COMP 7070 Advanced Topics in AI and ML April 09, 2024

Lecture 12: Diffusion Model

Instructor: Yifan Chen *Scribes: Lanjing Yi* *Proof reader: N/A*

Note: *LaTeX template courtesy of UC Berkeley EECS dept.*

12.1 Variational Diffusion Models

12.1.1 Markovian Hierarchical Variational Autoencoders

A Hierarchical Variational Autoencoder (HVAE) is a generalization of a VAE that extends to multiple hierarchies over latent variables. Under this formulation, latent variables themselves are interpreted as generated from other higher-level, more abstract latents.

Whereas in the general HVAE with T hierarchical levels, each latent is allowed to condition on all previous latents, in this work we focus on a special case called a Markovian HVAE (MHVAE).

In a MHVAE, the generative process is a Markov chain; that is, each transition down the hierarchy is Markovian, where decoding each latent z_t only conditions on previous latent $z_t + 1$. Intuitively and visually, this can be seen as simply stacking VAEs on top of each other, as depicted in Figure 1; another appropriate term describing this model is a Recursive VAE.

Mathematically, we represent the joint distribution and the posterior of a Markovian HVAE as:

$$p(\boldsymbol{x}, \boldsymbol{z}_{1:T}) = p(\boldsymbol{z}_T) p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}_1) \prod_{t=2}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t)$$

and

$$q_\phi(\boldsymbol{z}_{1:T}|\boldsymbol{x}) = q_\phi(\boldsymbol{z}_1|\boldsymbol{x}) \prod_{t=2}^{T} q_\phi(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})$$

Then, we can easily write the ELBO as:

$$\begin{aligned}
\log p(x) &= \log \int \frac{p(x, z_{1-T})}{q_\phi(z_{1-T}|X)} \cdot q_\phi(z_{1-T}|X) dz_{1-T} \\
&= \log E_{q\phi}(z_{1-T}|X) \frac{p(x, z_{1-T})}{q_\phi(z_{1-T}|X)} \\
&\geq E_{q\phi}(z_{1-T}|X) \log \frac{p(x, z_{1-T})}{q_\phi(z_{1-T}|X)} \\
&= E_{q\phi}(z_{1-T}|x) \log \frac{\prod_{t=0}^{T-1} p(z_t|z_{t+1}) \cdot p(z_t)}{\prod_{t=0}^{T-1} q_\phi(z_{t+1}|z_t)}
\end{aligned}$$

we can further plug our joint distribution and posterior into above Equation to produce an alternate form:

$$\mathbb{E}_{q_\phi(\boldsymbol{z}_{1:T}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{x},\boldsymbol{z}_{1:T})}{q_\phi(\boldsymbol{z}_{1:T}|\boldsymbol{x})}\right] = \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}_{1:T}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{z}_T)p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}_1)\prod_{t=2}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t)}{q_{\boldsymbol{\phi}}(\boldsymbol{z}_1|\boldsymbol{x})\prod_{t=2}^{T} q_{\boldsymbol{\phi}}(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})}\right]$$

12.1.2 Make MHVAE becomes Variational Diffusion Models

The easiest way to think of a Variational Diffusion Model (VDM) is simply as a Markovian Hierarchical Variational Autoencoder with three key restrictions:

• The latent dimension is exactly equal to the data dimension.

• The structure of the latent encoder at each timestep is not learned; it is pre-defined as a linear Gaussian model. In other words, it is a Gaussian distribution centered around the output of the previous timestep .

• The Gaussian parameters of the latent encoders vary over time in such a way that the distribution of the latent at final timestep T is a standard Gaussian.

From the first restriction, with some abuse of notation, we can now represent both true data samples and latent variables as $\boldsymbol{x}_t$, where $t = 0$ represents true data samples and $t \in [1, T]$ represents a corresponding latent with hierarchy indexed by t. The VDM posterior can now be rewritten as:

$$q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0) = \prod_{t=1}^{T} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$$

From the second assumption, we know that the distribution of each latent variable in the encoder is a Gaussian centered around its previous hierarchical latent. Unlike a Markovian HVAE, the structure of the encoder at each timestep t is not learned; it is fixed as a linear Gaussian model, where the mean and standard deviation can be set beforehand as hyper-parameters or learned as parameters.

