Goal-directed Generation of Molecules with Conditional Generative Models

Amina Mollaysa[1,4], Brooks Paige[2,3], Alexandros Kalousis[1,4]

[1] University of Geneva

[2] The Alan Turing Institute

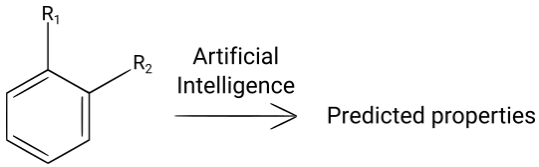
[3] University College London

[4] Data Mining and Machine Learning group, HES-SO

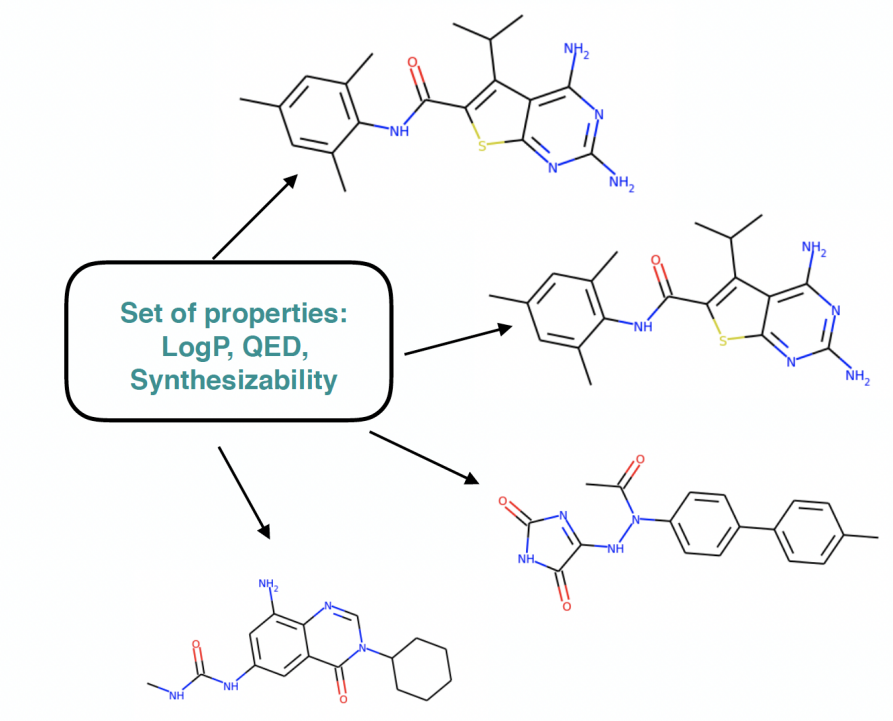
AMLD EPFL 2022

Motivation

- Molecule property prediction



- What about the inverse? → Goal-directed generation of molecules



Problem setting

- Suppose we have:
  - training set $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N}$, where $\mathbf{x}_i$ represents the molecules and $\mathbf{y}_i \in \mathcal{Y}$ is the corresponding property vector
  - and we have access to oracle function f that map $\mathbf{x}_i$ to $\mathbf{y}_i$.

- Goal: Learn a a generative model that can model $p_\theta(\mathbf{x}|\mathbf{y}) \approx \tilde{p}(\mathbf{x}|\mathbf{y})$

Maximizing expected reward

When there is a natural notion of distance $D(\mathbf{y}, \mathbf{y}')$ for values in $\mathcal{Y}$, then we can define:

- A reward $R(\mathbf{x}; \mathbf{y})$ for each $\mathbf{x}$ and for a given state $\mathbf{y}$

$$R(\mathbf{x}; \mathbf{y}) = \exp\{-D(f(\mathbf{x}), \mathbf{y})\}. \tag{1}$$

- $p_\theta(\mathbf{x}|\mathbf{y})$ defines a stochastic policy which is a distribution over $\mathbf{x}$ for a given state $\mathbf{y}$.

