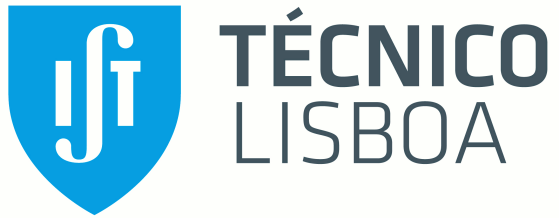
Lecture 12: Deep Generative Models

André Martins, Francisco Melo, Mário Figueiredo



Deep Learning Course, Winter 2023-2024

Announcements

- Thursday (January 4): guest lecture by Chrysoula Zerva.

  Grande Auditório, Centro de Congressos, 13:30

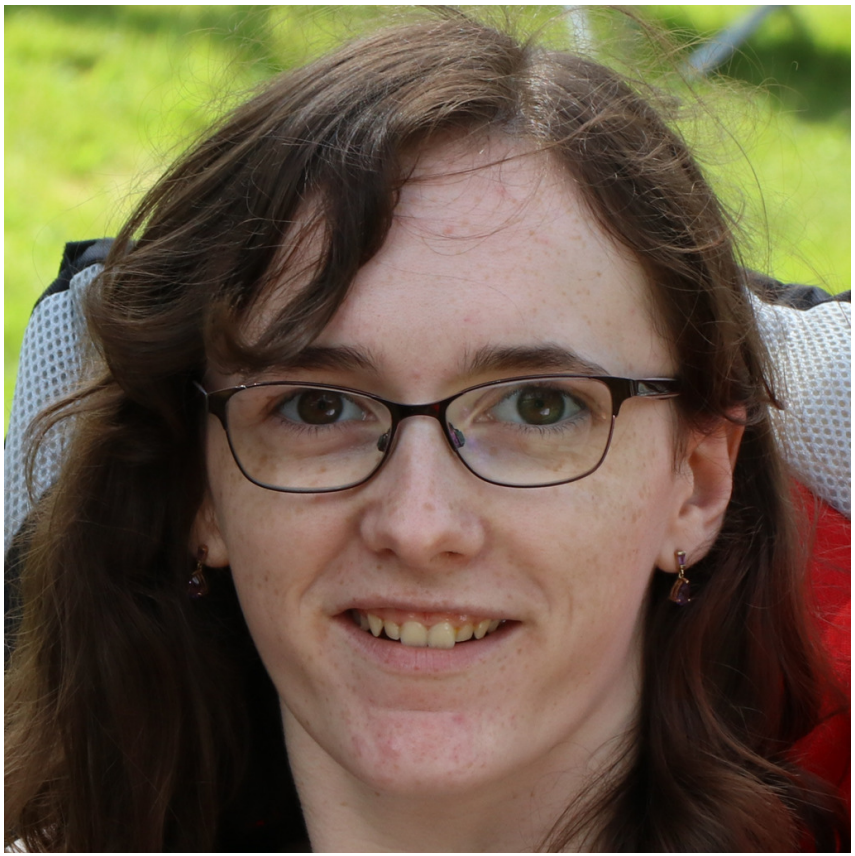
Today's Roadmap

Most of the course was about supervised learning.

Today we'll talk about deep generative models and unsupervised learning.

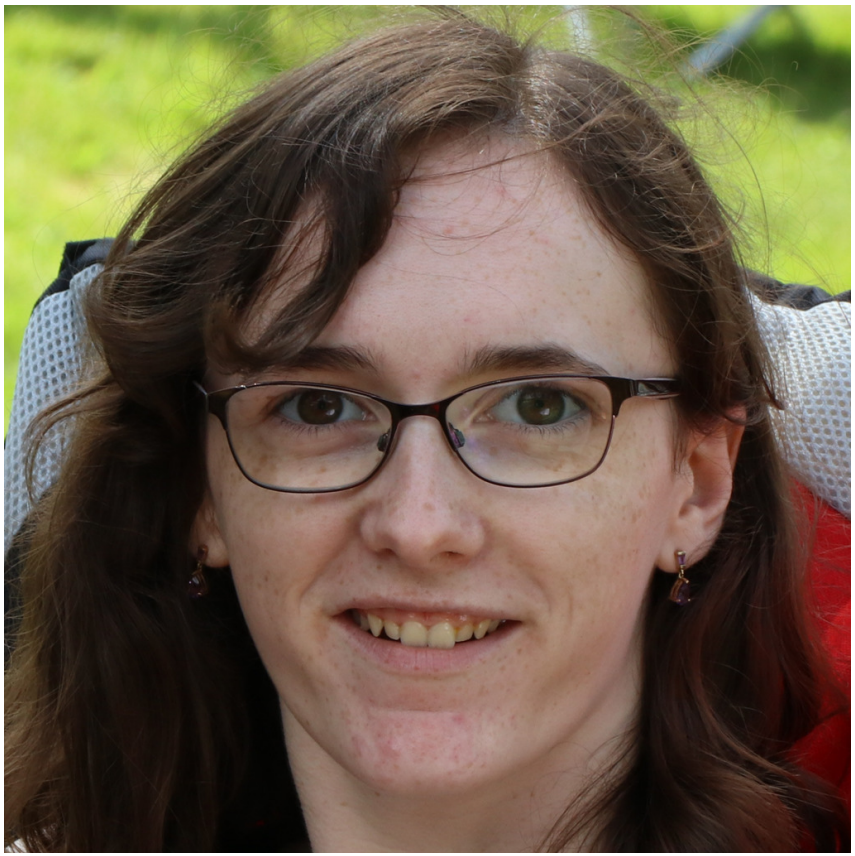
- Deep auto-regressive models
- Evidence lower bound (ELBO) and variational inference
- Variational auto-encoders
- Generative adversarial networks

Which of these people is real?



(http://www.whichfaceisreal.com)

Which of these people is real?



(http://www.whichfaceisreal.com)

Which of these people is real?

(http://www.whichfaceisreal.com)

Which of these people is real?

