Deep Learning - Foundations and Concepts

Chapter 20. Diffusion Models

nonlineark@github

April 21, 2025

Outline

Forward encoder

Suppose we take an image from the training set, which we will denote by x, and blend it with Gaussian noise independently for each pixel to give a noise-corrupted image z_1 defined by:

$$z_1 = \sqrt{1-\beta_1}x + \sqrt{\beta_1}\epsilon_1 \qquad \epsilon_1 \sim \mathcal{N}(\epsilon_1; 0, I)$$
$$q(z_1|x) = \mathcal{N}(z_1; \sqrt{1-\beta_1}x, \beta_1 I)$$

where $\beta_1 < 1$ is the variance of the noise distribution.

Forward encoder

We then repeat the process with additional independent Gaussian noise steps to give a sequence of increasingly noisy images $z_1, \dots, z_T$:

$$z_t = \sqrt{1-\beta_t} z_{t-1} + \sqrt{\beta_t}\epsilon_t \qquad \epsilon_t \sim \mathcal{N}(\epsilon_t; 0, I)$$
$$q(z_t|z_{t-1}) = \mathcal{N}(z_t; \sqrt{1-\beta_t} z_{t-1}, \beta_t I)$$

The values of the variance parameters $\beta_t \in (0, 1)$ are set by hand and are typically chosen such that the variance values increase through the chain according to a prescribed schedule such that $\beta_1 < \cdots < \beta_T$.

Diffusion kernel

Using induction, it's straightforward to verify that:

$$z_t = \sqrt{\alpha_t}x + \sqrt{1-\alpha_t}\epsilon_t \qquad \epsilon_t \sim \mathcal{N}(\epsilon_t; 0, I)$$
$$q(z_t|x) = \mathcal{N}(z_t; \sqrt{\alpha_t}x, (1-\alpha_t)I)$$

where we have defined:

$$\alpha_t = \prod_{\tau=1}^{t}(1-\beta_\tau)$$

We call $q(z_t|x)$ the diffusion kernel. After many steps the image becomes indistinguishable from Gaussian noise, and in the limit $T \to \infty$ we have:

$$q(z_T|x) = \mathcal{N}(z_T; 0, I)$$

Conditional distribution

Our goal is to learn to undo the noise process, and so it is natural to consider the reverse of the conditional distribution $q(z_t|z_{t-1})$:

$$q(z_{t-1}|z_t) = \frac{q(z_t|z_{t-1})q(z_{t-1})}{q(z_t)}$$

But $q(z_{t-1})$ is difficult to calculate:

- Evaluation of the integral $\int q(z_{t-1}|x)p(x)\mathrm{d}x$ is intractable, because we must integrate over the unknown data density $p(x)$.
- If we approximate the integration using samples from the training data set, we obtain a complicated distribution expressed as a mixture of Gaussians.

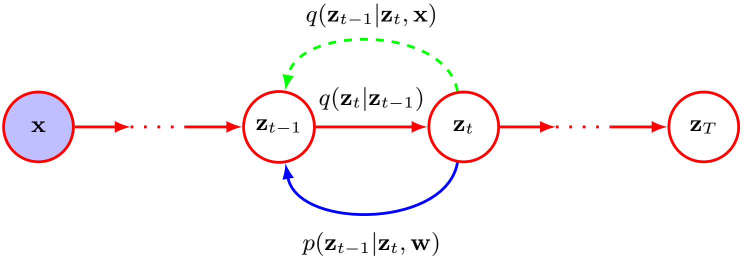
Conditional distribution

Instead, we consider the conditional version of the reverse distribution, conditioned on the data vector x, defined by $q(z_{t-1}|z_t, x)$:

$$
\begin{aligned}
q(z_{t-1}|z_t, x) &= \frac{q(z_t|z_{t-1}, x)q(z_{t-1}|x)}{q(z_t|x)} = \frac{q(z_t|z_{t-1})q(z_{t-1}|x)}{q(z_t|x)} \\
&= \mathcal{N}(z_{t-1}; m_t(x, z_t), \sigma_t^2 I) \\
m_t(x, z_t) &= \frac{(1-\alpha_{t-1})\sqrt{1-\beta_t}z_t + \sqrt{\alpha_{t-1}}\beta_t x}{1-\alpha_t} \\
\sigma_t^2 &= \frac{\beta_t(1-\alpha_{t-1})}{1-\alpha_t}
\end{aligned}
$$