We parameterize the Gaussian encoder with mean: $\boldsymbol{\mu}_t(\boldsymbol{x}_t) = \sqrt{\alpha_t}\boldsymbol{x}_{t-1}$ and variance: $\boldsymbol{\Sigma}_t(\boldsymbol{x}_t) = (1-\alpha_t)\mathbf{I}$ where the form of the coefficients are chosen such that the variance of the latent variables stays at a similar scale; in other words, the encoding process is variance-preserving. Mathematically, encoder transitions are denoted as:

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{\alpha_t}\boldsymbol{x}_{t-1}, (1-\alpha_t)\mathbf{I})$$

For more details on the variance-preserving,

$$\begin{aligned} Var_q(X_t) &= E_{X_{t-1}} Var_q(X_t|X_{t-1}) + Var_q E(X_t|X_{t-1}) \\ &= E_{X_{t-1}}(1-\alpha_t)I + Var_q(\sqrt{a_t}X_{t-1}) \\ &= E_{X_{t-1}}(1-\alpha_t)I + a_t Var_q(X_{t-1}) \end{aligned}$$

Then we can derive it as

$$Vur_q(z_T) = I => Vor_q(z_t) = I, \quad \forall t$$

Like any HVAE, the VDM can be optimized by maximizing the ELBO, which can be derived as:

$$\begin{aligned} ELBO &= \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log \frac{p(\boldsymbol{x}_T)p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1)\prod_{t=1}^{T-1} p_{\boldsymbol{\theta}}(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})}{q(\boldsymbol{x}_T|\boldsymbol{x}_{T-1})\prod_{t=1}^{T-1} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})}\right] \\ &= \mathbb{E}_{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1)\right] + \mathbb{E}_{q(\boldsymbol{x}_{T-1},\boldsymbol{x}_T|\boldsymbol{x}_0)}\left[\log \frac{p(\boldsymbol{x}_T)}{q(\boldsymbol{x}_T|\boldsymbol{x}_{T-1})}\right] + \sum_{t=1}^{T-1}\mathbb{E}_{q(\boldsymbol{x}_{t-1},\boldsymbol{x}_t,\boldsymbol{x}_{t+1}|\boldsymbol{x}_0)}\left[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})}{q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})}\right] \\ &= \mathbb{E}_{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1)\right] + \mathbb{E}_{q(\boldsymbol{x}_{T-1}|\boldsymbol{x}_0)} - \mathrm{KL}(q(\boldsymbol{x}_T|\boldsymbol{x}_{T-1})\|p(\boldsymbol{x}_T)) \\ &+ \mathbb{E}_{q(\boldsymbol{x}_{t-1},\boldsymbol{x}_{t+1}|\boldsymbol{x}_0)}\left[\mathrm{KL}(q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_t|\boldsymbol{x}_{t+1}))\right] \end{aligned}$$

The derived form of the ELBO can be interpreted in terms of its individual components:

$\mathbb{E}_{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1)\right]$ can be interpreted as a *reconstruction term, predicting the log probability of* the original data sample given the frst-step latent. This term also appears in a vanilla VAE, and can be trained similarly.

$\mathbb{E}_{q(\boldsymbol{x}_{T-1}|\boldsymbol{x}_0)}\left[\mathrm{KL}(q(\boldsymbol{x}_T|\boldsymbol{x}_{T-1})\|p(\boldsymbol{x}_T))\right]$ is a prior matching term; it is minimized when the final latent distribution matches the Gaussian prior. This term requires no optimization, as it has no trainable parameters; furthermore, as we have assumed a large enough T such that the final distribution is Gaussian, this term effectively becomes zero.

