Molecule Design by Latent Space Energy-Based Modeling and Gradual Distribution Shifting (Supplementary Materials)

Deqian Kong[†1] **Bo Pang**[†2] **Tian Han**[3] **Ying Nian Wu**[1]

[1]Department of Statistics, University of California, Los Angeles
[2]Salesforce Research
[3]Department of Computer Science, Stevens Institute of Technology

1 DETAILS ABOUT MODEL AND LEARNING

Our model is of the form $p_\alpha(z)p_\beta(x|z)p_\gamma(y|z)$. The marginal distribution of (x, y) is

$$p_\theta(x, y) = \int p_\theta(x, y, z)dz = \int p_\alpha(z)p_\beta(x|z)p_\gamma(y|z)dz.$$

We use $p_\theta(x, y)$ to approximate the data distribution of (x, y).

For the data distribution of (x, y), y is a deterministic function of x. However, a machine learning method usually cannot learn the deterministic function exactly. Instead, we can only learn a probabilistic $p_\theta(y|x)$. Our model $p_\theta(x, y)$ seeks to approximate the data distribution $p(x, y)$ by maximum likelihood. A learnable and flexible prior model $p_\alpha(z)$ helps to make the approximation more accurate than a fixed prior model such as that in VAE.

Let the training data be $\{(x_i, y_i), i = 1, ..., n\}$. The log-likelihood function is $L(\theta) = \sum_{i=1}^n \log p_\theta(x_i, y_i)$. The learning gradient is $L'(\theta) = \sum_{i=1}^n \nabla_\theta \log p_\theta(x_i, y_i)$. In the following, we provide details for calculating $\nabla_\theta \log p_\theta(x, y)$ for a single generic training example (x, y) (where we drop the subscript $_i$ for notation simplicity).

$$\begin{aligned}
\nabla_\theta \log p_\theta(x, y) &= \frac{1}{p_\theta(x, y)} \nabla_\theta p_\theta(x, y) \\
&= \frac{1}{p_\theta(x, y)} \int \nabla_\theta p_\theta(x, y, z)dz \\
&= \frac{1}{p_\theta(x, y)} \int p_\theta(x, y, z) \nabla_\theta \log p_\theta(x, y, z)dz \\
&= \int \frac{p_\theta(x, y, z)}{p_\theta(x, y)} \nabla_\theta \log p_\theta(x, y, z)dz \\
&= \int p_\theta(z \mid x, y) \nabla_\theta \log p_\theta(x, y, z)dz \\
&= \mathbb{E}_{p_\theta(z|x,y)} \left[\nabla_\theta \log p_\theta(x, y, z)\right] \\
&= \mathbb{E}_{p_\theta(z|x,y)} \left[\nabla_\theta (\log p_\alpha(z) + \log p_\beta(x|z) + \log p_\gamma(y|z))\right].
\end{aligned}$$

[†]Equal contribution

Accepted for the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023).

For the prior model,

$$\begin{aligned}\nabla_\alpha \log p_\alpha(z) &= \nabla_\alpha f_\alpha(z) - \nabla_\alpha \log Z(\alpha) \\ &= \nabla_\alpha f_\alpha(z) - \frac{1}{Z(\alpha)} \nabla_\alpha Z(\alpha) \\ &= \nabla_\alpha f_\alpha(z) - \frac{1}{Z(\alpha)} \int \nabla_\alpha \exp(f_\alpha(z)) p_0(z) dz \\ &= \nabla_\alpha f_\alpha(z) - \int \nabla_\alpha f_\alpha(z) \frac{1}{Z(\alpha)} \exp(f_\alpha(z)) p_0(z) dz \\ &= \nabla_\alpha f_\alpha(z) - \mathbb{E}_{p_\alpha(z)}[\nabla_\alpha f_\alpha(z)].\end{aligned}$$

Thus the learning gradient for α given an example (x, y) is

$$\delta_\alpha(x, y) = \nabla_\alpha \log p_\theta(x, y) = \mathbb{E}_{p_\theta(z|x,y)}[\nabla_\alpha f_\alpha(z)] - \mathbb{E}_{p_\alpha(z)}[\nabla_\alpha f_\alpha(z)]. \tag{1}$$

The above equation has an empirical Bayes nature. $p_\theta(z|x, y)$ is based on the empirical observation (x, y), while p_α is the prior model. For the generation model,

$$\delta_\beta(x, y) = \nabla_\beta \log p_\theta(x, y) = \mathbb{E}_{p_\theta(z|x,y)}[\nabla_\beta \log p_\beta(x|z)]. \tag{2}$$