(http://www.whichfaceisreal.com)

Which of these people is real?

(http://www.whichfaceisreal.com)

Which of these people is real?

(http://www.whichfaceisreal.com)

Which of these people is real?

(http://www.whichfaceisreal.com)

Which of these people is real?

(http://www.whichfaceisreal.com)

Generative Modeling

- Modelling high-dimensional data (e.g., images) is a hard problem.

Generative Modeling

- Modelling high-dimensional data (e.g., images) is a hard problem.
- Deep generative models are currently making remarkable progress.

Generative Modeling

- Modelling high-dimensional data (e.g., images) is a hard problem.
- Deep generative models are currently making remarkable progress.
- Unsupervised learning: modelling $\mathbb{P}(\boldsymbol{x})$.
- Supervised learning: modelling $\mathbb{P}(\boldsymbol{x}, \boldsymbol{y})$.

Generative Modeling

- Modelling high-dimensional data (e.g., images) is a hard problem.
- Deep generative models are currently making remarkable progress.
- Unsupervised learning: modelling $\mathbb{P}(\boldsymbol{x})$.
- Supervised learning: modelling $\mathbb{P}(\boldsymbol{x}, \boldsymbol{y})$.
- Often, generative models use latent variables $\boldsymbol{h}$, such that

$$\mathbb{P}(\boldsymbol{x}) = \sum_{\boldsymbol{h}} \mathbb{P}(\boldsymbol{x}, \boldsymbol{h}) \quad \text{or} \quad \mathbb{P}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{\boldsymbol{h}} \mathbb{P}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{h})$$

  where $\mathbb{P}(\boldsymbol{x}, \boldsymbol{h})$ and $\mathbb{P}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{h})$ are "simple".

Examples of Deep Generative Models

- Auto-regressive networks
- Restricted Boltzmann machines
- Deep belief networks
- Deep Boltzmann machines
- Gaussian-Bernoulli RBMs
- Convolutional Boltzmann machines
- Variational auto-encoders (VAE)
- Generative adversarial networks (GAN)
- Denoising diffusion models

Examples of Deep Generative Models

- Auto-regressive networks
- Restricted Boltzmann machines
- Deep belief networks
- Deep Boltzmann machines
- Gaussian-Bernoulli RBMs
- Convolutional Boltzmann machines
- Variational auto-encoders (VAE)
- Generative adversarial networks (GAN)
- Denoising diffusion models

Outline

Energy Based Models

- A probability distribution (mixing observed and latent variables) via an energy function $E(\boldsymbol{x}, \boldsymbol{h}; \boldsymbol{\theta})$:

$$\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{h}) = \frac{\exp(-E(\boldsymbol{x}, \boldsymbol{h}; \boldsymbol{\theta}))}{Z(\boldsymbol{\theta})}, \quad \text{where } Z(\boldsymbol{\theta}) = \sum_{\boldsymbol{x}, \boldsymbol{h}} \exp(-E(\boldsymbol{x}, \boldsymbol{h}; \boldsymbol{\theta}))$$

Energy Based Models

- A probability distribution (mixing observed and latent variables) via an energy function $E(\boldsymbol{x}, \boldsymbol{h}; \boldsymbol{\theta})$:

$$\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{h}) = \frac{\exp(-E(\boldsymbol{x}, \boldsymbol{h}; \boldsymbol{\theta}))}{Z(\boldsymbol{\theta})}, \quad \text{where } Z(\boldsymbol{\theta}) = \sum_{\boldsymbol{x}, \boldsymbol{h}} \exp(-E(\boldsymbol{x}, \boldsymbol{h}; \boldsymbol{\theta}))$$

- Maximizing probability corresponds to minimizing the energy.

Energy Based Models

- A probability distribution (mixing observed and latent variables) via an energy function $E(\boldsymbol{x}, \boldsymbol{h}; \boldsymbol{\theta})$:

$$\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{h}) = \frac{\exp(-E(\boldsymbol{x}, \boldsymbol{h}; \boldsymbol{\theta}))}{Z(\boldsymbol{\theta})}, \quad \text{where } Z(\boldsymbol{\theta}) = \sum_{\boldsymbol{x}, \boldsymbol{h}} \exp(-E(\boldsymbol{x}, \boldsymbol{h}; \boldsymbol{\theta}))$$

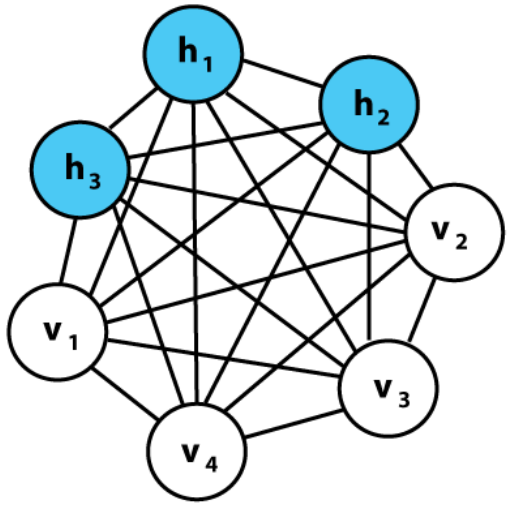
- Maximizing probability corresponds to minimizing the energy.