Reverse decoder

- The forward encoder model is defined by a sequence of Gaussian conditional distribution $q(z_t|z_{t-1})$ but inverting this directly leads to a distribution $q(z_{t-1}|z_t)$ that is intractable.
- Instead, we will learn an approximation to the reverse distribution by using a distribution $p(z_{t-1}|z_t; w)$ governed by a deep neural network, where w represents the network weights and biases.



Reverse decoder

For $\beta_t \ll 1$, the distribution $q(z_{t-1}|z_t)$ will be approximately a Gaussian distribution over z_{t-1} (further references). We therefore model the reverse process using a Gaussian distribution of the form:

$$p(z_{t-1}|z_t; w) = \mathcal{N}(z_{t-1}; \mu(z_t, t; w), \beta_t I)$$

where $\mu(z_t, t; w)$ is a deep neural network governed by a set of parameters w. Note:

- The network takes the step index t explicitly as an input so that it can account for the variation of the variance β_t across different steps of the chain.
- It is also possible to learn the covariances of the denoising process by incorporating further outputs in the network to account for the curvature in the distribution $q(z_{t-1})$ in the neighborhood of z_t.

Reverse decoder

The overall reverse denoising process then takes the form of a Markov chain given by:

$$p(x, z_1, \ldots, z_T; w) = p(z_T) \prod_{t=2}^{T} p(z_{t-1}|z_t; w) p(x|z_1; w)$$

$$p(z_T) = q(z_T) = \mathcal{N}(z_T; 0, I)$$

Once the model has been trained, sampling is straightforward, we just follow the chain $z_T, \ldots, z_1, x$ in turn.

Training the decoder

The likelihood function is given by:

$$p(x;w) = \int p(x, z_1, \ldots, z_T; w) \mathrm{d}z_1 \cdots \mathrm{d}z_T$$

We see that the likelihood involves integrating over all possible trajectories by which noise samples could give rise to the observed data point. The integrals are intractable.

Evidence lower bound

- Since the exact likelihood is intractable, we can adopt a similar approach to that used with variational autoencoders and maximize the evidence lower bound.
- With diffusion models, we choose $q(z)$ to be given by the fixed distribution $q(z_1, \ldots, z_T|x)$, and so the only adjustable parameters are those in the model $p(x, z_1, \ldots, z_T; w)$ for the reverse Markov chain.

Evidence lower bound

$$
\begin{aligned}
\mathcal{L}(w) &= \int q(z) \log \frac{p(x,z;w)}{q(z)} \mathrm{d}z \\
&= E_q(\log \frac{p(z_T) \prod_{t=2}^{T} p(z_{t-1}|z_t;w) p(x|z_1;w)}{q(z_1|x) \prod_{t=2}^{T} q(z_t|z_{t-1},x)}) \\
&= E_q(\log p(z_T) - \log q(z_1|x) + \log p(x|z_1;w) \\
&\qquad + \sum_{t=2}^{T} \log \frac{p(z_{t-1}|z_t;w)}{q(z_t|z_{t-1},x)})
\end{aligned}
$$

Evidence lower bound

- The first and second terms are independent of w and can be omitted.
- The third term can be evaluated by approximating the expectation by a Monte Carlo estimate: $E_q(\log p(x|z_1; w)) \approx \frac{1}{L}\sum_{l=1}^{L} \log p(x|z_1^{(l)}; w)$, where $z_1^{(l)} \sim \mathcal{N}(z_1; \sqrt{1-\beta_1}x, \beta_1 I)$.
- For the fourth term:
  - Although we can sample from $q(z_{t-1}|x)$ and $q(z_t|z_{t-1})$, the use of pairs of sampled values creates very noisy estimates with high variance, so that an unnecessarily large number of samples is required.
  - Instead, we rewrite the ELBO in a form that can be estimated by sampling just one value per term.

