COMP396

SMILES string molecular descriptors

Summary

1. Description
2. SMILES +LSTM
3. SMILES + CNN
4. Questions (feedback on the report and findings)
5. Next steps?

Description SMILES String:

SMILES (Simplified molecular-input line-entry system) strings are a compact way of representing molecules. In this project, we use SMILES strings in order to predict target properties from the QM9 benchmark. The SMILES that are retrieved from the xyz files are canonicalized and thus have a unique representation.

The main advantage of using SMILES string-based molecular descriptors is that they are less sophisticated then graph neural networks and perform relatively well.

SMILES representations can also be modified to include chiral indications. However, we only consider non-isomeric molecules from the QM9 dataset in this milestone.

We experiment with two different ways of building a molecular descriptor with the SMILES strings.

NB: The issue we encountered in our first attempts to reproduce the SMILES string with LSTM model was due to the incorrect encoding of the SMILES. The one-hot encoding dimensionality and sparsity were too high.

As you recommended, using **lookup embeddings** solved the problem. With the one-hot encoding alone we were not able to encode the semantic similarity of the SMILES.

With Pytorch, we used an embedding layer that maps that integer indices to dense vectors.

Smiles String-based molecular descriptor Inspiration from -> Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules https://arxiv.org/pdf/1610.02415.pdf

SMILES + LSTM

Implementation:

Dataset: QM9

Training, validation and testing size: 11000, 1000,1000

The models were trained using a mean squared error (MSE) loss function, stochastic gradient descent (SGD) with the Adam optimizer, and an initial learning rate of 0.001 for 50 epochs.

Mu: https://colab.research.google.com/drive/1VZk2owf4zxyR0LwZ1NLRJLj8-igp2mYn

Alpha: https://colab.research.google.com/drive/15KakRofEDmeTf7qc6sCTXAPBW6f_Kjda

HOMO: https://colab.research.google.com/drive/1kTa8oH9AcmBD6Wj84XVPtrtCWv9n07sr

LUMO:https://colab.research.google.com/drive/1YfDmU7nVT51OwWe7IRxNJ5HAbjW-QUlf

Gap: https://colab.research.google.com/drive/1G-Purxn8C_CP1aK3_2pilUd30_S8ORa4

R2 :https://colab.research.google.com/drive/1R5E6a16mDECJ_aY4siol2W0HWrT_4gGc

Zpve :https://colab.research.google.com/drive/1qn5IPSo_3Jdo1NRRdjeRnfiVXQyfS27Q

U0:https://colab.research.google.com/drive/1GMOF8_LqMv4Jjg5cZksnTrJKreuHjTHz

U:https://colab.research.google.com/drive/1T0jAGXvJ6Q2mvQm-8t9Xy-aD0TcxPYLx

H:https://colab.research.google.com/drive/120Usw7PzVkS1Ud3JXilnhlY7EGybpl1x

G:https://colab.research.google.com/drive/1FD-Dqh0Tk-Daz65ihtVH6danxOy3kG0C

Target	Mu	Alpha	HOMO	LUMO	gap	R2	ZPVE	U0	U	H	G	cv

MAE	0.8129	3.1636	0.00871	0.02378	0.0248	84.8628	0.0151	15.4144	15.5463	15.3164	15.3164	1.0816

SMILES + CNN

Implementation:

Dataset: QM9

Training, validation and testing size: 11000, 1000,1000

The models were trained using a mean squared error (MSE) loss function, stochastic gradient descent (SGD) with the Adam optimizer, and an initial learning rate of 0.001 for 50 epochs.

