Curiosity in exploring chemical space: Intrinsic rewards for deep molecular reinforcement learning

Luca A. Thiede  
Department of Physics, University of Göttingen, Germany  
Department of Computer Science, University of Toronto, Canada  
luca.thiede@yahoo.com

Mario Krenn  
Chemical Physics Theory Group, Department of Chemistry, University of Toronto, Canada  
Department of Computer Science, University of Toronto, Canada  
Vector Institute for Artificial Intelligence, Toronto, Canada  
mario.krenn@utoronto.ca

Akshat Kumar Nigam  
Department of Computer Science, University of Toronto, Canada  
Chemical Physics Theory Group, Department of Chemistry, University of Toronto, Canada  
akshat.nigam@mail.utoronto.ca

Alán Aspuru-Guzik  
Chemical Physics Theory Group, 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) Lebovic Fellow, Toronto, Canada  
alan@aspuru.com

Abstract

Computer-aided design of molecules has the potential to disrupt the field of drug and material discovery. Machine learning, and deep learning, in particular, have been topics where the field has been developing at a rapid pace. Reinforcement learning is a particularly promising approach since it allows for molecular design without prior knowledge. However, the search space is vast and efficient exploration is desirable when using reinforcement learning agents. In this study, we propose an algorithm to aid efficient exploration. The algorithm is inspired by a concept known in the literature as curiosity. We show on three benchmarks that a curious agent finds better performing molecules. This indicates an exciting new research direction for reinforcement learning agents that can explore the chemical space out of their own motivation. This has the potential to eventually lead to unexpected new molecules that no human has thought about so far.

1 Introduction

The development of new drugs and functional materials is an important but expensive undertaking. To bring the cost down, the scientific community introduced computational screening methods that frame the discovery process as an optimization problem of desired properties over a large molecule database or chemical space. This is also known as the inverse molecular design problem [1, 2]. However, the search space is enormous [3, 4] rendering exhaustive search of all possible molecules
There are many different ways to train a policy. Throughout this paper, we will use a policy gradient. For both formulations the cumulative reward is the property of the final molecule (or as the reward at every time step can be formulated in two ways. Either as

\[ R_t := R(s_t, a_t) \]

for some property \( p \). The goal is to find a policy so that

\[ J(\theta) = \mathbb{E}_{s_0 \sim \mu_0} \left[ V^\pi(s_0) \right] \]

is maximized.

Other approaches like genetic algorithms or reinforcement learning (RL) allows designs potentially far away from any known data distribution. Instead of a dataset, only a reward function is needed, that measures how good a generated molecule is. However, due to the vast chemical space, efficient exploration is necessary.

Here we take inspiration from the field of RL for video games, where the use of a method called curiosity has led to impressive results without access to actual rewards from an environment. Curiosity falls under the wider category of intrinsic motivation techniques, which are loosely modeled after human curiosity. In this work, we propose for the first time the use of intrinsic motivation for molecular design and show that the more curious agents in our study perform better than their less curious counterparts on three commonly used benchmarks.

1.1 Reinforcement Learning basics

Reinforcement Learning is a technique used to find a policy \( \pi_\theta \) parameterized by \( \theta \) that maximizes state-action trajectories in an environment. Formally, the environment is described as a Markov decision process \( M = (S, A, T, \mu_0, \gamma, R, T) \). Here, \( S \) is the state space, \( A \) is the action space, \( T : S \times A \to S \) is the transition function, \( \mu_0 \) is the initial state distribution, \( \gamma \in (0, 1] \) is the discount factor. \( R : S \times A \to \mathbb{R} \) is the reward function. We define \( r_t := R(s_t, a_t) \) and \( T \) as the maximal length of an episode. For every policy \( \pi \), the expected reward is defined as the reward that an agent will collect when it is in a certain state

\[ V^\pi(s_t) = \mathbb{E}_{\pi} \left( \sum_{t=0}^{T} \gamma^t R(s_t, a_t) \right) \]

This quantity is also called the value of the state \( s_t \). Analogously we define the Q value of an action in a state as

\[ Q^\pi(s_t, a_t) = \mathbb{E}_{\pi} \left( \sum_{t=0}^{T} \gamma^t R(s_t, a_t) \right) \]

. The goal is to find a policy so that

\[ J(\theta) = \mathbb{E}_{s_0 \sim \mu_0} \left[ V^\pi(s_0) \right] \]

is maximized.

