

https://github.com/rusty1s/pytorch_geometric/blob/master/torch_geometric/datasets/qm9.py

- Pau Riba & Anjan Dutta->https://github.com/priba/nmp_qc

Note: From our understanding, the first implementation is not complete yet and we currently cannot rely on the second one (. see issues listed below). However, we find that they are both great starting points.

QM9 Graph Representation :

In the current pytorch geometric representation of the QM9 dataset, each molecule is represented as an undirected graph $G=(V,E)$ with vertices representing the atoms and edges the bonds between the atoms. Each atom has distinct features shown on both Figure 1 and Table 1 below. The target variable Y is the same for all atoms in the same molecule. All of the properties in a molecule are wrapped in a Data class which also has information about the edge_index(i.e unidirectional bonds between atoms), the positions and the edge attributes.

The second repository has a similar representation with differences in how the edges attributes are implemented. As we have already mentioned it is also more complete and contains information about how to compute distance bins and raw distance for the edge representations.

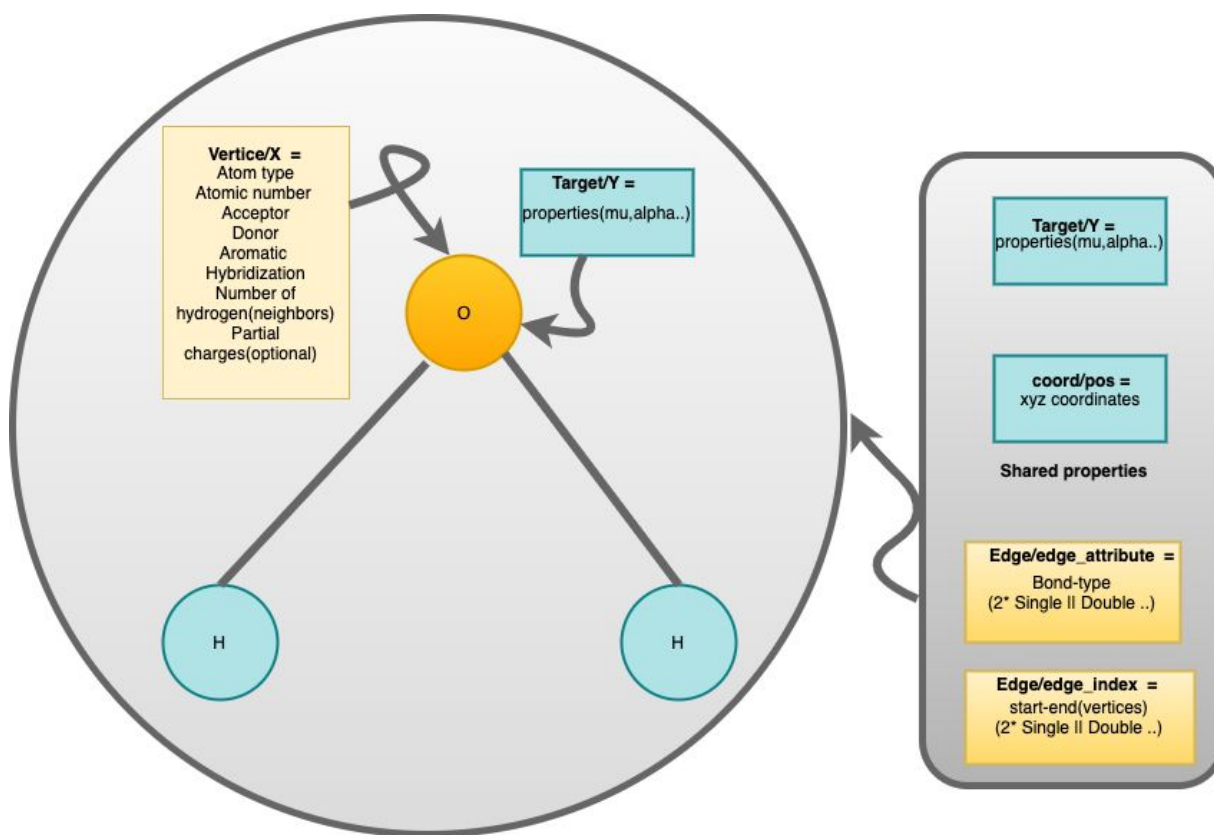


Figure1: Pytorch geometric graph representation

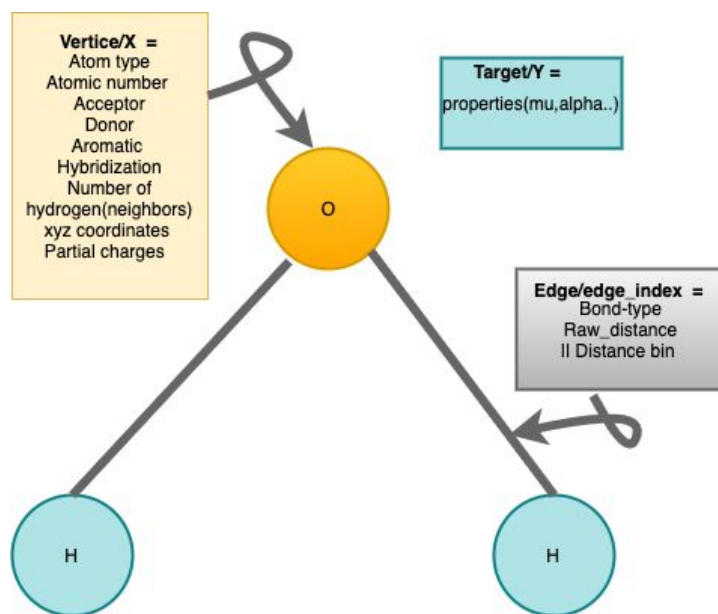


Figure2:Pau Riba &Anjan Dutta graph representation

Tools: The main tool that was used to retrieve information about the features of the molecules is RDKit.