$\mathbb{E}_{q(\boldsymbol{x}_{t-1},\boldsymbol{x}_{t+1}|\boldsymbol{x}_0)}\left[\mathrm{KL}(q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_t|\boldsymbol{x}_{t+1}))\right]$is a consistency term; it endeavors to make the distribution at x_t consistent, from both forward and backward processes. That is, a denoising step from a noisier image should match the corresponding noising step from a cleaner image, for every intermediate timestep; this is reflected mathematically by the KL Divergence.

Under this derivation, all terms of the ELBO are computed as expectations and can, therefore, be approximated using Monte Carlo estimates. However, actually optimizing the ELBO using the terms we just derived might be suboptimal; because the consistency term is computed as an expectation over two random variables $\{x_{t-1}, x_{t+1}\}$ for every timestep, the variance of its Monte Carlo estimate could potentially be higher than a term that is estimated using only one random variable per timestep. As it is computed by summing up $T-1$ consistency terms, the final estimated value of the ELBO may have high variance for large T values.

Let us instead try to derive a form for our ELBO where each term is computed as an expectation over only one random variable at a time.The key insight is that we can rewrite encoder transitions as $q(X_t|X_{t-1}) = q(X_t|X_{t-1}, X_0)$, where the extra conditioning term is superfluous due to the Markov property. Then, according to Bayes rule, we can rewrite each transition as:

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{x}_0) = \frac{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)q(\boldsymbol{x}_t|\boldsymbol{x}_0)}{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)}$$

Armed with this new equation, we can retry the derivation resuming from the ELBO Equation:

$$\begin{aligned}
ELBO &= \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_T)p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1)}{q(\boldsymbol{x}_1|\boldsymbol{x}_0)} + \log \prod_{t=2}^{T} \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{x}_0)}\right] \\
&= \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log \frac{p(\boldsymbol{x}_T)p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1)}{q(\boldsymbol{x}_1|\boldsymbol{x}_0)} + \log \prod_{t=2}^{T} \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{\frac{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)q(\boldsymbol{x}_t|\boldsymbol{x}_0)}{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)}}\right] \\
&= \mathbb{E}_{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1)\right] + \mathbb{E}_{q(\boldsymbol{x}_T|\boldsymbol{x}_0)}\left[\log \frac{p(\boldsymbol{x}_T)}{q(\boldsymbol{x}_T|\boldsymbol{x}_0)}\right] + \sum_{t=2}^{T} \mathbb{E}_{q(\boldsymbol{x}_t,\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)}\left[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)}\right] \\
&= \mathbb{E}_{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1)\right] - \mathrm{KL}(q(\boldsymbol{x}_T|\boldsymbol{x}_0) \parallel p(\boldsymbol{x}_T)) - \sum_{t=2}^{T} \mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[\mathrm{KL}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))\right]
\end{aligned}$$

We have successfully derived an interpretation for the ELBO that can be estimated with lower variance, as each term is computed as an expectation of at most one random variable at a time.

As a side note, one observes that in the process of both ELBO derivations,only the Markov assumption is used; as a result, these formulae will hold true for any arbitrary Markovian HVAE.

12.1.3 How to compute the loss

In this derivation of the ELBO, the bulk of the optimization cost once again lies in the summation term, which dominates the reconstruction term. Whereas each KL Divergence term is difficult to minimize for arbitrary posteriors in arbitrarily complex Markovian HVAEs due to the added complexity of simultaneously learning the encoder, in a VDM we can leverage the Gaussian transition assumption to make optimization tractable. By Bayes rule, we have:

$$q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) = \frac{q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{x}_0)q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)}{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}$$

