Lecture 19: Generative Networks
CS480/680 Intro to Machine Learning

2023-3-21

Pascal Poupart
David R. Cheriton School of Computer Science

UNIVERSITY OF WATERLOO

Generative networks

- Neural networks are typically used for classification or regression
  - Input: data
  - Output: class or prediction

- **Can we design neural networks that can generate data?**
  - Input: random vector
  - Output: data

Generative networks

- Several types of generative networks
  - Boltzmann machines
  - Sigmoid belief networks
  - **Variational autoencoders**
  - **Generative adversarial networks**
  - Entropic autoencoder
  - Generative moment matching networks
  - Sum-product networks
  - **Normalizing flows**
  - **Diffusion models**
  - …

Recall Probabilistic Autoencoder

- Let f and g represent conditional distributions

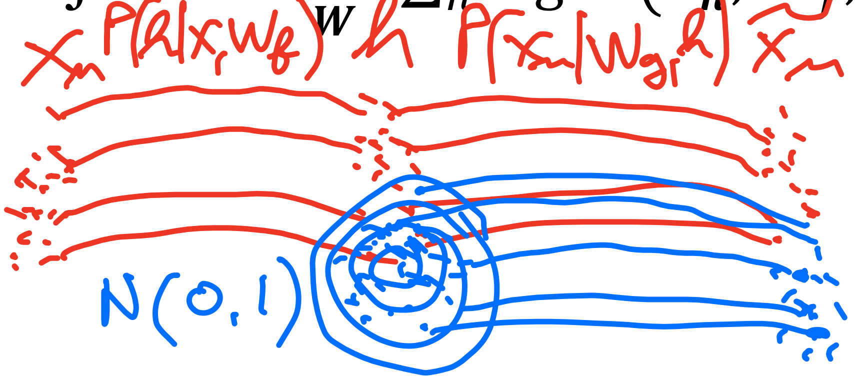
$$f: \Pr(\boldsymbol{h}|\boldsymbol{x}; \boldsymbol{W}_f) \quad \text{and} \quad g: \Pr\left(\boldsymbol{x}\middle|\boldsymbol{h}; \boldsymbol{W}_g\right)$$

- The decoder g can be treated as a generative model
  1. Sample $\boldsymbol{h}$ from $\Pr(\boldsymbol{h})$
  2. Sample $\boldsymbol{x}$ from $\Pr\left(\boldsymbol{x}\middle|\boldsymbol{h}; \boldsymbol{W}_g\right)$

- Question: how do we choose $\Pr(\boldsymbol{h})$?
  NB: We cannot use $\Pr(\boldsymbol{h}|\boldsymbol{x}; \boldsymbol{W}_f)$ since it is conditioned on $\boldsymbol{x}$, which we are trying to generate.

Variational Autoencoders

- Idea: train encoder $\Pr(\boldsymbol{h}|\boldsymbol{x};\boldsymbol{W}_f)$ to approach a simple and fixed distribution, e.g., $N(\boldsymbol{h};\boldsymbol{0},\boldsymbol{I})$
- This way we can set $\Pr(\boldsymbol{h})$ to $N(\boldsymbol{h};\boldsymbol{0},\boldsymbol{I})$
- Objective: $\max_{\boldsymbol{W}} \sum_n \log \Pr(\boldsymbol{x}_n;\boldsymbol{W}_f,\boldsymbol{W}_g) - c\, \underbrace{KL(\Pr(\boldsymbol{h}|\boldsymbol{x}_n;\boldsymbol{W}_f)\,||\,N(\boldsymbol{h};\boldsymbol{0},\boldsymbol{I}))}$



Kullback-Leibler divergence
Distance measure for distributions

UNIVERSITY OF WATERLOO

Variational Autoencoder Likelihood

- How do we compute $\Pr(\boldsymbol{x}_n; \boldsymbol{W}_f, \boldsymbol{W}_{\boldsymbol{g}})$?

