Deep Learning (BEV033DLE)
Lecture 12 Variational Autoencoders

Czech Technical University in Prague

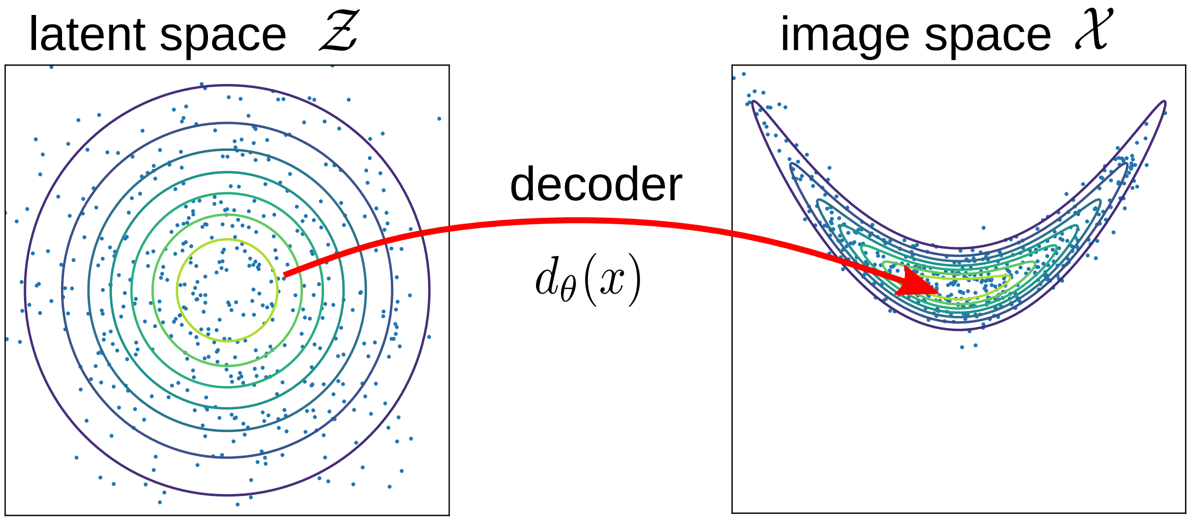
- Generative models in machine learning
- Variational autoencoders (VAE)
- Hierarchical VAE & diffusion models

Generative models

Generative models: Given training data $\mathcal{T} = \{x_j \mid j = 1, \ldots, \ell\}$ drawn i.i.d. from an unknown distribution $p_d(x)$, the goal is to learn a DNN model that allows to generate random instances of x similar to $x \sim p_d(x)$.

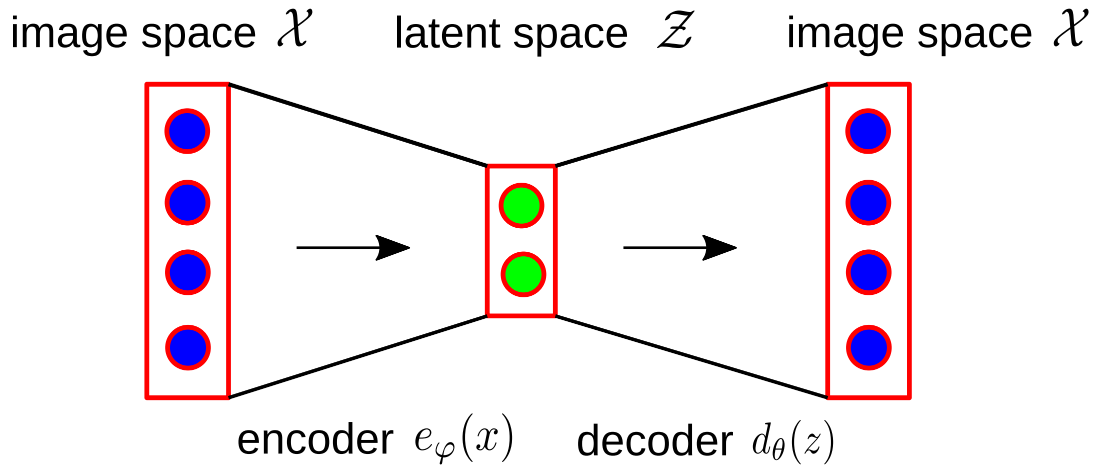
Approach this task by using *latent variable models:*

- fix a latent noise space $\mathcal{Z}$ and a distribution $p(z)$ on it,
- design a neural network d_θ that maps $\mathcal{Z}$ to the feature space $\mathcal{X}$,
- learn its parameters θ so that the resulting distribution $p_\theta(x)$ "reproduces" the data distribution.



Generative models

Classical autoencoder networks



e.g. with learning criterion $\mathbb{E}_{\mathcal{T}} \| x - d_\theta \circ e_\varphi(x) \|^2$. However,

- the distribution in the latent space is beyond our control,
- the model can not be used for sampling/generating x instances.

