Use of Reinforcement Learning in de novo Drug Discovery

El Mehdi Oudaoud
Montreal, Quebec
oudaoud.mehdi@gmail.com

Abstract

AI based Drug Discovery has become one of the most promissing fields in Digital Pathology. It is indeed much more efficient in both time and ressources than the traditionnal lab-related Drug Discovery. The results of this new approach can be very useful not only for medical providers but also for patients. In fact, the industry is now becoming a multi-billion dollar industry (50$ estimated for 2030) with different pharmaceutical companies investing more and more in computer based molecule discovery. Most approaches focus on the use of Graph Neural Networks (GNN) to extract meaninful embeddings from the molecules. However, another method is the use of Reinforcement Learning, especially using a Generator Predictor architecture, in which the Generator, as its name indicates, generates molecules, and the Predictor predicts its properties. The role of Reinforcement Learning is to link both the models. In this paper, I will discuss how to use Reinforcement Learning in the generation of molecules that have a high pIC-50 which is the negative logarithm of half maximal inhibitory concentration value for the Janus Kinase 2 disease.

1 Reinforcement Learning for Drug Design :

In order to create a model that generates molecules that answer to a certain criteria, we need to have a encoder decoder architecture, that will allow us to create the molecules. After doing so, the molecules need to have a certain property that should be guaranted with the use of the Predictor.

1.1 Generator Architecture:

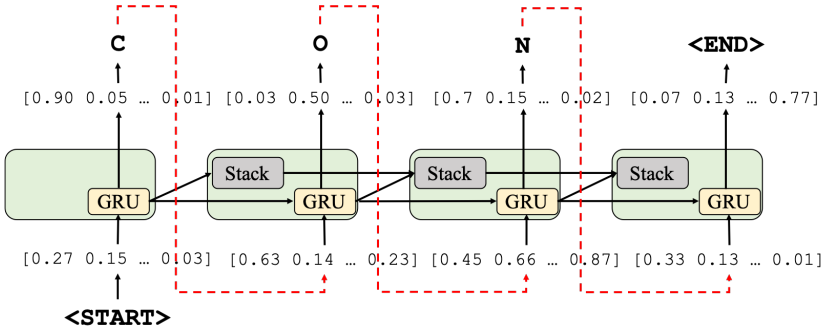

Figure 1: Architecture of GRU (RNN) [1]

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

The Generator Model **G** (Figure 1) that has been used is a RNN GRU Model in which the Molecules are presented in their SMILES[2] representation, as shown in Figure 1. Hence, the model here is the same as the one used for NLP tasks. We train the model to predict the next letter of a SMILES molecule, and once the model is trained, it would be able to generate SMILES representations of actual molecules.

As presented in the paper [1], the model **G** is a Stacked RNN GRU. In fact, the Regular RNN architecture is not very efficient for context-free languages such as the SMILES representation [2]. The stack memory is actually essential to capture better **long term dependencies**.

1.1.1 The DataSet :

The Dataset used here is the one found in *Chembl*[3] library. It contains over 1.5 million molecules. but since it is very memory consuming, I only restricted my training on a random set of 500,000 molecules.

For embedding the molecules, we use a fixed dictionnary of tokens (45 possible ones), and each letter is refered to as its postions in the dictionnary. Hence, every molecule is embedded into a vector of the same size, with integers from 1 to 45.

1.1.2 Training and Results :

Since the dataset is quite large, and we are training an encoder architecture, one has to use a large number of epochs (10,000 in this case). the learning rate used here is $\alpha = 0.01$, which was the best empirical choice that minimizes the loss without risking an overfitting of the model.

Speaking of overfitting, there are few ways to test the effectiveness of the Generator model **G**. The best one in our case would be the effectiveniss rate E_R

$$E_R = \frac{R_{mol}}{T_{mol}}$$

where R_{mol} is **the number of real generated molecules** and T_{mol} is **The total number of generated molecules**. In my case, this pourcentage was around 27%. One has to keep in mind that there are a very large number of possible combinations (way above 1.5 Million)

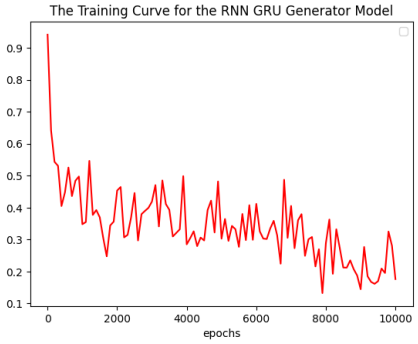

Figure 2: Tuned Learning curve of **G**

Figure 3: Some molecules generated by **G**

1.2 Predictor Architecture:

Here the dataset used for The predictor by the paper [1], is not available in public. I have used a smaller dataset [4] that contains 2000 molecules with their respective **pIC-50** values.

