Denoising Diffusion Probabilistic Models

Liu Yang

We will introduce the Denoising Diffusion Probabilistic Models (DDPM) in this lecture [1]. The notation might be slightly from the original paper.

1 Markov Chain with Gaussian Noise

Consider a Markov chain with the following kernel:

$$q(x_s|x_t) = \mathcal{N}(A_{s,t}x_t, B_{s,t}^2)$$

for $s > t$, we have

$$\begin{aligned} A_{r,t} &= A_{r,s}A_{s,t} \\ B_{r,t}^2 &= A_{r,s}^2 B_{s,t}^2 + B_{r,s}^2 \end{aligned}$$

for $r > s > t$.

Consider a forward discrete diffusion process:

$$q(x_{t+1}|x_t) = \mathcal{N}(A_{t+1,t}x_t, B_{t+1,t}^2)$$

If we set $A_{t+1,t} = \sqrt{\alpha_{t+1}}$, $B_{t+1,t} = \sqrt{1-\alpha_{t+1}}$, $\beta_t = 1 - \alpha_t$, which is the notation used in [1], i.e.

$$q(x_t|x_{t-1}) = \mathcal{N}(\sqrt{1-\beta_t}x_{t-1}, \beta_t)$$

then

$$\begin{aligned} A_{s,t} &= \sqrt{\alpha_{t+1}...\alpha_s} \\ B_{s,t} &= \sqrt{1-\alpha_{t+1}...\alpha_s} \\ A_{s,t}^2 + B_{s,t}^2 &= 1 \end{aligned}$$

If we fix α_i as a constant, then as $s \to \infty$, we have $A_{s,t} \to 0$, $B_{s,t} \to 1$, and $q(x_s|x_t) \to \mathcal{N}(0,1)$ for any x_t.

In [1], we denote $\bar{\alpha}_t = \Pi_{s=1}^t \alpha_s$. If we use this notation, then

$$\begin{aligned} A_{s,0} &= \sqrt{\alpha_1...\alpha_s} = \sqrt{\bar{\alpha}_s} \\ B_{s,0} &= \sqrt{1-\alpha_1...\alpha_s} = \sqrt{1-\bar{\alpha}_s} \\ q(x_s|x_0) &= \mathcal{N}(\sqrt{\bar{\alpha}_s}x_0, 1-\bar{\alpha}_s) \end{aligned}$$

2 Reverse Diffusion Process

We want to learn another Markov chain $p_\theta(x_{0:T})$ (reverse diffusion) defined by

$$\begin{aligned} p_\theta(x_T) &= \mathcal{N}(0, I) \\ p_\theta(x_{t-1}|x_t) &= \mathcal{N}(\mu_\theta(x_t, t), \Sigma_\theta(x_t, t)), t = T, ...2, 1 \end{aligned}$$

so that $p_\theta(x_0)$ approximates $q(x_0)$. $p_\theta(x_{0:T})$ is not Gaussian due to the nonlinearity of μ_θ. Actually $p_\theta(x_t)$ becomes more and more expressive as t decreases from T to 0.

In order to train $p_\theta(x_{t-1}|x_t)$, we need to decompose $q(x_{0:T})$ in a reverse order.

First perspective:

$$q(x_{0:T}) = q(x_T)q(x_{T-1}|x_T)q(x_{T-2}|x_{T-1})...q(x_0|x_1)$$

But it's hard formulate $q(x_{t-1}|x_t)$.

Fortunately, we have a second perspective:

$$q(x_{0:T}) = q(x_0)q(x_T|x_0)q(x_{T-1}|x_T, x_0)q(x_{T-2}|x_{T-1,x_0})...q(x_1|x_2, x_0)$$

An important observation is that $q(x_t|x_s, x_0)$ is Gaussian for $s > t$, and we have the analytical expression:

$$\begin{aligned} q(x_t|x_s, x_0) &= q(x_s|x_t, x_0)q(x_t|x_0)/q(x_s|x_0) \\ &= q(x_s|x_t)q(x_t|x_0)/q(x_s|x_0) \\ &\propto \exp(-\frac{(x_s - A_{s,t}x_t)^2}{2B_{s,t}^2} - \frac{(x_t - A_{t,0}x_0)^2}{2B_{t,0}^2} + \frac{(x_s - A_{s,0}x_0)^2}{2B_{s,t}^2}) \end{aligned}$$

