Generative Machine Learning
for Particle and Statistical Physics

Vipul Arora

IIT Kanpur

MADHAV lab
Machine Analysis of Data for Human Audition and Vision

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

The Need

- Probability density function (pdf)
- E.g., Boltzmann distribution

$$\hat{p}(x) = e^{-\frac{H(x)}{k_B T}}$$

$$p(x) = \frac{\hat{p}(x)}{Z}; Z = \int \hat{p}(x) dx$$

- Variants:

$$\hat{p}(x) = e^{-s(x)}$$

The Need

- We need expectation values over pdf's
- Examples: Macroscopic quantities
  - Average energy
  - Average magnetization
  - Susceptibility

Monte Carlo Approximation

- How to estimate expected values?

$$\mathbb{E}_{x\sim p_X(x)}[f(x)] = \int f(x)\, p_X(x)\, dx$$

- For many problems, integrating a pdf may be difficult or practically impossible
- Monte Carlo approximation:

$$\mathbb{E}_{x\sim p_X(x)}[f(x)] = \frac{1}{M}\sum_{s=1}^{M} f(x_s); \quad x_s \sim p_X(x)$$

Applications

- Statistical Physics
- Particle Physics
- Condensed Matter Physics
- Biological sciences
- Environmental sciences

Sampling Methods

Sampling a standard pdf

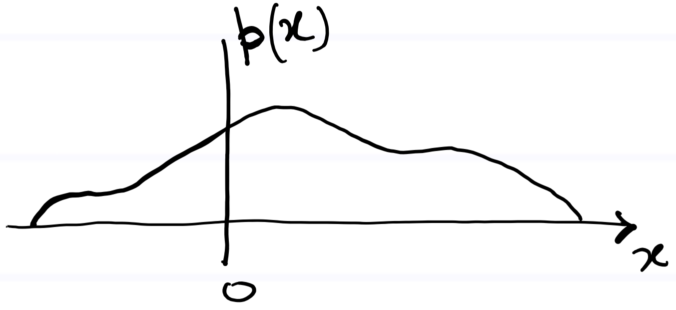
- We need samples from standard $p_X(x) = \mathcal{U}(x; 0,1)$
- Available libraries to sample from uniform distribution
- What if $p_X(x)$ is not standard?
- Methods
  - Variable transformation
  - Rejection sampling
  - Importance sampling (to compute expectations)

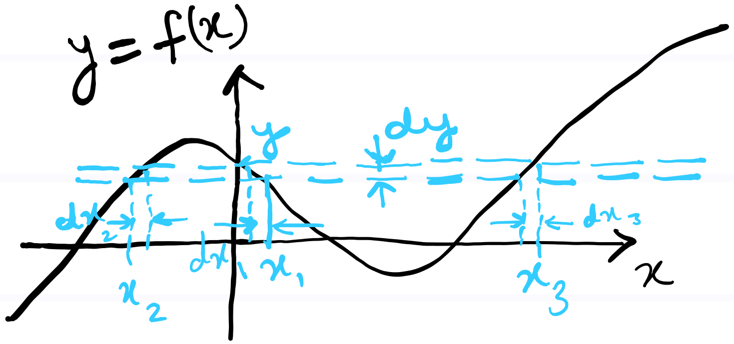
Transformation of Random Variables

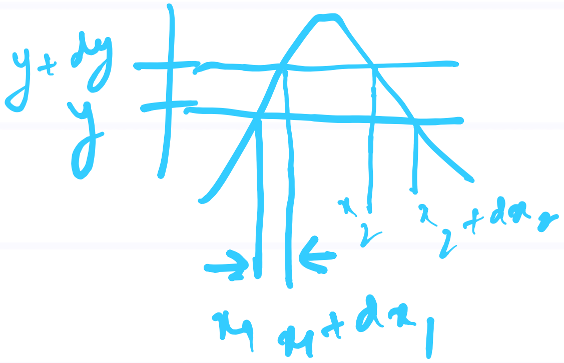
- Given $p_x(x)$ and $y = f(x)$, find $p_y(y)$
- At some x_1, $y = f(x_1)$
- Probability mass is conserved

$$p_y(y)|dy| = p_x(x_1)|dx_1|$$

- But there may be other x too that map to same y, say $f(x_i) = y$ for $i = 1,2, \dots, N$
- $p_y(y)|dy| = p_x(x_1)|dx_1| + p_x(x_2)|dx_2| + \cdots$
- $p_y(y) = \sum_i \frac{p_x(x)}{\left|\frac{dy}{dx}\right|}\Bigg|_{x_i}$







