GAN vs VAE

Simonazzi Gian Marco

Corso di Fondamenti dell’Intelligenza Artificiale

A.A. 2023/24

Reti generative differenziabili

Definiscono una funzione $\mathrm{g}: \mathrm{z} \rightarrow \mathrm{x}$ su dei parametri θ.

Chiamiamo *variabili latenti* i valori del vettore z.

Partendo da z (attraverso un suo campionamento) è possibile generare una distribuzione su x.

L'obiettivo è di ottenere la migliore approssimazione possibile, perciò vogliamo che la likelihood $\mathcal{L}(\theta \mid x)$ sia massima, ossia trovare

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}}\, \mathcal{L}(\theta \mid x)$$

Reti neurali generative

Trovare il parametro θ adatto è difficile, possiamo usare una rete neurale.

Ma anche con una rete, il training della rete non è banale: variabili latenti e meccanismi di probabilità.

Tra le soluzioni più importanti per questo problema troviamo:

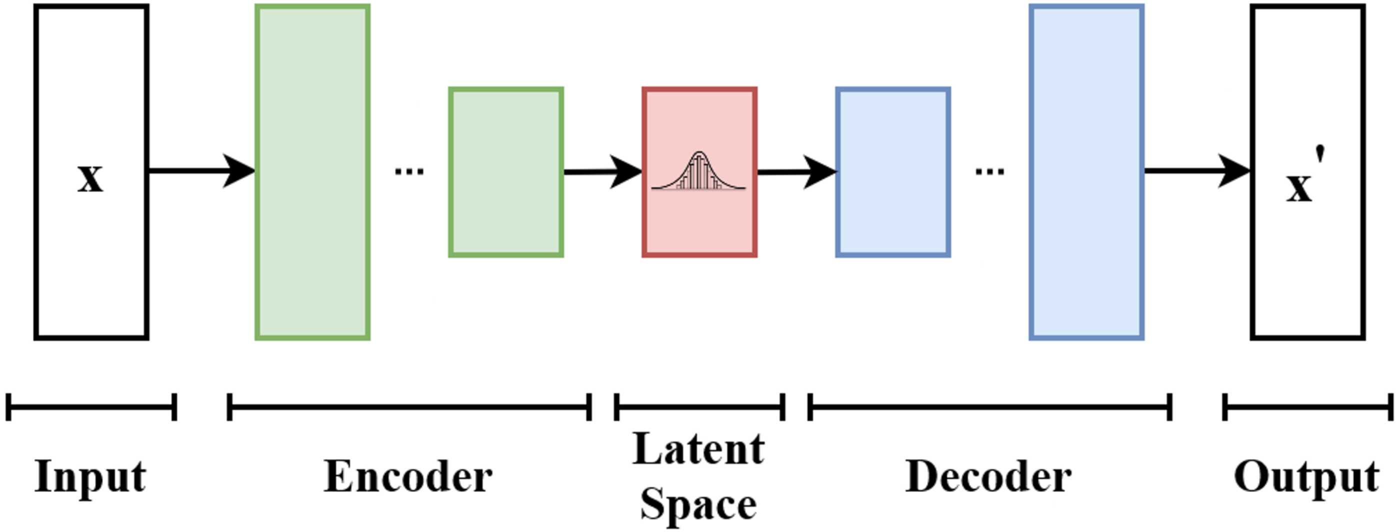
- Variational AutoEncoders (VAE)
- Generative Adversarial Networks (GAN)

Variational Auto Encoders

Nel 2014, viene pubblicato l'articolo Auto-Encoding Variational Bayes (1).

Viene descritto un meccanismo simile agli auto-encoders:

- Encoder: prende un input e lo comprime
- Decoder: decomprime verso l'output



https://en.wikipedia.org/wiki/Variational_autoencoder#/media/File:VAE_Basic.png

Il caso generativo

Consideriamo un campione x da una distribuzione X.

Assumiamo che coinvolga una variabile latente z.

1. Si ottiene un campione di z da $p_\theta(z)$
2. Si genera x dalla distribuzione condizionata $p_\theta(x \mid z)$

I valori di θ e z non sono noti.