Mu: https://colab.research.google.com/drive/155w-FCDB-wF_EJDwLd6rEiVPPdXkUuVA

Alpha: https://colab.research.google.com/drive/1K0ajlZRLVx1qszfnPli_tgPji4_NWPLj

HOMO: https://colab.research.google.com/drive/1RycsoqU37YGYjLC-NQqyJi3Qjp_KG0Tt

LUMO:https://colab.research.google.com/drive/1lEPKwY3tpuyjVx5aln-6-AVFiMNKDlHa

Gap: https://colab.research.google.com/drive/117Toe3yvLQcycG-BWL10QkzXcpHLkP5X

R2 :https://colab.research.google.com/drive/1gOr3YMzvp8ocx5zf2V3mOOGkNvZahhLu

Zpve :https://colab.research.google.com/drive/1PbqH0xVW7uXCn6xeah7kF_HbchTxYnVi

U0:https://colab.research.google.com/drive/1856iZXZ5f6XetkOz4pVQ-rsAB-iQurGe

U:https://colab.research.google.com/drive/1Uf-NEx0UsxVnvXgVl6U1Fqg5TOjOzo9H

H:https://drive.google.com/open?id=120Usw7PzVkS1Ud3JXilnhlY7EGybpl1x

G:https://colab.research.google.com/drive/15rCErVd0-QEHlPyFy9X7FE9ENsFmgezB

cv:https://colab.research.google.com/drive/1x6WUFM94eyYl4tAzP_JaPuZSbofRbY9T

Target	Mu	Alpha	HOMO	LUMO	gap	R2	ZPVE	U0	U	H	G	cv
MAE	0.8435	3.1797	0.0083	0.0198	0.0225	78.8795	0.0146	16.259	14.407	14.7347	14.0354	1.2181

Comparison of models

Target	Mu	Alpha	HOMO	LUMO	gap	R2	ZPVE	U0	U	H	G	cv
LSTM	0.8129	3.1636	0.00871	0.02378	0.0248	84.8628	0.0151	15.4144	15.5463	15.3164	15.3164	1.0816
CNN	0.8435	3.1797	0.0083	0.0198	0.0225	78.8795	0.0146	16.259	14.407	14.7347	14.0354	1.2181

Target	BOB	CM	GG-NN	MPNN
mu	0.6724	0.8078	0.7639	**0.1523**
alpha	0.7782	1.4809	0.9228	**0.3847**
HOMO	0.0074	0.017	0.0081	**0.0034**
LUMO	0.0106	0.0171	0.0111	**0.0037**
gap	0.0122	0.0196	0.0118	**0.0065**
R2	22.2674	37.7441	76.7278	**2.5781**
ZPVE	0.0007	0.0010	0.0007	**0.0004**
U0	0.7360	3.4899	**0.3955**	0.5545
U	0.7360	3.4895	**0.2621**	0.5218
H	0.7360	3.4898	**0.3899**	0.3991
G	0.7360	3.4901	**0.4603**	0.4632
CV	0.4156	0.6600	0.5121	**0.1516**

Observations:- The training is a lot faster than that of graph neural networks.

- A bit of variability during the training
- MAE is higher than all other models. Increasing the amount of data could possibly resolve this.

Questions/ Feedback on Report (if time permits)

-https://github.com/bmbodj/COMP396/blob/master/Fall_2019/COMP396_report.pdf

-Is the calculation of my MAE correct?

MAE=MAE* standard deviation

-Are there ideas/logic that do not make sense or that can be enhanced in my report?

- Is the edge network/ or any graph model that handles edge attributes a must for efficient quantum property prediction tasks on molecules? (OR) Would it depend on the task (node, graph or edge, level)?

-Which graph model would you consider as the state of the art for quantum property predictions?

I found a paper that uses DGI to maximize information between edge states and transform parameters.-> Utilizing Edge Features in Graph Neural Networks

via Variational Information Maximization https://arxiv.org/pdf/1906.05488.pdf

Next Steps?

- Use more data for Smiles to increase accuracy?

- Double-check models and attempt to correct training

-Use a visualization toolkit (i.e tensorBoard) to inspect training loss

-Attempt autoencoder if model is correct

~~- Start working on DGI~~