Learn the model $p_\theta(\mathbf{x}|\mathbf{y})$ by optimizing the expected reward:

- Expected reward:

$$\mathcal{J} = \mathbb{E}_{\tilde{p}(\mathbf{y})}\mathbb{E}_{p_\theta(\mathbf{x}|\mathbf{y})}[R(\mathbf{x}; \mathbf{y})] \tag{2}$$

Score-function gradient estimator

Standard way: use score function estimator to avoid non-differentiablity and taking derivative of the expectation:

$$\nabla_\theta \mathcal{J} = \mathbb{E}_{\tilde{p}(\mathbf{y})} \mathbb{E}_{p_\theta(\mathbf{x}|\mathbf{y})} [R(\mathbf{x};\mathbf{y}) \nabla_\theta \log p_\theta(\mathbf{x}|\mathbf{y})]. \tag{3}$$

- Noisy gradient $\Rightarrow$ Sample inefficient
- Need warm start from a pretrained model.
- Need control variate

A computationally efficient alternative

Assume we have a finite non-negative reward function $R(\mathbf{x};\mathbf{y})$, with $0 \leq R(\mathbf{x};\mathbf{y}) < \infty$, and let $c(\mathbf{y}) = \sum_{\mathbf{x}} R(\mathbf{x};\mathbf{y})$

We can then rewrite the objective in Eq. (2), observing that

$$\mathbb{E}_{\hat{p}(\mathbf{y})}\mathbb{E}_{p_\theta(\mathbf{x}|\mathbf{y})}[R(\mathbf{x};\mathbf{y})] = \mathbb{E}_{\hat{p}(\mathbf{y})}c(\mathbf{y})\mathbb{E}_{\bar{R}(\mathbf{x}|\mathbf{y})}[p_\theta(\mathbf{x}|\mathbf{y})],$$

where $\bar{R}(\mathbf{x}|\mathbf{y}) = R(\mathbf{x};\mathbf{y})/c(\mathbf{y})$

Reward function

Normalised reward distribution

The gradient is now:

$$\nabla_\theta \mathcal{J} = \mathbb{E}_{\hat{p}(\mathbf{y})}c(\mathbf{y})\mathbb{E}_{\bar{R}(\mathbf{x}|\mathbf{y})}[\nabla_\theta p_\theta(\mathbf{x}|\mathbf{y})]. \quad (4)$$

- Sample efficient.
- No need for control variate, warm start.
- Need an $\bar{R}$ that defines a reward distribution where we can sample from.

Sampling from the normalized reward distribution

- Re-express the normalized reward distribution in terms of a distribution over training indices.

$$\mathbb{E}_{\bar{R}(\mathbf{x}|\mathbf{y}_i)}[\log p_\theta(\mathbf{x}|\mathbf{y}_i)] \approx \sum_{j=1}^{N} \bar{R}(\mathbf{x}_j|\mathbf{y}_i) \log p_\theta(\mathbf{x}_j|\mathbf{y}_i) \approx \mathbb{E}_{p(j|i)}[\log p_\theta(\mathbf{x}_j|\mathbf{y}_i)] \tag{5}$$

where the distribution over indices $p(j|i)$ is defined as

$$p(j|i) = \frac{R(\mathbf{x}_j; \mathbf{y}_i)}{\sum_{j=1}^{N} R(\mathbf{x}_j; \mathbf{y}_i)}. \tag{6}$$

$$\bar{R}(\mathbf{x}_j|\mathbf{y}_i) = \frac{R(\mathbf{x}_j; \mathbf{y}_i)}{\sum_{\mathbf{x}} R(\mathbf{x}; \mathbf{y}_i)} \tag{7}$$

Experiment settings

- Tasks
  - proof of concept: generating python integer expressions that evaluate to a given value
  - Final goal: generating molecules which should exhibit a given set of properties

Toy dataset

- Synthetic data for expressions
- Properties: evaluate to certain values
- Examples of generated expression that evaluates to 100:

```
((66+41)+10)-17
39+19+42
65+35
81+19
58+42
83+17
```