As we already know that

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{x}_0) = q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{\alpha_t}\boldsymbol{x}_{t-1}, (1-\alpha_t)\mathbf{I})$$

what remains is deriving for the forms of $q(\boldsymbol{x}_t|\boldsymbol{x}_0)$ and $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)$ Fortunately, these are also made tractable by utilizing the fact that the encoder transitions of a VDM are linear Gaussian models. Recall that under the reparameterization trick, samples $\boldsymbol{x}_t \sim q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ can be rewritten as:

$$\boldsymbol{x}_t = \sqrt{\alpha_t}\boldsymbol{x}_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon} \quad \text{with} \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon}; \mathbf{0}, \mathbf{I})$$

and that, similarly, samples $\boldsymbol{x}_{t-1} \sim q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_{t-2})$ can be rewritten as:

$$\boldsymbol{x}_{t-1} = \sqrt{\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1-\alpha_{t-1}}\boldsymbol{\epsilon} \quad \text{with} \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon}; \mathbf{0}, \mathbf{I})$$

Then, the form of $q(\boldsymbol{x}_t|\boldsymbol{x}_0)$can be recursively derived through repeated applications of the reparameterization trick. Suppose that we have access to $2T$random noise variables$\{\boldsymbol{\epsilon}_t^*, \boldsymbol{\epsilon}_t\}_{t=0}^T \overset{\text{iid}}{\sim} \mathcal{N}(\boldsymbol{\epsilon}; \mathbf{0}, \mathbf{I})$. Then, for an arbitrary sample $x_t \sim q(\boldsymbol{x}_t|\boldsymbol{x}_0)$, we can rewrite it as:

$$\begin{aligned}
x_t &= \sqrt{\alpha_t}x_{t-1} + \sqrt{1-\alpha_t}\epsilon_{t-1}^* \\
&= \sqrt{\prod_{i=1}^{t}\alpha_i}\boldsymbol{x}_0 + \sqrt{1-\prod_{i=1}^{t}\alpha_i}\boldsymbol{\epsilon}_0 \\
&= \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_0 \\
&\sim \mathcal{N}(\boldsymbol{x}_t; \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0, (1-\bar{\alpha}_t)\mathbf{I})
\end{aligned}$$

We have therefore derived the Gaussian form of $q(\boldsymbol{x}_t|\boldsymbol{x}_0)$. This derivation can be modified to also yield the Gaussian parameterization describing $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)$. Now, knowing the forms of both q(xt—x0) and $q(\boldsymbol{x}_t|\boldsymbol{x}_0)$, we can proceed to calculate the form of $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)$ by substituting into the Bayes rule expansion:

$$\begin{aligned}
q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) &= \frac{q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{x}_0)q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)}{q(\boldsymbol{x}_t|\boldsymbol{x}_0)} \\
&= \frac{\mathcal{N}(\boldsymbol{x}_t; \sqrt{\alpha_t}\boldsymbol{x}_{t-1}, (1-\alpha_t)\mathbf{I})\mathcal{N}(\boldsymbol{x}_{t-1}; \sqrt{\bar{\alpha}_{t-1}}\boldsymbol{x}_0, (1-\bar{\alpha}_{t-1})\mathbf{I})}{\mathcal{N}(\boldsymbol{x}_t; \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0, (1-\bar{\alpha}_t)\mathbf{I})} \\
&= \mathcal{N}(\boldsymbol{x}_{t-1}; \mu_q(\boldsymbol{x}_t, \boldsymbol{x}_0) = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\boldsymbol{x}_0}{1-\bar{\alpha}_t}, \Sigma_q(t) = \frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{I})
\end{aligned}$$

In order to match approximate denoising transition step $p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ to ground-truth denoising transition step $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ as closely as possible, we can also model it as a Gaussian. Furthermore, we must parameterize its mean $\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$ as a function of x_t, however, since $p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ does not condition on x_0.Recall that the KL Divergence between two Gaussian distributions is:

$$\text{KL}(\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x) \parallel \mathcal{N}(\boldsymbol{y}; \boldsymbol{\mu}_y, \boldsymbol{\Sigma}_y)) = \frac{1}{2}\left[\log\frac{|\boldsymbol{\Sigma}_y|}{|\boldsymbol{\Sigma}_x|} - d + \text{tr}(\boldsymbol{\Sigma}_y^{-1}\boldsymbol{\Sigma}_x) + (\boldsymbol{\mu}_y - \boldsymbol{\mu}_x)^T\boldsymbol{\Sigma}_y^{-1}(\boldsymbol{\mu}_y - \boldsymbol{\mu}_x)\right]$$