Transformation of Random Vectors

- $x = G(z)$
- For random variables, $p_X(x) = \sum_{z=G^{-1}(x)} p_Z(z) \frac{1}{\left|\frac{dG(z)}{dz}\right|}$
- For random vectors,
  - $J_G(z) = \frac{dG(z)}{dz}$ Jacobian matrix
  - $p_X(x) = \sum_{z=G^{-1}(x)} p_Z(z) |\det J_G(z)|^{-1}$

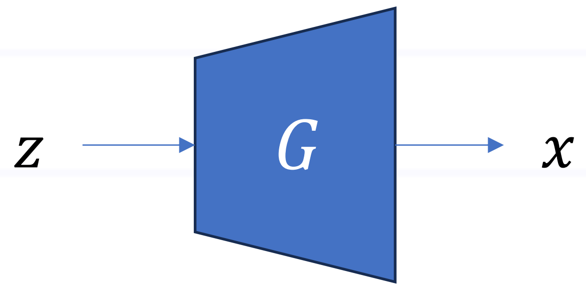
Deep Transformations for Sampling

Approximate samples (no guarantees)

Generative Adversarial Network (GAN)

[Goodfellow *et al.*, Generative Adversarial Nets, NIPS 2014]

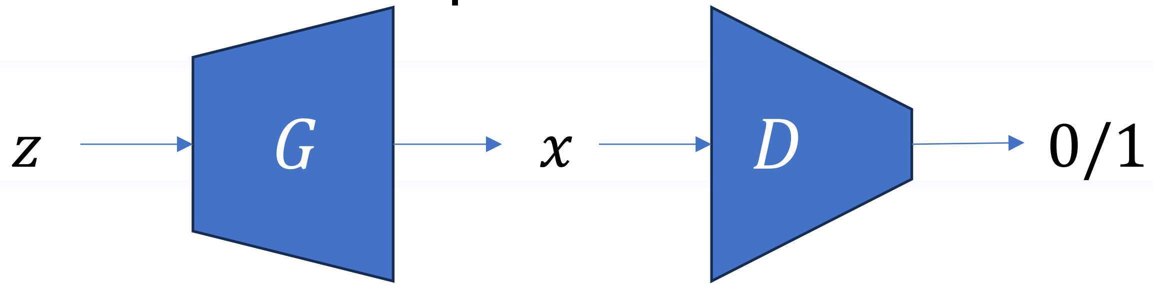
- Given: $p_X(x)$ is the target distribution. Only samples from $p_X(x)$ are available.
- Goal: get more samples from $p_X(x)$
- Solution: Choose $p_Z(z)$ that is easy to sample from and learn a transformation G to get x



Training a GAN

- Set a discriminator to distinguish real and fake samples

$$D(x) = \begin{cases} 0; & x = G(z) \\ 1; & x \sim p_X(x) \end{cases}$$



- Learning objective (to train parameters of G and D)

$$-\mathcal{L} = \mathbb{E}_{x \sim p_X(x)}[\log D(x)] + \mathbb{E}_{\substack{x = G(z) \\ z \sim p_Z(z)}}[1 - \log D(x)]$$

- D tries to maximize the objective function
- G tries to minimize the objective

Algorithm

- Training
  i. for k steps:
     i. get M samples of $z \sim p_Z(z); x = G(z)$
     ii. get M samples of $x \sim p_X(x)$
     iii. Update D by gradient descent to maximize $-\mathcal{L}$
  ii. for 1 step:
     i. get M samples of $z \sim p_Z(z); x = G(z)$
     ii. Update G by gradient descent to minimize $-\mathcal{L}$
  iii. Go back to step i until convergence
- Sampling
  - Sample $z \sim p_Z(z); x = G(z)$