- Generation performance

Objective	Valid	Unique	Novel
ML	0.9888 ± 0.0002	0.9681 ± 0.0004	0.9301 ± 0.0003
Ours	0.9903 ± 0.0003	0.9635 ± 0.0006	0.9271 ± 0.0005

Table 1: Python integer expression generation results

- Conditional generation performance

	MAE	Accuracy	Within $\pm$ 3	$-\log p(\mathbf{x}\|\mathbf{y})$
ML	13.917 ± 0.117	$\mathbf{0.166} \pm 0.001$	0.596 ± 0.001	$\mathbf{1.830}$
Ours	$\mathbf{11.823} \pm 0.145$	$\mathbf{0.166} \pm 0.001$	$\mathbf{0.682} \pm 0.001$	1.986

Table 2: Python integer expression *conditional* generation results

Conditional generation of molecules

Experimental setup:

- Dataset: QM9 [Ramakrishnan et al., 2014], ChEMBL [Mendez et al., 2018]
- Molecule representation: SMILES strings [Weininger, 1988]
- Properties: Nine molecule properties including both continuous and discrete values [1]

How often the model generate valid, unique, novel molecules?

	QM9			ChEMBL		
Model	Valid	Unique	Novel	Valid	Unique	Novel
ML	0.962	**0.967**	**0.366**	0.895	**0.999**	**0.990**
Ours	**0.989**	0.963	0.261	**0.945**	0.9986	0.981

Table 3: Molecule generation quality.

[1] #rotatable bonds, #aromatic rings, logP, QED-score, tpsa, bertz, molecule weight, atom Counter, #ringstion

Generation performance

- How plausible the generated molecules?

  Generation quality filter [Brown et al., 2019]: aim to detect those which are *"potentially unstable, reactive, laborious to synthesize, or simply unpleasant"*

Generated molecules	Random sample from ChEMBL test set
71.3%	72.2%

Table 4: Percentage of the molecules that pass the filter.

- Example of generated molecules from the ChEMBL model

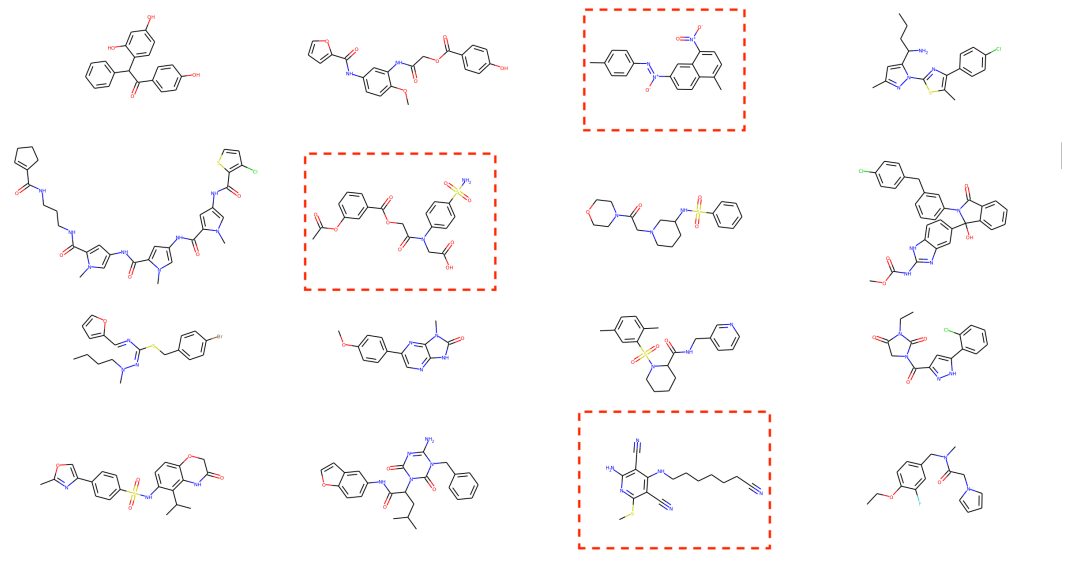


Figure 1: Those which fail to pass the quality filter are marked in red.