Rewriting the ELBO

Following our discussion of the ELBO for the variational autoencoder, our goal here is to write the ELBO in terms of Kullback-Leibler divergences:

$$q(z_t|z_{t-1},x) = \frac{q(z_{t-1}|z_t,x)q(z_t|x)}{q(z_{t-1}|x)}$$

$$\log\frac{p(z_{t-1}|z_t;w)}{q(z_t|z_{t-1},x)} = \log\frac{p(z_{t-1}|z_t;w)}{q(z_{t-1}|z_t,x)} + \log\frac{q(z_{t-1}|x)}{q(z_t|x)}$$

The second term is independent of w and can be omitted:

$$\mathcal{L}(w) = E_q(\log p(x|z_1;w)) + \sum_{t=2}^{T} E_q(\log\frac{p(z_{t-1}|z_t;w)}{q(z_{t-1}|z_t,x)})$$

Rewriting the ELBO

$$E_q(\log p(x|z_1;w)) = \int q(z_1|x)p(x|z_1;w)\mathrm{d}z_1 = \text{reconstruction term}$$

$$\begin{aligned}
&E_q(\log \frac{p(z_{t-1}|z_t;w)}{q(z_{t-1}|z_t,x)}) \\
&= \int \left(\int \left(\int q(z_1|x) \cdots q(z_{t-1}|z_{t-2}) \mathrm{d}z_1 \cdots \mathrm{d}z_{t-2} \right) \right. \\
&\qquad \left. \frac{q(z_{t-1}|z_t,x)}{q(z_{t-1}|x)} \log \frac{p(z_{t-1}|z_t;w)}{q(z_{t-1}|z_t,x)} \mathrm{d}z_{t-1} \right) q(z_t|x)\mathrm{d}z_t \\
&= -\int \mathrm{KL}(q(z_{t-1}|z_t,x) || p(z_{t-1}|z_t;w)) q(z_t|x)\mathrm{d}z_t \\
&= \text{consistency term}
\end{aligned}$$

Rewriting the ELBO

The consistency terms are defined between pairs of Gaussian distributions and therefore can be expressed in closed form:

$$\mathrm{KL}(q(z_{t-1}|z_t, x)||p(z_{t-1}|z_t; w)) = \frac{1}{2\beta_t}||m_t(x, z_t) - \mu(z_t, t; w)||^2 + \text{const}$$

$$-\int \mathrm{KL}(q(z_{t-1}|z_t, x)||p(z_{t-1}|z_t; w))q(z_t|x)\mathrm{d}z_t$$

$$= -\frac{1}{2\beta_t}\int ||m_t(x, z_t) - \mu(z_t, t; w)||^2 q(z_t|x)\mathrm{d}z_t + \text{const}$$

$$\approx -\frac{1}{2\beta_t}\frac{1}{L}\sum_{l=1}^{L} ||m_t(x, z_t^{(l)}) - \mu(z_t^{(l)}, t; w)||^2 + \text{const}$$

where $z_t^{(l)} \sim \mathcal{N}(z_t; \sqrt{\alpha_t}x, (1-\alpha_t)I)$, and any additive terms that are independent of the network parameters w have been absorbed into the constant term.

Predicting the noise

Let's further simplify the training objective $\mathcal{L}(w)$ by changing the role of the neural network so that instead of predicting the denoised image at each step of the Markov chain it predicts the total noise component that was added to the original image to create the noisy image at that step:

$$z_t = \sqrt{\alpha_t}x + \sqrt{1-\alpha_t}\epsilon_t \implies x = \frac{1}{\sqrt{\alpha_t}}z_t - \frac{\sqrt{1-\alpha_t}}{\sqrt{\alpha_t}}\epsilon_t$$

$$\begin{aligned} m_t(x, z_t) &= \frac{(1-\alpha_{t-1})\sqrt{1-\beta_t}z_t + \sqrt{\alpha_{t-1}}\beta_t x}{1-\alpha_t} \\ &= \frac{1}{\sqrt{1-\beta_t}}(z_t - \frac{\beta_t}{\sqrt{1-\alpha_t}}\epsilon_t) \end{aligned}$$