Convergence

- Does this method ensure that the samples $x = G(z); z \sim p_Z(z)$ represent $p_X(x)$?
- Ans: Yes. Proof given in [Goodfellow 2014]
- Provided the training happens perfectly – but that is difficult

Problems

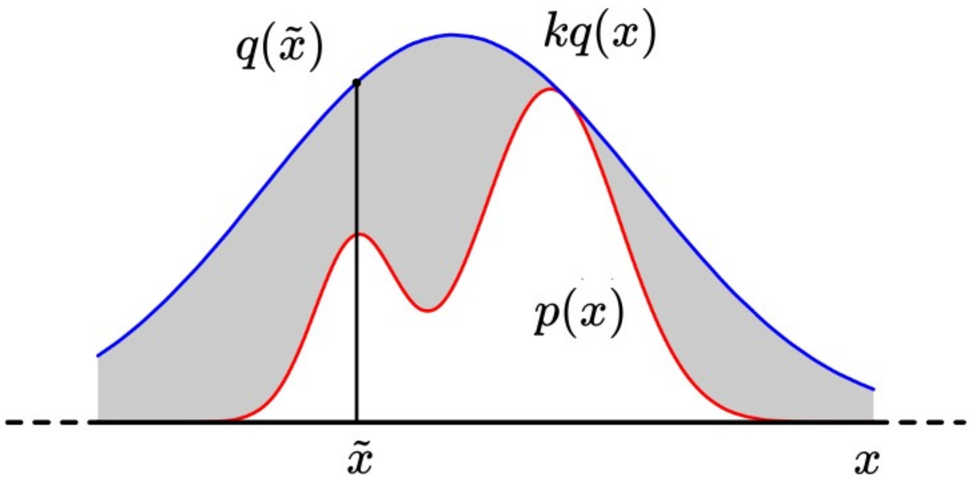
- Samples are not precise
- Hence, there could be unknown biases in the expectations

Precise Sampling

Guaranteed

Rejection sampling

- $p(x)$ is difficult to sample from
- Choose $q(x)$ that is easy to sample from
- Find k such that $kq(x) \geq p(x)\ \forall x$
- Sampling algorithm
    i. Sample $\tilde{x} \sim q(x)$
    ii. Accept $\tilde{x}$ with a probability $\frac{p(\tilde{x})}{kq(\tilde{x})}$



Source: PRML, ch. 11

Rejection sampling

- Acceptance Rate

- $\mathbb{E}[\text{accept}] = \int \left\{\frac{p(x)}{kq(x)}\right\} q(x)dx$

$$= \int \frac{p(x)}{k} dx = \frac{1}{k}$$

- For efficiency:
  - k should be as small as possible.
  - Thus, $q(x)$ should be similar to $p(x)$

Rejection sampling

- If only $\hat{p}(x)$ is given, as in Boltzmann distribution, rejection sampling still works
- To find k
  - $kq(x) \geq \frac{\hat{p}(x)}{Z}$
  - $(kZ)q(x) \geq \hat{p}(x)$
  - $\hat{k}q(x) \geq \hat{p}(x)$
- With increase in dimensions of x, k increases

Question

- Can we apply rejection sampling over GAN samples?
- No
- Because $q(x)$ is not known
- What to do to get $q(x)$ for GAN?
- We need to know all $\{z: z = G^{-1}(x)\}$
- This is not easy
- Moreover, finding k is another challenge

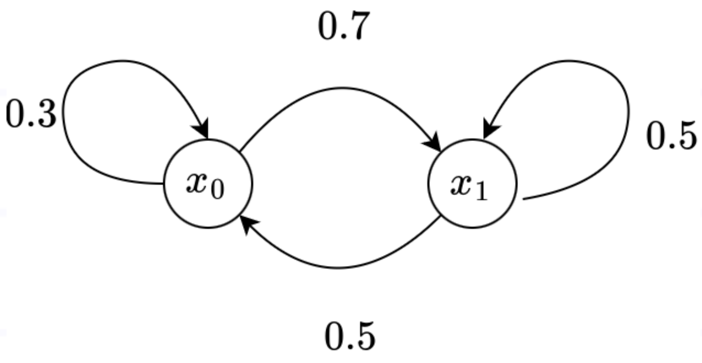
Iteratively Sampling Methods

Random Walks

Markov Chain Monte Carlo (MCMC)