Si assume che

$$\mathrm{p}_\theta(\mathrm{x}) = \int \mathrm{p}_\theta(\mathrm{z})\mathrm{p}_\theta(\mathrm{x} \mid \mathrm{z})\mathrm{dz} \quad \text{e} \quad \mathrm{p}_\theta(\mathrm{z} \mid \mathrm{x}) = \mathrm{p}_\theta(\mathrm{x} \mid \mathrm{z})\mathrm{p}_\theta(\mathrm{z})/\mathrm{p}_\theta(\mathrm{x})$$

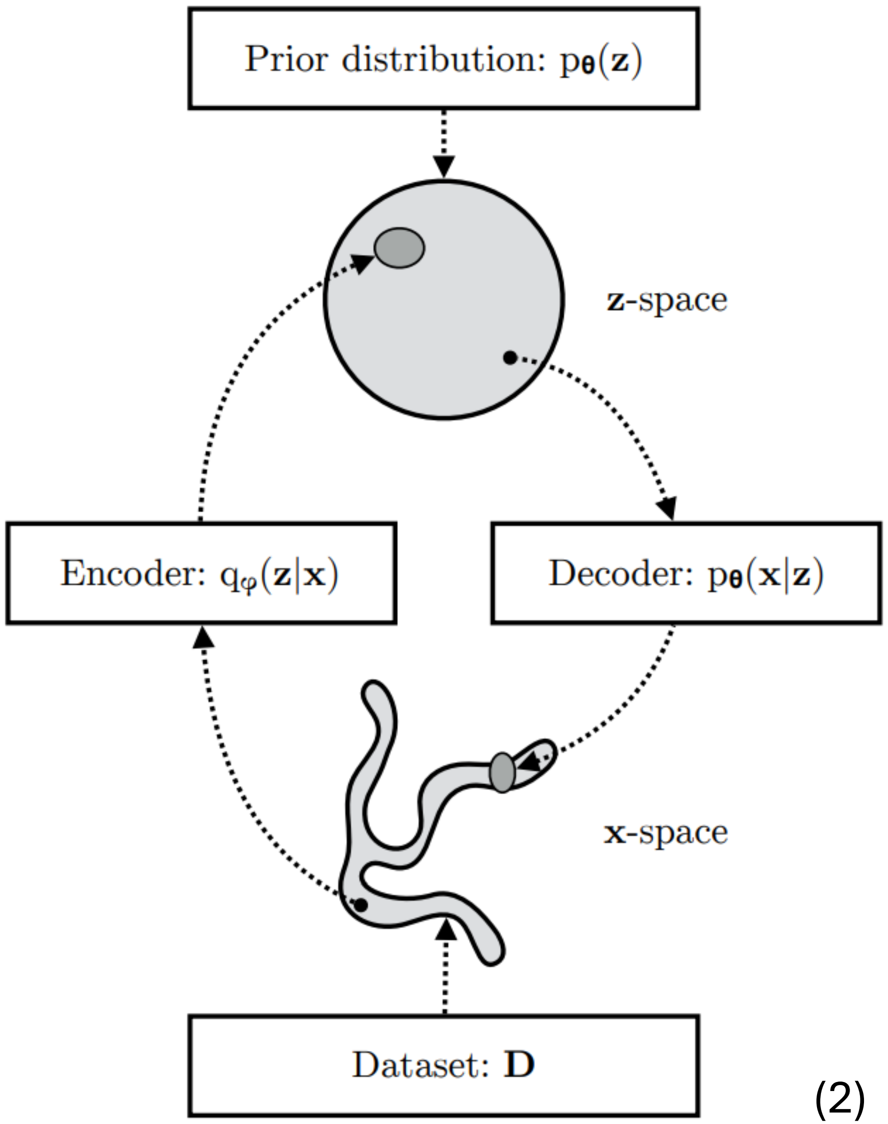
siano intrattabili.

Approssimazione di $p_\theta(z \mid x)$

Si introduce una approssimazione $q_\varphi(z \mid x)$, basata sul parametro φ.

Possiamo definire le due componenti:

1. Encoder: $q_\varphi(z \mid x)$
2. Decoder: $p_\theta(x \mid z)$



(2)

Evidence lower bound (ELBO)

La log-likelihood $\log p_\theta(x)$ può essere riscritta come

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) = D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}^{(i)})) + \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)})$$

con

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[-\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) + \log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})\right]$$

Siccome D_{KL} è positivo, possiamo dedurre che

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) \geq \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)})$$

Infine possiamo definire l'equazione

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) = \underbrace{-D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z}))}_{\text{Distanza tra z trovata e originale}} + \overbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z})\right]}^{\text{Similarità tra input e output}}$$

Reparameterization trick

I parametri θ e φ non possono ancora essere ottimizzati con SGD.

