Chem263: Machine Learning for Chemical and Dynamical Data

Lecture 15 — Generative models that construct probability distributions
Last time

- Applications of generative models
  - VAE (variational autoencoders) for chemical design

- Inverting property prediction
  - Gaussian process regression

\[ \text{GP}(\mu(x), k(x,x')) \]\n
Latent Space \rightarrow \text{Properties}
Generative models

Goal of unsupervised learning:

\[ D = \prod_{i=1}^{n} x_i \sim p(x) = \lim_{n \to \infty} \frac{1}{n} \sum_{i=1}^{n} S x_i \]

But we can approximate:

\[ p(x) \propto e^{-\beta E_0(x)} \]

Energy based model

\[ p(x | x') = \text{VAE}(x) \]

Variational Autoencoder
Sampling from VAEs

Sampling from a trained VAE is conditioned on data:

\[ \mu, \sigma^2 = \text{Encoder}(x) \]
\[ z \sim N(\mu, \sigma^2) \]
\[ x' = \text{Decoder}(z) \]

What is \( p(x') \)?

\[ p(x') = p(x) \frac{q(z|x)}{\text{still intractable?}} \]
Unconditional sampling?

Many applications, we want to sample $p(x)$ directly.

\[ p_0 (x) \approx p(x) \]

Sample via Langevin: \[ x = x_t + \nabla_x \log p_0 (x_t) \Delta t + \sqrt{2} \frac{\Delta t}{\xi} \]

$P_{\text{Langevin}}(x) \rightarrow e^{\log p_0 (x)} = p_0 (x)$

Metastability
Bayesian estimation

Bayes' Theorem: \( p(\theta) = p(\theta|\mathcal{D}) p_0(\theta) \)

Typically, we know \( \log p(\theta|\mathcal{D}) = x(\theta) \)

\[
p(\theta) = z^{-1} x(\theta) p_0(\theta) \quad z = \int \, x(\theta) p_0(\theta) \, d\theta
\]

Sampling \( p(\theta) \) requires normalization constant

(or MCMC)
Markov Chain Monte Carlo

Ubiquitous computational technique

1. Propose random displacement:

\[ \pi(x' | x) \]

2. Accept or reject:

\[
\text{Accept } [x \rightarrow x'] = \min \left[ 1, \frac{p(x') \pi(x | x')}{p(x) \pi(x' | x)} \right]
\]

Familiar form: \[ \min \left[ 1, e^{-\beta E} \right] \]

\( \pi \) - symmetric
\( p \) - Boltzmann
Criteria for a normalizing flow

Learned mapping: \( T : \Omega \rightarrow \Omega \)

Generative model: \( z \sim p_B(z) \)

Base distribution

\( T(z) \rightarrow x \)

Base \hspace{1cm} Target

Push forward distribution:

\[ p_B(x) = p_B(T^{-1}(x)) | \nabla T^{-1}(x) | \]

"diffeomorphism"

Map must be continuous, invertible, differentiable

Normalizing Flow
Building flexible maps

Reminder:

**Jacobian:**

\[
\begin{pmatrix}
\frac{\partial T_1}{\partial x_1} & \cdots & \frac{\partial T_1}{\partial x_d} \\
\vdots & \ddots & \vdots \\
\frac{\partial T_d}{\partial x_1} & \cdots & \frac{\partial T_d}{\partial x_d}
\end{pmatrix}
\]

\[\det |\nabla T| \approx \frac{\text{Vol}(dx)}{\text{Vol}(dz)}\]

Simplest strategy: Composition of simple maps:

\[T = T_k \circ T_{k-1} \circ \cdots \circ T_1 \quad \Rightarrow \quad \det |\nabla T(z_0)| = \frac{k}{\prod_{k=1}^{K} \det |\nabla T(z_{k-1})|} \]