$$\Pr(\boldsymbol{x}_n; \boldsymbol{W}_f, \boldsymbol{W}_{\boldsymbol{g}}) = \int_{\boldsymbol{h}} \Pr(\boldsymbol{x}_n | \boldsymbol{h}; \boldsymbol{W}_g) \Pr(\boldsymbol{h} | \boldsymbol{x}_n; \boldsymbol{W}_f)\, d\boldsymbol{h}$$

- Since $\Pr(\boldsymbol{h}|\boldsymbol{x}_{\boldsymbol{n}}; \boldsymbol{W}_f)$ should approach $N(h; 0, I)$, then force $\Pr(\boldsymbol{h}|\boldsymbol{x}_{\boldsymbol{n}}; \boldsymbol{W}_f)$ to be Gaussian

$$\Pr(\boldsymbol{h} | \boldsymbol{x}_n; \boldsymbol{W}_f) = N(\boldsymbol{h}; \mu_n(\boldsymbol{x}_n; \boldsymbol{W}_f), \sigma_n(\boldsymbol{x}_n; \boldsymbol{W}_f)\boldsymbol{I})$$

where the mean μ_n and variance σ_n are obtained by a neural net in $\boldsymbol{x}_n$ parametrized by $\boldsymbol{W}_f$

Variational Autoencoder Likelihood

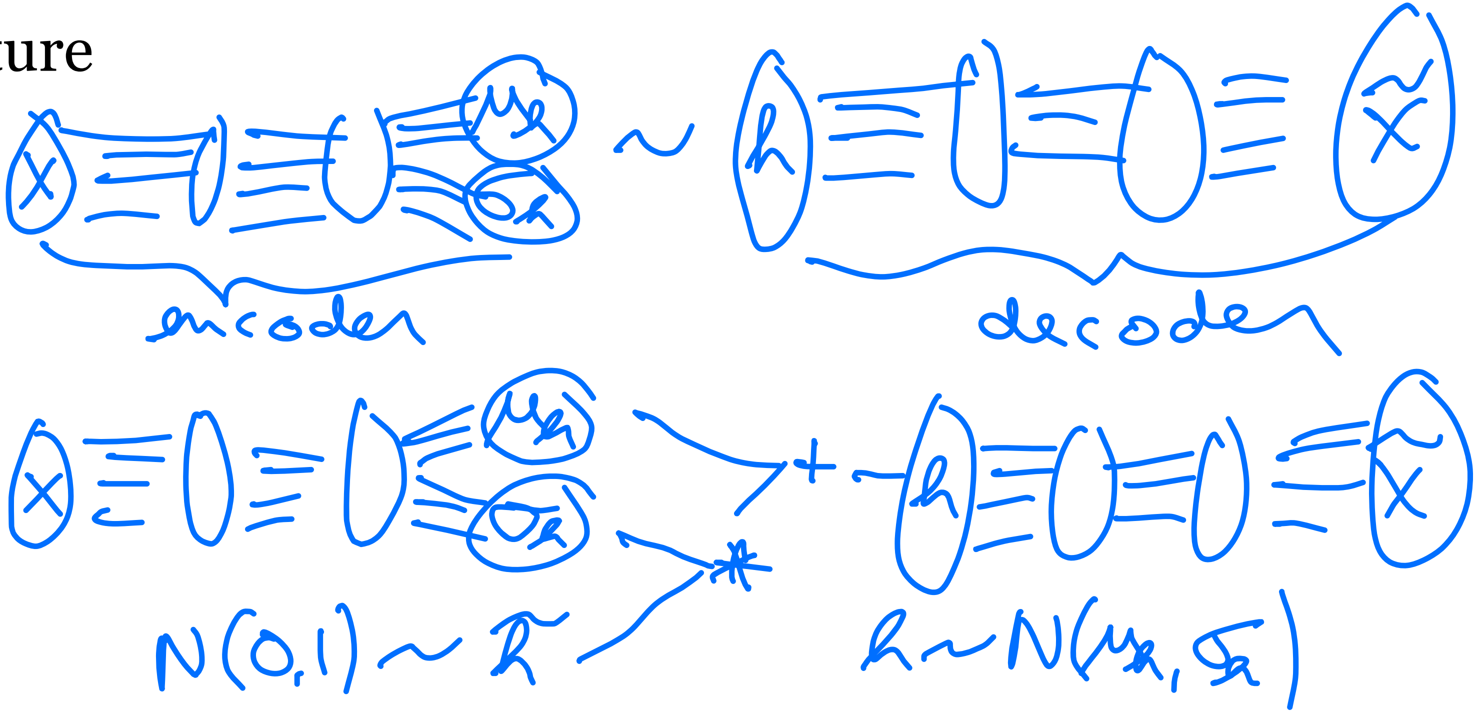
- Approximate the integral over $\boldsymbol{h}$