Similarly, for the regression model,

$$\delta_\gamma(x, y) = \nabla_\gamma \log p_\theta(x, y) = \mathbb{E}_{p_\theta(z|x,y)}[\nabla_\gamma \log p_\gamma(y|z)]. \tag{3}$$

Estimating expectations in the above equations requires Monte Carlo sampling of the prior model $p_\alpha(z)$ and the posterior distribution $p_\theta(z|x, y)$. If we can draw fair samples from the two distributions, and use these Monte Carlo samples to approximate the expectations, then the gradient ascent algorithm based on the Monte Carlo samples is the stochastic gradient ascent algorithm or the stochastic approximation algorithm of Robbins and Monro [Robbins and Monro, 1951], who established the convergence of such an algorithm to a local maximum of the log-likelihood.

For MCMC sampling using Langevin dynamics, the finite step or short-run Langevin dynamics may cause bias in Monte Carlo sampling. The bias was analyzed in Pang et al. [2020]. The resulting algorithm is an approximate maximum likelihood learning algorithm.

2 TRAINING TIME

The training of joint distribution of molecule and its properties takes around 4 hours with 25 iterations on a single Nvidia Titan XP GPU with batch size 2048. For non-biological single-objective property optimization, it takes around 20 minutes to do 30 distribution shifting (SGDS) iterations. If we use SGDS without warm start, it takes around half an hour. For biological binding affinity maximization, the optimization time is mainly dependent on the number of queries of AutoDock-GPU. We do 30 and 20 SGDS iterations for the single-objective and multi-objective tasks, respectively, which cost 10 hours and 8 hours with warm start, and cost 14 hours and 10 hours without warm start. For biological property optimization tasks, we use two Nvidia Titan XP GPUs, one for running our code and the other for running AutoDock-GPU. We have added a table to compare with previous methods.

Model	Penalized-logP/QED	Single binding affinity
JT-VAE	24	–
GCPN	8	6
MolDQN	24	6
GraphDF	8	12
Mars	12	6
LIMO	1	1
SGDS without warm start	4.5	18
SGDS with warm start	4.3	14

Table 1: Comparison of molecule generation time in (hrs). Results obtained from [Eckmann et al., 2022].

Even if we use MCMC sampling-based methods, our training speed is affordable comparing to existing methods. That is because our designed latent space EBM is low-dimensional (i.e. dim(z)=100) and we use short-run MCMC (i.e. with fixed iteration steps 20) in our experiments. The major bottleneck of the training speed is the time of querying the property compute engines.

3 GENERATED SAMPLES

3.1 BIOLOGICAL PROPERTY OPTIMIZATION

Figure 1 and Figure 2 show generated molecules with high binding affinities towards ESR1 and ACAA1 respectively in single-objective property design experiments.

Figure 3 and Figure 4 show generated molecules with high binding affinities towards ESR1 and ACAA1 respectively in multi-objective property design.

Comparing to the previous state-of-the-art methods, SGDS is able to produce more high quality molecules than top-3 molecules because after gradual distribution shifting, the joint distribution locates at the area supported by molecules with high binding affinities.

Meanwhile, compared to previous generative model-based methods, we use Langevin dynamics to infer the posterior distribution $p(z|x, y_1, \ldots, y_n)$ without bothering to design different encoders when facing different combination of properties.