Predicting the noise

We introduce a neural network $g(z_t, t; w)$ that aims to predict the total noise that was added to x to generate z_t:

$$\mu(z_t, t; w) = \frac{1}{\sqrt{1-\beta_t}}(z_t - \frac{\beta_t}{\sqrt{1-\alpha_t}} g(z_t, t; w))$$

We see that:

$$\mathrm{KL}(q(z_{t-1}|z_t, x)||p(z_{t-1}|z_t; w)) = \frac{1}{2\beta_t}||m_t(x, z_t) - \mu(z_t, t; w)||^2 + \mathrm{const}$$

$$= \frac{\beta_t}{2(1-\alpha_t)(1-\beta_t)}||g(z_t, t; w) - \epsilon_t||^2 + \mathrm{const}$$

Predicting the noise

Furthermore, for the reconstruction term, we have:

$$\begin{aligned}\log p(x|z_1;w) &= -\frac{1}{2\beta_1}||x-\mu(z_1,1;w)||^2+\text{const}\\ &= -\frac{1}{2(1-\beta_1)}||g(z_1,1;w)-\epsilon_1||^2+\text{const}\end{aligned}$$

Now the reconstruction and consistency terms can be combined:

$$\mathcal{L}(w) = -\sum_{t=1}^{T}\frac{\beta_t}{2(1-\alpha_t)(1-\beta_t)}\int ||g(z_t,t;w)-\epsilon_t||^2 q(z_t|x)\mathrm{d}z_t$$

Predicting the noise

It is found empirically that performance is further improved simply by omitting the factor $\frac{\beta_t}{2(1-\alpha_t)(1-\beta_t)}$, so that all steps in the Markov chain have equal weighting. If we only do one sample for each ϵ_t, then we have:

$$\mathcal{L}(w) = -\sum_{t=1}^{T} ||g(\sqrt{\alpha_t}x + \sqrt{1-\alpha_t}\epsilon_t, t; w) - \epsilon_t||^2$$

Predicting the noise

Algorithm 1: Training a denoising diffusion probabilistic model

for $t \leftarrow 1$ **to** T **do**
    $\alpha_t \leftarrow \prod_{\tau=1}^{t}(1-\beta_\tau)$;
end
repeat
    $x \sim \mathcal{D}$;
    $t \sim \{1, \ldots, T\}$;
    $\epsilon \sim \mathcal{N}(\epsilon; 0, I)$;
    $z_t \leftarrow \sqrt{\alpha_t}x + \sqrt{1-\alpha_t}\epsilon$;
    $\mathcal{L}(w) \leftarrow ||g(z_t, t; w) - \epsilon||^2$;
    Take optimizer step
until *converged*;
return w;

Generating new samples

Algorithm 2: Sampling from a denoising diffusion probabilistic model

$z_T \sim \mathcal{N}(z_T; 0, I)$;

for $t \leftarrow T$ **to** 2 **do**

  $\alpha_t \leftarrow \prod_{\tau=1}^{t}(1-\beta_\tau)$;

  $\mu(z_t, t; w) \leftarrow \frac{1}{\sqrt{1-\beta_t}}\left(z_t - \frac{\beta_t}{\sqrt{1-\alpha_t}} g(z_t, t; w)\right)$;

  $\epsilon \sim \mathcal{N}(\epsilon; 0, I)$;

  $z_{t-1} \leftarrow \mu(z_t, t; w) + \sqrt{\beta_t}\epsilon$;

end

$x \leftarrow \frac{1}{\sqrt{1-\beta_1}}\left(z_1 - \frac{\beta_1}{\sqrt{1-\alpha_1}} g(z_1, 1; w)\right)$;

return x;