Generative Modelling

Goal:

Learn the distribution $p(x)$ over images x s.t

(1) generation: $x_{new} \sim p(x)$ is "good"

(2) density estimation: $p(x)$ should be high if x is similar to the data

(3) unsupervised representation learning: learn features that our datapoints have in common.

- Question: how to represent $p(x)$?

Bayes Networks:

Recall:

(1) $p(x_1, \dots, x_n) = p(x_1)\, p(x_2 \mid x_1) \cdots p(x_n \mid x_1, \dots, x_{n-1})$ → Here $x_1, \dots, x_n$ could be pixels of an image.

Suppose $p(x_1)$ requires 1 parameter

$p(x_2 \mid x_1 = 0)$ requires 1 param, $p(x_2 \mid x_1 = 1)$ requires 1 param

Continuing this way, we require

$1 + 2 + \dots + 2^{n-1} = 2^n - 1$ parameters.

This is a lot of parameters!

But if we impose a markov assumption...

$$p(x_1, \dots, x_n) = p(x_1)\, p(x_2 \mid x_1) \cdots p(x_n \mid x_1, \dots, x_{n-1})$$
$$= p(x_1)\, p(x_2 \mid x_1)\, p(x_3 \mid x_2) \cdots p(x_n \mid x_{n-1})$$

This requires $2n-1$ parameters.

With this idea, we introduce Bayes Networks

for each random variable X_i, we learn $P(X_i \mid X_{A_i})$ where X_{A_i} is a set of random variables.

Then,

$$P(X_1, \dots, X_n) = \prod_i P(X_i \mid X_{A_i})$$

We won't dig into the details any further.

- **Naive Bayes**

Example: $Y = 1$ implies the email is a spam, $Y = 0$ otherwise

$(X_1, \dots, X_n)$ = an email.

Then $P(Y, X_1, \dots, X_n) = P(Y) \prod_{i=1}^{n} P(X_i \mid Y)$ by the Naive Bayes assumption.

We learn these parametres from the data:

$$P(Y = 1 \mid X_1, \dots, X_n) = \frac{P(Y=1) \prod_{i=1}^{n} P(X_i \mid Y=1)}{\sum_{y \in \{0,1\}} P(Y=y) \prod_{i=1}^{n} P(X_i \mid Y=y)}$$

Issue: The naive Bayes assumption is often too strong.

Here the Naive Bayes's assumption is that

$$X_i \perp X_{-i} \mid Y$$

i.e given Y, X_i is independent of X_{i-j}, $j \in \{1, \dots, i\}$

Note: Discriminative vs Generative Models

Generative models: learn $P(Y, X)$ and $P(X)$

Then, $P(Y|X) = \frac{P(Y, X)}{P(X)}$

Discriminative models: Directly learn $P(Y|X)$ — no need to learn $P(X)$

Discriminative Models:

$P(Y=1 \mid X; \theta) = f_\theta(x)$ ⟶ a parametrized function with parameters θ.

Often, we impose structure such as:

(1) linear dependence:

define $z(\theta, x) = \theta_0 + \sum_{i=1}^{n} \theta_i x_i$

Then $P(Y=1 \mid X; \theta) = \sigma(z(\theta, x))$

$$= \frac{1}{1 + \exp(-\theta_0 - \sum_{i=1}^{n} \theta_i x_i)}$$

Then, the decision boundary can be set to be $P(Y=1 \mid X; \theta) > 0.5$

(2) non-linear dependence:

let $h(A, b, x) = f(Ax+b)$ be a non-linear transformation of the inputs

$$P_{Neural}(Y=1 \mid X; A, b, \theta) = f(\theta_0 + \sum_{i=1}^{n} \theta_i h_i(A, b, x))$$

└ these have more parameters to learn but are also more flexible.

Autoregressive Models

We order all the random variables like ordering the pixels from top-left to bottom-right.

Then,

$$P(x_1, \ldots, x_n) = P(X_1)\, P(X_2 \mid X_1) \cdots P(X_n \mid X_1, \ldots, X_{n-1})$$

Maximum Likelihood Estimation

Assume that the data comes from P_{data}.
Dataset, D, of m samples from P_{data}.
Assume that the data instances are i.i.d.
We have a family of models $\mathcal{M}$. We learn the optimal model $P_\theta \in \mathcal{M}$.

Goal is to learn P_{data} itself so we can answer any probabilistic inference query afterwards.
How do we do so?

(i) Minimize $D_{KL}(P \| Q) = \sum_x P(x) \log \frac{P(x)}{Q(x)}$

Note: • $D_{KL}(P\|Q)$ is ≥ 0 and $= 0$ iff $P = Q$

This is because

$$D_{KL}(P\|Q) = \mathbb{E}_{x\sim P}\left[-\log \frac{Q(x)}{P(x)}\right] \geq -\log \mathbb{E}_{x\sim P}\left[\frac{Q(x)}{P(x)}\right] \text{ (by Jensen)}$$
$$= -\log(1) = 0$$

• Also, $D_{KL}(P\|Q) \neq D_{KL}(Q\|P)$

So, we $\boxed{\text{minimize } D_{KL}(P_{data} \| P_\theta)}$ which is 0 iff $P_{data} = P_\theta$.

Note: $D_{KL}(P_{data} \| P_\theta) = \mathbb{E}_{x\sim P_{data}}\left[\log\left(\frac{P_{data}(x)}{P_\theta(x)}\right)\right]$

$$= \underbrace{\mathbb{E}_{x\sim P_{data}}\left[\log(P_{data}(x))\right]}_{\text{independent of } P_\theta} - \mathbb{E}_{x\sim P_{data}}\left[\log(P_\theta(x))\right]$$

$$\therefore \arg\min_{P_\theta} D_{KL}(P_{data}\|P_\theta) = \arg\max_{P_\theta} \mathbb{E}_{x\sim P_{data}}\left[\log(P_\theta(x))\right]$$

↑ minimize KL divergence between P_{data} and P_θ ⟷ ↑ Maximise log probability of samples in the dataset!