Rearranging the first and second order coefficients, we denote

$$q(x_t|x_s, x_0) = \mathcal{N}(C_{s,t}x_s + D_{s,t}x_0, E_{s,t}^2)$$

Note that the mean is linear to x_s and x_0, while the std is independent of x_s and x_0. $C_{s,t}, D_{s,t}, E_{s,t}$ can be calculated with patience.

$$\begin{aligned} C_{s,t} &= A_{s,t}B_{t,0}^2/B_{s,0}^2 \\ D_{s,t} &= A_{t,0}B_{s,t}^2/B_{s,0}^2 \\ E_{s,t}^2 &= B_{s,t}^2B_{t,0}^2/B_{s,0}^2 \end{aligned}$$

A special case of $q(x_{s-1}|x_s, x_0)$ is given in [1]:

$$q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}_t(x_t, x_0), \tilde{\beta}_t),$$
$$\text{where } \tilde{\mu}_t(x_t, x_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1 - \bar{\alpha}_t}x_0 + \frac{\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t}x_t, \tilde{\beta}_t := \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t}\beta_t$$

3 Loss function

3.1 Decomposition

Intuition for training $p_\theta(x_{0:T})$: Let's write the decomposed q and p_θ

$$
\begin{aligned}
q(x_T|x_0) &\sim p_\theta(x_T) \\
q(x_{t-1}|x_t, x_0) &\sim p_\theta(x_{t-1}|x_t)
\end{aligned}
$$

It's encouraging that the above terms are all Gaussian, so maybe we can train RHS to fit LHS. But we still have a question: the LHS is conditioned on x_0 but the RHS is not. To what extent can we make the approximation?

Recall that we wish $p_\theta(x_0)$ approximates $q(x_0)$, so the loss could be negative log likelihood

$$
\begin{aligned}
L_{NNL} &= -E_{q(x_0)} \log p_\theta(x_0) \\
&= -E_{q(x_0)} \log(\int p_\theta(x_{0:T}) dx_{1:T}) \\
&= -E_{q(x_0)} \log(E_{q(x_{1:T}|x_0)} \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)}) \\
&\leq -E_{q(x_0)} E_{q(x_{1:T}|x_0)} \log(\frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)}) \quad \text{(Jensen's Inequality)} \\
&= -E_{q(x_{0:T})} \log(\frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)}) =: L
\end{aligned}
$$

Here importance sampling is motivated by leveraging the density and samples for $q(x_{1:T}|x_0)$, which we already have.

In L_{NNL} we only care about the marginal $p_\theta(x_0)$ and $q(x_0)$. However, L actually requires $p_\theta(x_{0:T})$ to approximate $q(x_{0:T})$, i.e., the reverse diffusion will follow the density of forward diffusion.

Now swap and decompose p_θ and q

$$
\begin{aligned}
L &= E_{q(x_{0:T})} \log(\frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})}) \\
&= E_{q(x_{0:T})} \log(\frac{q(x_T|x_0)}{p_\theta(x_T)}) \\
&+ \sum_{t=1}^{T-1} E_{q(x_{0:T})} \log(\frac{q(x_t|x_{t+1}, x_0)}{p_\theta(x_t|x_{t+1})}) \\
&+ E_{q(x_{0:T})} \log(\frac{1}{p_\theta(x_0|x_1)}) \\
&=: \sum_{t=0}^{T} l_t
\end{aligned}
$$

1. l_T is irrelevant to training, since $p_\theta(x_T)$ is fixed.

2. $l_t = E_{q(x_{t+1},x_0)} KL(q(x_t|x_{t+1}, x_0) \| p_\theta(x_t|x_{t+1}))$ as we will show later.

3. $l_0 = -E_{q(x_1,x_0)} \log(p_\theta(x_0|x_1))$

We can train θ with L, but we want to look further into l_t for the following reasons: (1) What does this loss function mean? (2) The usage of log could lead to instability; we want to simplify the loss function.