- Challenges:
  - Computing the partition function $Z(\boldsymbol{\theta})$
  - Computing the evidence $\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{\boldsymbol{h}} \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{h})$
  - Computing the posterior $\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h} \mid \boldsymbol{x}) = \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{h}) / \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x})$
  - Sampling from $\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{h})$ or from $\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x})$

Boltzmann Machine (Ackley et al., 1985)

- Energy-based model over binary vectors

- Some variables are observed ($\boldsymbol{v}$), others are latent/hidden ($\boldsymbol{h}$)



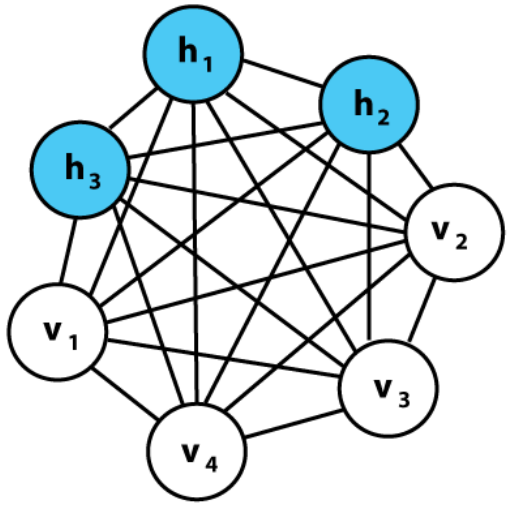
Boltzmann Machine (Ackley et al., 1985)

- Energy-based model over binary vectors

- Some variables are observed ($\boldsymbol{v}$), others are latent/hidden ($\boldsymbol{h}$)

- Probability distribution over $(\boldsymbol{v}, \boldsymbol{h}) \in \{0,1\}^{N+M}$:

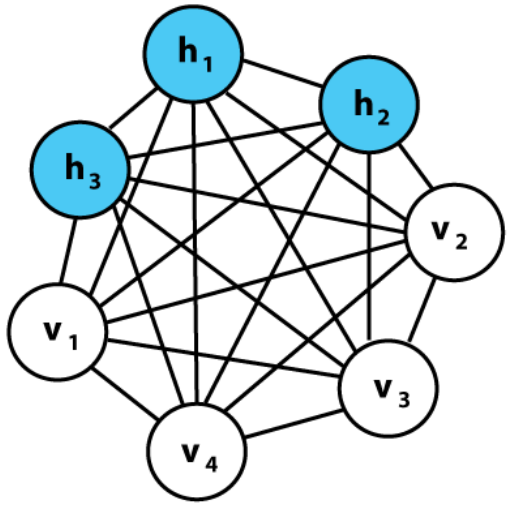
$$\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{v}, \boldsymbol{h}) = \frac{\exp(-E(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta}))}{Z(\boldsymbol{\theta})}$$



Boltzmann Machine (Ackley et al., 1985)

- Energy-based model over binary vectors

- Some variables are observed ($\boldsymbol{v}$), others are latent/hidden ($\boldsymbol{h}$)



- Probability distribution over $(\boldsymbol{v}, \boldsymbol{h}) \in \{0,1\}^{N+M}$:

$$\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{v}, \boldsymbol{h}) = \frac{\exp(-E(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta}))}{Z(\boldsymbol{\theta})}$$

- Energy function, with $\boldsymbol{\theta} = (\boldsymbol{R}, \boldsymbol{W}, \boldsymbol{S}, b, \boldsymbol{c})$,

$$E(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta}) = -\boldsymbol{v}^\top \boldsymbol{R} \boldsymbol{v} - \boldsymbol{v}^\top \boldsymbol{W} \boldsymbol{h} - \boldsymbol{h}^\top \boldsymbol{S} \boldsymbol{h} - b^\top \boldsymbol{v} - \boldsymbol{c}^\top \boldsymbol{h}$$

Boltzmann Machine

- Boltzmann machine (BM): universal approximator of probability mass functions over discrete variables (Le Roux and Bengio, 2008)

Boltzmann Machine

- Boltzmann machine (BM): universal approximator of probability mass functions over discrete variables (Le Roux and Bengio, 2008)
- Emulates the idea in Hebbian learning: "neurons that fire together wire together."

Boltzmann Machine

- Boltzmann machine (BM): universal approximator of probability mass functions over discrete variables (Le Roux and Bengio, 2008)
- Emulates the idea in Hebbian learning: "neurons that fire together wire together."
- However, in general,
  - Sampling is hard,
  - Inference is hard,
  - Learning is hard.

Boltzmann Machine

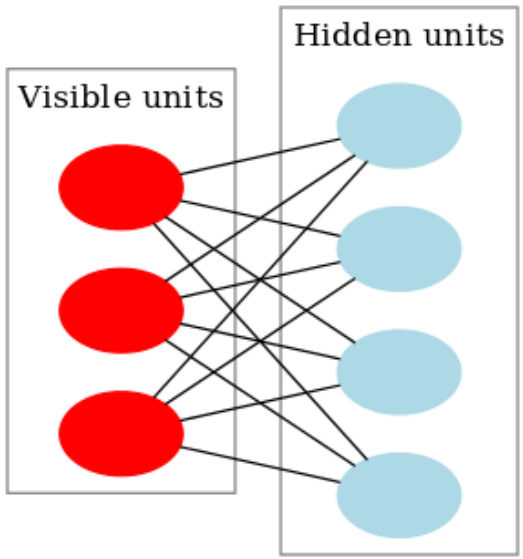
- Boltzmann machine (BM): universal approximator of probability mass functions over discrete variables (Le Roux and Bengio, 2008)
- Emulates the idea in Hebbian learning: "neurons that fire together wire together."
- However, in general,
  - Sampling is hard,
  - Inference is hard,
  - Learning is hard.
- In summary: learning a general BM is usually very challenging, so we typically use a particular version.

Boltzmann Machine

- Boltzmann machine (BM): universal approximator of probability mass functions over discrete variables (Le Roux and Bengio, 2008)
- Emulates the idea in Hebbian learning: "neurons that fire together wire together."
- However, in general,
  - Sampling is hard,
  - Inference is hard,
  - Learning is hard.
- In summary: learning a general BM is usually very challenging, so we typically use a particular version.
- Not covered here, but check Goodfellow et al. (2016, Chapter 18).