This provides the justification for maximizing log likelihood in ML.

■ Issues: since we ignore $H(P_{data}) = -\mathbb{E}_{x\sim P_{data}}\left[\log P_{data}(x)\right]$,
we do not know how close we are to the optimum.
Also, we don't really know P_{data}, so how do we maximize log probability.

■ Solutions:
Option 1: approximate $\mathbb{E}_{x\sim P_{data}}\left[\log(P_\theta(x))\right]$
with $\mathbb{E}_D\left[\log(P_\theta(x))\right] = \frac{1}{|D|}\sum_{x\in D}\log(P_\theta(x))$

Then, maximum likelihood learning is

$$\max_{p_\theta} \frac{1}{|D|} \sum_{x \in D} \log p_\theta(x)$$

for autoregressive models, we maximize the likelihood function

$$L(\theta, D) = \prod_{j=1}^{m} \prod_{i=1}^{n} p_{neural}(x_i^{(j)} | x_{<i}^{(j)}; \theta)$$

Aside: this is an example of Monte Carlo Estimation

approximation

$$\mathbb{E}_{x \sim p}[g(x)]$$

with

$$\frac{1}{T} \sum_{t=1}^{T} g(x^t) =: \hat{g}$$

where x^t are independent samples from p.

→ this is an unbiased estimation:

$$\mathbb{E}_p[\hat{g}] = \mathbb{E}_p[g(x)]$$

→ convergence: by weak law of large no.

$$\hat{g} \longrightarrow \mathbb{E}_p[g(x)] \text{ as } T \to \infty.$$

→ variance:

$$V_p[\hat{g}] = V_p\left[\frac{1}{T} \sum_{i=1}^{T} g(x^i)\right] = \frac{V_p[g(x)]}{T}$$

So, $V_p[\hat{g}] \longrightarrow 0$ as $T \longrightarrow \infty$

Variational Autoencoders

Latent Variable Models

For our datapoints X, we aim to learn high-level features Z. If the features are learned properly, learning $P(X|Z)$ could be easier than learning $P(X)$ itself.

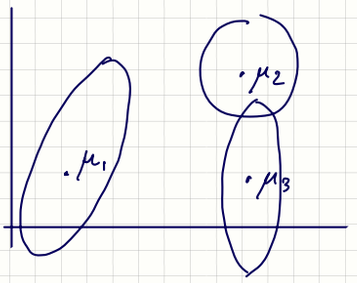
Example setup:
(1) features $Z \sim \mathcal{N}(0, I)$. We hope that Z corresponds to meaningful features
(2) Train $p_\theta(X|Z) := \mathcal{N}(x | \mu_\theta(z), \Sigma_\theta(z))$

where μ_θ and Σ_θ are neural networks.

Now, we look at some examples of latent variable models.

(1) Mixture of Gaussians

$$Z \sim \text{Categorical}(1, \ldots, k)$$
$$p_\theta(x|z=k) := \mathcal{N}(x | \mu_k, \Sigma_k)$$



Then, we can produce new data as

$$\begin{aligned} p(x) &= \sum_z p(x, z) \\ &= \sum_z p(z)\, p(x|z) \\ &= \sum_{i=1}^{k} p(z=i)\, \mathcal{N}(x | \mu_i, \Sigma_i) \end{aligned}$$

→ Alternative motivation is that each $\mathcal{N}(x | \mu_i, \Sigma_i)$ is a simple model and we are combining them.

→ Example construction:

$$\mu_\theta(z) = \sigma(Az + c)$$
$$\Sigma_\theta(z) = \text{diag}\left(\exp\left(\sigma(Bz + d)\right)\right) = \begin{pmatrix} \exp(\sigma(b_1 z + d_1)) & 0 \\ 0 & \exp(\sigma(b_2 z + d_2)) \end{pmatrix}$$

Issue: these are fairly difficult to learn

We will use the following elements:
(1) ELBO
(2) Reparametrization
(3) Amortization

(2) Partially observed data: In most real-world settings, the datapoints X are observed in dataset $\mathcal{D}$ and the variables Z are not observed.

If we learned $p_\theta(X, Z)$, we could do max. log likelihood:

$$\log \prod_{x \in \mathcal{D}} p_\theta(x) = \sum_{x \in \mathcal{D}} \log p_\theta(x) = \sum_{x \in \mathcal{D}} \log \left(\sum_z p_\theta(x, z) \right)$$

But evaluating $\log \left(\sum_z p_\theta(x, z) \right)$ is intractable: if we have 30 binary features / latents, we need to sum over 2^{30} terms and if Z is continuous, then this is often intractable + gradient is hard to compute.

We have a few approaches them.

For computing $p_\theta(x)$:

Approach 1: Naive Monte Carlo

$$p_\theta(x) = \sum_z p_\theta(x, z) \approx |\mathcal{Z}| \sum_{z \in \mathcal{Z}} \frac{1}{|\mathcal{Z}|} p_\theta(x, z)$$

$$= |\mathcal{Z}| \mathop{\mathbb{E}}_{z \sim \text{Uniform}} \left[p_\theta(x, z) \right]$$

Then,

$$\sum_z p_\theta(x, z) \approx |\mathcal{Z}| \frac{1}{k} \sum_{j=1}^{k} p_\theta(x, z^{(j)})$$

Issue: for most z, $p_\theta(x, z)$ is very low

Approach 2: Importance sampling

$$p_\theta(x) = \sum_z p_\theta(x, z) = \sum_z \frac{q(z)}{q(z)} p_\theta(x, z) = \mathop{\mathbb{E}}_{z \sim q(z)} \left[\frac{p_\theta(x, z)}{q(z)} \right]$$

Then, we can do Monte Carlo to compute this:

$$p_\theta(x) \approx \frac{1}{k} \sum_{j=1}^{k} \frac{p_\theta(x, z^{(j)})}{q(z^{(j)})}$$

For computing $\log p_\theta(x)$:

How do we compute $\log p_\theta(x)$?

