Data Mining: Final Project

Tuesday 20th April, 2021

For the project you are going to work in groups of 2.

1 Aim

The aim of this project is to predict molecular properties. You are going to work with molecular data and you have to predict 4 molecular properties, the Octanol-water partition coefficient (logP), the number of rotatable bonds (RBN), the molecular weight (MW) and the number of the rings (RN).

2 Data Description

You are going to work using a subset the QM9 dataset. The original QM9 dataset (Ramakrishnan et al., 2014), contains $134k$ stable small organic molecules with up to 9 heavy atoms.

The data consists of the following files:

1. QM9.txt
2. properties_QM9.npz

2.1 Input Data

In QM9.txt the molecules are represented as SMILES (Simplified molecular-input line-entry system). SMILES strings proposed by Weininger (1988), are non-unique representations which encode the molecular graph into a sequence of ASCII characters using a depth-first graph traversal.

2.2 Properties Description

For this project we ask you to predict 4 molecular properties, the Octanol-water partition coefficient (logP), the number of rotatable bonds (RBN), the molecular weight (MW) and the number of the rings (RN). File properties_QM9.npz contains these 4 properties for the QM9.txt data.

1. logP: represents a measure of the tendency of a compound to move from the aqueous phase into lipids
2. Number of rotatable bonds (RBN): the number of bonds which allow free rotation around themselves
3. Molecular weight (MW): the weight of a molecule based on the atomic masses of all atoms in the molecule
4. Number of the rings (RN): the number of connected set of atoms and bonds in which every atom and bond is a member of a cycle

3 Model

Before starting your analysis it is very important to understand the data and the objective of this project.

You are free to chose the representation of the data that you are going to use. If you want to work with one-hot-encoding data (see project.pdf for details) you can find the code to convert a single smile string to a one-hot encoding in smile_to_hot() function at utils.py. You can even use as input data different properties and descriptors (number of bonds, number of atoms, number of C, number of O e.t.c.) that you can extract from your data manually or using RDKit. (The properties that we mention are random examples, there are much more properties and descriptors for the molecules and you have to think what it makes sense for you to use).

You are free to try different approaches and models, you can use ready libraries for your algorithm. The main focus is to see the model you have come up but also the other approaches that you tried. You have to understand deeply all the algorithms that you have tried.
Using load_data() function in utils.py split the data and the corresponding properties into train and test. Use **cross-validation** to select the best model based on **statistical significant test**. Only for the best model you will use the test data (of course you have to include in your report and present all the models that you have tried). **It is mandatory to use three or more different algorithms in addition to a baseline**[1].

The final model we would expect is a model that can work on universal data, which means it can give a reasonable prediction on different molecular datasets.

[1]The baseline is a naive baseline algorithm, providing context on just how good a given method actually is.

4 Report

You have to submit a formal report . Your report has to include the approaches that you followed and main results not only for your final model but for all the models you have tried. We want a full picture of what exactly you have done and how. You should also discuss the different performances you have with your methods and explain why these work or not. What is important is to show us that you have a good understanding of the problem and of how to model it, what are the problems you encountered and how you solved them.

Note that this is an open project, you can try many different approaches as long as they make sense to get the best performance (creative ideas are always welcome).

5 Final submission

For your final submission you have to submit on Moodle a folder named using your names (ex. NAME1_NAME2_DM_project) which should include your code (all the scripts), the dataset and your report (the report should also be saved using your names: NAME1_NAME2_DM_project.pdf). If the size of your submission is big you can upload your submission on SWITCH drive and put on Moodle the shared link.

6 Installation Instructions

Below you will find some instructions about how to install some packages that required for the project through Conda.

Conda is a package manager that works on all the popular operating systems. If you do not already have it installed (e.g. through Anaconda) you can install it via Miniconda by following the instructions here – it doesn't matter which version of Python you pick at this stage. We can then setup the particular environment using the Conda yml file I put in the folder of the project.

Assuming you have Conda installed, install the environment by:

1. conda env create -f DM_project_env.yml
2. conda activate DM_project
3. And then finally check it worked correctly by running $ conda env list .

To **activate your environment** type: conda activate DM_project and to deactivate it you should type: conda deactivate.

If you you need to **add an additional package** to your environment you can follow the instruction [here.

7 APPENDIX

7.1 RDKit

RDKit is an open source Cheminformatics Software. It is a collection of cheminformatics and machine learning tools written in C++ and Python. It allows to work with many representations of chemical data and has a power to extract almost each chemical descriptor from the data you have.

Some examples of using RDKit in order to visualize molecules and to extract different molecular descriptors you can find in smiles-rdkit.ipynb file. For more Informations about how to use RDKit and examples, you can find in the RDKit documentation.

7.2 SMILES representations

A molecule can be considered as an undirected graph G with a set of edges E and set of vertices V. There are several ways to represent graphs for machine learning. The most popular way is the SMILES string representation. SMILES strings are a non-unique representation which encode the molecular graph into a sequence of ASCII characters using a depth-first graph traversal.

Atoms of chemical elements are represented by chemical symbols in capital letter, hydrogen is usually ignored. Single bonds are not displayed; for double, triple and quadruple bonds we shall use '=', '#', '$' respectively. Atoms that are bonded must stand nearby. Ring structures are written by breaking each ring at an arbitrary point (although some choices will lead to a more legible SMILES than others) to make a 'straight non-ring' structure (as if it wasn't a ring) and adding numerical ring closure labels to show connectivity between non-adjacent atoms. Aromacity is commonly illustrated by writing the constituent B, C, N, O, P and S atoms in lower-case forms b, c, n, o, p and s, respectively.

SMILES are typically first converted into a one-hot based representation. The representation of a molecule through a SMILE string is not unique and the non-uniqueness of SMILES arises from a fundamental ambiguity about which atom to start the SMILES string construction. However, it is possible to transform a smile into canonical form (using specific tools such as RDKit).

7.3 One Hot Encoding

One-hot encoding is a sparse way of representing data in a binary string in which only a single bit can be 1, while all others are 0. In the case of molecular graphs represented as a string, one-hot-encoding consists of encoding each character of the string with a binary vector.

For one-hot-encoding it is necessary to fix or create a dictionary containing all the characters present in the dataset in order to fix a binding between integer and character.
If you want to work with one-hot-encoding data you can find the code to convert a single smile string to a one-hot encoding in smile_to_hot() function at utils.py