Review - paper #48
On the use of RNN in construction of aimed molecules for inverse QSAR problem

Lupașcu Marian
Syntactic Modeling of Biological Systems
University of Bucharest

March 9, 2020

Context

The authors propose a solution of the inverse QSAR problem (that is, determining a molecular descriptor that can recover the structure of a molecule that has a desired activity or property - in paper, the molecular mass and the activity value of the molecule [Ki]) with a Recurrent Neural Network (RNN)).

The RNN implemented in the paper has as output SMILES sequences (linear notation for describing the structure of a molecule of a chemical substance) that describe compounds with the given properties at input. An approach with RNNs makes the most sense of all approaches with Deep Learning techniques (NN, CNN, Syamese Net, etc.) because the output is variable, molecules with equal masses (in paper) can be obtained but different number of atoms (eg N [C @ H] (Cc1c [nH] c2ccccc12) C (= O) O (Tryptophan) and C = c2c1ccccc1c (= C) c3ccccc23 (Anthracene, 9,10-bis-methylene) have the same mass (204) but different structures and different lengths in the SMILES string).

Suggestions

- Use LSTM cells with memory and then say that they consume a lot of memory and that learning 10 epochs takes 3 months. Have you tried GRU cells? They are faster, memory-efficient, and have similar performance to LSTM [Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling](https://arxiv.org/pdf/1412.3555v1.pdf). In addition, both are used to avoid vanishing and exploding gradients.
- Use Deep Learning techniques (RNNs) on insufficient data, 2346 chemical compounds. Did you increase your data (data augmentation), that you don't specify anywhere?

- Do not give any details about the architecture of your network but explanation how you implemented the LSTM cell and how to divide the data by train and test.
- You used PyTorch, explained what loss function you used but did not give any details about the optimizer used (the learning algorithm you used - Adam, SGD, Adagrad, etc)
- The validation step is missing during the learning so that you are sure to avoid overfitting. I understand you missed it because of the lack of data but I think it is necessary (and it was possible to solve the lack of data with an increase before learning).
- It is not clear to me why working with batches of 20 compounds and not the power of 2 (16, 32 or 64 for example) is proven to be faster.
- It is not clear what is revealed that you set the number of times to 100 then turn off the algorithm to 10 epochs because it takes too long. You could say from the first that you ran 10 epochs and so on.
- Why 256 the number of hidden layers, is a hyperparameter that must be tuned and obtained on the experimental one. It's unclear why 256. It's unclear why Dropout probably 1/4 why not 1/2? And here we have a hyperparameter that needs to be tuned, and on paper you say nothing about it.
- In Figure 3 a histogram is more relevant than your figure.
- Do not specify anything about the accuracy after the 10 epochs.
- You didn't compare yourself to other results that solved the same thing.

Overall evaluation: -1

Reviewer's confidence: 2