Chain Rule
Building flexible maps

Auto-regressive Flows

\[
\begin{align*}
&\left(\begin{array}{c}
  z_1 \\
  z_2 \\
  \vdots \\
  z_{i-1} \\
  z_i \\
  z_{i+1} \\
  \vdots \\
  z_d
\end{array}\right) \\
&\left(\begin{array}{c}
  z_1 \\
  z_2 \\
  \vdots \\
  z_{i-1} \\
  z_i \\
  z_{i+1} \\
  \vdots \\
  z_d
\end{array}\right) \\
&\left(\begin{array}{c}
  z_1 \\
  z_2 \\
  \vdots \\
  z_{i-1} \\
  z_i \\
  z_{i+1} \\
  \vdots \\
  z_d
\end{array}\right) \\
&\left(\begin{array}{c}
  z_1 \\
  z_2 \\
  \vdots \\
  z_{i-1} \\
  z_i \\
  z_{i+1} \\
  \vdots \\
  z_d
\end{array}\right)
\end{align*}
\]

\[i^\text{th} \text{ conditioner}
\]

\[C_i (z_1, \ldots, z_{i-1}) \equiv h_i
\]

\[z_i' = f (z_i, h_i)
\]

invertible

(monotonic) map

\[\Rightarrow J(T) = \begin{pmatrix}
\frac{\partial f}{\partial z_i} & 0 \\
\vdots & \vdots \\
L(z) & df/2d
\end{pmatrix}
\]

\[= \log \det J = \sum_{i=1}^d \log \left| \frac{\partial f}{\partial z_i} \right|
\]

Inverse is easy to compute sequentially

\[z_i = f^{-1} (z_i', h_i)
\]

\[h_i = C_i (z_1, \ldots, z_{i-1})
\]

Sequentially compute
Building flexible maps

\[ f \text{ must be a monotonic function: many options} \]

**Affine:**
\[ f(z_i ; h_i) = a_i z_i + b_i \quad h_i = \{ \tilde{a}_i, b_i \} \]

**Integration:**
\[ f(z_i , h_i) = \int_0^{z_i} g(z, \tilde{a}_i) \, dz + b_i \quad h_i = \{ \tilde{a}_i, b_i \} \]

Any positive valued function

Train w/ MLE:
\[ \ell(\theta) = \text{D}_{KL}(p_{\text{data}}(x) \| p_B^{\theta \perp T(z)}) \leftarrow \text{SGD} \]
Boltzmann generators

Target distribution $e^{-\beta E(x)}$ for some physical system

Can we design strategies for sampling of NFs in the low data regime?

Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning

Frank Noé*, Simon Olson*, Jonas Köhler*, Hao Wu
Sampling molecular configurations

Training samplers for biomolecules —

Many constraints

1) Gaussianize the data

\[
\frac{1}{d} x^T x R = R \Lambda
\]

Variance principal components

Data dependent \( \Lambda \)

\[
W(x) = \Lambda^{-1/2} R^T x
\]

"PCA whitening"
Permutation invariance

By default, there is none...

Possible solution (Wünsberger et al.)

Transformer (again) \( x_i \rightarrow h_i \) (linear, sum \( D_i \))

\( h_i \leftarrow h_i + \text{Self-attention}(h_1, \ldots, h_n) \)

\( h_i \leftarrow \text{MLP}(h_i) \)

\( \uparrow \) some MLP peremes \( D_i \)

\( \uparrow \) Matrix ops

\( \Rightarrow \) permutation equivariant

residual

self-attention
Performance of Boltzmann generator

Large speed up relative to MCMC (when there's data to train on)

"Few shot" learning requires building in more physics

Grant M. Rotskoff

Stanford University
Challenges with “learned” generators

Complicated, high-dimensional

One good solution: build more physics into \( p_B \)

Example: Brownian bridge

Similar strategies for hard core repulsions, bonds?
Building in equivariance

Boltzmann generator: rotate and align → just eliminate rotations / translations

Can we design a \( T \) such that

\[
T(Rz + \alpha) = RT(z) + \alpha
\]