The following tables provide information about the features and functions that were used to retrieve them.

Nodes

Node information	Description	Rdkit function (needs GetAtomIdx())
Atom type	H, C, N, O, ,F (one-hot or null).	GetSymbol()
Atomic number	Integer electronic charge.	GetAtomicNum()
Partial Charge	-	-
Acceptor	If the atom accepts electrons	GetFamily()
Donor	If the atom donates electrons	GetFamily()
Aromatic	If the atom is part of an aromatic system.	GetIsAromatic()
Hybridization	sp, sp2 , or sp3 (one-hot or null).	GetHybridization()

Edges

Types	Description	Rdkit function (needs GetAtomIdx())
Chemical graph	Bond type	GetBondType()
Distance bin	-	-
Raw distance	Magnitude of the vector	-

How message passing neural networks work

After representing QM9 data, create an adjacency matrix of size $(V \times V)$ to store relationships between nodes and edges. (Will need to be indexed to work in batches)

Hidden states at each vertex are updated over T message passing steps using message functions M_t and update functions U_t

- a. Message passing phase
 - There are three types of messages functions (matrix multiplication, pair message and edge network)

Ex: $M(h_v, h_w, e_{vw}) = (A_{evw}) h_w$.

$A_{evw} \rightarrow$ neural network with hidden layers

- b. Update phase

Updates node h_v given message

- c. Readout phase

Readout graph given node values (Two types : one in GG-N and set 2Set)

NB: I do not completely understand these functions and how they are implemented yet.

Data exploration for graph representation

Google colab links

NB: Please keep in mind that these are just drafts/ exploration I have been working on to test the tools and that they are not well organized and possibly have multiple errors.

-Rdkit exploration ->

https://colab.research.google.com/drive/1roLxfpXrCX7k-XsoPX_bUEgGRx-xTC91

- Pytorch geometric exploration ->

<https://colab.research.google.com/drive/1Rd3Hx0SKRtlcAzdR4snuzxFqN8N8E-XA>

Smiles+ LSTM

We started doing more research about smiles and LSTMs and started working on the implementation.

In terms of the data representation for the smiles, most of the work we looked at suggested to use a one hot encoding.

The following is the link of the data representation for QM9 SMILES with the use of deepchem one-hot encoding functions:

https://colab.research.google.com/drive/1LKKwx6PLeGNxsMG_v7GviQE1oqlmH7BK

NB: I haven't added the LSTM and I'm also confused about the output of the LSTM

Related papers:

SMILES2VEC: <https://arxiv.org/pdf/1712.02034.pdf>

Summary: A deep RNN that automatically learns features from SMILES to predict chemical properties . It uses the Tox21, HIV, and FreeSolv dataset from the MoleculeNet benchmark for predicting toxicity, activity and solvation free energy respectively.

CHEMCeption: <https://arxiv.org/pdf/1706.06689.pdf>

Summary: Chemception is a deep CNN for the prediction of chemical properties, using the images of 2D drawings of SMILES

Issues /Difficulties

Difficulties/questions

- Need more time to get up to speed with Pytorch geometric
- Not sure how propagation phase is done in parallel (in batches)for message passing phases
- Not sure about how to use LSTMS here :

- As an encoder ? to predict directly the properties or to represent the features then use a regression model on top?


Issues:

- Inability to reproduce MPNN from 2nd repository due to tensor mismatch issues that apparently other users had.

[Code](#)
[Issues 2](#)
[Pull requests 1](#)
[Projects 0](#)
[Wiki](#)
[Security](#)
[Insights](#)

RuntimeError during default execution #3

[Open](#)
 AlexanderGri opened this issue on Dec 16, 2017 · 7 comments · May be fixed by #5


 AlexanderGri commented on Dec 16, 2017 · edited

Hello, thank you for your implementation!

I've just tried to run default experiment with

```
python main.py --no-cuda --epochs 1
```

and run into the following problem

```

/opt/conda/lib/python3.6/importlib/_bootstrap.py:205: RuntimeWarning: compile
Prepare files
Define model
    Statistics
    Create model
Optimizer
Logger
=> no best model found at './checkpoint/qm9/mpnn/model_best.pth'
Check cuda
Traceback (most recent call last):
  File "main.py", line 321, in <module>
    main()
  File "main.py", line 182, in main
    train(train_loader, model, criterion, optimizer, epoch, evaluation, logger
  File "main.py", line 242, in train
    output = model(g, h, e)
  File "/opt/conda/lib/python3.6/site-packages/torch/nn/modules/module.py", li
    result = self.forward(*input, **kwargs)
  File "/data/grishin/nmp_qc/models/MPNN.py", line 78, in forward
    m = self.m[0].forward(h[t], h_aux, e_aux)
  File "/data/grishin/nmp_qc/MessageFunction.py", line 43, in forward
    return self.m_function(h_v, h_w, e_vw, args)
  File "/data/grishin/nmp_qc/MessageFunction.py", line 175, in m_mpnn
    h_w_rows = h_w[...].expand(h_w.size(0), h_v.size(1), h_w.size(1)).cc
RuntimeError: The expanded size of the tensor (25) must match the existing siz

```

Am i doing something wrong? Thank you in advance.


Assignees
 No one assigned

Labels
 None yet

Projects
 None yet


Milestone
 No milestone

Notifications [Customize](#)
[Subscribe](#)
 You're not receiving notifications from this thread.

7 participants


Next Steps

- attempt implementing either MPNN or baseline model Gated graph neural network
- work on SMILES + LSTM ?



- or your suggestions?

-

■ ■ ■