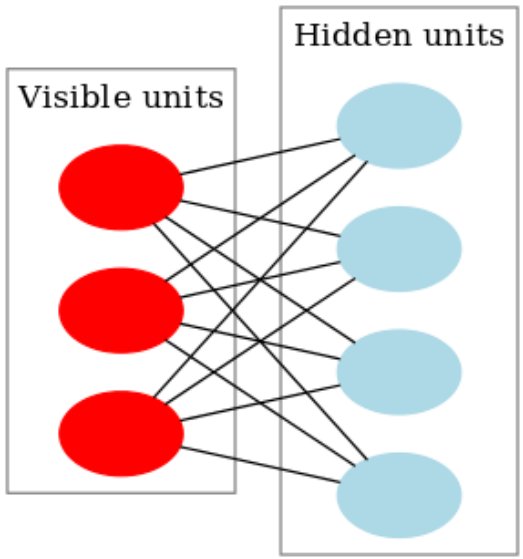
Particular Case: Restricted Boltzmann Machines

- Also called harmonium (Smolensky, 1986)
- A layer of observable variables
- A single layer of latent variables.



Particular Case: Restricted Boltzmann Machines

- Also called harmonium (Smolensky, 1986)
- A layer of observable variables
- A single layer of latent variables.

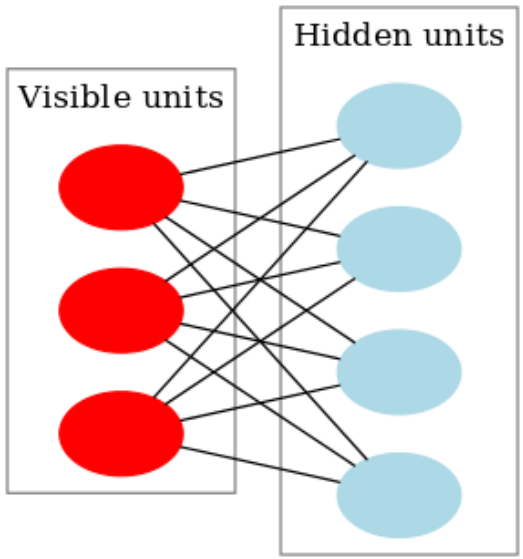


- Bipartite graph, no intra-layer connections: $\boldsymbol{R} = 0$; $\boldsymbol{S} = 0$.
- The energy function becomes:

$$E(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta}) = -\boldsymbol{v}^\top \boldsymbol{W}\boldsymbol{h} - \boldsymbol{b}^\top \boldsymbol{v} - \boldsymbol{c}^\top \boldsymbol{h}$$

Particular Case: Restricted Boltzmann Machines

- Also called harmonium (Smolensky, 1986)
- A layer of observable variables
- A single layer of latent variables.



- Bipartite graph, no intra-layer connections: $\boldsymbol{R} = 0$; $\boldsymbol{S} = 0$.
- The energy function becomes:

$$E(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta}) = -\boldsymbol{v}^\top \boldsymbol{W}\boldsymbol{h} - \boldsymbol{b}^\top \boldsymbol{v} - \boldsymbol{c}^\top \boldsymbol{h}$$

- What is the advantage?

Restricted Boltzmann Machines

- Unfortunately, the partition function $Z(\boldsymbol{\theta})$ is still intractable

Restricted Boltzmann Machines

- Unfortunately, the partition function $Z(\boldsymbol{\theta})$ is still intractable
- ... but the conditionals $\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h} \mid \boldsymbol{v})$ and $\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{v} \mid \boldsymbol{h})$ are now tractable!
  - easy to compute!
  - easy to sample!
  - using Markov-Chain Monte Carlo (MCMC) with Gibbs sampling.

Restricted Boltzmann Machines

- Unfortunately, the partition function $Z(\boldsymbol{\theta})$ is still intractable
- ... but the conditionals $\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h} \mid \boldsymbol{v})$ and $\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{v} \mid \boldsymbol{h})$ are now tractable!
  - easy to compute!
  - easy to sample!
  - using Markov-Chain Monte Carlo (MCMC) with Gibbs sampling.
- Why are these easy? Conditional independence (next slide)

Restricted Boltzmann Machines

- Without intra-layer connections, $h_1, \ldots, h_N$ are conditionally independent given $\boldsymbol{v}$:

$$\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h} \mid \boldsymbol{v}) = \prod_{j=1}^{M} \mathbb{P}(h_j \mid \boldsymbol{v})$$

where

$$\mathbb{P}_{\boldsymbol{\theta}}(h_j = 1 \mid \boldsymbol{v}) \quad = \quad \sigma\big(c_j + (\boldsymbol{W}\boldsymbol{v})_j\big), \quad j = 1, \ldots, M.$$

Restricted Boltzmann Machines

- Without intra-layer connections, $h_1, \ldots, h_N$ are conditionally independent given $\boldsymbol{v}$:

$$\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h} \mid \boldsymbol{v}) = \prod_{j=1}^{M} \mathbb{P}(h_j \mid \boldsymbol{v})$$

where

$$\mathbb{P}_{\boldsymbol{\theta}}(h_j = 1 \mid \boldsymbol{v}) \quad = \quad \sigma\big(c_j + (\boldsymbol{W}\boldsymbol{v})_j\big), \quad j = 1, \ldots, M.$$

- Reciprocally for $\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{v} \mid \boldsymbol{h})$.

Restricted Boltzmann Machines

- Without intra-layer connections, $h_1, \ldots, h_N$ are conditionally independent given $\boldsymbol{v}$:

$$\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h} \mid \boldsymbol{v}) = \prod_{j=1}^{M} \mathbb{P}(h_j \mid \boldsymbol{v})$$

where

$$\mathbb{P}_{\boldsymbol{\theta}}(h_j = 1 \mid \boldsymbol{v}) \quad = \quad \sigma\big(c_j + (\boldsymbol{W}\boldsymbol{v})_j\big), \quad j = 1, \ldots, M.$$

- Reciprocally for $\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{v} \mid \boldsymbol{h})$.
- RBMs are relatively easy to train by approximating $Z(\boldsymbol{\theta})$ (see Goodfellow et al. (2016, Chapter 18)).

