Ludwig-Maximilians-Universität München
Institut für Informatik
Prof. Dr. Matthias Schubert
Maximilian Bernhard, Niklas Strauß

Munich, 16.12.2024

Deep Learning and Artificial Intelligence

WS 2024/25

Exercise 9: Generative Models

Exercise 9-1 Variational Autoencoder

Remember the Kullback-Leibler divergence between two probability distributions q and p is given by

$$\begin{aligned} KL(q \,||\, p) &= -\int q(x) \log \frac{p(x)}{q(x)}\, dx \\ &= -\int q(x) \log p(x)\, dx + \int q(x) \log q(x)\, dx \end{aligned}$$

(a) The ELBO lower bound in variational inference is defined as

$$L = \int q(z) \log \frac{p(z,x)}{q(z)}\, dz.$$

Rearrange the above formula such that one can see that it is the expectation of $\log p(x|z)$ with respect to $q(z)$ minus the KL-divergence of $q(z)$ w.r.t. $p(z)$.

Detailed Solution

$$\begin{aligned} L &= \int_z q(z) \log \frac{p(x,z)}{q(z)} dz \\ &= \int_z q(z) \log \frac{p(x|z)p(z)}{q(z)} dz \\ &= \int_z q(z) \log p(x|z) dz + \int_z q(z) \log \frac{p(z)}{q(z)} dz \\ &= \underbrace{\mathbb{E}_{q(z)}\left[\log p(x|z)\right]}_{\text{Negative Reconstruction Loss}} \underbrace{- KL(\, q(z) \,||\, p(z)\,)}_{\text{KL loss}} \end{aligned}$$

(b) Explain intuitively why $-\mathbb{E}_{Q(z)}\left[\log P(x|z)\right]$ is called reconstruction loss.

Detailed Solution

The reconstructed input $\hat{x}$ is deterministically dependent of z. Thus, the probability distribution $P(x|z)$ can be re-written as $P(x|\hat{x})$. So basically if you are maximizing the ELBO lower bound you are maximizing the expectation of the log likelihood that the input will be re-constructed. If you assume that $P(x|\hat{x})$ is Gaussian distributed, it can be shown that the reconstruction error corresponds to the mean squared error between x and $\hat{x}$. If the distribution is Bernoulli, the reconstruction loss corresponds to the binary cross entropy.

Exercise 9-2 Generative Adversarial Networks (GANs)

(a) Explain the role of the generator G and the discriminator D in GANs.

Detailed Solution

D and G compete with each other in a two-player game. The discriminator tries to classify as accurately as possible whether a certain sample comes from the same distribution as the target data (i.e. is 'real') or whether it has been generated by a the generator (i.e. is 'fake'). $D(x) \in [0, 1]$ represents the probability that x came from the target data. The generator, on the other hand, tries to create samples that look like real examples by passing noise vectors z through a neural network and learning the weights that make the outputs $G(z)$ look like real samples.

With training, D will get better at detecting 'fake' data and G will become better at generating data that looks 'real'. In the equilibrium of this game, the generator produces perfect samples that come from the same distribution as the training data and the discriminator gives each sample the probability 1/2 to be 'real' or 'fake'. G can then be used as a generative model that creates samples that look like the training data.

(b) The loss for D is given as:

$$J^{(D)} = -\frac{1}{2}\mathbb{E}_{x\sim p_{data(x)}}[\log D(x)] + -\frac{1}{2}\mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z))].$$

Explain the terms in this loss!

Detailed Solution

The discriminator wants $D(x)$ to be 1 for 'real' samples x coming from p_{data} and $D(G(z))$ to be 0 for 'fake' samples $G(z)$. The loss enforces this: The term -$\log(D(x))$ is 0 when $D(x) = 1$, but is positive if $D(x) < 1$, i.e. it penalizes values smaller than 1.

Analogously, -$\log(1 - D(G(z))$ penalizes values of $D(G(z)) > 0$.

Since we have several samples, we have take the expected value over all of these. In training, the same number of 'real' and 'fake' samples is used, therefore the two expectations are weighted by the factor $\frac{1}{2}$.

Note that this loss corresponds to the normal binary cross-entropy loss.

(c) The generator tries to fool the discriminator, so its loss can be defined as:

$$J^{(G)} = -J^{(D)}$$

Write down this optimization problem as a minimax game!

Detailed Solution

$$\min_G \max_D V(D, G)$$

with

$$V(D, G) = \frac{1}{2}\mathbb{E}_{x\sim p_{data(x)}}[\log D(x)] + \frac{1}{2}\mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z))].$$

$V(D, G)$ is called the value function. Note that $V(D, G) = -J^{(D)} = J^{(G)}$.

