

Lecture 19: Generative Networks

CS480/680 Intro to Machine Learning

2023-3-21

Pascal Poupart
David R. Cheriton School of Computer Science



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

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

- The decoder g can be treated as a generative model

1. Sample \mathbf{h} from $\Pr(\mathbf{h})$
2. Sample \mathbf{x} from $\Pr(\mathbf{x}|\mathbf{h}; \mathbf{W}_g)$

- Question: how do we choose $\Pr(\mathbf{h})$?

NB: We cannot use $\Pr(\mathbf{h}|\mathbf{x}; \mathbf{W}_f)$ since it is conditioned on \mathbf{x} , which we are trying to generate.

Variational Autoencoders

- Idea: train encoder $\Pr(\mathbf{h}|\mathbf{x}; \mathbf{W}_f)$ to approach a simple and fixed distribution, e.g., $N(\mathbf{h}; \mathbf{0}, \mathbf{I})$
- This way we can set $\Pr(\mathbf{h})$ to $N(\mathbf{h}; \mathbf{0}, \mathbf{I})$

- Objective: $\max_{\mathbf{W}} \sum_n \log \Pr(\mathbf{x}_n; \mathbf{W}_f, \mathbf{W}_g) - c \underbrace{KL(\Pr(\mathbf{h}|\mathbf{x}_n; \mathbf{W}_f) || N(\mathbf{h}; \mathbf{0}, \mathbf{I}))}_{\text{Kullback-Leibler divergence}}$

Kullback-Leibler divergence
Distance measure for distributions

Variational Autoencoder Likelihood

- How do we compute $\Pr(\mathbf{x}_n; \mathbf{W}_f, \mathbf{W}_g)$?

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

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

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

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

Variational Autoencoder Likelihood

- Approximate the integral over \mathbf{h}

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

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

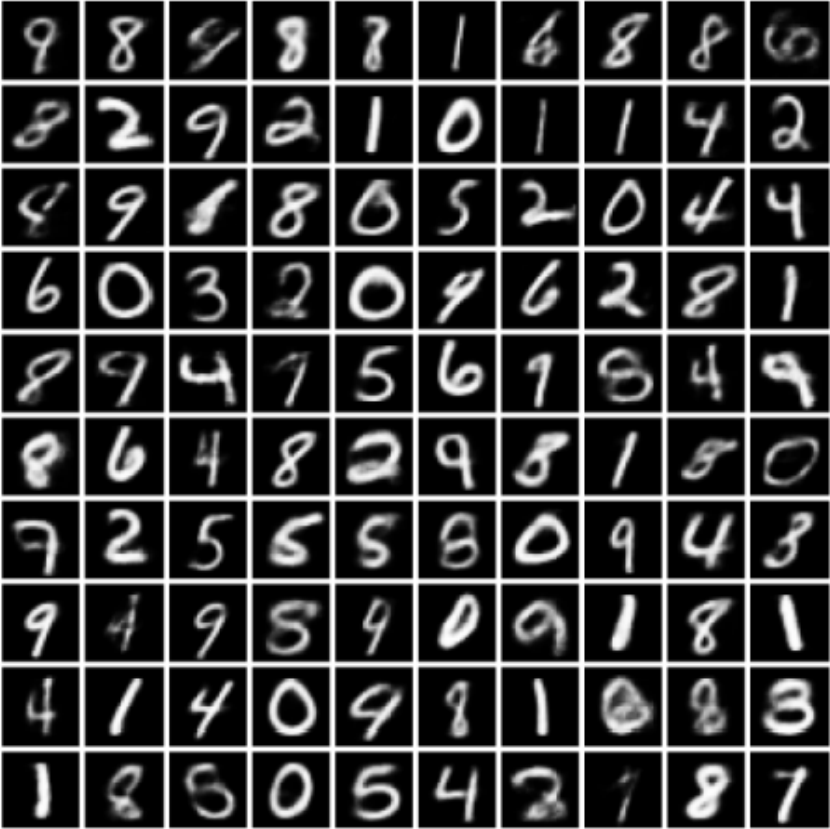
Variational Autoencoder Training

- 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 $g(\mathbf{z}; \mathbf{W}_g) \rightarrow \mathbf{x}$
 2. Discriminator $d(\mathbf{x}; \mathbf{W}_d) \rightarrow \Pr(\mathbf{x} \text{ is real})$
- Objective:

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

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

Generative Adversarial Networks

- Picture

GAN training

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

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

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

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

GAN training

- In the limit (with sufficiently expressive networks, sufficient data and global convergence)
 - $\Pr(\mathbf{x}|\mathbf{z}; \mathbf{W}_g) \rightarrow \text{true data distribution}$
 - $\Pr(\mathbf{x} \text{ is real}; \mathbf{W}_d) \rightarrow 0.5$ (for real and fake data)
- Problems in practice:
 - **Imbalance: one network may dominate the other**
 - **Local convergence**

Images generated with GANs

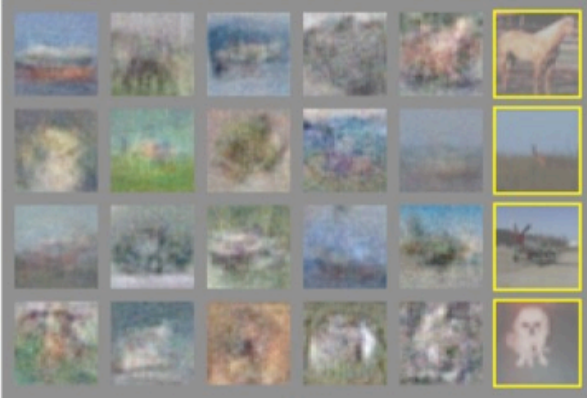
Right columns are nearest neighbour training examples of adjacent column



a)



b)



c)



d)