There are many different ways to train a policy. Throughout this paper, we will use a policy gradient method, in particular proximal policy optimization (PPO). We use the hyperparameter provided by the original paper which were tuned on atari games. The authors in conducted a thorough hyperparameter search and confirmed that these hyperparameters are also nearly optimal for molecular design and also other authors reported that they could not find better settings.

2 Reinforcement Learning for molecular design

For molecular design, we define the state \( s_t \) as the SELFIES string that is so far constructed. SELFIES is a 100% robust string-based representation of molecules. This is in contrast to SMILES or other string-based representations, which frequently produce semantically or syntactically invalid strings. The robustness of SELFIES is advantageous as our RL agent produces in every step a valid molecule. Thereby, the training doesn’t require any post-processing or filters, thus is simpler. The action \( a_t \) is the next character to be appended to the string. The molecule is finished either when the max number of steps is reached, which we set to 35 throughout our experiments, or the agents use the \([STOP]\) symbol.

For some property \( p \) that we wish to optimize, and by denoting the molecule at time step \( t \) as mol, the reward at every time step can be formulated in two ways. Either as

\[ r_t = \begin{cases} p(\text{mol}(T)), & \text{SELFIES string finished} \\ 0, & \text{otherwise} \end{cases} \]

or as

\[ r_t = p(\text{mol}(t)) - p(\text{mol}(t-1)) \]

For both formulations the cumulative reward is the property of the final molecule \( (\sum_{t=0}^{T} \gamma^t r_t = p(\text{mol}(T))) \) for \( \gamma = 1 \). The first formulation is sparse, but only needs one evaluation of the reward
The literature on reinforcement learning often distinguishes between intrinsic and extrinsic rewards. An extrinsic reward is anything that comes directly from the environment. Intrinsic rewards are any rewards that are generated by the agent itself. Pathak et. al [18] introduce an intrinsic reward called curiosity. Curiosity guides the exploration of an agent into regions of the state space, where it has not understood the effect of its actions on the environment. They introduced a separate neural network that tries to predict the next state the agent will be in given the action it took. Then, the total reward the agent gets is

$$r_{total}(t) = r_{extrinsic}(t) + \alpha r_{intrinsic}(t)$$  \hspace{1cm} (3)$$

where $r_{extrinsic}(t)$ is the normal reward provided by the environment, and $r_{intrinsic}(t)$ is the error of the prediction from the newly introduced neural network. The prediction network can also be seen as implicitly storing information in its weights about which areas of the state-action space have
already been visited, since the more often the agent is in a certain region of that space, the smaller the prediction error is going to be. By exploiting this information, the agent will continue exploring new regions and not get stuck in local optima. This situation is depicted in Figure 1.

4 Curiosity for molecular design

Our goal is to adapt the work described in the previous section to molecular design. For this, we use a prediction network that predicts the property of the next molecule and use the prediction error as the intrinsic reward (see Figure 1). In order to test different variations of this idea, we formulate it generally as:

\[ r_{\text{intrinsic}}(t) = \text{dist}\left(\hat{p}(\text{mol}(t, \theta), \eta), p(\text{mol}(t, \theta))\right) \]  

(4)

Here \(\hat{p}(\cdot, \eta)\) is the prediction network parameterized by \(\eta\) that tries to predict the real value of the considered property \(p\) of the molecule. \(\text{mol}(t, \theta)\) is the molecule the agent parameterized by \(\theta\) generates at time step \(t\) and \(\text{dist}(\cdot, \cdot)\) is a distance metric, for example \(L_1\) or \(L_2\). We also try an alternative formulation that we call “greedy curiosity”. It uses a mask function to be curious only in promising direction (see appendix A). For training the predictor network we consider two options: Either we update the predictor network each time the agent generates new molecules, or we collect them in a buffer and train on the whole buffer (see appendix B).