(Gaussian) Variational Autoencoders

- latent space $\mathcal{Z} = \mathbb{R}^m$, prior distribution $p(z)$: $\mathcal{N}(0, \mathbb{I})$
- image space $\mathcal{X} = \mathbb{R}^n$, conditional distribution $p_\theta(x \,|\, z)$: $\mathcal{N}(\mu_\theta(z), \sigma^2 \mathbb{I})$
  The mapping $\mathcal{Z} \ni z \mapsto \mu_\theta \in \mathcal{X}$ is modelled in terms of a (deep, convolutional) *decoder network* $d_\theta \colon \mathcal{Z} \to \mathcal{X}$.
- Learning goal: maximise data log-likelihood

$$L(\theta; \mathcal{T}) = \mathbb{E}_{\mathcal{T}} \log p_\theta(x) = \mathbb{E}_{\mathcal{T}} \log \int_{\mathcal{Z}} dz \, p_\theta(x \,|\, z) p(z)$$

Computing $L(\theta)$ or $\nabla_\theta L(\theta)$ is not tractable! It would require to integrate the decoder mapping $d_\theta(z)$ over the latent space $\mathcal{Z}$.

Proposal: Use ELBO, i.e. a lower bound of the data log-likelihood

$$L(\theta) \geqslant L_B(\theta, q) = \mathbb{E}_{\mathcal{T}} \mathbb{E}_{q(z \,|\, x)} \Big[\log p_\theta(x \,|\, z) - \log \frac{q(z \,|\, x)}{p(z)} \Big]$$

(Gaussian) Variational Autoencoders

$$L(\theta) \geqslant L_B(\theta, q) = \mathbb{E}_{\mathcal{T}} \mathbb{E}_{q(z\,|\,x)} \Big[\log p_\theta(x\,|\,z) - \log \frac{q(z\,|\,x)}{p(z)} \Big]$$

May be we can apply the *EM algorithm* directly?

EM-algorithm corresponds to block-coordinate ascent of $L_B(\theta, q)$ w.r.t. θ and q

E-step fix θ_t, set $q_t(z\,|\,x) = \arg\max_q L(\theta_t, q) \Rightarrow q_t(z\,|\,x) = p_{\theta_t}(z\,|\,x)$

M-step fix $q_t(z\,|\,x)$, maximise $\theta_{t+1} = \arg\max_\theta \mathbb{E}_{\mathcal{T}} \mathbb{E}_{q_t(z\,|\,x)} \log p_\theta(x\,|\,z)$

No, it is not feasible because computing

$$p_{\theta_t}(z\,|\,x) = \frac{p_{\theta_t}(x\,|\,z) p(z)}{\int dz'\, p_{\theta_t}(x\,|\,z') p(z')}$$

would require to integrate the decoder mapping.

(Gaussian) Variational Autoencoders

Way out: choose a class of *amortised inference* models $q_\varphi(z \,|\, x)$

$$z \,|\, x \sim \mathcal{N}\big(\mu_\varphi(x), \operatorname{diag}(\sigma_\varphi^2(x))\big)$$

The mapping $x \mapsto \mu_\varphi(x), \sigma_\varphi(x)$ is modelled in terms of a (deep, convolutional) *encoder network* $e_\varphi(x) = \big(\mu_\varphi(x), \sigma_\varphi(x)\big)$.

The ELBO criterion reads now

$$L_B(\theta, \varphi) = \mathbb{E}_\mathcal{T}\Big[\mathbb{E}_{q_\varphi(z \,|\, x)} \log p_\theta(x \,|\, z) - D_{KL}(q_\varphi(z \,|\, x) \,\|\, p(z))\Big]$$

Can we maximise it by gradient ascent w.r.t. θ and φ?

- $\mathbb{E}_\mathcal{T}$: SGD with mini-batches ✓
- $D_{KL}(q_\varphi(z \,|\, x) \,\|\, p(z))$: both Gaussians factorise and the KL-divergence decomposes into a sum over components $\sum_{i=1}^m D_{KL}(q_\varphi(z_i \,|\, x) \,\|\, p(z_i))$. The KL-divergence of univariate Gaussian distributions can be computed in closed form! ✓

(Gaussian) Variational Autoencoders

$$L_B(\theta,\varphi) = \mathbb{E}_{\mathcal{T}}\Big[\mathbb{E}_{q_\varphi(z\,|\,x)} \log p_\theta(x\,|\,z) - D_{KL}(q_\varphi(z\,|\,x) \parallel p(z))\Big]$$

- $\nabla_\theta \mathbb{E}_{q_\varphi(z\,|\,x)} \log p_\theta(x\,|\,z)$: use SGD by sampling $z \sim q_\varphi(z\,|\,x)$. ✓
- $\nabla_\varphi \mathbb{E}_{q_\varphi(z\,|\,x)} \log p_\theta(x\,|\,z)$: this gradient is *critical*.
  We can not replace $\mathbb{E}_{q_\varphi(z\,|\,x)}$ by a sample $z \sim q_\varphi(z\,|\,x)$, because it will depend on φ!