Equivariant to Euclidean transformations
Review of the E(n)–GNN

Message passing architecture:

\[ m_{ij} = \phi_e (h_i^e, h_j^e, \|x_i^e - x_j^e\|_2) \]

\[ m_i^e = \sum_{j \neq i} e_{ij} m_{ij} \]

\[ x_i^{e+1} = x_i^e + \sum_{j \neq i} (x_i^e - x_j^e) \phi_x (m_{ij}) \]

\[ h_i^{e+1} = \phi_h (h_i^e, m_i) \]
Adapting GNNs for normalizing flows

Continuous depth parameterization:

\[ \mathbf{z}_0, \mathbf{z}_h = (x(0), h(0)) + \int_0^1 \phi(x(t), h(t)) \, dt \]

\[ \dot{x}, \dot{h} = \phi(x, h) = x'(t) - x(t), h'(t) \]

Get these from EGNN
Performance of the E(n)-GNN NF

In Table 1 we report the cross-validated Negative Log Likelihood for the test partition. Our E-NF method and its non-equivariant variants (GNF, GNF-att, GNF-att-aug) consist of 3 layers each, with 32 features per layer, and SiLU activation functions. All reported numbers have been averaged over 3 runs. Further implementation details are provided in the Appendix A.1.

Table 1: Negative Log Likelihood comparison on the test partition over different methods on DW4

<table>
<thead>
<tr>
<th># Samples</th>
<th>DW4</th>
<th>10^2</th>
<th>10^3</th>
<th>10^4</th>
<th>10^5</th>
<th>10</th>
<th>10^2</th>
<th>10^3</th>
<th>10^4</th>
</tr>
</thead>
<tbody>
<tr>
<td>GNF</td>
<td>11.93</td>
<td>11.31</td>
<td>10.38</td>
<td>7.95</td>
<td>43.56</td>
<td>42.84</td>
<td>37.17</td>
<td>36.49</td>
<td></td>
</tr>
<tr>
<td>GNF-att</td>
<td>11.65</td>
<td>11.13</td>
<td>9.34</td>
<td>7.83</td>
<td>43.32</td>
<td>36.22</td>
<td>33.84</td>
<td>32.65</td>
<td></td>
</tr>
<tr>
<td>GNF-att-aug</td>
<td>8.81</td>
<td>8.31</td>
<td>7.90</td>
<td>7.61</td>
<td>41.09</td>
<td>31.50</td>
<td>30.74</td>
<td>30.93</td>
<td></td>
</tr>
<tr>
<td>Simple dynamics</td>
<td>9.58</td>
<td>9.51</td>
<td>9.53</td>
<td>9.47</td>
<td>33.67</td>
<td>33.10</td>
<td>32.79</td>
<td>32.99</td>
<td></td>
</tr>
<tr>
<td>Kernel dynamics</td>
<td>8.74</td>
<td>8.67</td>
<td>8.42</td>
<td>8.26</td>
<td>35.03</td>
<td>34.49</td>
<td>31.17</td>
<td>31.25</td>
<td></td>
</tr>
<tr>
<td>E-NF</td>
<td>8.31</td>
<td>8.15</td>
<td>7.69</td>
<td>7.48</td>
<td>33.12</td>
<td>30.99</td>
<td>30.56</td>
<td>30.41</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Neg. log-likelihood – log p_θ(x, h, M), atom stability and mol stability for the QM9 dataset.

<table>
<thead>
<tr>
<th># Methods</th>
<th>NLL</th>
<th>Atom stability</th>
<th>Mol stability</th>
</tr>
</thead>
<tbody>
<tr>
<td>GNF-attention</td>
<td>-28.2</td>
<td>72%</td>
<td>0.3%</td>
</tr>
<tr>
<td>GNF-attention-aug</td>
<td>-29.3</td>
<td>75%</td>
<td>0.5%</td>
</tr>
<tr>
<td>E-NF (ours)</td>
<td>-53.4</td>
<td>84%</td>
<td>4.2%</td>
</tr>
<tr>
<td>Data</td>
<td>-</td>
<td>99%</td>
<td>95.2%</td>
</tr>
</tbody>
</table>

Currently, no great new "standard" benchmark problems
One more lecture—what should we discuss?

Option 1
VAMP Nets / Analysis of MD trajectories

Option 2
Inverse problems, Cryo EM, scattering

Option 3
Variational Monte Carlo / Many-body Wave function architectures