Conditional generation performance

Model	rotatable bonds	aromatic rings	logP	QED	TPSA	bertz	mol weight	fluorine count	# rings
QM9: MSE									
ML	0.0468	0.0014	0.0390	0.0010	11.18	80.77	4.425	0.0023	0.0484
Ours	**0.0166**	**0.0005**	**0.0184**	**0.0004**	**3.859**	**63.67**	**1.184**	**0.0004**	**0.0120**
QM9: Correlation coefficient									
ML	0.9809	0.9944	0.9805	0.9063	0.9871	**0.9843**	0.9651	0.9783	0.9817
Ours	**0.9937**	**0.9972**	**0.9901**	**0.9634**	**0.9954**	0.9840	**0.9887**	**1.0000**	**0.9948**
ChEMBL: MSE									
ML	**0.1552**	0.0388	0.1450	0.0050	**27.64**	**1708.**	**103.9**	0.0128	0.0226
Ours	**0.1555**	**0.0268**	**0.1320**	**0.0046**	35.05	2512.	174.9	**0.0074**	**0.0191**
CheEMBL: Correlation coefficient									
ML	**0.9936**	0.9862	0.9777	0.9450	**0.9906**	0.9934	**0.9956**	0.9940	0.9931
Ours	0.9934	**0.9901**	**0.9796**	**0.9496**	0.9878	0.9902	0.9926	**0.9966**	**0.9943**

Table 5: Conditional generation performance for the molecules datasets.

Conditional generation performance

- Can the model generate various molecules for a fixed property vector?

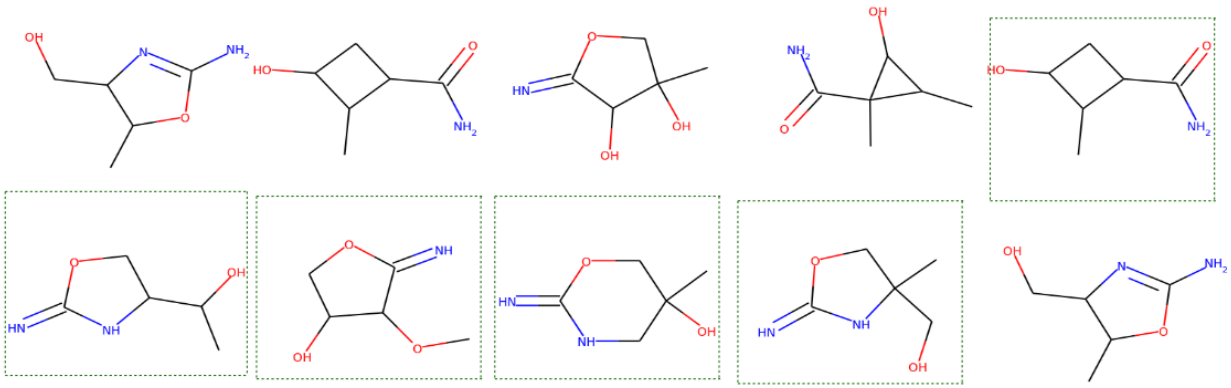


Figure 2: Molecules generated from a given property value vector for QM9 MODEL. The boxed ones are molecules that have not been seen before.

Sequence diversification

To encourage a more diverse exploration of the sequence space and thus more diverse generations, we couple our objective with an entropy regulariser:

$$\max_{\theta} \sum_{i=1}^{N} [\mathbb{E}_{\bar{R}(\mathbf{x}|\mathbf{y}_i)} [\log p_\theta(\mathbf{x}|\mathbf{y}_i)] + \lambda H(p_\theta(\mathbf{x}|\mathbf{y}_i))]. \tag{8}$$