Restricted Boltzmann Machines

- Without intra-layer connections, $h_1, \ldots, h_N$ are conditionally independent given $\boldsymbol{v}$:

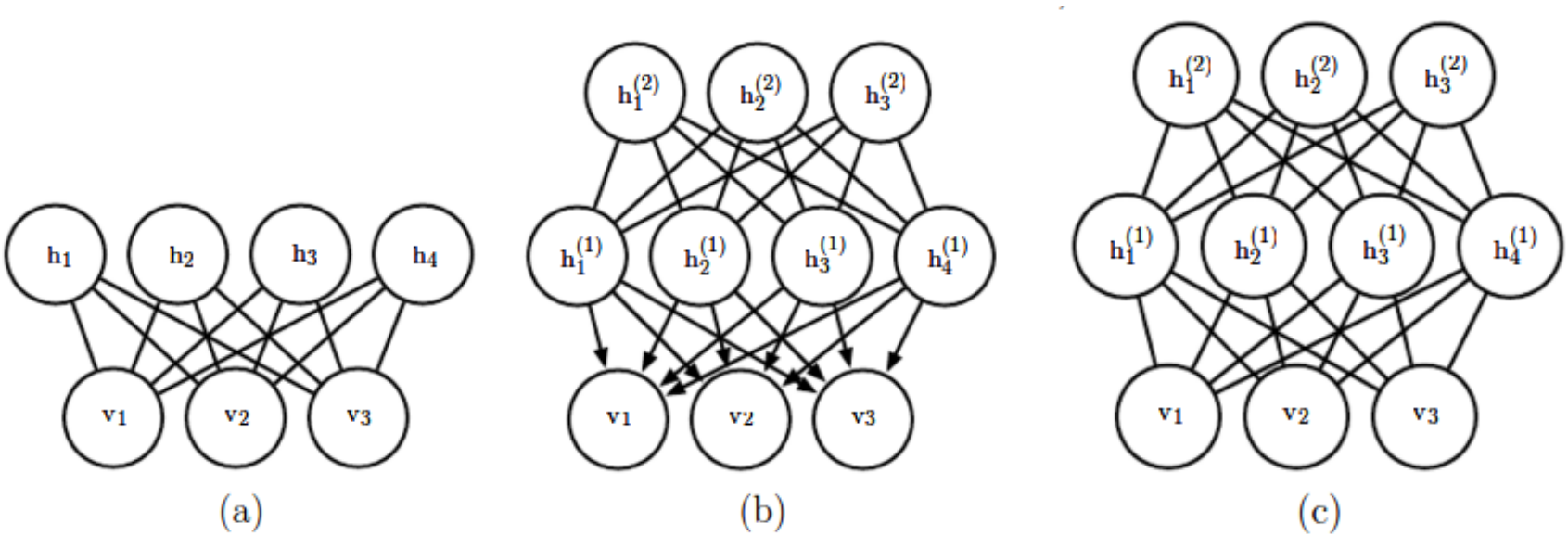
$$\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h} \mid \boldsymbol{v}) = \prod_{j=1}^{M} \mathbb{P}(h_j \mid \boldsymbol{v})$$

where

$$\mathbb{P}_{\boldsymbol{\theta}}(h_j = 1 \mid \boldsymbol{v}) \quad = \quad \sigma\big(c_j + (\boldsymbol{W}\boldsymbol{v})_j\big), \quad j = 1, \ldots, M.$$

- Reciprocally for $\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{v} \mid \boldsymbol{h})$.
- RBMs are relatively easy to train by approximating $Z(\boldsymbol{\theta})$ (see Goodfellow et al. (2016, Chapter 18)).
- RBMs may be stacked to form deeper models.

Some RBM's Friends



(Image from Goodfellow et al. (2016))

(a) Restricted Boltzmann machine (RBM)

(b) Deep belief network (DBN): hybrid directed/undirected GM with multiple latent layers

(c) Deep Boltzmann machine (DBM): undirected GM with several layers of latent variables.

Examples of Deep Generative Models

- Restricted Boltzmann Machines
- Deep Belief Networks
- Deep Boltzmann Machines
- Gaussian-Bernoulli RBMs
- Convolutional Boltzmann Machines
- Sigmoid Belief Nets
- Variational Auto-Encoders
- Generative Adversarial Networks
- Denoising diffusion models

Examples of Deep Generative Models

- Restricted Boltzmann Machines ✓
- Deep Belief Networks
- Deep Boltzmann Machines
- Gaussian-Bernoulli RBMs
- Convolutional Boltzmann Machines
- Sigmoid Belief Nets
- Variational Auto-Encoders
- Generative Adversarial Networks
- Denoising diffusion models

Outline

Differentiable Generator Networks

- Several recent models are based on differentiable generator networks.

Differentiable Generator Networks

- Several recent models are based on differentiable generator networks.
- This is a differentiable function $G(\boldsymbol{h}; \boldsymbol{\theta})$ that maps latent variables $\boldsymbol{h}$ into sample reconstructions $\boldsymbol{x}$ (or distributions $\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{h})$).

Differentiable Generator Networks

- Several recent models are based on differentiable generator networks.
- This is a differentiable function $G(\boldsymbol{h}; \boldsymbol{\theta})$ that maps latent variables $\boldsymbol{h}$ into sample reconstructions $\boldsymbol{x}$ (or distributions $\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{h})$).
- This idea underlies
  - Variational auto-encoders (VAE)
  - Generative adversarial networks (GAN)

Variational Auto-Encoders

- Many latent variable models have:
  - intractable evidence $\mathbb{P}(\boldsymbol{x})$
  - intractable posterior $\mathbb{P}(\boldsymbol{h} \mid \boldsymbol{x})$.

Variational Auto-Encoders

- Many latent variable models have:
  - intractable evidence $\mathbb{P}(\boldsymbol{x})$
  - intractable posterior $\mathbb{P}(\boldsymbol{h} \mid \boldsymbol{x})$.
- Variational inference is used to approximate these quantities.

Variational Auto-Encoders

- Many latent variable models have:
  - intractable evidence $\mathbb{P}(\boldsymbol{x})$
  - intractable posterior $\mathbb{P}(\boldsymbol{h} \mid \boldsymbol{x})$.
- Variational inference is used to approximate these quantities.
- Widely used in Bayesian inference, topic models, etc...