We have

$$
\begin{aligned}
l_T &= E_{q(x_{0:T})} \log(\frac{q(x_T|x_0)}{p_\theta(x_T)}) \\
&= E_{q(x_0)} E_{q(x_T|x_0)} E_{q(x_{1:T-1}|x_0,x_T)} \log(\frac{q(x_T|x_0)}{p_\theta(x_T)}) \\
&= E_{q(x_0)} E_{q(x_T|x_0)} \log(\frac{q(x_T|x_0)}{p_\theta(x_T)}) \\
&= E_{q(x_0)} KL(q(x_T|x_0) \| p_\theta(x_T))
\end{aligned}
$$

and for $t = 1, 2, ...T-1$

$$
\begin{aligned}
l_t &= E_{q(x_{0:T})} \log(\frac{q(x_t|x_{t+1}, x_0)}{p_\theta(x_t|x_{t+1})}) \\
&= E_{q(x_{t+1},x_0)} E_{q(x_t|x_{t+1},x_0)} E_{q(x_{others}|x_t,x_{t+1},x_0)} \log(\frac{q(x_t|x_{t+1}, x_0)}{p_\theta(x_t|x_{t+1})}) \\
&= E_{q(x_{t+1},x_0)} E_{q(x_t|x_{t+1},x_0)} \log(\frac{q(x_t|x_{t+1}, x_0)}{p_\theta(x_t|x_{t+1})}) \\
&= E_{q(x_{t+1},x_0)} KL(q(x_t|x_{t+1}, x_0) \| p_\theta(x_t|x_{t+1}))
\end{aligned}
$$

and finally

$$
\begin{aligned}
l_0 &= E_{q(x_{0:T})} \log(\frac{1}{p_\theta(x_0|x_1)}) \\
&= E_{q(x_0,x_1)} E_{q(x_{2:T}|x_0,x_1)} \log(\frac{1}{p_\theta(x_0|x_1)}) \\
&= E_{q(x_0,x_1)} \log(\frac{1}{p_\theta(x_0|x_1)}) \\
&= -E_{q(x_0,x_1)} \log(p_\theta(x_0|x_1))
\end{aligned}
$$

Or if you like KL divergence representation

$$
\begin{aligned}
l_0 &= E_{q(x_{0:T})} \log(\frac{1}{p_\theta(x_0|x_1)}) \\
&= E_{q(x_1)} E_{q(x_0|x_1)} E_{q(x_{2:T}|x_0,x_1)} \log(\frac{1}{p_\theta(x_0|x_1)}) \\
&= E_{q(x_1)} E_{q(x_0|x_1)} \log(\frac{1}{p_\theta(x_0|x_1)}) \\
&= E_{q(x_1)} [E_{q(x_0|x_1)} \log(\frac{q(x_0|x_1)}{p_\theta(x_0|x_1)}) - E_{q(x_0|x_1)} \log(q(x_0|x_1))] \\
&= E_{q(x_1)} KL(q(x_0|x_1) \| p_\theta(x_0|x_1)) + E_{q(x_1)} H(q(x_0|x_1))
\end{aligned}
$$

This is consistent with the equation (3) and (5) in [1].

3.2 Mean Squared Loss

We can further simplify the loss function, replacing KL divergence with mean squared loss, which is more stable.

1. We don't need to train l_T since $p_\theta(x_T) = \mathcal{N}(0, I)$ is fixed.

2. For $t = 2, ...T$,

$$l_{t-1} = E_{q(x_t,x_0)} KL(q(x_{t-1}|x_t, x_0) \| p_\theta(x_{t-1}|x_t))$$

where

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

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

The KL divergence between two Gaussian distributions has the analytical expression:

$$KL(\mathcal{N}(\mu_1, \Sigma_1) \| \mathcal{N}(\mu_2, \Sigma_2)) = \frac{1}{2}\left[\log\frac{|\Sigma_2|}{|\Sigma_1|} - k + \text{tr}(\Sigma_2^{-1}\Sigma_1) + (\mu_2 - \mu_1)^T \Sigma_2^{-1}(\mu_2 - \mu_1)\right]$$

For simplicity, we can fix the variance of $p_\theta(x_{t-1}|x_t)$ as $\sigma^2(t)$. According to [1], both $\sigma^2(t) = \beta_t$ and $\sigma^2(t) = \tilde{\beta}_t$ had similar experimental results. Denote the mean of $p_\theta(x_{t-1}|x_t)$ as $\mu_\theta(x_t, t)$, then

$$l_{t-1} = E_{q(x_t,x_0)}\left[\frac{\|\tilde{\mu}_t(x_t, x_0) - \mu_\theta(x_t, t)\|^2}{2\sigma^2(t)}\right] + C$$

for $t = 2, ...T$.