So, since the dataset is quite small LR algorithm would be much more effecient, which indeed was the case from what shows **Learning and Testing Cruve** as shown in Figure 4.

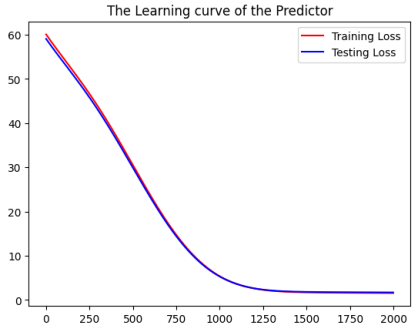

Figure 4: Learning curve of **P**

1.3 Reinforcement Learning:

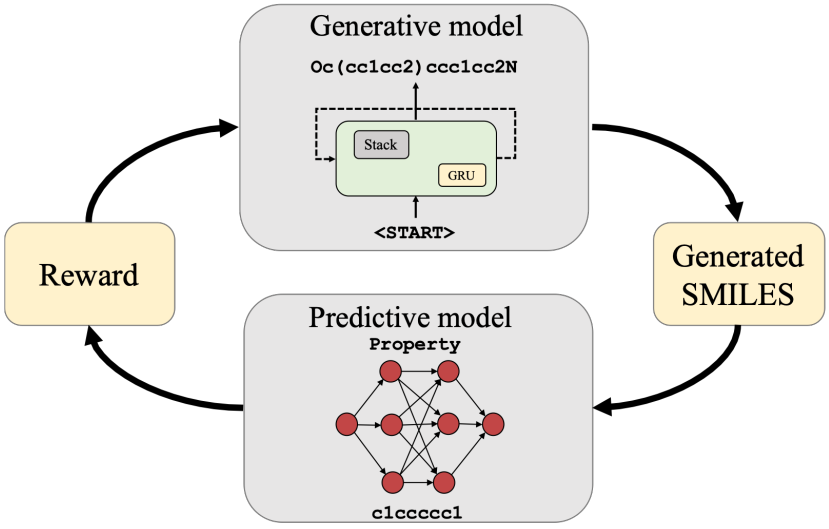

Figure 5: Architecture of the entire Model

Once both the models **(G)** and **(P)** properly trained, we can first test the pIC-50 value of a batch of generated molecules via **(G)**, the figure 5, shows that the mean is centered around 4.23.

Now we need to link the **(G)** and **(P)** using Reifocement Learning, as shown in Figure 5.

First, we have to define :

- The Generator **(G)** as the agent.
- The states s that are the molecules generated by **(G)**
- The actions a as the tokens used in the SMILES writing.
- the reward $R(s) := exp(\frac{P(s)}{3})$ where $P(s)$ is the property predicted by the Predictor **(P)** for the state s

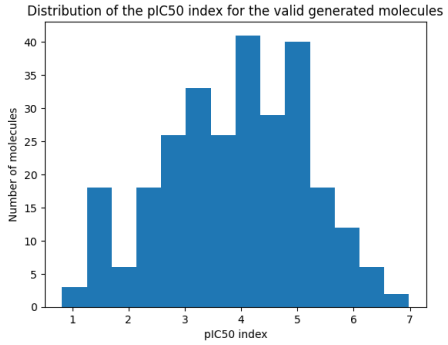

Figure 6: pIC-50 of the first batch of generated molecules

The last and most important element to finish the definition of the Reinforcement Learning Algorithm is **the policy gradient**. In the case of the study we use a batch of 10 molecules each time.