- Given transition probabilities $p(x^{t+1}|x^t)$, get samples from $\pi(x)$, i.e., equilibrium distribution
- Sampling algorithm
  i. Start from a random x^0
  ii. Sample multiple times from $p(x^{t+1}|x^t)$
  iii. After 1000 iterations, note down the state
  iv. Repeat i – iii multiple times

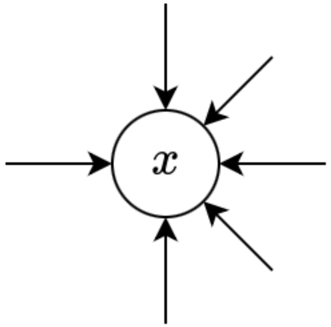
  The noted down samples are the samples from $\pi(x)$



Will a Markov Chain converge?

- $p(x^{t+1}|x^t)$ converges to a stationary distribution $\pi(x)$ if

$$\pi(x) = \sum_{x'} p(x|x')\pi(x')$$



- This is equivalent to the "detailed balance principle"

$$\pi(x')p(x|x') = \pi(x)p(x'|x)$$

Proof

$$\pi(x) = \sum_{x'} p(x|x')\pi(x')$$

$$\pi(x')p(x|x') = \pi(x)p(x'|x)$$

In first equation,

$$\text{RHS} = \sum_{x'} p(x|x')\pi(x')$$

$$= \sum_{x'} p(x'|x)\pi(x)$$

$$= \pi(x)\sum_{x'} p(x'|x)$$

$$= \pi(x) = \text{LHS}$$

Use of MCMC

- If $\pi(x)$ is given and difficult to sample from, one can construct a Markov chain $p(x^{t+1}|x^t)$ and get samples iteratively
- How to construct a Markov Chain?
- Methods:
  - Gibbs sampling
  - Metropolis Hastings

Metropolis Hastings (MH) Algorithm

- Applies rejection sampling to Markov chain
- Select a proposal distribution $q(x^{t+1}|x^t)$ that is easy to sample from
- Algorithm
  - for $k = 0,1, \dots$
    - Initialize x^0 randomly
    - Sample $\tilde{x}$ from $q(x^{t+1}|x^t)$
    - Accept $\tilde{x}$ with a probability $A(x^{t+1}|x^t)$, i.e.,

      $x^{t+1} = \tilde{x}$ if accepted, and $x^{t+1} = x^t$ otherwise

Metropolis Hastings (MH) Algorithm

- Overall transition probability is

$$p(x^{t+1}|x^t) = q(x^{t+1}|x^t)\,A(x^{t+1}|x^t)$$

- Now, find $A(x^{t+1}|x^t)$ such that $p(x^{t+1}|x^t)$ converges to the desired $\pi(x)$

Metropolis Hastings (MH) Algorithm

- Use condition for convergence

$$\pi(x')p(x|x') = \pi(x)p(x'|x)$$

$$\pi(x^t)q(x^{t+1}|x^t)\,A(x^{t+1}|x^t) = \pi(x^{t+1})q(x^t|x^{t+1})A(x^t|x^{t+1})$$

$$\frac{A(x^{t+1}|x^t)}{A(x^t|x^{t+1})} = \frac{\pi(x^{t+1})q(x^t|x^{t+1})}{\pi(x^t)q(x^{t+1}|x^t)} = \rho \ \ \text{(say)}$$

If $\rho < 1$, let $A(x^{t+1}|x^t) = \rho$ and $A(x^t|x^{t+1}) = 1$

If $\rho \geq 1$, let $A(x^{t+1}|x^t) = 1$ and $A(x^t|x^{t+1}) = \frac{1}{\rho}$

Thus,
$A(x^{t+1}|x^t) = \min(1, \rho)$

Metropolis Hastings (MH) Algorithm

for $k = 0,1,\ldots$

- Initialize x^0 randomly
- Sample $\tilde{x}$ from $q(x^{t+1}|x^t)$
- Accept $\tilde{x}$ with a probability $\min\left(1, \frac{\pi(x^{t+1})q(x^t|x^{t+1})}{\pi(x^t)q(x^{t+1}|x^t)}\right)$, i.e.,