$$\log(p_\theta(x)) \approx \log \left(\frac{1}{k} \sum_{j=1}^{k} \frac{p_\theta(x, z^{(j)})}{q(z^{(j)})} \right)$$

$$\overset{k=1}{\approx} \log \left(\frac{p_\theta(x, z^{(j)})}{q(z^{(j)})} \right)$$

But this would be a bad approximation: as

$$\log \left(\mathop{\mathbb{E}}_{z \sim q(z)} \left[\frac{p_\theta(x, z)}{q(z)} \right] \right) \neq \mathop{\mathbb{E}}_{z \sim q(z)} \left[\log \left(\frac{p_\theta(x, z)}{q(z)} \right) \right]$$

Approach 3: ELBO

$$\log p_\theta(x) = \log\left(\mathbb{E}_{z\sim q(z)}\left[\frac{p_\theta(x,z)}{q(z)}\right]\right) \geq \mathbb{E}_{z\sim q(z)}\left[\log\left(\frac{p_\theta(x,z)}{q(z)}\right)\right]$$

by importance sampling

by Jensen's inequality

Evidence Lower Bound (ELBO)

Recall: $q(z)$ can be any distribution over z

→ Note:

$$\log p_\theta(x) \geq \mathbb{E}_{z\sim q(z)}\left[\log \frac{p_\theta(x,z)}{q(z)}\right]$$

$$= \mathbb{E}_{z\sim q(z)}\left[\log p_\theta(x,z)\right] - \underbrace{\mathbb{E}_{z\sim q(z)}\left[\log q(z)\right]}_{\text{Entropy of } q}$$

→ Equality holds if $q = p_\theta(z|x)$

∴ Ideal choice of q is $q = p_\theta(z|x)$

Issue: what if $p_\theta(z|x)$ is hard to compute? How loose is this bound then?

→ Suppose $q(z)$ is any probability distribution.

Then,

$$D_{KL}(q(z) \,\|\, p_\theta(z|x)) = -\sum_z q(z)\log p_\theta(z,x) + \log p_\theta(x) - H(q) \geq 0$$

Thm, note that ELBO is equal to $\log p_\theta(x)$ if $q = p_\theta(z|x)$ (as $D_{KL}(q(z) \,\|\, p_\theta(z|x)) = 0$)

So, this KL-divergence term is a measure of the looseness of our bound.

→ we train $q_\phi(z)$

Eg: $q_\phi(z) = \mathcal{N}(\phi_1, \phi_2)$

Thm, we do variational inference over ϕ so that $q_\phi(z)$ is close to $p_\theta(z|x)$.

Recap:

$$\log p_\theta(x) \geq \sum_z q_\phi(z)\log p_\theta(x,z) + H(q_\phi(z)) = \mathcal{L}_{ELBO}(x;\theta,\phi)$$

and $$\log p_\theta(x) = \mathcal{L}_{ELBO}(x;\theta,\phi) + D_{KL}(q_\phi(z) \,\|\, p_\theta(z|x))$$

Now, this was for one datapoint i.e we made $q_\phi(z)$ close to $p_\theta(z|x)$. Then, for each datapoint $x^{(i)}$ we would have $\phi^{(i)}$ & train $q_{\phi^{(i)}}(z)$. If we did this, our MLE becomes

$$\ell(\theta;D) = \sum_{x^{(i)}\in D} \log p_\theta(x^{(i)}) \geq \sum_{x^{(i)}\in D} \mathcal{L}(x^{(i)};\theta,\phi^{(i)})$$

Issue : The problem is in optimizing this.

$$\mathcal{L}(x^{(i)}, \theta, \phi^{(i)}) = \mathbb{E}_{z \sim q_\phi^{(i)}(z)}\left[\log p_\theta(x^{(i)}|z) - \log q_\phi^{(i)}(z)\right]$$

$$\nabla_\theta \mathcal{L}(x^{(i)}, \theta, \phi^{(i)}) = \nabla_\theta \mathbb{E}_{z \sim q_\phi^{(i)}(z)}\left[\log p_\theta(x^{(i)}, z)\right]$$

$$= \mathbb{E}_{z \sim q_\phi^{(i)}(z)}\left[\nabla_\theta \log p_\theta(x^{(i)}, z)\right]$$

$$\approx \frac{1}{k} \sum_k \nabla_\theta \log p_\theta(x^{(i)}, z^k)$$

But $\nabla_\phi \mathcal{L}(x^{(i)}, \theta, \phi^{(i)})$ is hard as we have expectation wrt $q_\phi^{(i)}$.

Solution 1 : Reparametrization

In general suppose we want $\nabla_\phi \mathbb{E}_{z \sim q_\phi(z)}[r(z)]$ where z is continuous.

Now, note : if $q_\phi(z) = \mathcal{N}(\mu, \sigma^2 I)$

then

$$z = \mu + \sigma\varepsilon, \quad \varepsilon \sim \mathcal{N}(0, I)$$

$$\Rightarrow z = g_\phi(\varepsilon) \quad \text{where } g \text{ is deterministic}$$

Then,

$$\mathbb{E}_{z \sim q_\phi(z)}[r(z)] = \int q_\phi(z) r(z) dz$$

$$= \mathbb{E}_{\varepsilon \sim \mathcal{N}(0,I)}\left[r(g_\phi(\varepsilon))\right]$$

$$= \int \mathcal{N}(\varepsilon) r(\mu + \sigma\varepsilon) d\varepsilon$$

Then, $\nabla_\phi \mathbb{E}_{q_\phi(z)}[r(z)] = \mathbb{E}_\varepsilon\left[\nabla_\phi r(g_\phi(\varepsilon))\right]$

$$\approx \frac{1}{k} \sum_k \nabla_\phi r(g_\phi(\varepsilon^k))$$

In our case, we assume $z = \mu + \sigma\varepsilon$ and then use this reparametrisation technique.