Re-parametrisation trick: Simple solution for Gaussians:

$$z \sim \mathcal{N}(\mu,\sigma^2) \iff \epsilon \sim \mathcal{N}(0,1) \text{ and } z = \sigma\epsilon + \mu$$

Now, if μ and σ depend on φ:

$$\nabla_\varphi \mathbb{E}_{z\sim\mathcal{N}(\mu_\varphi,\sigma_\varphi^2)}[f(z)] = \mathbb{E}_{\epsilon\sim\mathcal{N}(0,1)}\big[\nabla_\varphi f(\sigma_\varphi\epsilon + \mu_\varphi)\big]$$

(Gaussian) Variational Autoencoders

Overall, the learning step for a (Gaussian) VAE is pretty simple:

Fetch a mini-batch x from training data

1. apply the encoder network $e_\varphi(x) \mapsto \mu_\varphi(x), \sigma_\varphi(x)$ and compute $q_\varphi(z \,|\, x)$
2. compute the KL-divergence $D_{KL}(q_\varphi(z \,|\, x) \,\|\, p(z))$
3. sample a batch $z \sim q_\varphi(z \,|\, x)$ with reparametrisation
4. apply the decoder network $d_\theta(z) \mapsto \mu_\theta(z)$ and compute $\log p_\theta(x \,|\, z)$
5. combine the ELBO terms and let PyTorch compute the derivatives and make an SGD step.

Strengths and weaknesses of VAEs

- concise model, simple objective (ELBO), can be optimised by SGD ✓
- local optima, *posterior collapse*: some latent components collapse to $q_\varphi(z_i \,|\, x) = p(z_i)$, i.e. they carry no information. ✗
- amortised inference models $q_\varphi(z \,|\, x)$ have not enough expressive power to close the gap between $L(\theta)$ and $L_B(\theta, \varphi)$ for complex data distributions ✗

Hierarchical Variational Autoencoders

Closing the gap between $L(\theta)$ and $L_B(\theta, \varphi)$:

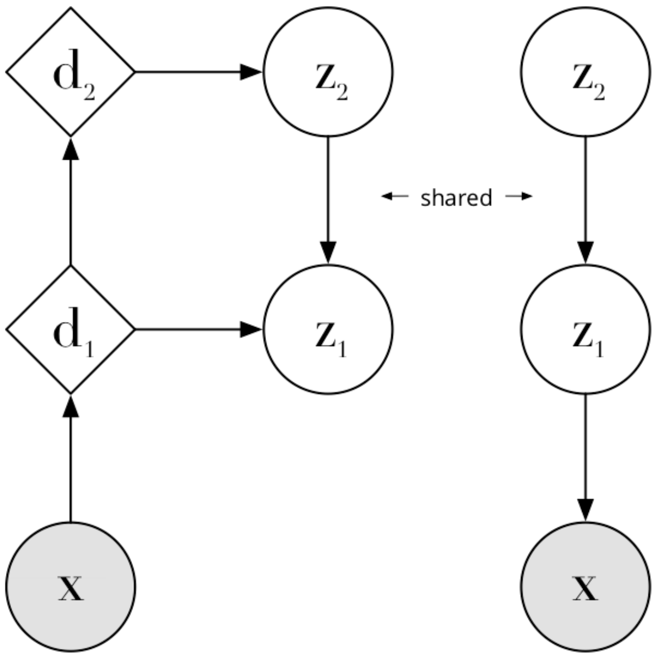
The hidden state z consists of groups $z_1, \ldots, z_m$.

$$p_\theta(x, z) = p(z_m) \prod_{i=1}^{m-1} p_\theta(z_i \,|\, z_{>i})\, p_\theta(x \,|\, z); \quad q_\varphi(z \,|\, x) = q_\varphi(z_m \,|\, x) \prod_{i=1}^{m-1} q_\varphi(z_i \,|\, z_{>i}, x).$$

The encoder shares parameters with the decoder, by assuming

$$q_{\theta,\varphi}(z_i \,|\, z_{>i}, x) \sim p_\theta(z_i \,|\, z_{>i}) d_i(z_i, x, \varphi),$$

where the functions d_i are hidden layer outputs of a deterministic encoder network whose forward direction is reverse to the factorisation order of the model.



Hierarchical VAEs can be learned by maximising ELBO.

Hierarchical Variational Autoencoders

A. Vahdat et al., NeurIPS 2020: A Deep Hierarchical VAE trained on CelebA data.

Hierarchical Variational Autoencoders

J. Ho et al., NeurIPS 2020, Denoising diffusion probabilistic models