  $x^{t+1} = \tilde{x}$ if accepted, and $x^{t+1} = x^t$ otherwise

Metropolis Hastings (MH) Algorithm

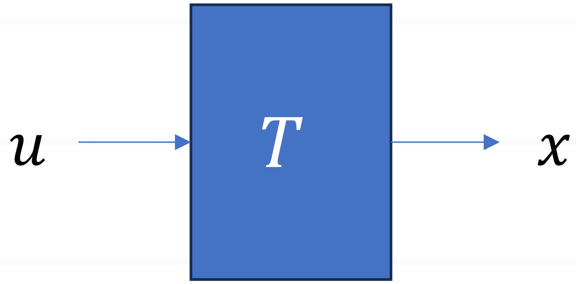
Choosing $q(x^{t+1}|x^t)$

- $q(x^{t+1}|x^t) > 0 \quad \forall x^{t+1}$
- Need to balance step size
  - too narrow $q(x^{t+1}|x^t)$: takes too long to cover the space
  - too wide $q(x^{t+1}|x^t)$: most samples are rejected

Normalizing Flows for Sampling

Precise samples (guaranteed)

Normalizing Flow (NF)



- $p_U(u)$ is easy to sample
- T should be invertible, i.e., bijective, and differentiable

$$q_X(x) = p_U(u)\left|\det J_T(u)\right|^{-1}\Big|_{u=T^{-1}(x)}$$

- NF has a parametric T which can be trained such that $q_X(x)$ matches $p_X(x)$

Example of T: scalar

- Scalar random variables

$$x = T(u) = au + b$$

- What is $q_X(x)$?

$$q_X(x) = p_U(u)\left|\frac{dx}{du}\right|^{-1} = p_U(u)\frac{1}{|a|}$$

Example of T: 2-D

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = T\left(\begin{bmatrix} u_1 \\ u_2 \end{bmatrix}\right) = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

- Is T differentiable?
- Is T invertible?
- What is $q_X(x)$?

$$q_X(x) = p_U(u) |\det J_T(u)|^{-1}$$

Example of T: 2-D

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = T\left(\begin{bmatrix} u_1 \\ u_2 \end{bmatrix}\right) = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

- What is $J_T(u)$?

$$J_T(u) = \begin{bmatrix} \frac{\partial x_1}{\partial u_1} & \frac{\partial x_1}{\partial u_2} \\ \frac{\partial x_2}{\partial u_1} & \frac{\partial x_2}{\partial u_2} \end{bmatrix}$$

$$J_T(u) = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

- What is $|\det J_T(u)|$?

$$|\det J_T(u)| = |ad - bc|$$

MADHAV

Example of T: non-affine or non-linear

$$x_1 = u_1$$

$$x_2 = u_2\sigma_2(u_1) + \mu_2(u_1)$$

$\sigma_2(u_1),\ \mu_2(u_1)$ could be implemented with a neural network

- Is it differentiable?
- Is it invertible?
- Given $\boldsymbol{x}$, $u_1 = x_1; u_2 = \frac{x_2 - \mu_2(x_1)}{\sigma_2(x_1)}$

Example of T: non-affine or non-linear

- What is $|\det J_T(u)|$?