Variational Auto-Encoders

- Many latent variable models have:
  - intractable evidence $\mathbb{P}(\boldsymbol{x})$
  - intractable posterior $\mathbb{P}(\boldsymbol{h} \mid \boldsymbol{x})$.
- Variational inference is used to approximate these quantities.
- Widely used in Bayesian inference, topic models, etc...
- Auto-encoders: effective to learn data representations or codes, i.e.,

$$\boldsymbol{x} \longrightarrow \boldsymbol{h} \longrightarrow \hat{\boldsymbol{x}}$$

Variational Auto-Encoders

- Many latent variable models have:
  - intractable evidence $\mathbb{P}(\boldsymbol{x})$
  - intractable posterior $\mathbb{P}(\boldsymbol{h} \mid \boldsymbol{x})$.
- Variational inference is used to approximate these quantities.
- Widely used in Bayesian inference, topic models, etc...
- Auto-encoders: effective to learn data representations or codes, i.e.,

$$\boldsymbol{x} \longrightarrow \boldsymbol{h} \longrightarrow \hat{\boldsymbol{x}}$$

- Key idea: combine auto-encoders with variational inference.

Assumptions

- We assume that:
  - prior $\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h})$ is tractable (e.g.. standard Gaussian, Bernoulli, ...)

Assumptions

- We assume that:
  - prior $\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h})$ is tractable (e.g.. standard Gaussian, Bernoulli, ...)
  - conditional $\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{h})$ is tractable (e.g., feed-forward NN or an RNN).

Assumptions

- We assume that:
  - prior $\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h})$ is tractable (e.g.. standard Gaussian, Bernoulli, ...)
  - conditional $\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{h})$ is tractable (e.g., feed-forward NN or an RNN).
- However,
  - evidence $\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x})$ (i.e. marginalizing out $\boldsymbol{h}$) is still intractable
  - posterior $\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h} \mid \boldsymbol{x})$ is still intractable.

Assumptions

- We assume that:
  - prior $\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h})$ is tractable (e.g.. standard Gaussian, Bernoulli, ...)
  - conditional $\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{h})$ is tractable (e.g., feed-forward NN or an RNN).
- However,
  - evidence $\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x})$ (i.e. marginalizing out $\boldsymbol{h}$) is still intractable
  - posterior $\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h} \mid \boldsymbol{x})$ is still intractable.
- Next: using variational inference to approximate these computations

Outline

Recap: Shannon's Entropy

- Let $\mathbb{P}$ be a distribution over $\mathcal{X}$. The entropy of $\mathbb{P}$ is

$$H(\mathbb{P}) = -\sum_{\boldsymbol{x} \in \mathcal{X}} \mathbb{P}(\boldsymbol{x}) \log \mathbb{P}(\boldsymbol{x})$$

Recap: Shannon's Entropy

- Let $\mathbb{P}$ be a distribution over $\mathcal{X}$. The entropy of $\mathbb{P}$ is

$$H(\mathbb{P}) = -\sum_{\boldsymbol{x}\in\mathcal{X}} \mathbb{P}(\boldsymbol{x}) \log \mathbb{P}(\boldsymbol{x}) = \mathbb{E}_{\mathbb{P}(\boldsymbol{x})}\left[-\log \mathbb{P}(\boldsymbol{x})\right]$$

Recap: Shannon's Entropy

- Let $\mathbb{P}$ be a distribution over $\mathcal{X}$. The entropy of $\mathbb{P}$ is

$$H(\mathbb{P}) = -\sum_{\boldsymbol{x}\in\mathcal{X}} \mathbb{P}(\boldsymbol{x}) \log \mathbb{P}(\boldsymbol{x}) = \mathbb{E}_{\mathbb{P}(\boldsymbol{x})}\big[-\log \mathbb{P}(\boldsymbol{x})\big]$$

- Always non-negative: $H(\mathbb{P}) \geq 0$
- $H(\mathbb{P}) = 0$ iff $\mathbb{P}(\boldsymbol{x}) = 1$, for some $\boldsymbol{x}$ and $\mathbb{P}(\boldsymbol{x}') = 0$, for any $\boldsymbol{x}' \neq \boldsymbol{x}$.

Recap: Shannon's Entropy

- Let $\mathbb{P}$ be a distribution over $\mathcal{X}$. The entropy of $\mathbb{P}$ is

$$H(\mathbb{P}) = -\sum_{\boldsymbol{x}\in\mathcal{X}} \mathbb{P}(\boldsymbol{x}) \log \mathbb{P}(\boldsymbol{x}) = \mathbb{E}_{\mathbb{P}(\boldsymbol{x})}\big[-\log \mathbb{P}(\boldsymbol{x})\big]$$

- Always non-negative: $H(\mathbb{P}) \geq 0$
- $H(\mathbb{P}) = 0$ iff $\mathbb{P}(\boldsymbol{x}) = 1$, for some $\boldsymbol{x}$ and $\mathbb{P}(\boldsymbol{x}') = 0$, for any $\boldsymbol{x}' \neq \boldsymbol{x}$.
- Upper bounded: $H(\mathbb{P}) \leq \log |\mathcal{X}|$
- $H(\mathbb{P}) = \log |\mathcal{X}|$ iff $\mathbb{P}(\boldsymbol{x}) = 1/|\mathcal{X}|$ (uniform distribution)

Recap: Shannon's Entropy

- Let $\mathbb{P}$ be a distribution over $\mathcal{X}$. The entropy of $\mathbb{P}$ is

$$H(\mathbb{P}) = -\sum_{\boldsymbol{x}\in\mathcal{X}} \mathbb{P}(\boldsymbol{x}) \log \mathbb{P}(\boldsymbol{x}) = \mathbb{E}_{\mathbb{P}(\boldsymbol{x})}\big[-\log \mathbb{P}(\boldsymbol{x})\big]$$