In a discrete sequence model the gradient of the entropy term can be computed as

$$\begin{aligned} \nabla_\theta H(p_\theta(\mathbf{x}|\mathbf{y})) &= -\nabla_\theta \mathbb{E}_{p_\theta(\mathbf{x}|\mathbf{y})} \log p_\theta(\mathbf{x}|\mathbf{y}) \\ &= -\mathbb{E}_{p_\theta(\mathbf{x}|\mathbf{y})} [(1 + \log p_\theta(\mathbf{x}|\mathbf{y})) \nabla_\theta \log p_\theta(\mathbf{x}|\mathbf{y})]; \end{aligned} \tag{9}$$

- En efficient approximation of the entropy:

$$H(p_\theta(\mathbf{x}|\mathbf{y})) \approx H[p_\theta(\mathbf{x}_1|\mathbf{y})] + \sum_{t=2}^{T} \left[H[p_\theta(\mathbf{x}_t|\mathbf{x}^*_{1:t-1}, \mathbf{y})] \right].$$

where $\mathbf{x}^*_t$ is the maximum probability element at each step.

More baselines

- Data augmentation on discrete sequences:
  RAML [Norouzi et al., 2016]: maximizes a conditional log probability of the augmented versions of the training instances.
- Edit distance augmentation

m	validity	unicity	MSE
One	0.265	0.322	194.945
Two	0.095	0.421	468.556
Three	0.046	0.393	725.128
Four	0.0276	0.422	985.451
Five	0.0204	0.480	1496.023
Six	0	-	-

Table 6: Edit distance augmentation evaluation on QM9 dataset

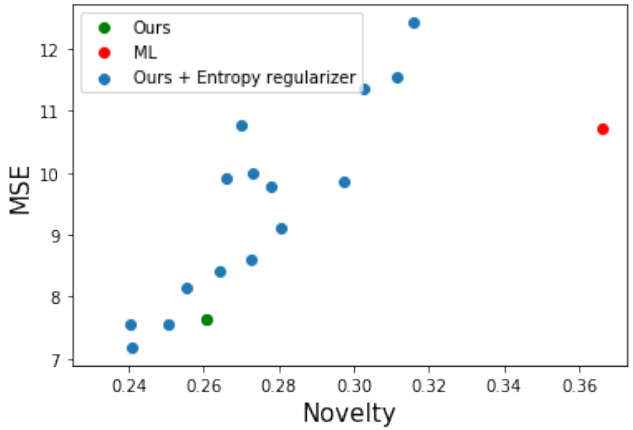
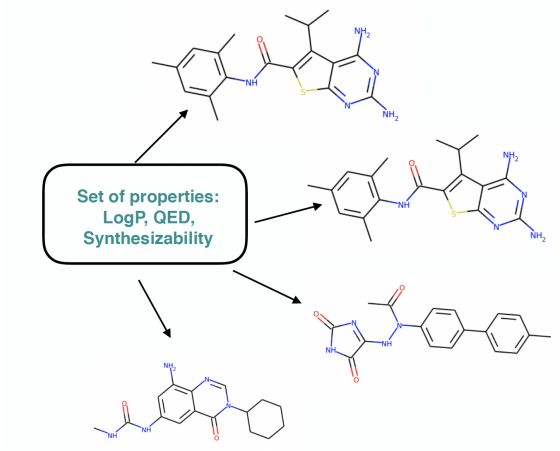


Table 7: Effect of the entropy on the generated sequences from the validation set.

Data augmentation fails when the underlying sequences properties are sensitive to local change.

De Novo Molecular Design

- Global optimization: generating from scratch



- Local optimization: starting from a prototype molecule

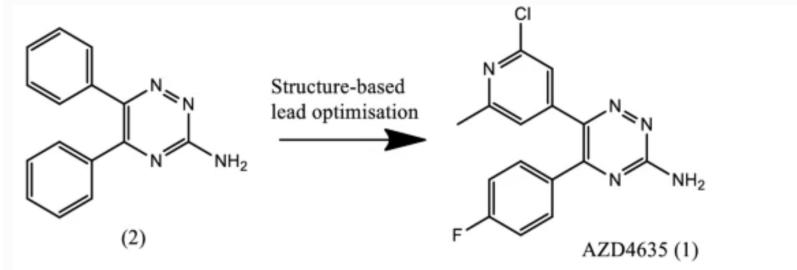


Figure 3: [Green et al., 2020]