$$x_1 = u_1$$
$$x_2 = u_2\sigma_2(u_1) + \mu_2(u_1)$$

$$J_T(u) = \begin{bmatrix} \frac{\partial x_1}{\partial u_1} & \frac{\partial x_1}{\partial u_2} \\ \frac{\partial x_2}{\partial u_1} & \frac{\partial x_2}{\partial u_2} \end{bmatrix}$$

$$J_T(u) = \begin{bmatrix} 1 & 0 \\ u_2\frac{\partial \sigma_2(u_1)}{\partial u_1} + \frac{\partial \mu_2(u_1)}{\partial u_1} & \sigma_2(u_1) \end{bmatrix}$$

$$|\det J_T(u)| = |\sigma_2(u_1)|$$

MADHAV

Example of T: RNVP Flow

- Real Non-Volume Preserving Flow

$$x_1 = u_1$$

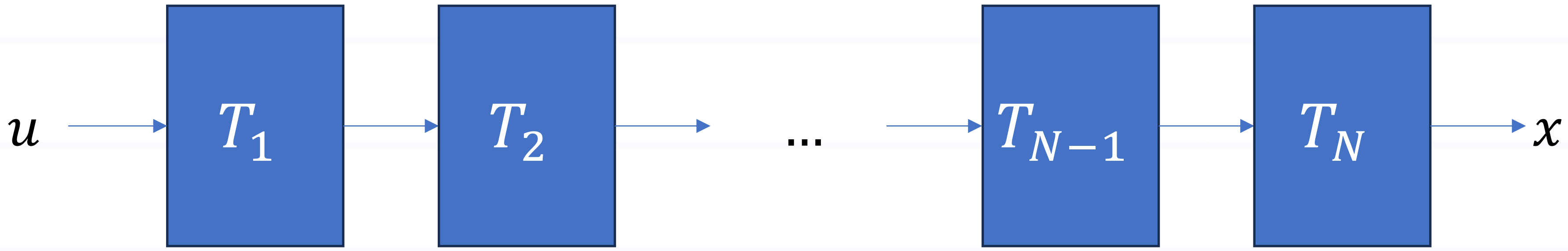
$$x_2 = u_2\sigma_2(u_1) + \mu_2(u_1)$$

$$x_3 = u_3\sigma_3(u_1, u_2) + \mu_3(u_1, u_2)$$

$$\dots$$

- $|\det J_T(u)| = |\sigma_2(u_1)\sigma_3(u_1, u_2)\dots|$

We can make it deep



- The overall composition is differentiable and invertible

Training the T

- $q(x;\theta)$ should match $p(x)$. Here, θ are parameters of T.
  Subscripts dropped from $q_X(x), p_U(u), p_X(x)$ for brevity.
- How to design the loss function?
- Kullback Leibler Divergence
- $D_{KL}[p(x) \parallel q(x;\theta)]$ or $D_{KL}[q(x;\theta) \parallel p(x)]$

Forward KL divergence

$D_{KL}[p(x) \parallel q(x;\theta)]$

$= \mathbb{E}_{x\sim p(x)}[\log p(x) - \log q(x;\theta)]$

$\log p(x)$ is independent of θ

and $q(x) = p(u)|\det J_T(u;\theta)|^{-1}|_{u=T^{-1}(x;\theta)} = p(u)|\det J_{T^{-1}}(x;\theta)|\big|_{u=T^{-1}(x;\theta)}$

$= \mathbb{E}_{x\sim p(x)}[-\log p(u) - \log|\det J_{T^{-1}}(x;\theta)|]; u = T^{-1}(x;\theta)$

$= \mathbb{E}_{x\sim p(x)}\left[-\log p\left(T^{-1}(x;\theta)\right) - \log|\det J_{T^{-1}}(x;\theta)|\right]$

Reverse KL divergence

$D_{KL}[q(x;\theta) \parallel p(x)]$

$= \mathbb{E}_{x\sim q(x;\theta)}[\log q(x;\theta) - \log p(x)]$

$= \mathbb{E}_{u\sim p(u)}[\log q(x;\theta) - \log p(x)]$ $\therefore$ T is bijective

Using $q(x) = p(u)|\det J_T(u;\theta)|^{-1}|_{u=T^{-1}(x;\theta)}$

$= \mathbb{E}_{u\sim p(u)}\left[\log p(u) - \log|\det J_T(u;\theta)| - \log p\big(T(u;\theta)\big)\right]$

Summary

- Transformation of random variables and vectors
- GANs for sampling (no guarantees)
- Sampling with guarantees
  - Rejection sampling
  - MH algorithm
  - Normalizing Flow (NF)