Amortization

Instead of learning $\phi^{(i)}$ for $x^{(i)}$, $i \in \{1, \ldots, |D|\}$, we learn a single f_λ s.t

$$f_\lambda(x^{(i)}) \approx \phi^{(i)} \text{ for some good parameter } \phi^{(i)}.$$

Then, we approximate $q(z|x^{(i)})$ using $q_{\phi^{(i)}}(z)$ which is approximated with

$$q_{f_\lambda(x^{(i)})}(z) = q(z; f_\lambda(x^{(i)}))$$

Putting all of this together ....

Autoencoders

loss function : $\mathcal{L}(x; \theta, \phi) = \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x,z) - \log q_\phi(z|x)\right]$

$= \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x,z) - \log p(z) + \log p(z) - \log q_\phi(z|x)\right]$

$= \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x|z) - D_{KL}\left(q_\phi(z|x) \,\|\, p(z)\right)\right]$

Recipe : Take $x^{(i)}$, map it to $z^{(i)}$ using $\underbrace{q(z^{(i)}; f_\lambda(x^{(i)}))}_{\text{encoder}}$

Reconstruct $x^{(i)}$ by sampling $p_\theta(x|z^{(i)})$

Energy-based Models (EBMs)

When we try to parametrize $p_\theta(x)$, what are the constraints?

(1) $p_\theta(x) \geq 0$

(2) $\sum_x p_\theta(x) = 1$ or $\int p_\theta(x)dx = 1$

The first is easy but generally $\sum_x p_\theta(x)$ need not be 1.

Solution:

Set $p_\theta(x) = \frac{1}{Z(\theta)} g_\theta(x) = \frac{1}{\int g_\theta(x)dx} g_\theta(x)$

Then, we choose g_θ s.t we know $Z(\theta) = \int g_\theta(x)dx$

Examples:

(1) $g_{\mu,\sigma}(x) = \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$ → Gaussian

$Z(\mu,\sigma) = \sqrt{2\pi\sigma^2}$

(2) $g_\lambda(x) = e^{-\lambda x}$ → Exponential

$Z(\lambda) = \frac{1}{\lambda}$

(3) generally....

$g_\theta(x) = h(x)\exp\{\theta^T T(x)\}$

$Z(\theta) = \exp(A(\theta)) = \int h(x)\exp(\theta^T T(x))dx$

→ Exponential family

Energy-based models: $p_\theta(x) = \frac{1}{\int \exp(f_\theta(x))dx} \cdot \exp(f_\theta(x))$

$= \frac{1}{Z(\theta)} \exp(f_\theta(x))$

→ called the partition function.

We use $\exp(f_\theta(x))$ to capture very large variations in probability + exponential family models have this structure.

Pros: flexibility in choosing $f_\theta(x)$

Cons: (1) Sampling difficulty

(2) Evaluating is hard

(3) No feature learning

(4) Curse of dimensionality: computing $Z(\theta)$

↳ however, some tasks don't require knowing $Z(\theta)$ like checking which sample is more likely:

$$\frac{p_\theta(x)}{p_\theta(x')} = \exp(f_\theta(x) - f_\theta(x'))$$

Goal : $\max \frac{\exp(f_\theta(x_{train}))}{Z(\theta)} = p_\theta(x_{train})$

Training Approach 1 : Monte Carlo estimate

instead of calculating $Z(\theta)$ exactly.
use contrastive divergence algorithm :

sample $x_{sample} \sim p_\theta$

Then, take step along

$$\nabla_\theta (f_\theta(x_{train}) - f_\theta(x_{sample}))$$

Make training data more likely than typical sample

This can be derived :

$$\nabla_\theta \log p_\theta(x) = \nabla_\theta f_\theta(x_{train}) - \nabla_\theta \log Z(\theta)$$

$$= \nabla_\theta f_\theta(x_{train}) - \frac{\nabla_\theta Z(\theta)}{Z(\theta)}$$

$$= \nabla_\theta f_\theta(x_{train}) - \frac{1}{Z(\theta)} \int \nabla_\theta \exp(f_\theta(x)) dx$$

$$= \nabla_\theta f_\theta(x_{train}) - \frac{1}{Z(\theta)} \int (\nabla_\theta f_\theta(x)) e^{f_\theta(x)} dx$$

$$= \nabla_\theta f_\theta(x_{train}) - \int p_\theta(x) \cdot \nabla_\theta f_\theta(x) dx$$

$$= \nabla_\theta f_\theta(x_{train}) - \mathbb{E}_{x \sim p_\theta}[\nabla_\theta f_\theta(x)]$$

$$\approx \nabla_\theta f_\theta(x_{train}) - \nabla_\theta f_\theta(x_{sample})$$

Sampling Approach

(1) Markov Chain Monte Carlo (MCMC) :

Algorithm :

Initialize x^0, $t = 0$

Let $x' = x^t + \text{noise}$:

if $f_\theta(x') > f_\theta(x^t)$, $x^{t+1} := x'$

else $x^{t+1} := x'$ with probability $\exp(f_\theta(x') - f_\theta(x^t))$

Issue : takes a long time to converge.

(2) Langevin MCMC :

Suppose we can compute the score function $\nabla_x \log p_\theta(x)$

Let $\pi(x)$ be a prior dist. that is easy to sample from.

Algorithm :

$x^0 \sim \pi(x)$

$x^{t+1} \sim x^t + \epsilon \nabla_x \log p_\theta(x^t) + \sqrt{2\epsilon}\, z^t$, $t = 0, \ldots, T-1$ where $z^t \sim \mathcal{N}(0, I)$

if $\epsilon \to 0$ & $T \to \infty$, then $x^T \sim p_\theta(x)$.