Unlike Pathak et. al. [18], we do not predict the next state. The reason is, that given the current state (the string so far), and the next action (the character to append), predicting the next state (the string so far with the new character appended) does not require to learn anything about the chemical space. Instead of storing information in the state-action space, our prediction network only remembers regions of the state space the agent visited.

We also consider a very simple alternative where we explicitly store the last \(N\) molecules into a buffer and calculate the average Tanimoto Similarity (TS) of the Morgan Fingerprints (MF) [25].

\[ r_{\text{intrinsic, alternative}}(t) = -\frac{1}{N} \sum_{i=0}^{N} \text{TS(MF(mol(t))), MF(mol_i))} \]  

(5)

This approach explicitly makes a choice of a particular distance metric. In contrast, curiosity defines one implicitly, because molecules with a low prediction error are close to previously encountered molecules in the learned problem-specific feature space of the predictor network. Since this feature space is by construction useful for predicting the property that the agent ought to optimize, we hope that the induced similarity metric is more useful for the optimization task at hand.

5 Experiments

We test our method on 3 different tasks. The three tasks are optimizing for Quantitative Estimate of Druglikeness [20] (QED), penalized logP [5] (plogP), and similarity (in terms of Tanimoto similarity of Morgan fingerprints) to the target molecule Celecoxib, a task from the Guacamol benchmark [27].

QED is a statistical measure to quantify if a specific set of 8 properties of a molecule fit those of 771 approved orally administered drugs. Penalized logP rewards the logP value and synthetic accessibility score of a molecule, and penalizes its ring count. This property is often used in drug design literature. For penalized logP the best-known optimum is the sulfur chain [12]. It turned out that the carbon chain was a good local minimum all agents were getting stuck in. Therefore we were providing the [S] symbol as the initial state so that some of the agents were able to find the sulfur chain.

For each task we train 3 agents for all possible combinations of \(\alpha = \{0, 0.01, 0.1, 1\}\), \(r_{\text{intrinsic, alternative}}\), the distance metric \(L_1/L_2\) and whether or not Greedy curiosity and the buffer are used. We divide the extrinsic and intrinsic rewards by their respective running averages so that their influence is on the same order of magnitude and the scaling factor \(\alpha\) is comparable across experiments. The agent consists of an LSTM [28] with 64 units to encode the state and a linear network to predict the action. The prediction network has the same architecture, but does not share any neural network weights.
With Curiosity

Without Curiosity

5.1 Results

The averaged results of the 3 runs for the best performing hyperparameter sets are shown in Table 1a-1c for the 3 different tasks. Additionally the best value for an agent without curiosity ($\alpha = 0$) and the best value for an agent using $r_{\text{intrinsic, alternative}}$ are shown.

The agents with curiosity perform the best, moreover, the best-performing agents always have the highest curiosity weight ($\alpha$) from all tested hyperparameter sets. For the pLogP task, only two agents, both of which use curiosity, have found the sulfur chain. This indicates that curiosity indeed can help to escape local optima. The alternative formulation of the intrinsic reward seems to help for the similarity task but not on the QED task and does not help to find the sulfur chain. The version where we optimize the predictor network after every step of the agent consistently performed better than the version where we used a buffer, which is probably due to the fact, that we trained the predictor only two times during the agent’s lifetime. In Figure 2 the best generated molecule for each task are shown.