- learning $p_\theta(\mathbf{x}|\mathbf{y})$
- learning $p_\theta(\mathbf{x}|\mathbf{y},\mathbf{z})$ where $\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})$, style transfer

- Conditional generation: learning $p_\theta(\mathbf{x}|\mathbf{y})$ with auto-regressive models [2]
- Style transfer: learning $p_\theta(\mathbf{x}|\mathbf{y},\mathbf{z})$ where $\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})$ with VAEs[3]

[2] A.Mollaysa, B.Paige, A.Kalousis, *Goal-directed Generation of Discrete Structures with Conditional Generative Models, NeurIPS 2020.*

[3] A.Mollaysa, B.Paige, A.Kalousis, *Conditional generation of molecules from disentangled representations, Machine Learning for Molecules Workshop NeurIPS 2020*

- Auto-regressive models
  - have very good validity and conditional generation performance
  - can not do style transfer
- Latent variable models
  - can do style transfer
  - poor validity and poor style transfer/conditional generation performance

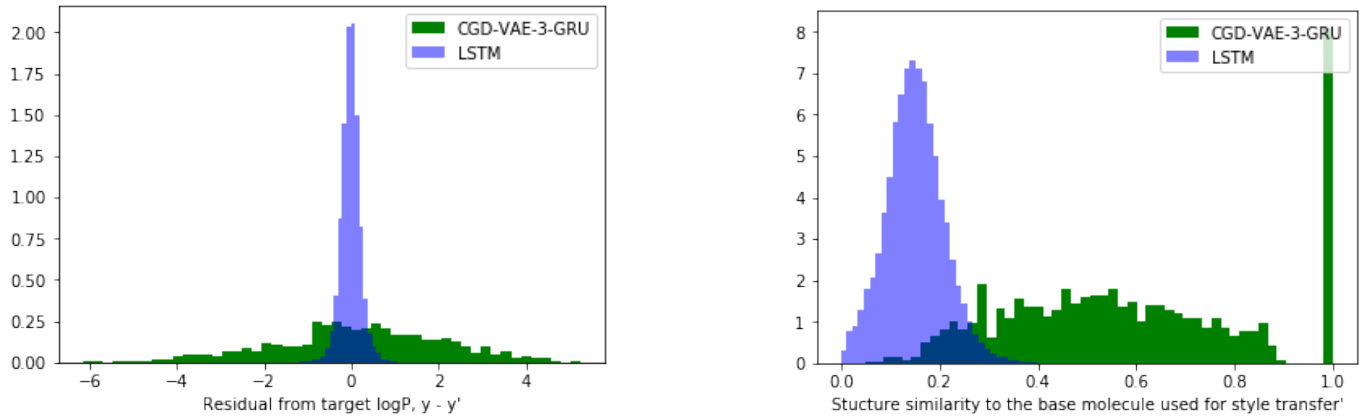


Figure 4: A comparison of simulated logP values and Tanimoto similarity to a target on the ZINC dataset.

Auto-regressive models VS latent variable models

Model	encoder	decoder	validity	Conditional generation	
			$\mathbf{z} \sim p(\mathbf{z})$	$\mathbf{z} \sim p(\mathbf{z})$	$\mathbf{z} \sim q(\mathbf{z}\|\mathbf{x})$
CVAE [Lim et al., 2018]	LSTM	LSTM w.o. Teacher forcing	0.6%	0.0012%	0.0147%
SSVAE [Kang and Cho, 2018]	bi- GRU	GRU w. Teacher forcing	99.3	75.6%	78.0%

Table 8: Comparison in terms of network structure and generation strategy

Combining the best of two world

- Possible solutions
  - provide supervision to for style transfer: using a pre-trained $\tilde{p}(\mathbf{x}|\mathbf{y})$ as a regularizer:

$$\max_{\theta} \mathbb{E}_{\mathbf{x}\sim p_\theta(\mathbf{x}|\mathbf{y},\mathbf{z})} \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})} \log \tilde{p}(\mathbf{x}|\mathbf{y}) \tag{10}$$

  - learn an auto-regressive model $p(\mathbf{x}|\mathbf{y},\mathbf{z})$ that condition on both $\mathbf{y}$ and $\mathbf{z}$, where $\mathbf{z}$ is a learned structural representations of the molecule.