Note: $\nabla_x \log p_\theta(x) = \nabla_x f_\theta(x) - \underbrace{\nabla_x \log Z(\theta)}_{0}$

$= \nabla_\theta f_\theta(x)$

Score Matching

Recall : energy based models : $p_\theta(x) = \frac{\exp(f_\theta(x))}{Z(\theta)}$

Def : (Stein) score function

$$s_\theta(x) := \nabla_x \log p_\theta(x)$$
$$= \nabla_x f_\theta(x) - \nabla_x \log Z(\theta)$$
$$= \nabla_x f_\theta(x)$$

Examples :

(1) Gaussian distribution :

$$p_\theta(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(x-\mu)^2/2\sigma^2}$$

Then, $s_\theta(x) = -\frac{x-\mu}{\sigma^2}$

(2) Gamma distribution :

$$p_\theta(x) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}$$

Then, $s_\theta(x) = \frac{\alpha-1}{x} - \beta$

Note :

(1) $s_\theta(x)$ is independent of $Z(\theta)$, the partition function.

Recall : fisher divergence between $p(x)$ and $q(x)$

$$D_F(p \| q) := \frac{1}{2} \mathbb{E}_{x\sim p}\left[\| \nabla_x \log p(x) - \nabla_x \log q(x) \|_2^2 \right]$$

Score matching :

Minimize the fisher divergence between $p_{data}(x)$ and the EBM $p_\theta(x) \propto \exp(f_\theta(x))$:

$$\min. \; D_F(p_{data} \| p_\theta)$$
$$= \frac{1}{2} \mathbb{E}_{x\sim p_{data}}\left[\| \nabla_x \log p_{data}(x) - s_\theta(x) \|_2^2 \right]$$
$$= \frac{1}{2} \mathbb{E}_{x\sim p_{data}}\left[\| \nabla_x \log p_{data}(x) - \nabla_x f_\theta(x) \|_2^2 \right]$$

How do we compute this? Consider the univariate case for simplicity:

$$\frac{1}{2}\,\mathbb{E}_{x\sim p_{data}}\left[\left(\nabla_x \log p_{data}(x) - \nabla_x \log p_\theta(x)\right)^2\right]$$

$$= \frac{1}{2}\int p_{data}(x)\left(\nabla_x \log p_{data}(x) - \nabla_x \log p_\theta(x)\right)^2 dx$$

$$= \frac{1}{2}\int p_{data}(x)\left(\nabla_x \log p_{data}(x)\right)^2 dx - \int p_{data}(x)\,\nabla_x \log p_{data}(x)\,\nabla_x \log p_\theta(x)\,dx + \frac{1}{2}\int p_{data}(x)\left(\nabla_x \log p_\theta(x)\right)^2 dx$$

Now

$$-\int p_{data}(x)\,\nabla_x \log p_{data}(x)\,\nabla_x \log p_\theta(x)\,dx$$

$$= -\int p_{data}(x)\,\frac{1}{p_{data}(x)}\,\nabla_x p_{data}(x)\,\nabla_x \log p_\theta(x)\,dx$$

$$= -\,p_{data}(x)\,\nabla_x \log p_\theta(x)\Big|_{-\infty}^{\infty} + \int p_{data}(x)\,\nabla_x^2 \log p_\theta(x)\,dx$$

(assuming $p_{data}(x) \to 0$, as $x \to \pm\infty$)

$$= \int p_{data}(x)\,\nabla_x^2 \log p_\theta(x)\,dx$$

Then,

$$\frac{1}{2}\,\mathbb{E}_{x\sim p_{data}}\left[\left(\nabla_x \log p_{data}(x) - \nabla_x \log p_\theta(x)\right)^2\right]$$

$$= \underbrace{\frac{1}{2}\int p_{data}(x)\left(\nabla_x \log p_{data}(x)\right)^2 dx}_{\text{constant wrt } \theta} + \frac{1}{2}\int p_{data}(x)\left(\nabla_x \log p_\theta(x)\right)^2 dx + \int p_{data}(x)\,\nabla_x^2 \log p_\theta(x)\,dx$$

$$= \mathbb{E}_{x\sim p_{data}}\left[\frac{1}{2}\left(\nabla_x \log p_\theta(x)\right)^2 + \nabla_x^2 \log p_\theta(x)\right] + \text{constant}$$

Similarly, in the multivariate case:

$$\frac{1}{2}\,\mathbb{E}_{x\sim p_{data}}\left[\left\|\nabla_x \log p_{data}(x) - \nabla_x \log p_\theta(x)\right\|_2^2\right]$$

$$= \mathbb{E}_{x\sim p_{data}}\left[\frac{1}{2}\left\|\nabla_x \log p_\theta(x)\right\|_2^2 + \operatorname{tr}\left(\nabla_x^2 \log p_\theta(x)\right)\right] + \text{constant}$$

Final objective: minimize:

$$\approx \frac{1}{n}\sum_{i=1}^{n}\left[\frac{1}{2}\left\|\nabla_x f_\theta(x_i)\right\|_2^2 + \operatorname{tr}\left(\nabla_x^2 f_\theta(x_i)\right)\right]$$

Issue: $\nabla_x^2 \log p_\theta(x)$ is computationally expensive

Score-based Models

The ideas in score matching for EBMs can be generalized:

Paper: Generative Modelling by Estimating Gradients of the data distribution (Score-based Models)

Score-based Generative Modeling

Dataset (iid samples): $\{x_i \in \mathbb{R}^D\}_{i=1}^N$

Data Distribution: $p_{data}(x)$

Score of a probability density $p(x)$: $\nabla_x \log p(x)$

Score Network: $s_\theta : \mathbb{R}^D \to \mathbb{R}^D$ → Train s_θ to approximate the score of $p_{data}(x)$

Goal: Learn a model to generate new samples from $p_{data}(x)$

2 steps → ① Score matching ② Langevin dynamics

① Score matching for score estimation:

Train NN $s_\theta(x)$ to estimate $\nabla_x \log p(x)$ without training a model to estimate $p_{data}(x)$ first. → ①

goal: $\underset{\theta}{\arg\min} \frac{1}{2} \mathbb{E}_{p_{data}(x)}\left[\| s_\theta(x) - \nabla_x \log p_{data}(x) \|_2^2 \right]$

② Denoising score matching:

