Exploring the chemical space without bias: data-free molecule generation with DQN and SELFIES

Théophile Gaudin
IBM Research Zürich, Switzerland
Department of Computer Science, University of Toronto, Canada
tga@zurich.ibm.com

AkshatKumar Nigam
Department of Computer Science, University of Toronto, Canada
akshat.nigam@mail.utoronto.ca

Alan Aspuru-Guzik
Department of Chemistry, University of Toronto, Canada
Department of Computer Science, University of Toronto, Canada
Vector Institute for Artificial Intelligence, Toronto, Canada
Canadian Institute for Advanced Research (CIFAR) Senior Fellow, Toronto Canada
alan@aspuru.com

Abstract

We propose a method for de novo molecule design that is free of human- and data-biases. We used a deep q-network [1] to train an agent to learn how to sequentially generate a molecule with SELFIES representation [2]. Using SELFIES instead of SMILES [3] frees the agent from having to learn a specific grammar, as even completely random SELFIES produce valid molecule. The network is rewarded according to the output molecule's value of the partition coefficient, logP, which is considered as the *Drosophila* of computational chemistry. After less than an hour of training on a single GPU, the agent was successfully producing molecules with the targeted property value. Bias-free design could lead to unrestricted exploration of the entire chemical space, and exploring regions that humans (and human-generated data) might have entirely overlooked so far. This property makes our algorithm directly applicable to other design questions in the physical sciences.

1 Introduction

Exploring the chemical space of compounds is a task that recently gained traction due to an increase in the amount of data available and to the advances in machine learning and AI. So far, most approaches [4–11] focus on using the available data to generate new molecules. As no database can encompass the entirety of the chemical space, while these methods have contributed to great advancements in chemical compounds generation, they are also biased by the data available to them.

It has already been shown that blind application of machine learning can amplify systemic mistakes [12]. Because data is limited by our current discoveries, it is restrained and biased by our current knowledge. In brief, molecule generation that relies on data is limited and biased to a particular area of the chemical space.

In this paper, we propose a method that does not rely on pre-collected static dataset and is, therefore, able to explore the chemical space in an unbiased way. Our method uses deep q-network (DQN) [1]

Second Workshop on Machine Learning and the Physical Sciences (NeurIPS 2019), Vancouver, Canada.

combined with a representation of molecules called SELFIES [2]. The combination of SELFIES and reinforcement learning is particularly interesting as SELFIES strings are always valid. The use of SELFIES instead of SMILES [3] also frees the network from having to learn any grammar. The reward is calculated according to targeted value property. In our experiment, we rewarded the molecule made by the agent according to a difference between the partition coefficient, logP, of the molecule and a targeted value.

2 Previous work

The application of machine learning to molecule generation has been amply researched in the past couple of years. However, to the best of our knowledge, only Zhou et al [13] proposed a method that is data-free. They suggested to use DQN to build molecule, although they incorporated chemistry domain knowledge in their model, which can bias the molecule generation.

We can categorize other previous approaches by the representation they are using: the graph-based representations [8–10] and the SMILES-based representations [4–7, 11]. Intuitively, it seems more natural to treat molecules as graphs. However, none of the methods using graphs are able to integrate the stereochemistry and the chirality within their representation.

Contrarily, SMILES-based approaches are able to encode both stereochemistry and chirality. Nevertheless, generating valid SMILES requires the algorithm to learn the SMILES' grammar first. To minimize this problem, Kusner et al proposed to use parse-trees [7]. More recently Krenn et al [2] proposed a new representation called SELFIES that is able to directly translate into a valid molecule. Not only this new representation is more robust than the SMILES, but it also improves the performance of the model since it doesn't have to learn any grammar.

3 Proposed method

Our method consists of casting the molecule generation process as a Markov decision process (MDP) [14] in order to use deep reinforcement learning. A Markov decision process is a decision-making process where an agent sequentially interacts with an environment $\mathcal{E}$. At each time step, the agent selects an action a_t and the environment respond by presenting a new state s_t. Additionally to this state, the environement also emits a reward r_t that quantifies how good the action taken was. The goal of the agent is to maximize the reward over time.

Q-learning is a value-based method [15]. Instead of directly trying to find the optimal policy π, it estimates the optimal value function $Q^*(s, a)$ which can be defined as

$$Q^*_\pi(s, a) = \max_\pi \mathbb{E}\left[\sum_{k=t}^{T} \gamma^{k-t} r_t | s_t = s, a_t = a, \pi\right].$$

Using Bellman equation, and assuming the optimal value $Q^*(s', a')$ for the next step is known, we can rewrite the previous equation as

$$Q^*_\pi(s, a) = \mathbb{E}_{s' \sim \mathcal{E}}\left[r + \gamma \max_{a'} Q^*(s', a') | s, a\right].$$

However, it is not possible to directly compute a tabular version of the Q-function due to the size of the action and states space. Fortunately, $Q^*(s, a)$ can be approximated using a parametrized value function $Q(s, a; \theta)$ where θ refers to the parameters of the model that defines the Q-values. In DQN [1], a neural network is used to estimate this value function. This network is trained by minimising the following loss:

$$L_i(\theta_i) = \mathbb{E}_{s,a \sim \rho(s,a)}[(y_i - Q(s, a; \theta_i))^2]$$

where $y_i = \mathbb{E}_{s' \sim \mathcal{E}}\left[r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) | s, a\right]$ and $\rho(s, a)$ is a probability distribution over sequence of states s and actions s. This loss can be differentiated according to the model parameters θ hence stochastic gradient descent can be used conveniently to optimize the network.