(d) Why might the usage of $J^{(G)} = -J^{(D)}$ as loss for the generator lead to slow learning?

Hint: What happens to the gradient of the losses if $D(G(z))$ is small? (No calculation required)

Detailed Solution

If $D(G(z))$ is close to 0 this means that the discriminator correctly classified $G(z)$ as 'fake' (which is likely at the beginning of training since G is still poor). In this case the gradient of $J^{(D)}$ will be very small (because D did something right). However, this means that also the gradient of $J^{(G)}$ will be small and thus the generator doesn't learn to become better.

One solution is to this problem is to use $J^{(G)} = -\log(D(G(z))$ as loss for the generator, such that G will try to maximize the probability of D being wrong (i.e. $D(G(z))$ close to 1) rather then minimizing the probability of D being right.

Exercise 9-3 Diffusion Process

The basic idea behind diffusion models for image generation is as follows: In the forward process, we incrementally add random noise to an image, making it noisier and noisier with every iteration. The backward process, i.e., the generation of less noisy images from noisy ones, is then learned by a neural network. After training, we can apply this neural network iteratively after starting with pure noise and will obtain a generated image that matches the distribution of the original training images.

We formalize the forward and backward process as depicted in the figure below.

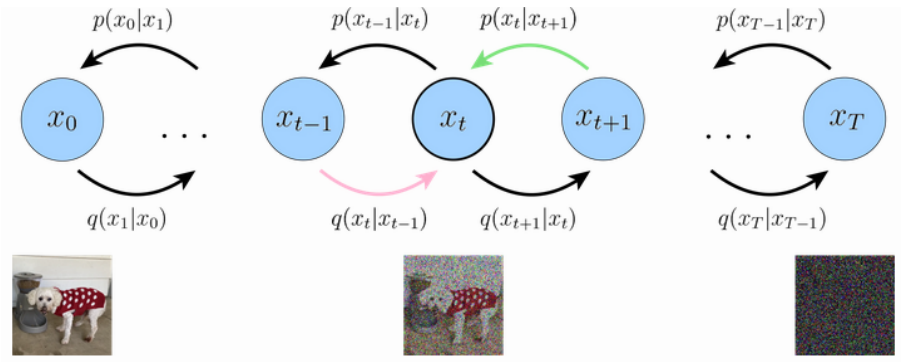

Figure 1: Source: `https://calvinyluo.com/2022/08/26/diffusion-tutorial.html`

(a) Let us assume a Gaussian distribution for $q(x_t|x_{t-1})$, i.e., $q(x_t|x_{t-1}) \sim \mathcal{N}(ax_{t-1}, b^2\mathbf{I})$. How can we sample $x_t|x_{t-1}$ under this assumption in practice?

Detailed Solution

We can use the reparameterization trick, i.e., we sample some noise $\epsilon_{t-1} \sim \mathcal{N}(0, \mathbf{I})$, and set $x_t = ax_{t-1} + b\epsilon_{t-1} \sim \mathcal{N}(ax_{t-1}, b^2\mathbf{I})$.

(b) How can we sample $x_t|x_0$ directly? Also, show that the distribution of $x_t|x_0$ will converge to $\mathcal{N}(0, \mathbf{I})$ as $t \to \infty$ if we set $a = \sqrt{\alpha}$ and $b = \sqrt{1-\alpha}$ for some $\alpha \in (0, 1)$.
Hint: Geometric series.
Note: In practice, α can be easily made dependent on t (i.e., α_t) without severe changes to the diffusion process.

Detailed Solution

We decompose x_t as

$$\begin{aligned} x_t &= \sqrt{\alpha} x_{t-1} + \sqrt{1-\alpha}\epsilon_{t-1} \\ &= \sqrt{\alpha}^2 x_{t-2} + \sqrt{1-\alpha}\epsilon_{t-1} + \sqrt{\alpha}\sqrt{1-\alpha}\epsilon_{t-2} \\ &= \sqrt{\alpha}^t x_0 + \sqrt{1-\alpha}\left(\epsilon_{t-1} + \sqrt{\alpha}\epsilon_{t-2} + \cdots + \sqrt{\alpha}^{t-1}\epsilon_0\right) \end{aligned}$$

The term including x_0 is fixed here because of the conditioning on x_0 and the remainder of the expression is a linear combination of independent standard normal distributions. That is, the resulting distribution will be a normal distribution with a mean of $\sqrt{\alpha}^t x_0$ and a covariance of

$$(1-\alpha)\left(1+\alpha+\cdots+\alpha^{t-1}\right)\mathbf{I} = (1-\alpha)\left(\frac{1-\alpha^t}{1-\alpha}\right)\mathbf{I} = (1-\alpha^t)\mathbf{I}.$$

Thus, we can directly sample $x_t|x_0$ with the same reparameterization trick from above and, for $t \to \infty$, the distribution clearly converges to $\mathcal{N}(0, \mathbf{I})$.