References I

Brown, N., Fiscato, M., Segler, M. H., and Vaucher, A. C. (2019).

Guacamol: benchmarking models for de novo molecular design.
Journal of chemical information and modeling, 59(3):1096–1108.

Green, D. V., Pickett, S., Luscombe, C., Senger, S., Marcus, D., Meslamani, J., Brett, D., Powell, A., and Masson, J. (2020).

Bradshaw: a system for automated molecular design.
Journal of computer-aided molecular design, 34(7):747–765.

Kang, S. and Cho, K. (2018).

Conditional molecular design with deep generative models.
Journal of chemical information and modeling, 59(1):43–52.

Lim, J., Ryu, S., Kim, J. W., and Kim, W. Y. (2018).

Molecular generative model based on conditional variational autoencoder for de novo molecular design.
Journal of cheminformatics, 10(1):1–9.

Mendez, D., Gaulton, A., Bento, A. P., Chambers, J., De Veij, M., Félix, E., Magariños, M. P., Mosquera, J. F., Mutowo, P., Nowotka, M., et al. (2018).
Chembl: towards direct deposition of bioassay data.
Nucleic acids research, 47(D1):D930–D940.

Norouzi, M., Bengio, S., Jaitly, N., Schuster, M., Wu, Y., Schuurmans, D., et al. (2016).

Reward augmented maximum likelihood for neural structured prediction.
In *Advances In Neural Information Processing Systems*, pages 1723–1731.

Ramakrishnan, R., Dral, P. O., Rupp, M., and Von Lilienfeld, O. A. (2014).

Quantum chemistry structures and properties of 134 kilo molecules.
Scientific data, 1:140022.

Weininger, D. (1988).

Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules.
Journal of chemical information and computer sciences, 28(1):31–36.

Numerical instability

To avoid numerical instability when $p_\theta(\mathbf{x}|\mathbf{y})$ takes very small values, we instead work in terms of log probabilities:

$$\arg\max_\theta \mathcal{J} = \arg\max_\theta \log \mathcal{J}, \tag{11}$$

where we then have

$$\log \mathcal{J} = \log \left(\mathbb{E}_{\tilde{p}(\mathbf{y})} c(\mathbf{y}) \mathbb{E}_{\bar{R}(\mathbf{x}|\mathbf{y})} [p_\theta(\mathbf{x}|\mathbf{y}_i)] \right) \geq \mathbb{E}_{\tilde{p}(\mathbf{y})} \mathbb{E}_{\bar{R}(\mathbf{x}|\mathbf{y})} [\log p_\theta(\mathbf{x}|\mathbf{y})] + const.$$

which motivates optimizing a lower-bound on $\log \mathcal{J}$,

$$\mathcal{L} = \mathbb{E}_{\tilde{p}(\mathbf{y})} \mathbb{E}_{\bar{R}(\mathbf{x}|\mathbf{y})} [\log p_\theta(\mathbf{x}|\mathbf{y})] \tag{12}$$

and whose gradient is simply

$$\nabla_\theta \mathcal{L} = \mathbb{E}_{\tilde{p}(\mathbf{y})} \mathbb{E}_{\bar{R}(\mathbf{x}|\mathbf{y})} [\nabla_\theta \log p_\theta(\mathbf{x}|\mathbf{y})]. \tag{13}$$

Conclusion

- We present a simple, tractable, and efficient algorithm to learn the conditional distribution of molecules.
- By sampling directly from the approximate normalized reward distribution,our approach sidesteps challenges of directly maximizing an expected reward.