$$\Pr(\boldsymbol{x}_n; \boldsymbol{W}_f, \boldsymbol{W}_{\boldsymbol{g}}) = \int_{\boldsymbol{h}} \Pr(\boldsymbol{x}_n | \boldsymbol{h}; \boldsymbol{W}_g) N(\boldsymbol{h}; \mu_n(\boldsymbol{x}_n; \boldsymbol{W}_f), \sigma_n(\boldsymbol{x}_n; \boldsymbol{W}_f)\boldsymbol{I})\, d\boldsymbol{h}$$

- by a single sample: $\Pr(\boldsymbol{x}_n; \boldsymbol{W}_f, \boldsymbol{W}_{\boldsymbol{g}}) \approx \Pr(\boldsymbol{x}_n | \boldsymbol{h}_n; \boldsymbol{W}_g)$
  where $\boldsymbol{h}_n \sim N(\boldsymbol{h}; \mu_n(\boldsymbol{x}_n; \boldsymbol{W}_f), \sigma_n(\boldsymbol{x}_n; \boldsymbol{W}_f)\boldsymbol{I})$

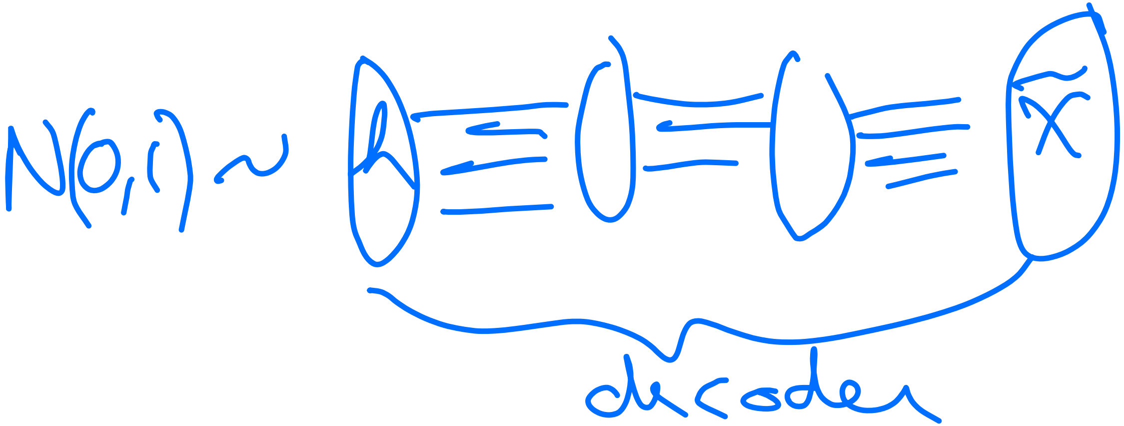
Variational Autoencoder Training

- Training by backpropagation
- Picture

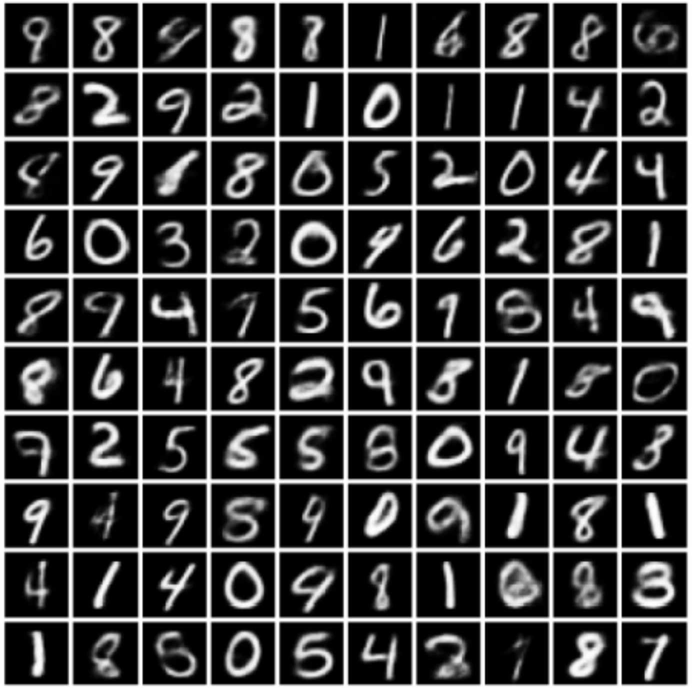


Variational Autoencoder Testing

- Testing corresponds to generating a data point
- Picture



Images generated with VAEs



Generative Adversarial Networks

- Approach based on game theory
- Two networks:
  1. Generator $g(\boldsymbol{z}; \boldsymbol{W}_g) \rightarrow \boldsymbol{x}$
  2. Discriminator $d(\boldsymbol{x}; \boldsymbol{W}_d) \rightarrow \Pr(\boldsymbol{x}\ is\ real)$

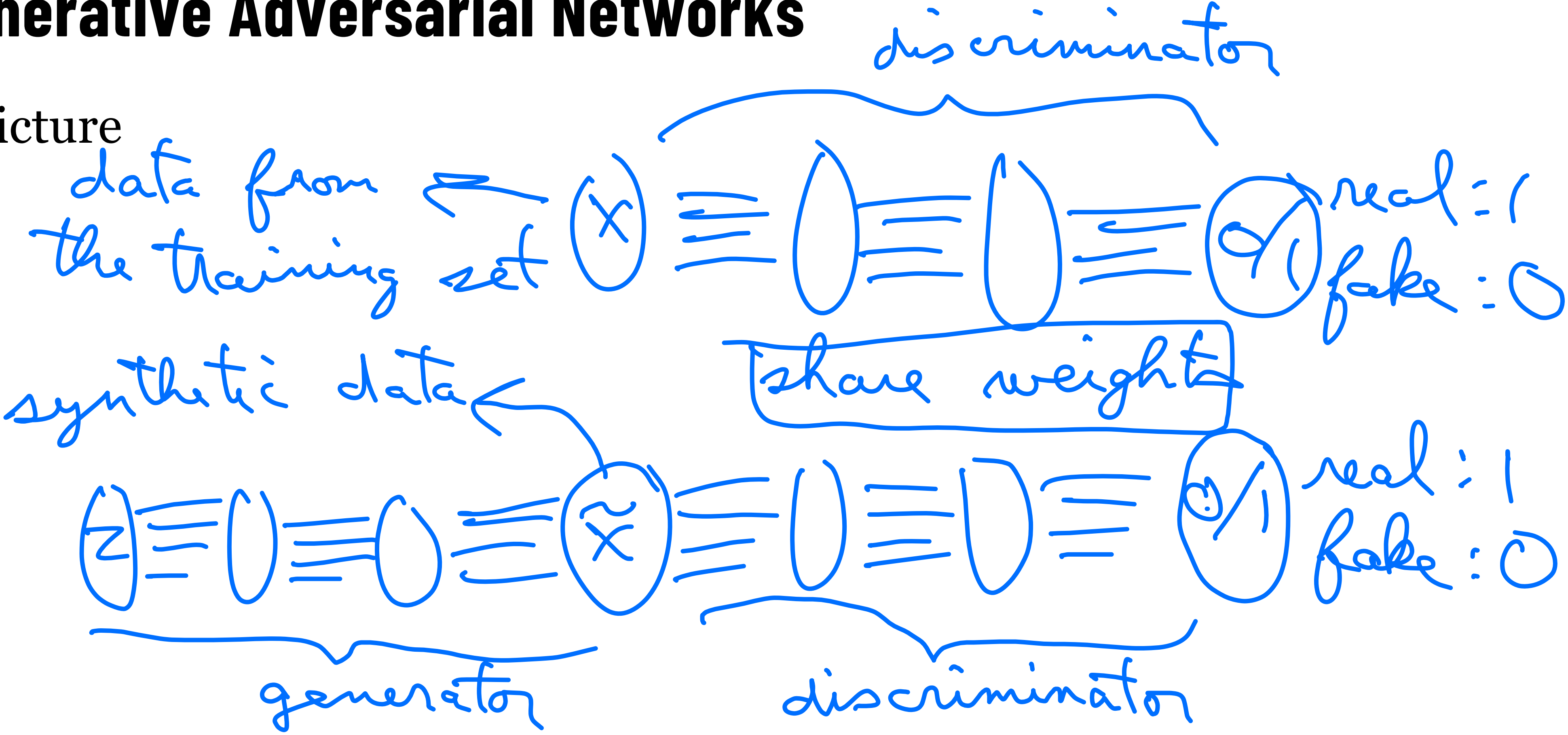
- Objective:

$$\min_{\boldsymbol{W}_g} \max_{\boldsymbol{W}_d} \sum_n \log \Pr(\boldsymbol{x}_n\ is\ real; \boldsymbol{W}_d) + \log \Pr\left(g\left(\boldsymbol{z}_n; \boldsymbol{W}_g\right)\ is\ fake; \boldsymbol{W}_d\right)$$

$$\equiv \min_{\boldsymbol{W}_g} \max_{\boldsymbol{W}_d} \sum_n \log d(\boldsymbol{x}_n; \boldsymbol{W}_d) + \log\left(1 - d\left(g(\boldsymbol{z}_n; \boldsymbol{W}_g); \boldsymbol{W}_d\right)\right)$$

Generative Adversarial Networks

- Picture



GAN training

- Repeat until convergence
  - For k steps do
    - Sample $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N$ from $\Pr(\boldsymbol{z})$
    - Sample $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ from training set
    - Update discriminator by ascending its stochastic gradient

$$\nabla_{\boldsymbol{W}_d}\left(\frac{1}{N}\sum_{n=1}^{N}\left[\log d(\boldsymbol{x}_{\boldsymbol{n}}; \boldsymbol{W}_d) + \log\left(1 - d\left(g\left(\boldsymbol{z}_n; \boldsymbol{W}_g\right); \boldsymbol{W}_d\right)\right)\right]\right)$$

  - Sample $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N$ from $\Pr(\boldsymbol{z})$
  - Update generator by descending its stochastic gradient

$$\nabla_{\boldsymbol{W}_g}\left(\frac{1}{N}\sum_{n=1}^{N}\log\left(1 - d\left(g\left(\boldsymbol{z}_n; \boldsymbol{W}_g\right); \boldsymbol{W}_d\right)\right)\right)$$

GAN training

- In the limit (with sufficiently expressive networks, sufficient data and global convergence)
  - $\Pr(\boldsymbol{x}|\boldsymbol{z};\boldsymbol{W}_g) \rightarrow \textit{true data distribution}$
  - $\Pr(\boldsymbol{x}\ \textit{is real};\boldsymbol{W}_d) \rightarrow 0.5$ (for real and fake data)

- Problems in practice:
  - **Imbalance: one network may dominate the other**
  - **Local convergence**

Images generated with GANs

Right columns are nearest neighbour training examples of adjacent column