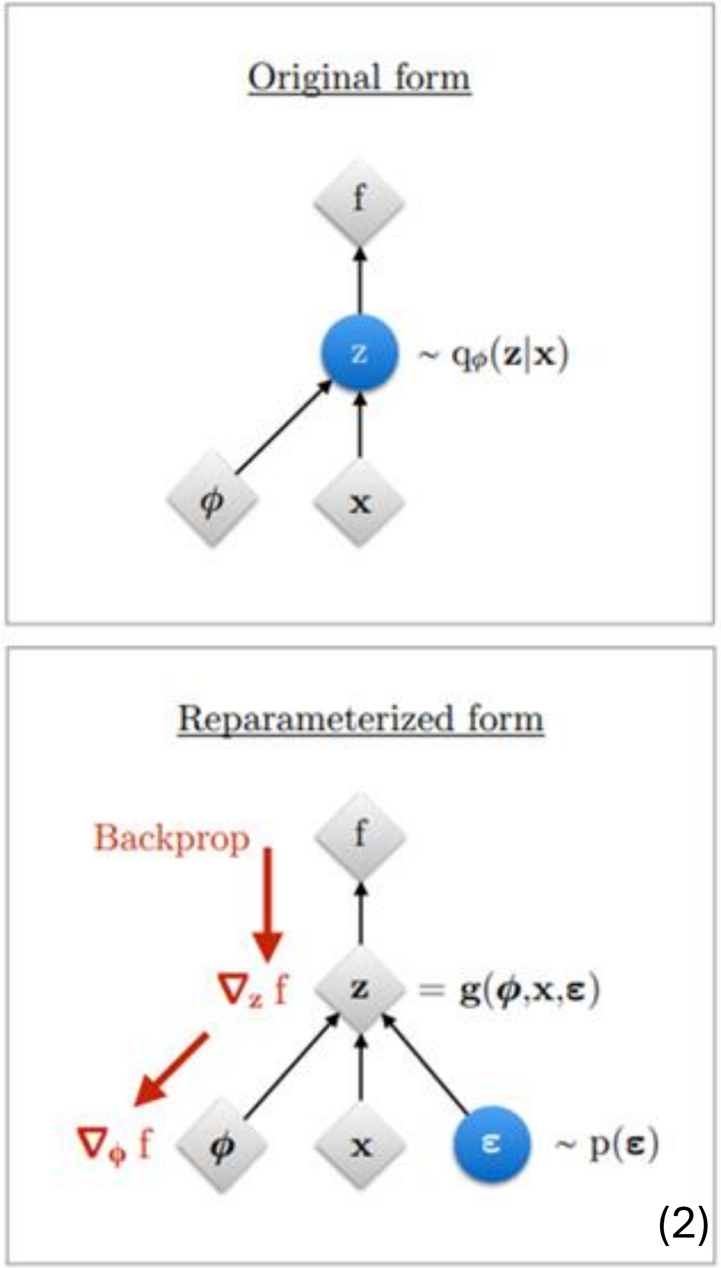
La variabile z impedisce la backpropagation.

Soluzione: si introduce una variabile stocastica

$$\epsilon \sim p(\epsilon)$$

rendendo quindi $\mathrm{z} = \mathrm{g}_{\varphi}(\epsilon, \mathrm{x})$

Si noti che rimane vero $\mathrm{z} \sim q_{\varphi}(\mathrm{z} \mid \mathrm{x})$



(2)

Auto-Encoding Variational Bayes algorithm

Un campione x passa per encoder e decoder, producendo un output.

La loss utilizzata è la seguente

$$\mathcal{L}(\boldsymbol{\theta}, \phi; \mathbf{x}^{(i)}) \simeq \frac{1}{2} \sum_{j=1}^{J} \left(1 + \log((\sigma_j^{(i)})^2) - (\mu_j^{(i)})^2 - (\sigma_j^{(i)})^2\right) + \frac{1}{L} \sum_{l=1}^{L} \log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)} | \mathbf{z}^{(i,l)})$$