First perturbs data point x with a pre-specified noise distribution: $q_\sigma(\tilde{x}|x)$ → (noising x)

$$\tilde{x} = x + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2 I)$$

$$q_\sigma(\tilde{x}|x) := \frac{1}{(2\pi)^{d/2}\sigma^d} \exp\left(-\frac{1}{2\sigma^2}\|\tilde{x} - x\|^2\right)$$

$$q_\sigma(\tilde{x}) := \int q_\sigma(\tilde{x}|x) p_{data}(x)\, dx$$

If $q_\sigma(\tilde{x}) \approx p_{data}(\tilde{x})$ (given noise is small), then we write ①

as (using Theorem 1)

$$\underset{\theta}{\arg\min} \frac{1}{2} \mathbb{E}_{q_\sigma(\tilde{x}|x) p_{data}(x)}\left[\| s_\theta(\tilde{x}) - \nabla_{\tilde{x}} \log q_\sigma(\tilde{x}|x) \|_2^2 \right]$$

Theorem 1:

$$\mathbb{E}_{q_\sigma(x,\tilde{x})}\left[\frac{1}{2}\left\| s_\theta(\tilde{x}) - \frac{\partial}{\partial \tilde{x}} \log q_\sigma(\tilde{x}|x)\right\|^2\right] = \mathbb{E}_{q_\sigma(\tilde{x})}\left[\frac{1}{2}\left\| s_\theta(\tilde{x}) - \frac{\partial}{\partial \tilde{x}} \log q_\sigma(\tilde{x})\right\|^2\right]$$

Proof:

$$\frac{\partial}{\partial \tilde{x}} \log q_\sigma(\tilde{x}|x) = \frac{1}{\sigma^2}(\tilde{x} - x)$$

Start from RHS

$$\mathbb{E}_{q_\sigma(\tilde{x})}\left[\frac{1}{2}\left\| s_\theta(\tilde{x})\right\|^2\right] - A(\theta) + C_2$$

where $C_2 := \mathbb{E}_{q_\sigma(\tilde{x})}\left[\frac{1}{2}\left\| \frac{\partial \log q_\sigma(\tilde{x})}{\partial \tilde{x}}\right\|^2\right]$ independant of θ

and $A(\theta) := \mathbb{E}_{q_\sigma(\tilde{x})}\left[\left\langle s_\theta(\tilde{x}), \frac{\partial \log q_\sigma(\tilde{x})}{\partial \tilde{x}}\right\rangle\right]$

$$= \int_{\tilde{x}} q_\sigma(\tilde{x}) \left\langle s_\theta(\tilde{x}), \frac{\partial \log q_\sigma(\tilde{x})}{\partial \tilde{x}}\right\rangle d\tilde{x}$$

$$= \int_{\tilde{x}} q_\sigma(\tilde{x}) \left\langle s_\theta(\tilde{x}), \frac{\frac{\partial}{\partial \tilde{x}} q_\sigma(\tilde{x})}{q_\sigma(\tilde{x})}\right\rangle d\tilde{x}$$

$$= \int_{\tilde{x}} \left\langle s_\theta(\tilde{x}), \frac{\partial}{\partial \tilde{x}} q_\sigma(\tilde{x})\right\rangle d\tilde{x}$$

$$= \int_{\tilde{x}} \left\langle s_\theta(\tilde{x}), \frac{\partial}{\partial \tilde{x}} \int_x \underbrace{q_0(x)}_{p_{data}(x)} q_\sigma(\tilde{x}|x)\, dx\right\rangle d\tilde{x}$$

$$= \int_{\tilde{x}} \left\langle s_\theta(\tilde{x}), \int_x q_0(x) \frac{\partial q_\sigma(\tilde{x}|x)}{\partial \tilde{x}} dx\right\rangle d\tilde{x}$$

$$= \int_{\tilde{x}} \left\langle s_\theta(\tilde{x}), \int_x q_0(x)\, q_\sigma(\tilde{x}|x) \frac{\partial}{\partial \tilde{x}} \log q_\sigma(\tilde{x}|x)\, dx\right\rangle d\tilde{x}$$

$$= \int_{\tilde{x}} \int_x q_0(x) q_\sigma(\tilde{x}|x) \left\langle s_\theta(\tilde{x}), \frac{\partial}{\partial \tilde{x}} \log q_\sigma(\tilde{x}|x)\right\rangle dx\, d\tilde{x}$$

$$= \int_{\tilde{x}} \int_x q_\sigma(\tilde{x}, x) \left\langle s_\theta(\tilde{x}), \frac{\partial}{\partial \tilde{x}} \log q_\sigma(\tilde{x}|x)\right\rangle dx\, d\tilde{x}$$

$$= \mathbb{E}_{q_\sigma(\tilde{x},x)}\left[\left\langle s_\theta(\tilde{x}), \frac{\partial}{\partial \tilde{x}} \log q_\sigma(\tilde{x}|x)\right\rangle\right]$$

$\therefore$ RHS becomes

$$\mathbb{E}_{q_\sigma(\tilde{x})}\left[\frac{1}{2}\left\| s_\theta(\tilde{x})\right\|^2\right] - \mathbb{E}_{q_\sigma(x,\tilde{x})}\left[\left\langle s_\theta(\tilde{x}), \frac{\partial}{\partial \tilde{x}} \log q_\sigma(\tilde{x}|x)\right\rangle\right] + C_2$$

Now, from LHS, we can write

$$\mathbb{E}_{q_\sigma(\tilde{x})}\left[\frac{1}{2}\left\| s_\theta(\tilde{x})\right\|^2\right] - \mathbb{E}_{q_\sigma(x,\tilde{x})}\left[\left\langle s_\theta(\tilde{x}), \frac{\partial}{\partial \tilde{x}} \log q_\sigma(\tilde{x}|x)\right\rangle\right] + C_3$$

② Langevin Dynamics (L.D)

L.D produces samples from $p(x)$ using only the score function $\nabla_x \log p(x)$.

fix a step size $\epsilon > 0$ and $\tilde{x}_0 \sim \pi(x)$ with π being a prior distribution

L.D method (recursively):

$$\tilde{x}_t := \tilde{x}_{t-1} + \frac{\epsilon}{2} \underbrace{\nabla_x \log p(\tilde{x}_{t-1})}_{\approx s_\theta(\tilde{x}_{t-1})} + \sqrt{\epsilon}\, z_t, \quad z_t \sim \mathcal{N}(0, I)$$

Distribution of $\tilde{x}_T \approx p(x)$ when $\epsilon \to 0$, $T \to \infty$

Diffusion Models

DDPM

Paper : Denoising Diffusion Probabilistic Models.

