DL $\quad$ L15: 30.06.2022

$\underline{x} \in \mathbb{R}^d$: Inputs

$\underline{z} = f_E(\underline{x}, \underline{\theta}_E) \in \mathbb{R}^c$: Latent variable / representation / code → hidden code of $\underline{x}$

$$\hat{\underline{x}} = f_D(\underline{z}, \underline{\theta}_D) = f_D(f_E(\underline{x}, \underline{\theta}_E), \underline{\theta}_D)$$
$$= f(\underline{x}, \underline{\theta}) \in \mathbb{R}^d$$

The output is a reconstruction of the input.

$\underline{\theta}_E, \underline{\theta}_D, \underline{\theta} = \begin{bmatrix} \underline{\theta}_E \\ \underline{\theta}_D \end{bmatrix}$ → Parameters to be learnt

$\underline{y} = \underline{x}$: Self-reconstruction

⇒ Self-supervised learning

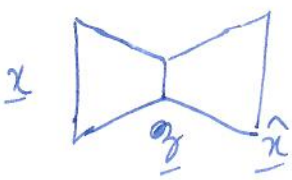
$\underbrace{\subset}_{\text{Subset}}$ unsupervised learning

↳ Supervised learning using unlabelled data (See Ch 10)

Encoder / Decoder :

- any DNN
- Symmetric structure

Always $\underset{\sim\sim\sim}{\text{under complete}}$ AE → $c < d$ [eg image compression]



bottleneck → NN learns the most important attributes

Overcomplete AE → $c > d$, never used

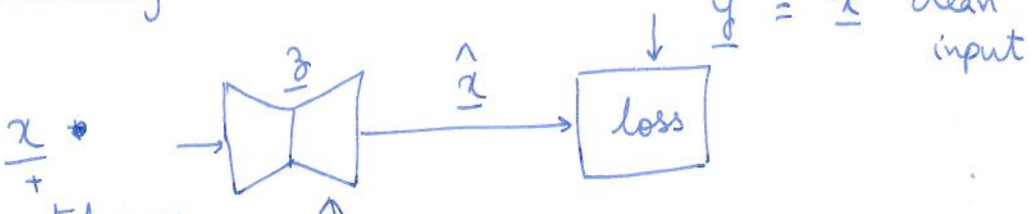
9-5

9-6

Applications of Auto Encoders

1. dimension reduction (data compression) $c < d$

2. Denoising AE



generated noise
or
corruption

noisy input

reduce noise corruption

$\underline{z}$: low-dimensional

→ keep only relevant information for clean input $\underline{x}$

→ drop noise / corruptions information as it is not there in output

9-7
⋮
9-9

Performs good on known examples but not good on unknown domains which are similar to the others but a lot vast also.

Theory of VAE:

$p(\underline{x})$: unknown distribution of $\underline{x}$,
only samples $\underline{x}(1) \cdots \underline{x}(N)$ available
We don't have $\underline{Y}$

$q(\underline{x}, \underline{\theta})$: parametric approximation of $p(\underline{x})$

$\underline{\theta}$: parameter vector of VAE

Goal : $\min_{\underline{\theta}} D_{KL}(p \| q)$

As in Ch 3.4 :

Cost fn: $L(\underline{\theta}) = D_{KL}(p \| q)$

$$= \underset{\underline{X} \sim p}{E}\left(\ln \frac{p(\underline{X})}{q(\underline{X}; \underline{\theta})} \right)$$

$$= \underbrace{\underset{\underline{X} \sim p}{E} \ln p(\underline{X})}_{\text{independent of } \underline{\theta}} - \underset{\underline{X} \sim p}{E} \ln q(\underline{X}; \underline{\theta})$$

unknown

Replace $p(\underline{x})$ by $\hat{p}(\underline{x}) = \frac{1}{N} \sum_{n=1}^{N} \delta(\underline{x} - \underline{x}(n))$

$$L(\theta) \approx \text{Constant} - \underset{\underline{X} \sim \hat{p}}{E} \ln q(\underline{X}; \underline{\theta})$$

$$= \text{Const} + \frac{1}{N} \sum_{n=1}^{N} \underbrace{- \ln q(\underline{x}(n); \underline{\theta})}_{\text{loss } l(\underline{x}(n); \underline{\theta}) \text{ of VAE}}$$

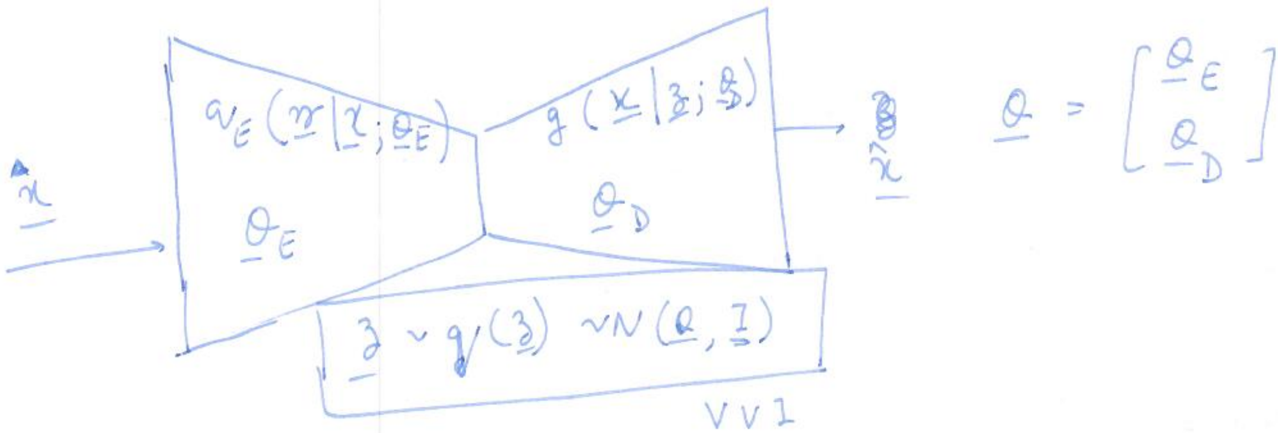
Diff ~~to~~ to the derivation supervised learning in Ch 3.4:

No y, $p(\underline{x}, \underline{y}) \to p(\underline{x})$ $\quad q(\underline{y} | \underline{x}; \underline{\theta}) \to q(\underline{x}; \underline{\theta})$

How to model $q(\underline{x};\underline{\theta})$?

VAE: $\underline{x}$ is generated by $\underline{z}$ with a known distribution

$$\underline{z} \sim q(\underline{z}) \quad \text{eg } \underline{z} \sim N(\underline{0}, \underline{\underline{I}})$$



$$\underline{\theta} = \begin{bmatrix} \underline{\theta}_E \\ \underline{\theta}_D \end{bmatrix}$$

i.e. $$q(\underline{x};\underline{\theta}) = \int q(\underline{x},\underline{z};\underline{\theta})\, d\underline{z}$$
$$= \int \underbrace{q(\underline{x}|\underline{z};\underline{\theta})}_{\text{decoder}}\ \underbrace{q(\underline{z})}_{\text{known}}\, dz$$

Since the distribution of the decoder is unknown due to multiple neural networks, the integration is hard to calculate. So, we can't calculate $\underline{\nabla} L$

1st trick of sol^n:

Replace $-\ln q(\underline{x};\underline{\theta})$ by its <u>variational upperbound</u>

Then we minimize the variational upperbound.

9 - 13

14

15