$L = \frac{1}{n} \sum_{i=1}^{10} \sum_{j=1}^{\text{length } (mol_i)} R_i \cdot \gamma^i \cdot \log p\left(mol_i \mid mol_0 \ldots mol_{i-1}\theta\right)$ [5]

where R is the reward and $p\left(mol_i \mid mol_0 \ldots mol_{i-1}\theta\right)$ is the distribution (Log_Softmax) from the Generator **(G)**.

With the use of the backrpopagation in the*pytorch library*, minimizing L, will change the weights of the Generator Model (The agent) .

1.3.1 The Results:

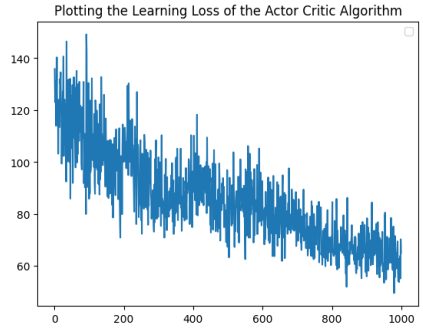

Figure 7: Learning curve of the RL

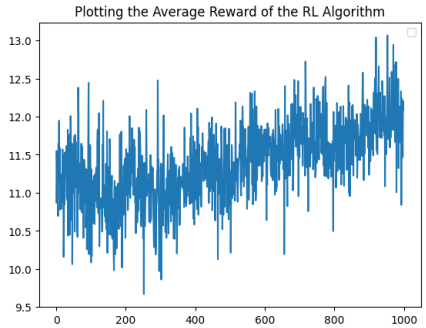

Figure 8: Average Reward Curve of the RL

Here we can notice that the loss is decreasing and that the reward is increasing, over 1000 epochs.

Now after modifying the weights of the Generator **(G)**, we recreate molecule using **(G)**, that will later on be passed to the predictor **(P)** to predict their pIC-50 value. The Figure 8 compares the two histogramms the initial one shown in Figure 6, and the new one after using RL.

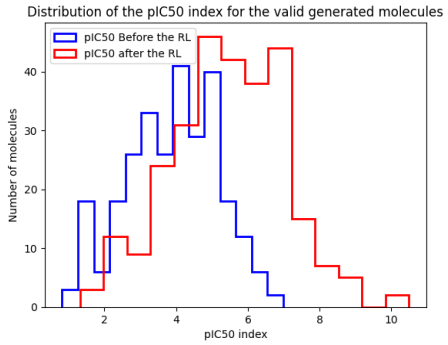

Figure 9: Comparing the Distibution of pIC-50 values between the use Generator (G) before the RL and the use of Generator (G) after the RL

In fact, one can notice that the new mean is now around 5.8, also the scalar E_R has also slightly improved to 29%

2 Thoughts and Opening

In fact, the E_R that I have found can be largey improved to a 75% in some cases. The main disavantage in my study is the restriction of the number of molecules in the dataset to a third of the initial number due to memory restrictions. Moreover, improving the score of the E_R is also going to increase the mean of the pIC-50 up tp 7 - 8. which is a really good value for a molecule.

Also, one has to keep in mind that Reinforcement Learning is rarely used in AI-Drug discovery, but surprisingly so, the paper[1], on which I have based my study shows promising results.

One way of improving these scores furthermore, would be to replace the Generator (G) based on NLP analogy with another Generator using the graph characteristics of the molecule. However, such a replacement would need a much bigger model, since the graphs are much more large than the SMILES represntation.

References

[1] Mariya Popova et al. ,Deep reinforcement learning for de novo drug design.Sci. Adv.4,eaap7885(2018).DOI:10.1126/sciadv.aap7885

[2] Li C, Feng J, Liu S, Yao J. A Novel Molecular Representation Learning for Molecular Property Prediction with a Multiple SMILES-Based Augmentation. Comput Intell Neurosci. 2022 Jan 28;2022:8464452. doi: 10.1155/2022/8464452. PMID: 35178082; PMCID: PMC8843876

[3] ChemBL database : https://chembl.gitbook.io/chembl-interface-documentation/downloads

[4] https://github.com/isayev/ReLeaSE/tree/b1bb40031ed4f9f5741bd006e3d3c96fae48bc93

[5] https://medium.datadriveninvestor.com/drug-design-made-fun-using-reinforcement-learning-212a4f867f33