Notation :

→ Data distribution : $x_0 \sim q(x_0)$

→ Learned parameterised distribution : $p_\theta(x)$

Forward Process (Noising)

$$q(x_{1:T} \mid x_0) := \prod_{t=1}^{T} q(x_t \mid x_{t-1})$$

where

$$q(x_t \mid x_{t-1}) := \mathcal{N}\left(x_t \,;\, \sqrt{1-\beta_t}\, x_{t-1},\, \beta_t I\right) \longrightarrow (1)$$

We can actually sample x_t, given x_0, in one step. To do so, define

$$\alpha_t = 1 - \beta_t$$

$$\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$$

Let $\epsilon_0, \ldots\ldots, \epsilon_{t-1} \sim \mathcal{N}(0, 1)$

Then,

$$x_t = \sqrt{1-\beta_t}\, x_{t-1} + \sqrt{\beta_t}\, \epsilon_{t-1} \qquad \text{(from (1))}$$

$$= \sqrt{\alpha_t}\, x_{t-1} + \sqrt{1-\alpha_t}\, \epsilon_{t-1}$$

$$= \sqrt{\alpha_t}\left(\sqrt{\alpha_{t-1}}\, x_{t-2} + \sqrt{1-\alpha_{t-1}}\, \epsilon_{t-2}\right) + \sqrt{1-\alpha_t}\, \epsilon_{t-1}$$

$$= \sqrt{\alpha_t \alpha_{t-1}}\, x_{t-2} + \sqrt{\alpha_t}\sqrt{1-\alpha_{t-1}}\, \epsilon_{t-2} + \sqrt{1-\alpha_t}\, \epsilon_{t-1}$$

Now, $\sqrt{\alpha_t}\sqrt{1-\alpha_{t-1}}\, \epsilon_{t-2} \sim \mathcal{N}(0,\, \alpha_t(1-\alpha_{t-1}))$

and $\sqrt{1-\alpha_t}\, \epsilon_{t-1} \sim \mathcal{N}(0,\, 1-\alpha_t)$

$$\therefore \sqrt{\alpha_t}\sqrt{1-\alpha_{t-1}}\, \epsilon_{t-2} + \sqrt{1-\alpha_t}\, \epsilon_{t-1} \sim \mathcal{N}(0,\, \alpha_t(1-\alpha_{t-1}) + (1-\alpha_t))$$

$$\therefore x_t = \sqrt{\alpha_t \alpha_{t-1}}\, x_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\, \epsilon_{t-2} \longrightarrow (2)$$

$$\therefore x_t \sim q(x_t \mid x_0) = \mathcal{N}\left(x_t \,;\, \sqrt{\bar{\alpha}_t}\, x_0,\, (1-\bar{\alpha}_t) I\right) \longrightarrow (3)$$

Backward Process (denoising)

The reverse process is also a Markov chain with learned Gaussian noise

Start by sampling $x_T \sim p(x_T) = \mathcal{N}(x_T ; 0, I)$

Then,

$$p_\theta(x_{0:T}) = p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1} | x_t)$$

where $p_\theta(x_{t-1}|x_t) = \mathcal{N}\left(x_{t-1} ; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)\right)$

(μ_θ, Σ_θ: learned.)

- Version 1 of training loss function:
  Maximize log probability:

$$\begin{aligned}
\log p_\theta(x_0) &= \log \int_{x_{1:T}} p_\theta(x_{0:T})\, dx_{1:T} \\
&= \log \int_{x_{1:T}} p_\theta(x_{0:T}) \frac{q(x_{1:T}|x_0)}{q(x_{1:T}|x_0)}\, dx_{1:T} \\
&= \log \mathbb{E}_{q(x_{1:T}|x_0)}\left[\frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)}\right] \\
&\geq \mathbb{E}_{q(x_{1:T}|x_0)}\left[\log \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)}\right]
\end{aligned}$$

Then, we minimize:

$$\begin{aligned}
-\mathbb{E}_{x_0 \sim q(\cdot)}\left[\log p_\theta(x_0)\right] &\leq \mathbb{E}_{q(\cdot)}\left[-\log \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)}\right] \\
&= \mathbb{E}_q\left[-\log p_\theta(x_T) - \sum_{t \geq 1} \log \frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1})}\right]
\end{aligned}$$

$\therefore$ loss function is:

$$L = \mathbb{E}_{q(x_{0:T})}\left[-\log p_\theta(x_T) - \sum_{t \geq 1} \log \frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1})}\right]$$

Version 2 (with variance reduction)

$$L = \mathbb{E}_q \left[-\log p_\theta(x_T) - \sum_{t \geq 1} \log \frac{p_\theta(x_{t-1} | x_t)}{q(x_t | x_{t-1})} \right]$$

$$= \mathbb{E}_q \left[-\log p_\theta(x_T) - \sum_{t>1} \log \frac{p_\theta(x_{t-1} | x_t)}{q(x_t | x_{t-1})} - \log \frac{p_\theta(x_0 | x_1)}{q(x_1 | x_0)} \right]$$

$$= \mathbb{E}_q \left[-\log p_\theta(x_T) - \sum_{t>1} \log \frac{p_\theta(x_{t-1} | x_t)}{q(x_{t-1} | x_t, x_0)} \frac{q(x_{t-1} | x_t, x_0)}{q(x_t | x_0)} - \log \frac{p_\theta(x_0 | x_1)}{q(x_1 | x_0)} \right]$$

