Comp Chem / Applied ML for Chemistry

* DeepChem

description => Deep learning models for drug discovery, quantum chemistry and life sciences.

Applications: chemistry, Biology, Material-science, life-science, drug-discovery

Documentation: https://deepchem.readthedocs.io/en/latest (website https://deepchem.io/)

Source Code: https://github.com/deepchem/deepchem

Install requirements (Libraries): Joblib, numpy, pandas, scikit-learn, scipy, rdkit

Relevant article: http://arxiv.org/abs/1703.00564

* Predictive modeling by machine learning [Cheminformatics]

Quantitative Structure-Activity/Property Relationship (QSAR/QSPR)

(85) eXtreme Gradient Boosting. https://github.com/dmlc/xgboost, Accessed: 2017-10-18.

RDKit ⇒ A useful python library when dealing with chemical data

```
pip install rdkit
from rdkit import Chem
from rdkit.Chem import Draw
from rdkit.Chem import Descriptors
from rdkit.Chem.Draw import IPythonConsole
from rdkit.Chem import AllChem
from rdkit import DataStructs
import numpy as np
```

Importing structures ①

```
mol = Chem.MolFromSmiles('CNCCC')
mol
      img = Draw.MolToImage(mol)
OR
      img
```

Structure Database.
www.cheminfo.org
www.cheminfo.org/Chemistry/Name_to_structure/

MolFromFASTA	MolFromMolBlock
MolFromHELM	MolFromMolFile
MolFromInchi	MolFromPDBBlock
MolFromMol2Block	MolFromPDBFil
MolFromMol2File	MolFromPNGFile
MolFromPNGString	MolFromTPLBlock
MolFromSequence	MolFromTPLFile
MolFromSmarts	MolFromXYZBlo
MolFromSmiles	MolFromXYZFile

Computing properties ②

If you have the molecule, then you can compute various interesting things

```
mw = Descriptors.MolWt(mol)
mw
```

Looping over list of structures ③

glycine
phenylalanine
histidine
cysteine

```
smiles_list = [' ', ' ', ...]
mol_list = []
for smiles in smiles_list:
    mol = Chem.MolFromSmiles(smiles)
    mol_list.append(mol)
img = Draw.MolsToGridImage(mol_list, molsPerRow = len(mol_list))
```

~~Developing Neural Network Force Fields for Aqueous Graphene-Oxide Interfaces at Different Oxidation Levels.~~

Developing Atomistic ML force Fields for HEA
python, atomic simulation environment (ase), high performance computing (hpc)

Searching for pattern / Sub-structure search

```
pattern = Chem.MolFromSmiles('S')               sulphur element present
pattern = Chem.MolFromSmiles('C(=O)O')          Carboxyl group present
pattern = Chem.MolFromSmiles('CC(N)C')
pattern = Chem.MolFromSmarts('[r]')             Search for a ring structure
pattern = Chem.MolFromSmarts('[r5]')            Search for 5-membered ring sub structure
for mol in mol_list:
    print(mol.HasSubstructMatch(pattern))
```

Fingerprints and Tanimoto similarity

```
fp = AllChem.GetMorganFingerprintAsBitVect(glycine, 2, nBits=1024)
DataStructs.ConvertToNumpyArray(fp, fp_arr)
```

2 → number of bonds you want to go up to

A long list of zeros and 1 (can be converted to a numpy array)
The presence of 1 specifies whether a certain substructure subgroup is there or not.

```
np.nonzero(fp_arr)
```

→ To see the nonzero element indices

```
prints = [(glycine, x, bi) for x in fp.GetOnBits()]
Draw.DrawMorganBits(prints, molsPerRow=4, legends=[str(x) for x in fp.GetOnBits()])
```

* Butina Clustering [rdkit.ML.Cluster.Butina] → Cluster by Tanimoto Similarity

* Tanimoto Similarity: Method to rank how similar compounds are based on common figerprint bits.