Please note that for this work we did not employ any synthesizability or stability screening of the final molecules. This results in structures that are not so pleasant to the eye of the trained chemist. Further work can be done to add these as suitable additional terms in the reinforcement learning framework or alternatively as post-selection filters.

6 Conclusion

In this work, we develop the first curious agents in the domain of molecular design and show that they outperform their lesser curious competitors in three distinct molecular design tasks. Our results point towards a new, efficient RL-based exploration strategy for identifying high-performance molecules and compounds. Since the agent explores the chemical space without any prior knowledge, in the future this can lead to the discovery of new molecules completely different from the ones scientists know of today. Interesting future experiments could further investigate the curiosity rewards, and
Table 1: The best values of the generated molecules, averaged over the 3 runs for the 3 best performing hyperparameter settings over all tasks. Additionally the average best value of an agent without curiosity (\(\alpha = 0\)), and one that uses \(r_{\text{intrinsic, alternative}}\) are shown. The best agents all used the intrinsic reward (\(\alpha = 1\)).

<table>
<thead>
<tr>
<th>Curiosity weight (\alpha)</th>
<th>(r_{\text{intrinsic, alternative}})</th>
<th>dist</th>
<th>Greedy curiosity</th>
<th>Use buffer</th>
<th>Best QED</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>False</td>
<td>(L_2)</td>
<td>False</td>
<td>False</td>
<td>0.918</td>
</tr>
<tr>
<td>1</td>
<td>False</td>
<td>(L_1)</td>
<td>False</td>
<td>False</td>
<td>0.916</td>
</tr>
<tr>
<td>0.1</td>
<td>False</td>
<td>(L_2)</td>
<td>False</td>
<td>False</td>
<td>0.898</td>
</tr>
<tr>
<td>0</td>
<td>False</td>
<td>-</td>
<td>False</td>
<td>-</td>
<td>0.889</td>
</tr>
<tr>
<td>0.1</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>-</td>
<td>0.883</td>
</tr>
</tbody>
</table>

(a) Results for the QED task

<table>
<thead>
<tr>
<th>Curiosity weight (\alpha)</th>
<th>(r_{\text{intrinsic, alternative}})</th>
<th>dist</th>
<th>Greedy curiosity</th>
<th>Use buffer</th>
<th>Best p. logP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>False</td>
<td>(L_2)</td>
<td>False</td>
<td>False</td>
<td>10.364</td>
</tr>
<tr>
<td>1</td>
<td>False</td>
<td>(L_1)</td>
<td>False</td>
<td>False</td>
<td>10.364</td>
</tr>
<tr>
<td>0</td>
<td>False</td>
<td>-</td>
<td>False</td>
<td>-</td>
<td>9.580</td>
</tr>
<tr>
<td>0.1</td>
<td>True</td>
<td>-</td>
<td>False</td>
<td>-</td>
<td>9.580</td>
</tr>
</tbody>
</table>

(b) Results for the penalized logP task

<table>
<thead>
<tr>
<th>Curiosity weight (\alpha)</th>
<th>(r_{\text{intrinsic, alternative}})</th>
<th>dist</th>
<th>Greedy curiosity</th>
<th>Use buffer</th>
<th>Best similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>False</td>
<td>(L_1)</td>
<td>False</td>
<td>False</td>
<td>0.239</td>
</tr>
<tr>
<td>1</td>
<td>False</td>
<td>(L_1)</td>
<td>True</td>
<td>False</td>
<td>0.237</td>
</tr>
<tr>
<td>1</td>
<td>False</td>
<td>(L_2)</td>
<td>False</td>
<td>False</td>
<td>0.236</td>
</tr>
<tr>
<td>0.1</td>
<td>True</td>
<td>(L_2)</td>
<td>True</td>
<td>-</td>
<td>0.224</td>
</tr>
<tr>
<td>0</td>
<td>True</td>
<td>-</td>
<td>False</td>
<td>-</td>
<td>0.186</td>
</tr>
</tbody>
</table>

(c) Results for the similarity task

explore how they are related or can be combined with other exploration strategies, such as count based exploration \cite{29, 30}, parameter space noise injection \cite{31}, Go-Explore \cite{32} and others.