(c) The training objective for denoising diffusion models is generally the Evidence Lower Bound (ELBO), which constitutes a lower bound for the log likelihood of the data $\log p(x)$. Under some favourable conditions and choice of distribution assumptions, it is given by

$$\begin{aligned} ELBO_\theta(x) =& \mathbb{E}_{q(x_1|x_0)}[\log p_\theta(x_0|x_1)] && (1) \\ &- \mathrm{KL}(q(x_T|x_0)||p(x_T)) && (2) \\ &- \sum_{t=2}^{T} \mathbb{E}_{q(x_t|x_0)}[\frac{1}{2\sigma_q^2(t)}||\mu_q(x_t, x_0) - \mu_\theta(x_t)||^2], && (3) \end{aligned}$$

for specific choices of μ_q and σ_q (for details see `https://arxiv.org/pdf/2403.18103`). For parameterizing p_θ, we use a neural network $\mu_\theta(\cdot)$ and define $p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}|\mu_\theta(x_t), \sigma_t^2(t)\mathbf{I})$.

Show that the inner part of reconstruction term (1) reduces to

$$\log p_\theta(x_0|x_1) = -\frac{||x_0 - \mu_{\theta(x_1)}||^2}{2\sigma_q^2(1)} - \frac{d}{2}\log(2\pi\sigma_q^2(1)).$$

How can we interpret this reconstruction term (1) with respect to optimization?

Detailed Solution

$$\begin{aligned} \log p_\theta(x_0|x_1) &= \log \mathcal{N}(x_0|\mu_{\theta(x_1)}, \sigma_q^2(1)\mathbf{I}) \\ &= \log \frac{1}{\sqrt{2\pi\sigma_q^2(1)}^d} \exp\left(-\frac{||x_0 - \mu_\theta(x_1)||^2}{2\sigma_q^2(1)}\right) \\ &= -\frac{||x_0 - \mu_{\theta(x_1)}||^2}{2\sigma_q^2(1)} - \frac{d}{2}\log(2\pi\sigma_q^2(1)) \end{aligned}$$

The reconstruction term refers to the initial block of the diffusion process and measures how well our neural network is able to reconstruct samples from our training data distribution in terms of the log-likelihood. The expectation is taken with respect to the samples that are generated with a single noising step in the forward process.

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
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** $t = T, \dots, 1$ **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** $\mathbf{x}_0$

Figure 2: Training and Sampling with denoising diffusion models. Source: `https://arxiv.org/pdf/2006.11239`

(d) Discuss the meaning of the prior matching term (2) and its impact on optimization.

Detailed Solution

The prior matching term measures how well the distribution of $q(x_T|x_0)$ matches $p(x_T)$, which is assumed to be a standard normal distribution. A sufficiently large T and an appropriate design of the forward diffusion process ensure a good prior matching and there is no optimization of a neural network involved here, i.e., we can omit this for training).

(e) Finally, interpret the consistency term (3) with respect to optimization. Why does it look the way it does?

Detailed Solution

This term ensures that p_θ is trained in a way such that it can reverse the forward process. This term also corresponds to a Kullback-Leibler divergence, namely between $q(x_{t-1}|x_t, x_0)$ and $p_\theta(x_{t-1}, x_t)$, which are both normal distributions with the same covariance ($\sigma_q^2(t)\mathbf{I}$). Therefore, only their means $\mu_q(x_t, x_0)$ (known) and $\mu_\theta(x_t)$ (neural network) have to be aligned.

(f) Do we use the same or multiple different neural networks for different steps t in the denoising process? Why?

Detailed Solution

The same network is used for every step t as training a separate model for every step would lead to an excessive number of models.

Conclusion: We have seen that the optimizable part of the ELBO as a training objective boils down to a sum of squared distances (under some assumptions). This makes the training workflow of diffusion models remarkably simple despite the sophisticated underlying probabilistic concepts. Algorithms for training and inference are provided below.

Exercise 9-4 Latent Diffusion Models

(a) What is the main disadvantage of diffusion models?

Detailed Solution

Diffusion models are compute-intensive because of the iterative process at inference.

(b) What are the advantages of performing the diffusion modeling in a latent space instead of the pixel space for images?