3. For $l_0 = -E_{q(x_0,x_1)} \log(p_\theta(x_0|x_1))$, we can set $p_\theta(x_0|x_1) = \mathcal{N}(\mu_\theta(x_1, 1), \sigma^2(1))$. Then

$$l_0 = E_{(x_0,x_1)\sim q(x_0,x_1)}\left[\frac{\|\mu_\theta(x_1, 1) - x_0\|^2}{2\sigma^2(1)}\right] + C$$

The choice of $\sigma^2(1)$ is tricky. Fortunately, as we will see later, it can be removed from the training loss and reverse diffusion process.

3.3 Reparameterization

Let's focus on l_{t-1} for $t = 2, ...T$. We aim to approximate $\tilde{\mu}_t(x_t, x_0)$ with the neural network $\mu_\theta(x_t, t)$. Note that $\mu_\theta(x_t, t)$ only takes x_t as input, while $\tilde{\mu}_t(x_t, x_0)$ is the linear combination of x_0 and x_t, and x_0 can also be represented by x_t and noise. So why not just use the neural network to approximate the noise?

In particular, with $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$, where ϵ is the standard Gaussian noise.

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

Now we can reparameterize μ_θ as

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

where $\epsilon_\theta(x_t, t)$ is the neural network aiming to approximate ϵ. With such reparameterization, we rewrite the loss as

$$l_{t-1} = E_{x_0,\epsilon}\left[\frac{\beta_t^2}{2\sigma^2(t)\alpha_t(1-\bar{\alpha}_t)}\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)\|^2\right] + C$$

for $t = 2, ...T$.

Similar reparameterization can also be applied to l_0.

$$l_0 = E_{(x_0,x_1)\sim q(x_0,x_1)}\left[\frac{1}{2\sigma^2(1)}\|\mu_\theta(x_1, 1) - x_0\|^2\right] + C$$

$$\text{reparameterize } \mu_\theta(x_1, 1) = \frac{1}{\sqrt{\alpha_1}}(x_1 - \frac{\beta_1}{\sqrt{1-\bar{\alpha}_1}}\epsilon_\theta(x_1, 1)).$$

$$x_0 = \frac{1}{\sqrt{\bar{\alpha}_1}}(x_1 - \sqrt{1-\bar{\alpha}_1}\epsilon)$$

With $\alpha_1 = \bar{\alpha}_1$, $\beta_1 = 1 - \alpha_1$, we rewrite

$$l_0 = E_{x_0,\epsilon}\left[\frac{\beta_1^2}{2\sigma^2(1)\alpha_1(1-\bar{\alpha}_1)}\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_1}x_0 + \sqrt{1-\bar{\alpha}_1}\epsilon, 1)\|^2\right] + C$$

3.4 Simplication

Furthermore, [1] found it beneficial to sample quality (and simpler to implement) to remove the weights in the above mean squared loss:

$$l_{t-1} = E_{x_0,\epsilon}\left[\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)\|^2\right]$$

for $t = 1, ...T$.

4 Sampling

After training, we can sample $p_\theta(x_0)$ with the reverse diffusion process. For $p_\theta(x_0|x_1) = \mathcal{N}(\mu_\theta(x_1, 1), \sigma^2(1))$ we can just set $\sigma^2(1) = 0$, i.e. noiseless sampling.

In summary, the training and sampling algorithm is in Figure 1.

Algorithm 1 Training

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

Algorithm 2 Sampling

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

Figure 1: DDPM training and sampling algorithm

5 Tutorial Code

https://github.com/Jmkernes/Diffusion/blob/main/diffusion/ddpm/diffusers.py

References

[1] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.