- Always non-negative: $H(\mathbb{P}) \geq 0$
- $H(\mathbb{P}) = 0$ iff $\mathbb{P}(\boldsymbol{x}) = 1$, for some $\boldsymbol{x}$ and $\mathbb{P}(\boldsymbol{x}') = 0$, for any $\boldsymbol{x}' \neq \boldsymbol{x}$.
- Upper bounded: $H(\mathbb{P}) \leq \log |\mathcal{X}|$
- $H(\mathbb{P}) = \log |\mathcal{X}|$ iff $\mathbb{P}(\boldsymbol{x}) = 1/|\mathcal{X}|$ (uniform distribution)
- Intuition: $H(\mathbb{P})$ measures how close to uniform the distribution is

Recap: Shannon's Entropy

- Let $\mathbb{P}$ be a distribution over $\mathcal{X}$. The entropy of $\mathbb{P}$ is

$$H(\mathbb{P}) = -\sum_{\boldsymbol{x}\in\mathcal{X}} \mathbb{P}(\boldsymbol{x}) \log \mathbb{P}(\boldsymbol{x}) = \mathbb{E}_{\mathbb{P}(\boldsymbol{x})}\big[-\log \mathbb{P}(\boldsymbol{x})\big]$$

- Always non-negative: $H(\mathbb{P}) \geq 0$
- $H(\mathbb{P}) = 0$ iff $\mathbb{P}(\boldsymbol{x}) = 1$, for some $\boldsymbol{x}$ and $\mathbb{P}(\boldsymbol{x}') = 0$, for any $\boldsymbol{x}' \neq \boldsymbol{x}$.
- Upper bounded: $H(\mathbb{P}) \leq \log |\mathcal{X}|$
- $H(\mathbb{P}) = \log |\mathcal{X}|$ iff $\mathbb{P}(\boldsymbol{x}) = 1/|\mathcal{X}|$ (uniform distribution)
- Intuition: $H(\mathbb{P})$ measures how close to uniform the distribution is
- Coding perspective: expected number of bits (using $\log_2$) to optimally encode $\boldsymbol{x} \sim \mathbb{P}(\boldsymbol{x})$

Recap: Kullback-Leibler (KL) Divergence

- Let $\mathbb{P}$ and $\mathbb{Q}$ be two distributions over $\mathcal{X}$.

$$KL(\mathbb{P}\|\mathbb{Q}) = \sum_{\boldsymbol{x}} \mathbb{P}(\boldsymbol{x}) \log \frac{\mathbb{P}(\boldsymbol{x})}{\mathbb{Q}(\boldsymbol{x})}$$

Recap: Kullback-Leibler (KL) Divergence

- Let $\mathbb{P}$ and $\mathbb{Q}$ be two distributions over $\mathcal{X}$.

$$\begin{aligned} KL(\mathbb{P}\|\mathbb{Q}) &= \sum_{\boldsymbol{x}} \mathbb{P}(\boldsymbol{x}) \log \frac{\mathbb{P}(\boldsymbol{x})}{\mathbb{Q}(\boldsymbol{x})} \\ &= \mathbb{E}_{\mathbb{P}(\boldsymbol{x})}\big[-\log \mathbb{Q}(\boldsymbol{x})\big] + \mathbb{E}_{\mathbb{P}(\boldsymbol{x})}\big[\log \mathbb{P}(\boldsymbol{x})\big] \end{aligned}$$

Recap: Kullback-Leibler (KL) Divergence

- Let $\mathbb{P}$ and $\mathbb{Q}$ be two distributions over $\mathcal{X}$.

$$\begin{aligned} KL(\mathbb{P}\|\mathbb{Q}) &= \sum_{\boldsymbol{x}} \mathbb{P}(\boldsymbol{x}) \log \frac{\mathbb{P}(\boldsymbol{x})}{\mathbb{Q}(\boldsymbol{x})} \\ &= \mathbb{E}_{\mathbb{P}(\boldsymbol{x})}\big[-\log \mathbb{Q}(\boldsymbol{x})\big] + \mathbb{E}_{\mathbb{P}(\boldsymbol{x})}\big[\log \mathbb{P}(\boldsymbol{x})\big] \\ &= \mathbb{E}_{\mathbb{P}(\boldsymbol{x})}\big[-\log \mathbb{Q}(\boldsymbol{x})\big] - H(\mathbb{P}) \end{aligned}$$

Recap: Kullback-Leibler (KL) Divergence

- Let $\mathbb{P}$ and $\mathbb{Q}$ be two distributions over $\mathcal{X}$.

$$\begin{aligned} KL(\mathbb{P}\|\mathbb{Q}) &= \sum_{\boldsymbol{x}} \mathbb{P}(\boldsymbol{x}) \log \frac{\mathbb{P}(\boldsymbol{x})}{\mathbb{Q}(\boldsymbol{x})} \\ &= \mathbb{E}_{\mathbb{P}(\boldsymbol{x})}\big[-\log \mathbb{Q}(\boldsymbol{x})\big] + \mathbb{E}_{\mathbb{P}(\boldsymbol{x})}\big[\log \mathbb{P}(\boldsymbol{x})\big] \\ &= \mathbb{E}_{\mathbb{P}(\boldsymbol{x})}\big[-\log \mathbb{Q}(\boldsymbol{x})\big] - H(\mathbb{P}) \end{aligned}$$

- Always non-negative: $KL(\mathbb{P}\|\mathbb{Q}) \geq 0$

Recap: Kullback-Leibler (KL) Divergence

- Let $\mathbb{P}$ and $\mathbb{Q}$ be two distributions over $\mathcal{X}$.

$$\begin{aligned} KL(\mathbb{P}\|\mathbb{Q}) &= \sum_{\boldsymbol{x}} \mathbb{P}(\boldsymbol{x}) \log \frac{\mathbb{P}(\boldsymbol{x})}{\mathbb{Q}(\boldsymbol{x})} \\ &= \mathbb{E}_{\mathbb{P}(\boldsymbol{x})}\big[-\log \mathbb{Q}(\boldsymbol{x})\big] + \mathbb{E}_{\mathbb{P}(\boldsymbol{x})}\big[\log \mathbb{P}(\boldsymbol{x})\big] \\ &= \mathbb{E}_{\mathbb{P}(\boldsymbol{x})}\big[-\log \mathbb{Q}(\boldsymbol{x})\big] - H(\mathbb{P}) \end{aligned}$$