Detailed Solution

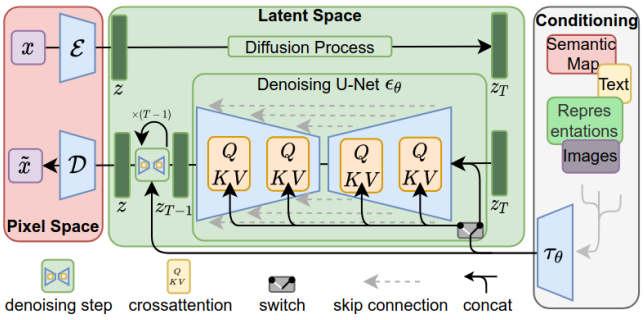

Figure 3: Latent diffusion model from https://arxiv.org/pdf/2112.10752.

In the latent space, information is represented in a more condensed and less redundant way, allowing the model to better learn and generate important features.

Furthermore, the latent space is lower dimensional than the pixel space, which greatly reduces the computational load of the diffusion model.

(c) Do we have control over the content or type of the generated samples in a diffusion model as seen in the previous exercise? How is this addressed in the latent diffusion model described in the Figure above?

Detailed Solution

In the diffusion model of the previous exercise, we have no control over the generated samples as we start with random noise and do not feed any other inputs into the model. Note that we do not have any idea which parts in the noise the model associates with which semantic features.

In order to gain control over the generated samples, we can condition the diffusion model on additional information. Probably the most famous example of this are text-to-image models where the input to the model is both random noise and a text prompt. However, other types of conditioning information are possible, e.g., image layouts.

On a technical level, this conditioning is implemented by encoding the conditioning information to an embedding (τ_θ) with a dedicated encoder and then letting the latent features of the diffusion model cross-attend to this embedding.

Exercise 9-5 Score-Based Perspective

The score function is the derivative of the log-likelihood function with respect to the data, i.e., $\nabla_x \log p(x)$. That is, given a point x, the score function tells us in which direction we can move to obtain a more likely point. In score-based generative modeling, the score function is learned such that we can start with an arbitrary point and, then, iteratively follow the score until we reach a mode in the data distribution (Langevin dynamics, see https://calvinyluo.com/2022/08/26/diffusion-tutorial.html#score-based-generative-m

Discuss the correspondences between score-based generative modeling and denoising diffusion models. What does the score function correspond to in diffusion models? What distribution is considered for the score function and what are its advantages?

Detailed Solution

If we consider the denoising diffusion model as the score function, we can naturally interpret the backward diffusion process as an instance of score-based generative models: We start with random noise and use a neural network to iteratively describe a path toward more and more likely locations.

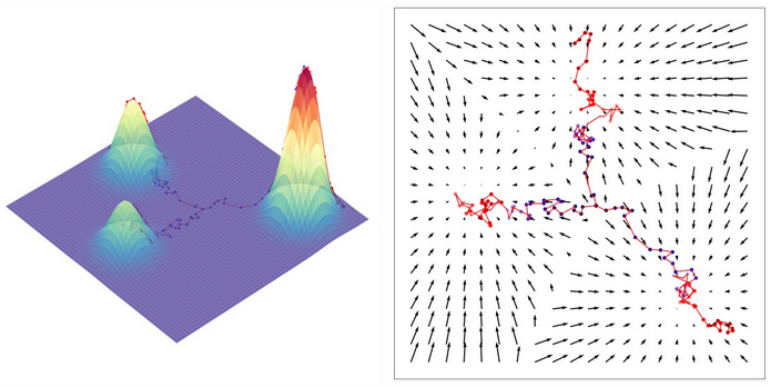

Figure 4: Score-based generative modeling. The score function is represented by the vector field on the right-hand side. The three trajectories correspond to three random variations of an iterative score-based sampling process with the same starting point. Source: `https://calvinyluo.com/2022/08/26/diffusion-tutorial.html#score-based-generative-models`

However, note that we do not model the score of the actual data distribution but the distribution containing multiple levels of Gaussian noise. This has the advantage that we have useful scores in regions that would have an extremely low density in the actual data distribution.

To gain more intuition for this, consider images as an example. Images lie on a manifold with a lower dimension than the pixel space, i.e., the vast majority of regions in distribution space $\mathbb{R}^{H \times W \times 3}$ do not contain valid images. In other words, if you randomly sample some pixel values, it is practically certain that you will not obtain a proper image. Therefore, starting from random pixel values, it is not feasible to tell how we can obtain a proper image from it as all surrounding locations do not represent proper images either (they are all equally unlikely). With the diffusion process, this problem is mitigated by adding multiple intermediate noise levels. In a way, the addition of Gaussian noise achieves that the data points used for training do not only lie in a low-dimensional and hard-to-reach manifold but expand to other regions of $\mathbb{R}^{H \times W \times 3}$ (up to pure Gaussian noise). From these regions, we can easily sample randomly and we can actually predict useful scores, i.e., directions for how to obtain more likely images (likely w.r.t. the distribution including multiple noise levels).