In our case, where we can set the variances of the two Gaussians to match exactly, optimizing the KL divergence term reduces to minimizing the difference between the means of the two distributions:

$$\begin{aligned}
&\arg\min \text{KL}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)) \\
&= \arg\min_{\boldsymbol{\theta}} D_{\text{KL}}(\mathcal{N}(\boldsymbol{x}_{t-1}; \boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q(t)) \parallel \mathcal{N}(\boldsymbol{x}_{t-1}; \boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\Sigma}_q(t))) \\
&= \arg\min_{\boldsymbol{\theta}} \frac{1}{2}\left[(\boldsymbol{\mu}_{\boldsymbol{\theta}} - \boldsymbol{\mu}_q)^T\boldsymbol{\Sigma}_q(t)^{-1}(\boldsymbol{\mu}_{\boldsymbol{\theta}} - \boldsymbol{\mu}_q)\right] \\
&= \arg\min_{\boldsymbol{\theta}} \frac{1}{2\sigma_q^2(t)}\left[\left\|\boldsymbol{\mu}_{\boldsymbol{\theta}} - \boldsymbol{\mu}_q\right\|_2^2\right]
\end{aligned}$$

where we have written $\boldsymbol{\mu}_q$ as shorthand for $\boldsymbol{\mu}_q(\boldsymbol{x}_t, \boldsymbol{x}_0)$, and $\boldsymbol{\mu}_{\boldsymbol{\theta}}$ as shorthand for $\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$ for brevity. In other words, we want to optimize a $\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$ that matches $\boldsymbol{\mu}_q(\boldsymbol{x}_t, \boldsymbol{x}_0)$,takes the form:

$$\boldsymbol{\mu}_q(\boldsymbol{x}_t, \boldsymbol{x}_0) = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\boldsymbol{x}_0}{1-\bar{\alpha}_t}$$

As $\mu_\theta(x_t, t)$ also conditions on x_t, we can match $\mu_q(x_t, x_0)$ closely by setting it to the following form:

$$\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\hat{\boldsymbol{x}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)}{1-\bar{\alpha}_t}$$

where $\hat{\boldsymbol{x}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$ is parameterized by a neural network that seeks to predict x_0 from noisy image x_t and time index t.Then, the optimization problem simplifies to:

$$\begin{aligned}
&\arg\min \mathrm{KL}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)) \\
=&\arg\min_{\boldsymbol{\theta}} \frac{1}{2\sigma_q^2(t)} \frac{\bar{\alpha}_{t-1}(1-\alpha_t)^2}{(1-\bar{\alpha}_t)^2} \left[\|\hat{\boldsymbol{x}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) - \boldsymbol{x}_0\|_2^2\right]
\end{aligned}$$

Therefore, optimizing a VDM boils down to learning a neural network to predict the original ground truth image from an arbitrarily noisified version of it. Furthermore, minimizing the summation term of our derived ELBO objective across all noise levels can be approximated by minimizing the expectation over all timesteps:

$$\arg\min_{\boldsymbol{\theta}} \mathbb{E}_{t\sim U\{2,T\}} \left[\mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)} \left[D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))\right]\right]$$

which can then be optimized using stochastic samples over timesteps.

12.1.4 How to connect VDM with score function

In English, Tweedie's Formula states that the true mean of an exponential family distribution, given samples drawn from it, can be estimated by the maximum likelihood estimate of the samples (aka empirical mean) plus some correction term involving the score of the estimate. In the case of just one observed sample, the empirical mean is just the sample itself. It is commonly used to mitigate sample bias; if observed samples all lie on one end of the underlying distribution, then the negative score becomes large and corrects the naive maximum likelihood estimate of the samples towards the true mean.