con

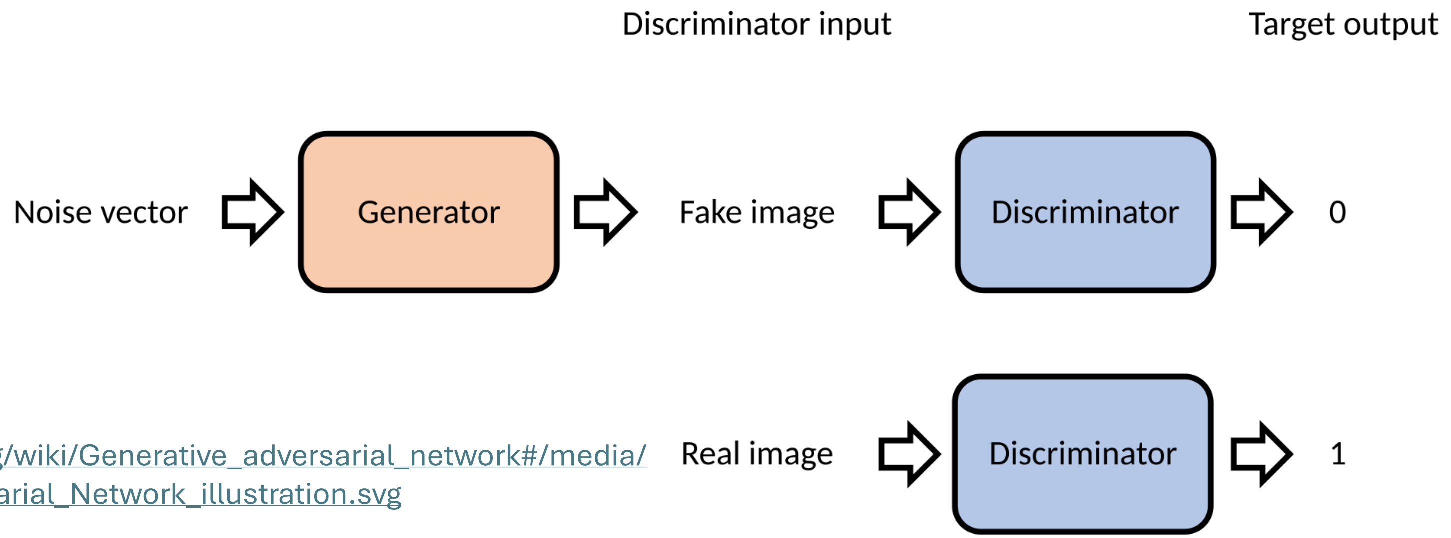
$$\mathbf{z}^{(i,l)} = \boldsymbol{\mu}^{(i)} + \boldsymbol{\sigma}^{(i)} \odot \boldsymbol{\epsilon}^{(l)} \qquad\qquad \boldsymbol{\epsilon}^{(l)} \sim \mathcal{N}(0, \mathbf{I})$$

Generative Adversarial Network

Questo metodo è stato proposto nel 2014 nel paper dello stesso nome (4).

Il training della rete generativa prevede due entità:

- Generatore: la rete generativa da addestrare.
- Detector: un classificatore che discrimina campioni veri da quelli generati.



https://en.wikipedia.org/wiki/Generative_adversarial_network#/media/File:Generative_Adversarial_Network_illustration.svg

Il meccanismo di gioco

Il detector definisce la likelihood $D(x)$ che x sia un campione vero.

Il generatore definisce una funzione $G: z \longmapsto x$.

Il detector vuole massimizzare $\log D(x)$ sugli x veri.

Il generatore vuole minimizzare $\log(1 - D(G(z)))$.

Questo meccanismo può essere sintetizzato come

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

Il training

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right].$$

end for

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

Goodfellow, Ian, et al. "Generative adversarial nets." *Advances in neural information processing systems* 27 (2014).

Risultato ottimale

Si dimostra che il detector ottimale avrà $D_G(x) = \frac{1}{2}$

Con

$$p_g = p_{data}$$

E perciò si avrà il minimo del generatore

$$C(G) = \max_D V(G, D) = -\log 4$$

Differenze tra VAE e GAN - training

VAE	GAN
• Inferenza senza Markov Chain Monte Carlo • Singola loss function di due componenti • Tradeoff tra mixing e generazione • Latent space più strutturato • Generazione da latent space • Obiettivo: generare campione simile all’originale	• Training senza inferenza • Due loss function da ottimizzare • Sincronizzazione tra detector e generatore • Latent space meno strutturato • Generazione da rumore • Obiettivo: generare campione verosimile

Riferimenti

1. Kingma, Diederik P., and Max Welling. "Auto-encoding variational bayes." *arXiv preprint arXiv:1312.6114* (2013).
2. Kingma, Diederik P., and Max Welling. "An introduction to variational autoencoders." *Foundations and Trends® in Machine Learning* 12.4 (2019): 307-392.
3. Doersch, Carl. "Tutorial on variational autoencoders." *arXiv preprint arXiv:1606.05908* (2016).
4. Goodfellow, Ian, et al. "Generative adversarial nets." *Advances in neural information processing systems* 27 (2014).