To prevent instabilities during the training, Mnih et al. [1] replaced $Q(s', a'; \theta_{i-1})$ by $Q(s', a'; \theta_j)$ where θ_j is only updated to θ_i every τ iterations.

3.1 Environment

Our environment emits observations consisting of a SELFIES string. At the beginning of an episode, the observation is initialized to be a single symbol corresponding to a carbon atom `[C]`. To interact with the environment, the agent has to provide an index and an action. The index corresponds to the position where the action will take place in the string. The action can either be to change the selected symbol to another one or to insert a new symbol at the position indicated by the index.

The environment uses the following SELFIES symbols:

`[C]`, `[=C]`, `[#C]`, `[O]`, `[=O]`, `[N]`, `[=N]`, `[#N]`, `[F]`, `[epsilon]`, `[Ring1]`, `[Ring2]`, `[Branch1_1]`, `[Branch1_2]`, `[Branch1_3]`, `[Branch2_1]`, `[Branch2_2]`, `[Branch2_3]`

In total, there are 36 possible actions. An episode is finished after the agent took an arbitrary number of step, which we set to 64.

3.2 Model

The model architecture corresponds to the encoder of a Transformer [16]. This architecture has now become mainstream, thus we will omit an exhaustive model description here. Our implementation uses PyTorch [17] and is largely based on a blog post [18]. The notable difference between the original model and ours is the size: we used only 2 layers of attention of size 128, with 4 attention heads and an inner layer of the point-wise feedforward of 256.

The Transformer encoder outputs a representation for each token of the input. During training, a token of the representation is randomly selected and an additional linear transformation is performed giving a single output for each action.

3.3 Training

We trained the agent for 1500 episodes on a single GPU with a learning rate of 0.0001. We used experience replay with a memory large enough to contain the entirety of the training sequence and an ϵ-greedy policy with ϵ decaying exponentially between 1 and 0.001. We sampled batch from the memory of size 64. The reward discount γ was set to 0.4. We updated the target network every 10 episodes. Training time is notably short and generally falls to about one hour.

3.4 Reward function

The reward is a function of the distance d between the target logP p_t and the current logP p. We used the partition coefficient logP, connected to the solvability of the molecule, as it can be easily computed for any molecule [19] using RDKit [20]. The reward was given at each time step.

4 Results and discussion

Figure 1 shows that the agent is able to learn how to generate molecules with a particular target of logP. In this particular example, the target logP was set to 5. Similar results were obtained with targeted logP values ranging between -5 to 20.

The agent quickly learned to produce plausible structures for molecules as shown on the right of Figure 1. Moreover, we noticed the generation of rings occurred naturally in the molecule generated, which is not the case of other approaches [4] that have an additional term in their reward function accounting rings.

This approach opens up to a large number of possible experiments: we could constraints part of the substructure and optimize the rest of the structure to reach a particular value of logP. We could also have the model interacting with a chemist that could choose parts of the index and action to be taken. By design, the agent and environment are flexible enough to allow that.

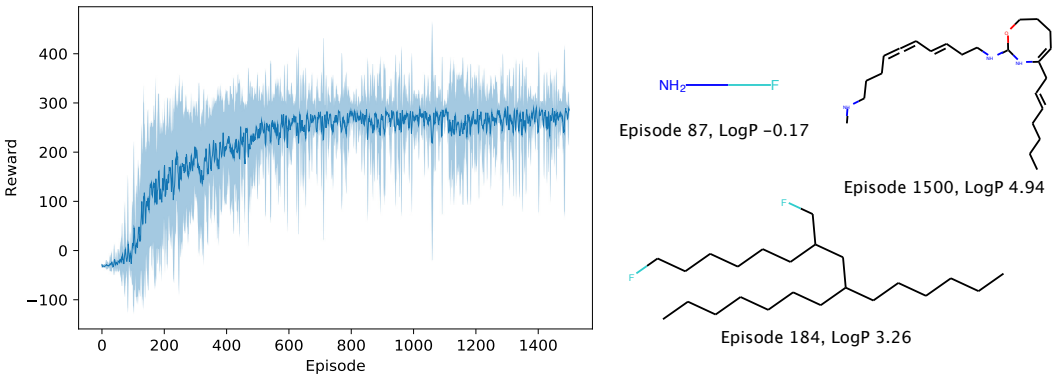

Figure 1: Reward per episode (left) and depiction of some molecules generated (right). The results are averaged over 5 runs. The line represents the mean value of the runs and the shaded area is the standard deviation. A Savitzky-Golay filter has been applied to both mean and standard deviation with a window-length of 7.