Mathematically, for a Gaussian $variable z \sim \mathcal{N}(z; \mu_z, \Sigma_z)$, Tweedie's Formula states that:

$$\mathbb{E}\left[\boldsymbol{\mu}_z|\boldsymbol{z}\right] = \boldsymbol{z} + \boldsymbol{\Sigma}_z \nabla_z \log p(\boldsymbol{z})$$

In this case, we apply it to predict the true posterior mean of x_t given its samples.By Tweedie's Formula, we have:

$$\mathbb{E}\left[\boldsymbol{\mu}_{x_t}|\boldsymbol{x}_t\right] = \boldsymbol{x}_t + (1-\bar{\alpha}_t)\nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t)$$

recall that we are aiming to minimize the difference between the means of the two distributions and the core term here is: $\|\mu_q(X_t, X_0) - \mu_\theta(X_t, t)\|^2$

$$VDM \Rightarrow \|x_0 - \hat{x}_\theta(x_t, t)\|^2$$

$$DDPM \Rightarrow \|\varepsilon_t - \hat{\varepsilon}_\theta(x_t, t)\|^2$$

$$ScoreFunction \Rightarrow \|\nabla \log p(x) - \hat{s}_\theta(x_t, t)\|^2$$

12.2 Score-based Generative Models

We have shown that a Variational Diffusion Model can be learned simply by optimizing a neural network $s_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$ to predict the score function $\nabla \log p(\boldsymbol{x}_t)$. However, in our derivation, the scoring term arrived from applying Tweedie's Formula; this doesn't necessarily provide us with great insight or insight into the score function or why it is worth modeling. Fortunately, we can look to another class of generative models, Score-based Generative Models, for exactly this intuition. As it turns out, we can show that the VDM formulation we have previously derived has an equivalent Score-based Generative Modeling formulation that allows us to switch between these two interpretations flexibly.

12.2.1 Energy based model

To begin to understand why optimizing a score function makes sense, we introduce energy-based models. Arbitrarily flexible probability distributions can be written in the form:

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \frac{1}{Z_{\boldsymbol{\theta}}} e^{-f_{\boldsymbol{\theta}}(\boldsymbol{x})}$$

where $f_{\boldsymbol{\theta}}(\boldsymbol{x})$ is an arbitrarily flexible, parameterizable function called the energy function, often modeled by a neural network, and Z_θ is a normalizing constant to ensure that $\int p_{\boldsymbol{\theta}}(\boldsymbol{x}) d\boldsymbol{x} = 1$. One way to learn such a distribution is maximum likelihood; however, this requires tractably computing the normalizing constant $Z_\theta = \int e^{-f_{\boldsymbol{\theta}}(\boldsymbol{x})} d\boldsymbol{x}$, which may not be possible for complex $f_{\boldsymbol{\theta}}(\boldsymbol{x})$functions.

One way to avoid calculating or modeling the normalization constant is by using a neural network $s_{\boldsymbol{\theta}}(\boldsymbol{x})$ to learn the score function $\nabla \log p(\boldsymbol{x})$ of distribution $p(x)$ instead.

$$\begin{aligned}
\nabla_x \log p_\theta(\boldsymbol{x}) &= \nabla_x \log(\frac{1}{Z_\theta} e^{-f_\theta(x)}) \\
&= \nabla_x \log \frac{1}{Z_\theta} + \nabla_x \log e^{-f_\theta(x)} \\
&= -\nabla_x f_{\boldsymbol{\theta}}(\boldsymbol{x}) \\
&\approx s_\theta(x)
\end{aligned}$$

which can be freely represented as a neural network without involving any normalization constants. The score model can be optimized by minimizing the Fisher Divergence with the ground truth score function:

$$\begin{aligned}
&\mathbb{E}_{p(\boldsymbol{x})} \left[\| \nabla \log p(\boldsymbol{x}) - s_{\boldsymbol{\theta}}(\boldsymbol{x}) \|_2^2 \right] \\
=&||\mathbf{s}_\theta(\mathbf{x})||^2 - 2(\nabla_{\mathbf{x}} \log p(\mathbf{x}))^T \mathbf{s}_\theta(\mathbf{x})] d\mathbf{x} \\
=&\int p(\mathbf{x}) ||\mathbf{s}_\theta(\mathbf{x})||^2 d\mathbf{x} - 2(\nabla_{\mathbf{x}} \log p(\mathbf{x}))^T \mathbf{s}_\theta(\mathbf{x})] d\mathbf{x} \\
=&\int p(\mathbf{x}) ||\mathbf{s}_\theta(\mathbf{x})||^2 d\mathbf{x} + 2 \int p(\mathbf{x}) \mathrm{tr}(\nabla_{\mathbf{x}} \mathbf{s}_\theta(\mathbf{x})) d\mathbf{x} \\
=&\mathbb{E}_{p(\mathbf{x})}[||\mathbf{s}_\theta(\mathbf{x})||^2 + 2\mathrm{tr}(\nabla_{\mathbf{x}} \mathbf{s}_\theta(\mathbf{x}))].
\end{aligned}$$

12.2.2 Score Matching Langevin Dynamics (SMLD)

We have now learned the score function of the data distribution through neural networks. So how do we obtain samples from this data distribution using the score function? The answer is Langevin Dynamics sampling.

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \epsilon \nabla_{\mathbf{x}} \log p(\mathbf{x}) + \sqrt{2\epsilon} \mathbf{z}_i, \quad \mathbf{z}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad i = 0, 1, \cdots K$$

The sampling here is an iterative process. ϵ is a very small quantity. x_0 is randomly initialized and updated through the iteration formula above. When the number of iterations K is sufficiently

large, x converges to a sample from the distribution. In this way, we essentially obtain a generative model. We can first train a network to estimate the score function, and then use Langevin Dynamics along with the network-estimated score function to sample, which allows us to obtain samples from the original distribution. Because the entire method consists of two parts: score matching and Langevin Dynamics, it is called SMLD.

Now we have obtained the SMLD generative model, but in fact, this model has three significant problem.

$$\mathcal{L} = \mathbb{E}_{p(\mathbf{x})}[||\nabla_{\mathbf{x}} \log p(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x})||^2] = \int p(\mathbf{x})||\nabla_{\mathbf{x}} \log p(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x})||^2 d\mathbf{x}$$

12.2.3 Three drawbacks of SMLD

Firstly, the score function is ill-defined when x lies on a low-dimensional manifold in a high-dimensional space. This can be seen mathematically; all points not on the low-dimensional manifold would have probability zero, the log of which is undefined.

Secondly, the estimated score function trained via vanilla score matching will not be accurate in low-density regions. We can see that the L2 term is actually weighted by $p(x)$.and explicitly trained on samples from it, the model will not receive an accurate learning signal for rarely seen or unseen examples. This is problematic since our sampling strategy involves starting from a random location in the high-dimensional space, which is most likely random noise, and moving according to the learned score function. Since we are following a noisy or inaccurate score estimate, the final generated samples may be suboptimal as well, or require many more iterations to converge on an accurate output.

Lastly, Langevin dynamics sampling may not mix, even if it is performed using the ground truth scores.Suppose that the true data distribution is a mixture of two disjoint distributions:

$$p(\boldsymbol{x}) = c_1 p_1(\boldsymbol{x}) + c_2 p_2(\boldsymbol{x})$$

Then, when the score is computed, these mixing coefficients are lost, since the log operation splits the coefficient from the distribution and the gradient operation zeros it out. Langevin dynamics sampling from the depicted initialization point has a roughly equal chance of arriving at each mode, despite the bottom right mode having a higher weight in the actual Mixture of Gaussians.

12.2.4 Add multiple Gaussian noise to fix it

It turns out that these three drawbacks can be simultaneously addressed by adding multiple levels of Gaussian noise to the data. Firstly, as the support of a Gaussian noise distribution is the entire space, a perturbed data sample will no longer be confined to a low-dimensional manifold. Secondly, adding large Gaussian noise will increase the area each mode covers in the data distribution, adding more training signal in low density regions. Lastly, adding multiple levels of Gaussian noise with increasing variance will result in intermediate distributions that respect the ground truth mixing coefficients.