(Bayes Law)

$$= \mathbb{E}_{q(x_{0:T})} \left[-\log \frac{p_\theta(x_T)}{q(x_T | x_0)} - \log p_\theta(x_0 | x_1) - \sum_{t>1} \log \frac{p_\theta(x_{t-1} | x_t)}{q(x_{t-1} | x_t, x_0)} \right]$$

$$= \mathbb{E}_{q(x_{0:T})} \left[D_{KL}\left(q(x_T | x_0) \,\|\, p_\theta(x_T) \right) - \log p_\theta(x_0 | x_1) + \sum_{t>1} D_{KL}\left(q(x_{t-1} | x_t, x_0) \,\|\, p_\theta(x_{t-1} | x_0) \right) \right]$$

Now,

$$q(x_{t-1} | x_t, x_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}_t(x_t, x_0), \tilde{\beta}_t I)$$

where $\tilde{\mu}_t(x_t, x_0) = \frac{\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_t} \beta_t x_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} x_t$

and $\tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \beta_t$

Thus,

$$L = \mathbb{E}_{q(x_{0:T})} \left[\overbrace{D_{KL}\left(q(x_T | x_0) \,\|\, p(x_T) \right)}^{L_T} \overbrace{- \log p_\theta(x_0 | x_1)}^{L_0} + \underbrace{\sum_{t>1} D_{KL}\left(q(x_{t-1} | x_t, x_0) \,\|\, p_\theta(x_{t-1} | x_0) \right)}_{L_{t-1}} \right]$$

Version 3 (DDPM) :

Consider β_t to be constant for all t.

Thm. L_T is independent of θ (recall: $p_\theta(x_T) = \mathcal{N}(0, I)$) so we ignore it.

Recall:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}\left(x_{t-1};\ \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)\right)$$

$$q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1};\ \tilde{\mu}_t(x_t, x_0),\ \tilde{\beta}_t I)$$

where $\tilde{\mu}_t(x_t, x_0) = \frac{\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_t}\beta_t x_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} x_t$

and $\tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$

Set $\Sigma_\theta(x_t, t) := \sigma_t^2 I$

and set $\sigma_t^2 = \beta_t$ or $\sigma_t^2 = \tilde{\beta}_t$ → either works

↑ Better for $x_T \sim \mathcal{N}(0, 1)$

So, $\Sigma_\theta(x_t, t)$ is not learned.

Then, $p_\theta(x_{t-1}|x_t) = \mathcal{N}\left(x_{t-1};\ \mu_\theta(x_t, t),\ \sigma_t^2 I\right)$

So,

$$L_{t-1} = \mathbb{E}_q\left[D_{KL}\left(q(x_{t-1}|x_t, x_0) \,\|\, p_\theta(x_{t-1}|x_0)\right)\right]$$

$$= \mathbb{E}_q\left[\frac{1}{2\sigma_t^2}\left\|\tilde{\mu}_t(x_t, x_0) - \mu_\theta(x_t, t)\right\|_2^2\right] + C$$

↑ constant

Now, in the forward process, we know

$$x_t \sim q(x_t|x_0) = \mathcal{N}\left(x_t;\ \sqrt{\bar{\alpha}_t}\, x_0,\ (1-\bar{\alpha}_t) I\right)$$

So, $x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\,\varepsilon$

So, $x_0 = \frac{x_t - \sqrt{1-\bar{\alpha}_t}\,\varepsilon}{\sqrt{\bar{\alpha}_t}\, x_0}$

Then,

$$L_{t-1} - C = \mathbb{E}_{\varepsilon, x_0 \sim q}\left[\frac{1}{2\sigma_t^2}\left\|\tilde{\mu}_t\left(x_t, \frac{x_t - \sqrt{1-\bar{\alpha}_t}\,\varepsilon}{\sqrt{\bar{\alpha}_t}\, x_0}\right) - \mu_\theta(x_t, t)\right\|_2^2\right]$$

$$= \mathbb{E}_{\epsilon, x_0 \sim q}\left[\frac{1}{2\sigma_t^2}\left\|\frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\,\varepsilon\right) - \mu_\theta(x_t, t)\right\|_2^2\right]$$

Since x_t is input to μ_θ, we set μ_θ to be:

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\,\varepsilon_\theta(x_t, t)\right)$$

where ε_θ is a function approximator to predict ε from x_t.

Then,

$$L_{t-1} = \mathbb{E}_{x_0, \varepsilon}\left[\frac{\beta_t^2}{2\sigma_t^2 \alpha_t (1-\bar{\alpha}_t)} \left\| \varepsilon - \varepsilon_\theta\left(\sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\, \varepsilon, t\right)\right\|^2 \right]$$

What about L_0?

We set the last step of the reverse process to be an independent discrete decoder derived from the Gaussian $\mathcal{N}(x_0;\ \mu_\theta(x_1, 1),\ \sigma_1^2 I)$:

$$p_\theta(x_0 | x_1) := \prod_{i=1}^{d} \int_{\delta_-(x_0^i)}^{\delta_+(x_0^i)} \mathcal{N}(x;\ \mu_\theta^i(x_1, 1),\ \sigma_1^2)\, dx$$

where

$$\delta_+(x) = \begin{cases} \infty & \text{if } x = 1 \\ x + \frac{1}{255} & \text{if } x < 1 \end{cases}$$

$$\delta_-(x) = \begin{cases} -\infty & \text{if } x = -1 \\ x - \frac{1}{255} & \text{if } x > -1 \end{cases}$$

and d = data dimensionality

At the last step, we sample $\mu_\theta(x_1, 1)$ noiselessly.

Altogether, Simplified training objective becomes:

$$L(\theta) := \mathbb{E}_{t, x_0, \varepsilon}\left[\left\| \varepsilon - \varepsilon_\theta\left(\sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\, \varepsilon, t\right)\right\|^2 \right]$$

$$t \sim \text{Uniform}\{1, \ldots, T\}$$

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \ldots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\right\|^2$
6: **until** converged

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \ldots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$