$$T(A,B) = \frac{N_c}{N_A + N_B - N_c}$$

N_A = "on" features in Structure A

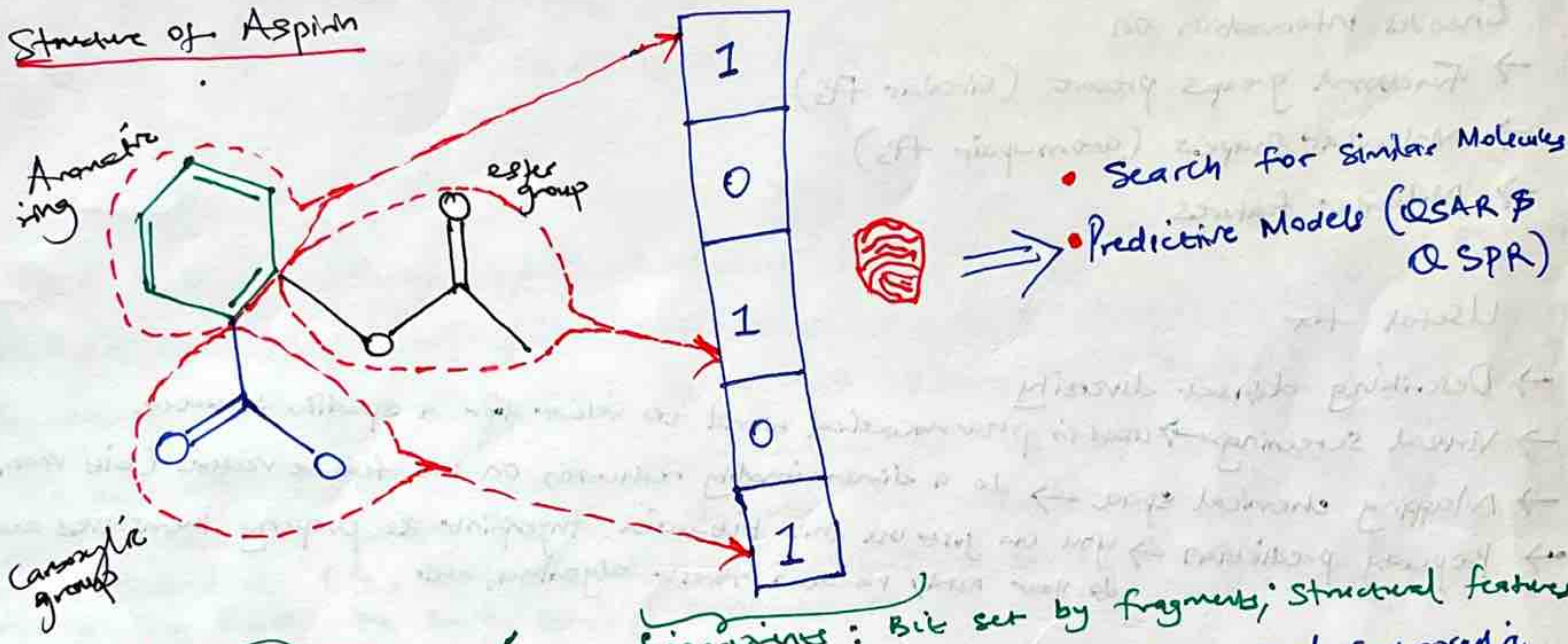
N_B = "on" features in Structure B

N_c = "on" features in both structure A and B

Molecular Fingerprints: Clashing & Clustering

Molecular fingerprints are a type of molecular descriptors that encode molecular features/fragments of a molecule in the form of a binary digit (0 or 1). So, we can use molecular ~~Fingerprints~~ fingerprints to perform computations. A bit is 0 or 1 if a certain fragment is found in a molecule structure.

Structure of Aspirin



Aromatic ring
ester group
carboxylic group
1
0
1
0
1
Molecular Structure
fingerprints: Bit set by fragments, structural features
• Search for Similar Molecules
• Predictive Models (QSAR & QSPR)

The bit is ON for ester functional group, aromatic ring and carboxylic functional group. The ON bits represents the structural features present.