Formally, we can choose a positive sequence of noise levels $\{\sigma_t\}_{t=1}^T$ and define a sequence of progressively perturbed data distributions:

$$p_{\sigma_t}(\boldsymbol{x}_t) = \int p(\boldsymbol{x}) \mathcal{N}(\boldsymbol{x}_t; \boldsymbol{x}, \sigma_t^2 \mathbf{I}) d\boldsymbol{x}$$

Then, a neural network $s_\theta(x, t)$ is learned using score matching to learn the score function for all noise levels simultaneously:

$$\arg\min_{\boldsymbol{\theta}} \sum_{t=1}^{T} \lambda(t) \mathbb{E}_{p_{\sigma_t}(\boldsymbol{x}_t)} \left[\| s_{\boldsymbol{\theta}}(\boldsymbol{x}, t) - \nabla \log p_{\sigma_t}(\boldsymbol{x}_t) \|_2^2 \right]$$

Note that this objective almost exactly matches the objective to train a Variational Diffusion Model. Furthermore, the authors propose annealed Langevin dynamics sampling as a generative procedure, in which samples are produced by running Langevin dynamics for each $t = T, T-1, ..., 2, 1$ in sequence.

The initialization is chosen from some fixed prior (such as uniform), and each subsequent sampling step starts from the final samples of the previous simulation. Because the noise levels steadily decrease over timesteps t, and we reduce the step size over time, the samples eventually converge into a true mode. This is directly analogous to the sampling procedure performed in the Markovian HVAE interpretation of a Variational Diffusion Model, where a randomly initialized data vector is iteratively refined over decreasing noise levels.

Therefore, we have established an explicit connection between Variational Diffusion Models and Score-based Generative Models, both in their training objectives and sampling procedures.

12.3 Unify VDM and SMLD by SDE

In fact, different Stochastic Differential Equations (SDEs) correspond to different ways of adding noise. We have already found that the training objective functions of SMLD (Score-based Model with Langevin Dynamics) and DDPM (Denoising Diffusion Probabilistic Models) are actually the same in form. With the SDE framework, we can see that both SMLD and DDPM can actually be formulated in the form of SDEs.

12.3.1 SMLD

$$X_{i+1} = X_i + \Delta t \cdot \nabla \log p(X_i) + \sqrt{2\Delta_t} \cdot Z_i, Z_i \sim N(0,1)$$

$$dX = dt \cdot \nabla \log p(X_t) + \sqrt{2} \cdot dW$$

The process of sample generation is the solution to the reverse SDE.

12.3.2 From SDE to ODE

Consider the movement of particles in a domain; the Lagrangian form focuses on how each particle moves.

$$\Longrightarrow SDE : dX = f(x,t)dt + g(t) \cdot dw$$

the Euler form focuses on the density $\rho(x,t)$,Fokker-Planck Equation:$\rho_0 \longrightarrow \rho_T$.

$$\Longrightarrow SDE : \mathrm{d}\mathbf{X} = [\mathbf{f}(\mathbf{x},t) - \frac{1}{2}(g^2(t) - \sigma^2(t))\nabla_{\mathbf{x}} \log p_t(\mathbf{x})]\mathrm{d}\mathbf{t} + \sigma(t)\mathrm{d}\mathbf{w}.$$

Because the two SDEs are equivalent, their corresponding $p_t(x)$ is the same, meaning that we can alter the variance $\sigma(t)$ of the second SDE. When we set $\sigma(t) = 0$, we obtain an Ordinary Differential Equation (ODE).

$$\mathrm{d}\mathbf{x} = [\mathbf{f}(\mathbf{x},t) - \frac{1}{2}g^2(t)\nabla_{\mathbf{x}} \log p_t(\mathbf{x})]\mathrm{d}\mathbf{t}.$$