These positive results indicate that curiosity, and intrinsic rewards in general, have the potential to significantly improve de-novo molecular design. We imagine three very interesting domains of application. First, the exploration of the large chemical space could be improved leading to better solutions. Second, if scientists are interested in exceptionally unique solutions that are very different from known structures, curiosity-based optimization techniques might be a way to obtain these. Lastly, our main motivation is to explore the potential advantage of intrinsic curiosity-based rewards in RL environment. We show in three examples how curious agents outperform their non-curious friends. As future research, it will be interesting to see how these insights can be combined to build algorithms \cite{33} that compete in a wide variety of different molecular design tasks.

Importantly, to understand how to exploit the advantage of curiosity in realistic applications, we will need to train on other more complex molecular properties, as well as to make the model aware of synthesizability and stability constraints by adding filters \cite{34} and/or numerical approximations \cite{35} of the feasible regions within the chemical space. These two aspects, in general, should be a major area of focus for most of the work in this field.

7 Acknowledgements

The authors thank Theophile Gaudin and Dr. Si Yue Guo for interesting discussions. A. A.-G. acknowledges generous support from the Canada 150 Research Chairs Program, Tata Steel, Anders G Froseth, and the Office of Naval Research. L.A.T. acknowledges support from Mitacs via project FR44234. M.K. acknowledges support from the Austrian Science Fund (FWF) through the Erwin Schrödinger fellowship No. J4309. We acknowledge supercomputing support from SciNet.
References


A Greedy curiosity

We tried out another alternative formulation that we call greedy curiosity. The intuition behind greedy curiosity is that the agent is only curious in promising directions and ignores directions in the state space which have a high prediction error due to a very low reward. We achieve this by multiplying the prediction error by a mask function. Equation (4) becomes

$$ r_{\text{intrinsic}}(t) = \text{dist} \left( \widehat{p} (\text{mol}(t, \theta), \eta), p (\text{mol}(t, \theta)) \right) \cdot \text{mask} (\text{mol}(t, \theta)_{1 \ldots \text{batch size}}) \quad (6) $$

The mask function $\text{mask}(\text{mol}(t, \theta)_{1 \ldots \text{batch size}})$ gets the whole batch of molecules as input, and masks off the curiosity reward for all molecules which target properties are worse than the average in the batch:

$$ \text{mask}(\text{mol}(t, \theta)_{i=1 \ldots \text{batch size}})_i = \begin{cases} 1, & \text{if } p(\text{mol}(t, \theta)_i) - \text{mean}(p(\text{mol}(t, \theta)_{1 \ldots \text{batch size}})) > 0 \\ 0, & \text{otherwise} \end{cases} \quad (7) $$
B Curiosity training details

For training the predictor network we consider two options: The first is to update the predictor network after every episode with the new batch of generated samples. A potential downside of this is, that the predictor might forget about older samples. The second option is to use a buffer and collect and train the predictor on all samples. One can either reinitialize the predictor every time before training, which makes it very resource-intensive, or once can do warm starts. However, old samples will be seen more often than new samples, leading to overfitting. Thus we opted for reinitializing and training the predictor only two times, after 200, and again after 500 episodes. In our setup, when using equation (1) we have access to the property of the molecule at every time step. Thus, we can use all molecules at every time step to train the prediction network. If we use equation (2) instead, we have access to molecules only at the end of each trajectory, and therefore use only the last molecule of the trajectory to train the predictor.

The alternative reward formulation has the downside, that the runtime scales with the size of the memory and becomes the most time consuming step even for a small number of stored molecules $N$. Therefore we were only storing the last 2 batches of molecules in the memory which was giving it about the same computational budget as the prediction network.