Application of fingerprints:

⇒ Structure, substructure, and similarity search for virtual ligand screening from compound database.

⇒ Machine learning — Quantitative Structure-Activity Relationship (QSAR) and Quantitative Structure-Property Relationship (QSPR).

Some common fingerprints available in RDKit:

→ Molecular Access System Keys or MACCS-Keys
→ Avalon fingerprint
→ Atom-pair fingerprint
→ Topological-Torsions fingerprint
→ Morgan fingerprint or Circular fingerprint
→ RDKit fingerprint

A molecular fingerprint gives us a unique encoding of a molecule. It's a representation of a molecule as a series of zeos and ones (0's and 1's). It's basically a one-hot encoding of a molecule. It is done in different ways. There are different algorithms and different types of fingerprints. Morgan or Circular fingerprints is commonly used. <u>If</u> a specific functional group is present or some pattern is present, that gets encoded in bit vector as a 1.

A method to represent a molecule as a series of 0s and 1s

Encodes information on

→ Functional groups present (Circular FPs)

→ Molecular shapes (atom-pair FPs)

→ Molecular features

Useful for

→ Describing dataset diversity

→ Virtual screening → used in pharmaceutical world to select for a specific structure

→ Mapping chemical space → do a dimensionality reduction on the feature vectors (bit vecto

→ Property prediction → you can just use this bit-wise fingerprint as property themselves and do your NN, random forest algorithm, etc.

Limitations

→ Difficult to return to original structure

→ Interpretability

Fingerprints can be different length eg 1024 bits, 2048 bits

Bit Clashing For most fingerprints, the length of your fingerprint might be 1024 bit vector. The algorithm will generate your vector of such length. Sometimes if your is not long enough, multiple substructures get encoded int the same position in your bit vector (bit clashing). When multiple substructu are represented at the same position in the vector (bit collisions). We optimally want to basically have each bit (each position) in that vector, be it's on unique substructure – in that way we can tell differences between them and the might contribute differently to whatever you're trying to predict

Computing Fingerprints | Avoiding bit clashing | Using in a Statistical models

FP Generation
Bit Clashing
FP clustering for property prediction

```
mol1 = Chem.MolFromSmiles('...')
mol2 = Chem.MolFromSmiles('...')
Draw.MolsToGridImage([mol1, mol2], subImgSize=(400,400))
```

```
info1, info2 = {}, {}
fp1 = AllChem.GetMorganFingerPrintAsBitVect(mol1, 3, nBits=4096, bitInfo=info1, useFeatures=True)
fp2 = AllChem.GetMorganFingerPrintAsBitVect(mol2, 3, nBits=4096, bitInfo=info2, useFeatures=True)
```

generating a morgan fingerprint as a bit vector of 4096 bits.
This can be converted to a numpy array of 1's and 0's.
We can also see which functional groups get encoded as 1's and which are encoded as 0's. Molecules with the same "on" index in the fingerprint vector means they have the same/similar substructure.

```
arr = np.zeros((0,))
Chem.DataStructs.ConvertToNumpyArray(fp1, arr)
arr[0:100]
```

```
Draw.DrawMorganBit(mol1, 2136, info1, useSVG=True)
Draw.DrawMorganBit(mol2, 2136, info2, useSVG=True)
```

Fingerprints Clashing

With some of these fingerprints embedding, maybe we don't have enough bits for all of the structural diversity to be represented. One way to manage this is by increasing the number of bits that we have.

Check how many bits you have in use, across all of our fingerprints

It looks like our maximum number of occupied bits plateaus at 163, if we want maximal diversity in our fingerprints with no overlapping.

Fingerprints – Based Clustering

Butina, D. J. Chem Info. Comput. Sci. 1999, 39, 747–750.

DOI: 10.1021/ci9803381