5 Conclusion

In this paper, we present a new bias and data-free method that trains an agent to generate molecules that have specific property value. As we do not use any data, it is not bound to any domain in specific and can be used for any kind of property that we can compute. Moreover, our method is fast and uses little computational resources: it produces molecules with targeted property value within an hour using a single GPU.

To explore an even wider area of the chemical space, one could combine the current algorithm with curiosity or other *intrinsic rewards* [21, 22] to encourage the agent to efficiently explore the chemical space.

References

[1] Mnih, V. *et al.* Playing Atari with Deep Reinforcement Learning. *arXiv:1312.5602 [cs]* (2013). URL `http://arxiv.org/abs/1312.5602`. ArXiv: 1312.5602.

[2] Krenn, M., Häse, F., Nigam, A., Friederich, P. & Aspuru-Guzik, A. SELFIES: a robust representation of semantically constrained graphs with an example application in chemistry. *arXiv:1905.13741 [physics, physics:quant-ph, stat]* (2019). URL `http://arxiv.org/abs/1905.13741`. ArXiv: 1905.13741.

[3] Weininger, D. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *Journal of Chemical Information and Computer Sciences* **28**, 31–36 (1988). URL `https://pubs.acs.org/doi/abs/10.1021/ci00057a005`.

[4] Popova, M., Isayev, O. & Tropsha, A. Deep reinforcement learning for de novo drug design. *Science Advances* **4**, eaap7885 (2018). URL `https://advances.sciencemag.org/content/4/7/eaap7885`.

[5] Sanchez-Lengeling, B., Outeiral, C., Guimaraes, G. L. & Aspuru-Guzik, A. Optimizing distributions over molecular space. An Objective-Reinforced Generative Adversarial Network for Inverse-design Chemistry (ORGANIC) (2017). URL `https://chemrxiv.org/articles/ORGANIC_1_pdf/5309668`.

[6] Gómez-Bombarelli, R. *et al.* Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules. *ACS Central Science* **4**, 268–276 (2018). URL `https://pubs.acs.org/doi/10.1021/acscentsci.7b00572`.

[7] Kusner, M. J., Paige, B. & Hernández-Lobato, J. M. Grammar Variational Autoencoder. *arXiv:1703.01925 [stat]* (2017). URL `http://arxiv.org/abs/1703.01925`. ArXiv: 1703.01925.

[8] Jin, W., Barzilay, R. & Jaakkola, T. Junction Tree Variational Autoencoder for Molecular Graph Generation. In *arXiv:1802.04364 [cs, stat]* (2018). URL http://arxiv.org/abs/1802.04364. ArXiv: 1802.04364.

[9] De Cao, N. & Kipf, T. MolGAN: An implicit generative model for small molecular graphs. *arXiv:1805.11973 [cs, stat]* (2018). URL http://arxiv.org/abs/1805.11973. ArXiv: 1805.11973.

[10] You, J., Liu, B., Ying, Z., Pande, V. & Leskovec, J. Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation. 12 (2018).

[11] Ståhl, N., Falkman, G., Karlsson, A., Mathiason, G. & Boström, J. Deep Reinforcement Learning for Multiparameter Optimization in de novo Drug Design. *Journal of Chemical Information and Modeling* **59**, 3166–3176 (2019). URL https://doi.org/10.1021/acs.jcim.9b00325.

[12] Caliskan, A., Bryson, J. J. & Narayanan, A. Semantics derived automatically from language corpora contain human-like biases. *Science* **356**, 183–186 (2017). URL https://science.sciencemag.org/content/356/6334/183.

[13] Zhou, Z., Kearnes, S., Li, L., Zare, R. N. & Riley, P. Optimization of Molecules via Deep Reinforcement Learning. *Scientific Reports* **9**, 1–10 (2019). URL https://www.nature.com/articles/s41598-019-47148-x.

[14] BELLMAN, R. A Markovian Decision Process. *Journal of Mathematics and Mechanics* **6**, 679–684 (1957). URL https://www.jstor.org/stable/24900506.

[15] Watkins, C. *Learning from delayed rewards*. Ph.D. thesis, King's College, Cambridge (1989).

[16] Vaswani, A. *et al.* Attention is All you Need 11 (2017).

[17] Paszke, A. *et al.* Automatic differentiation in PyTorch (2017). URL https://openreview.net/forum?id=BJJsrmfCZ.

[18] Rush. The Annotated Transformer (2018). URL http://nlp.seas.harvard.edu/2018/04/03/attention.html.

[19] Wildman, S. A. & Crippen, G. M. Prediction of Physicochemical Parameters by Atomic Contributions. *Journal of Chemical Information and Computer Sciences* **39**, 868–873 (1999). URL https://doi.org/10.1021/ci9903071.

[20] Greg, L. RDKit. URL http://www.rdkit.org/.

[21] Pathak, D., Agrawal, P., Efros, A. A. & Darrell, T. Curiosity-Driven Exploration by Self-Supervised Prediction. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 488–489 (IEEE, Honolulu, HI, USA, 2017). URL http://ieeexplore.ieee.org/document/8014804/.

[22] Burda, Y. *et al.* Large-Scale Study of Curiosity-Driven Learning 15 (2019).