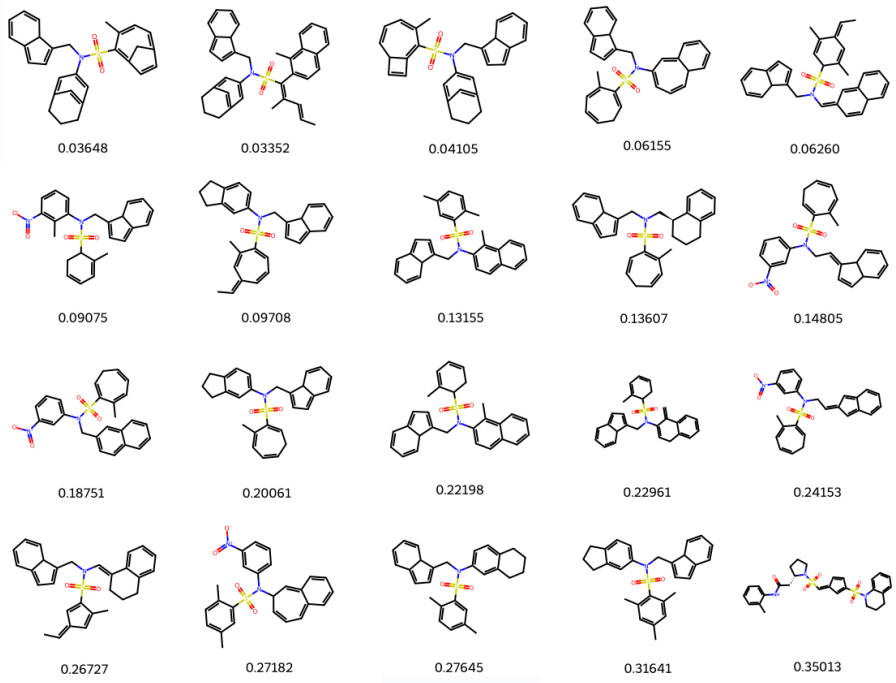

Figure 1: Generated molecules in singe-objective esr1 binding affinity maximization experiments with corresponding $\mathrm{K_D}(\downarrow)$ in nmol/L.

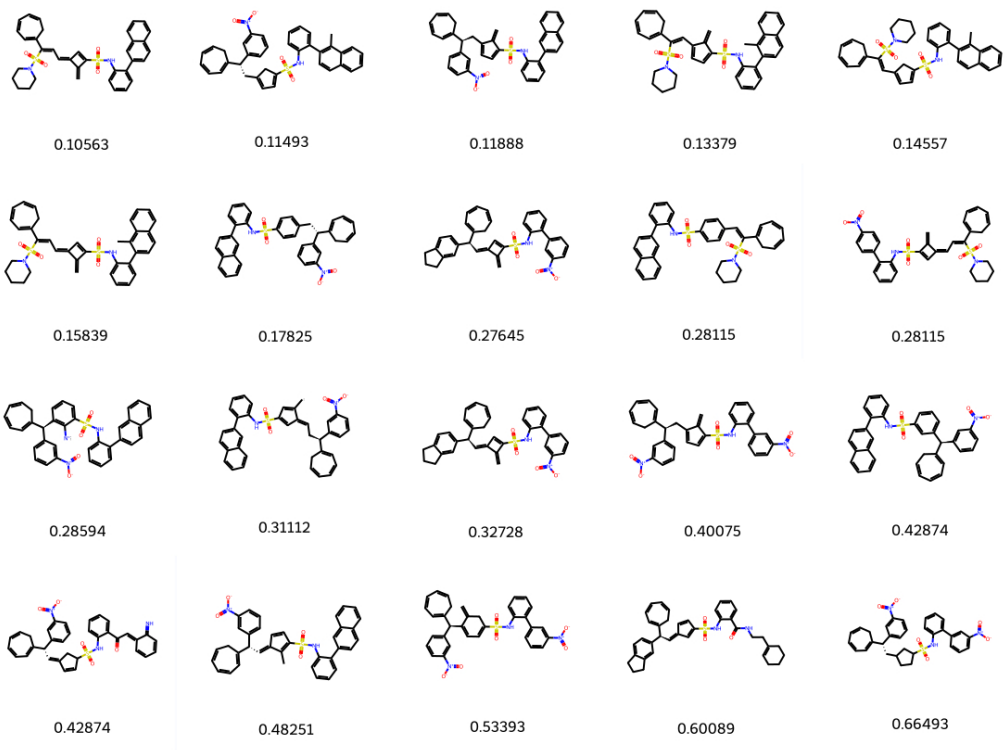

Figure 2: Generated molecules in singe-objective acaa1 binding affinity maximization experiments with corresponding $\mathrm{K_D}(\downarrow)$ in nmol/L.

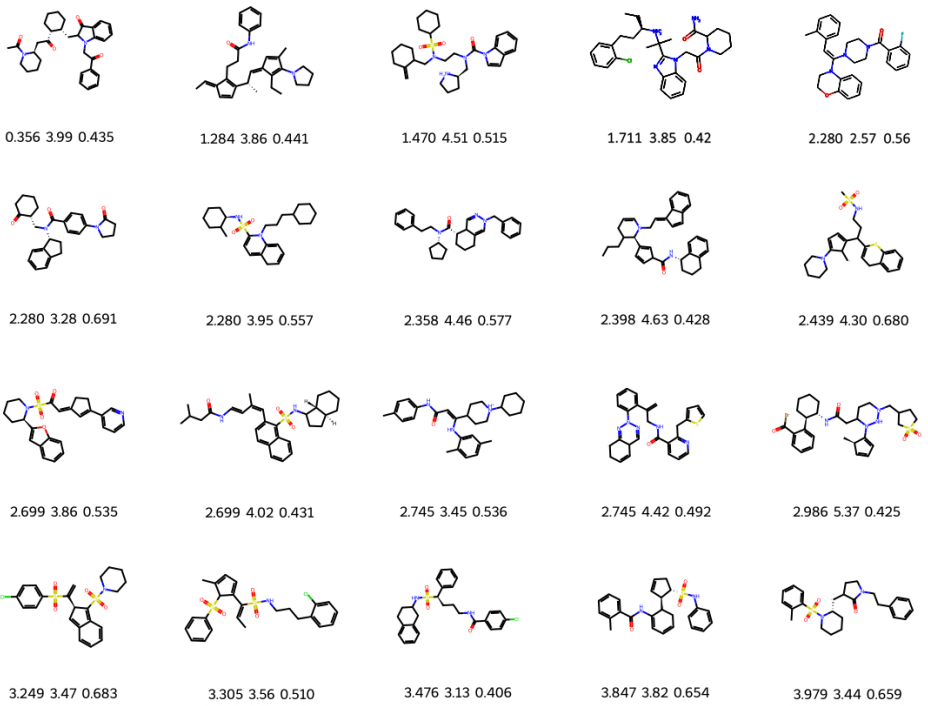

Figure 3: Generated molecules in multi-objective esr1 binding affinity maximization experiments with corresponding $\mathrm{K_D}(\downarrow)$ in nmol/L, SA$(\downarrow)$ and QED$(\uparrow)$ respectively.

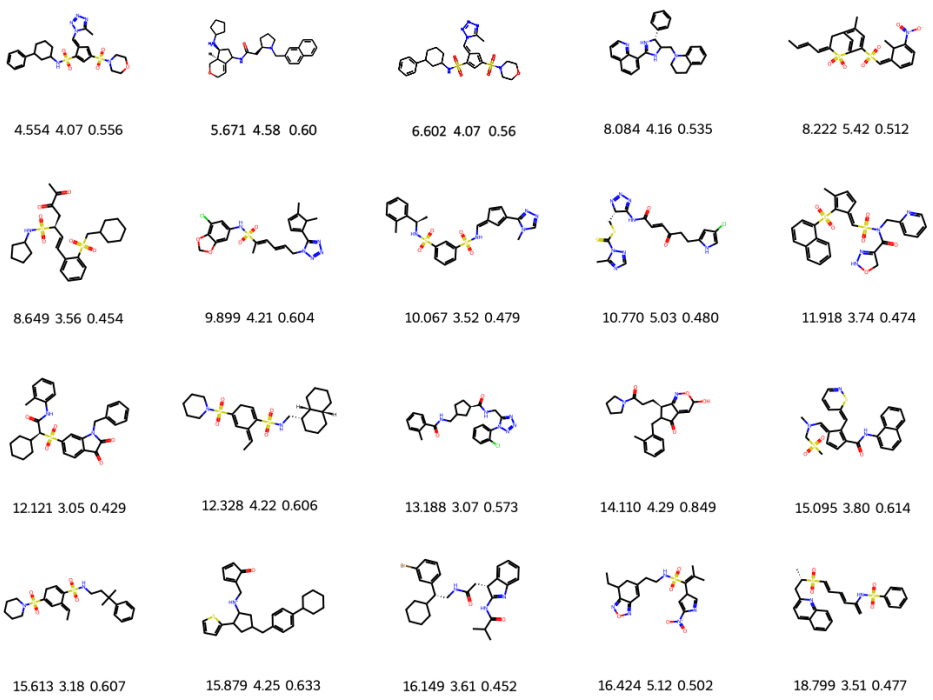

Figure 4: Generated molecules in multi-objective acaa1 binding affinity maximization experiments with corresponding $K_D(\downarrow)$ in nmol/L, SA($\downarrow$) and QED($\uparrow$) respectively.

3.2 P-LOGP AND QED OPTIMIZATION

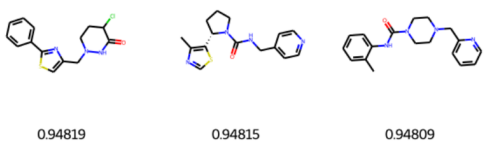

Figure 5: Top-3 molecules in single-objective QED maximization.

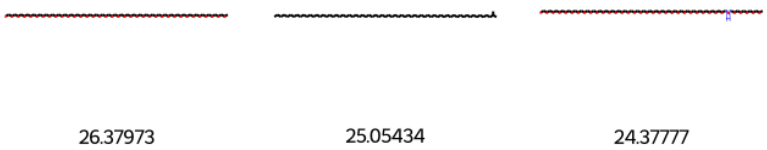

Figure 6: Top-3 molecules in single-objective p-logP maximization.

4 ILLUSTRATION OF SAMPLING WITH GRADUAL DISTRIBUTION SHIFTING (SGDS)

Figures 7 to 9 show property densities of sampled molecules of the distribution shifting process in single-objective penalized logP, esr1 and acaa1 optimization respectively. SGDS is implemented with warm start. We can see the model distribution is gradually shifting towards the region supported by molecules with high property values. To better visualize the shifting process, we plot the docking scores rather than $\mathrm{K_D}$. The increase in docking scores corresponds to the exponential decrease in $\mathrm{K_D}$.

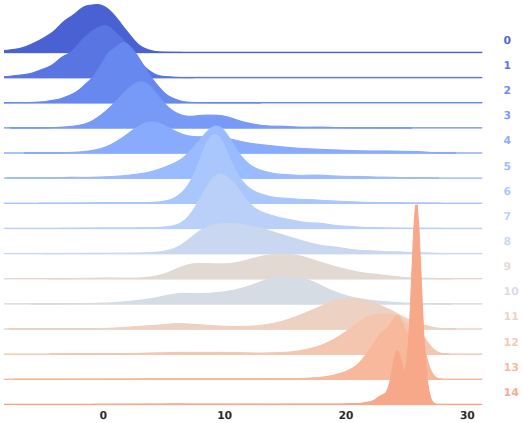

Figure 7: Illustration of SGDS in a single-objective penalized logP optimization experiment.

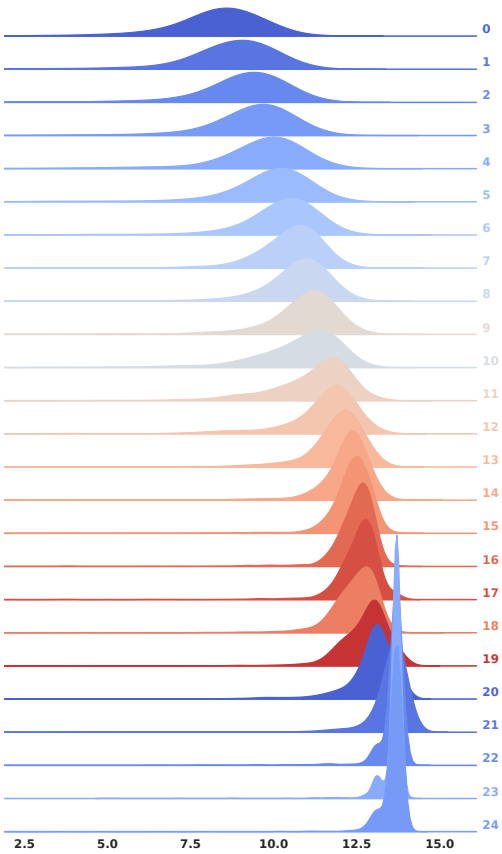

Figure 8: Illustration of SGDS in a single-objective esr1 optimization experiment.

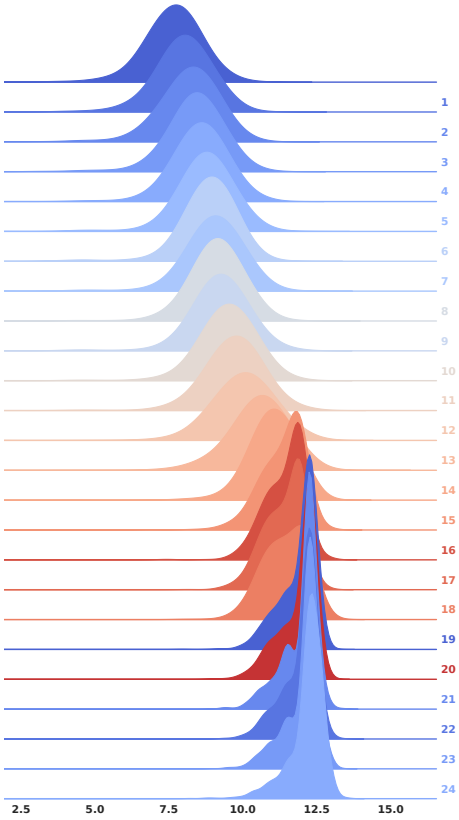

Figure 9: Illustration of SGDS in a single-objective acaa1 optimization experiment.

References

Peter Eckmann, Kunyang Sun, Bo Zhao, Mudong Feng, Michael K Gilson, and Rose Yu. Limo: Latent inceptionism for targeted molecule generation. *arXiv preprint arXiv:2206.09010*, 2022.

Bo Pang, Tian Han, Erik Nijkamp, Song-Chun Zhu, and Ying Nian Wu. Learning latent space energy-based prior model. *arXiv preprint arXiv:2006.08205*, 2020.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.