- Always non-negative: $KL(\mathbb{P}\|\mathbb{Q}) \geq 0$
- $KL(\mathbb{P}\|\mathbb{Q}) = 0$ iff $\mathbb{P}(\boldsymbol{x}) = \mathbb{Q}(\boldsymbol{x})$, for all $\boldsymbol{x} \in \mathcal{X}$

Recap: Kullback-Leibler (KL) Divergence

- Let $\mathbb{P}$ and $\mathbb{Q}$ be two distributions over $\mathcal{X}$.

$$\begin{aligned} KL(\mathbb{P}\|\mathbb{Q}) &= \sum_{\boldsymbol{x}} \mathbb{P}(\boldsymbol{x}) \log \frac{\mathbb{P}(\boldsymbol{x})}{\mathbb{Q}(\boldsymbol{x})} \\ &= \mathbb{E}_{\mathbb{P}(\boldsymbol{x})}\big[-\log \mathbb{Q}(\boldsymbol{x})\big] + \mathbb{E}_{\mathbb{P}(\boldsymbol{x})}\big[\log \mathbb{P}(\boldsymbol{x})\big] \\ &= \mathbb{E}_{\mathbb{P}(\boldsymbol{x})}\big[-\log \mathbb{Q}(\boldsymbol{x})\big] - H(\mathbb{P}) \end{aligned}$$

- Always non-negative: $KL(\mathbb{P}\|\mathbb{Q}) \geq 0$
- $KL(\mathbb{P}\|\mathbb{Q}) = 0$ iff $\mathbb{P}(\boldsymbol{x}) = \mathbb{Q}(\boldsymbol{x})$, for all $\boldsymbol{x} \in \mathcal{X}$
- Not symmetric: in general, $KL(\mathbb{P}\|\mathbb{Q}) \neq KL(\mathbb{Q}\|\mathbb{P})$

Recap: Kullback-Leibler (KL) Divergence

- Let $\mathbb{P}$ and $\mathbb{Q}$ be two distributions over $\mathcal{X}$.

$$\begin{aligned} KL(\mathbb{P}\|\mathbb{Q}) &= \sum_{\boldsymbol{x}} \mathbb{P}(\boldsymbol{x}) \log \frac{\mathbb{P}(\boldsymbol{x})}{\mathbb{Q}(\boldsymbol{x})} \\ &= \mathbb{E}_{\mathbb{P}(\boldsymbol{x})}\big[-\log \mathbb{Q}(\boldsymbol{x})\big] + \mathbb{E}_{\mathbb{P}(\boldsymbol{x})}\big[\log \mathbb{P}(\boldsymbol{x})\big] \\ &= \mathbb{E}_{\mathbb{P}(\boldsymbol{x})}\big[-\log \mathbb{Q}(\boldsymbol{x})\big] - H(\mathbb{P}) \end{aligned}$$

- Always non-negative: $KL(\mathbb{P}\|\mathbb{Q}) \geq 0$
- $KL(\mathbb{P}\|\mathbb{Q}) = 0$ iff $\mathbb{P}(\boldsymbol{x}) = \mathbb{Q}(\boldsymbol{x})$, for all $\boldsymbol{x} \in \mathcal{X}$
- Not symmetric: in general, $KL(\mathbb{P}\|\mathbb{Q}) \neq KL(\mathbb{Q}\|\mathbb{P})$
- Intuition: $KL(\mathbb{P}\|\mathbb{Q})$ measures how different $\mathbb{Q}$ is from $\mathbb{P}$

Recap: Kullback-Leibler (KL) Divergence

- Let $\mathbb{P}$ and $\mathbb{Q}$ be two distributions over $\mathcal{X}$.

$$\begin{aligned} KL(\mathbb{P}\|\mathbb{Q}) &= \sum_{\boldsymbol{x}} \mathbb{P}(\boldsymbol{x}) \log \frac{\mathbb{P}(\boldsymbol{x})}{\mathbb{Q}(\boldsymbol{x})} \\ &= \mathbb{E}_{\mathbb{P}(\boldsymbol{x})}\big[-\log \mathbb{Q}(\boldsymbol{x})\big] + \mathbb{E}_{\mathbb{P}(\boldsymbol{x})}\big[\log \mathbb{P}(\boldsymbol{x})\big] \\ &= \mathbb{E}_{\mathbb{P}(\boldsymbol{x})}\big[-\log \mathbb{Q}(\boldsymbol{x})\big] - H(\mathbb{P}) \end{aligned}$$

- Always non-negative: $KL(\mathbb{P}\|\mathbb{Q}) \geq 0$
- $KL(\mathbb{P}\|\mathbb{Q}) = 0$ iff $\mathbb{P}(\boldsymbol{x}) = \mathbb{Q}(\boldsymbol{x})$, for all $\boldsymbol{x} \in \mathcal{X}$
- Not symmetric: in general, $KL(\mathbb{P}\|\mathbb{Q}) \neq KL(\mathbb{Q}\|\mathbb{P})$
- Intuition: $KL(\mathbb{P}\|\mathbb{Q})$ measures how different $\mathbb{Q}$ is from $\mathbb{P}$
- Coding perspective: expected number of extra bits needed to encode $\boldsymbol{x} \sim \mathbb{P}(\boldsymbol{x})$ using a code that is optimal for $\mathbb{Q}(\boldsymbol{x})$.

Recap: Entropy and KL Divergence in the Continuous Case

- Let $\mathbb{P}$ be a probability density over $\mathcal{X}$. The differential entropy of $\mathbb{P}$ is

$$H(\mathbb{P}) = -\int_{\mathcal{X}} \mathbb{P}(\boldsymbol{x}) \log \mathbb{P}(\boldsymbol{x})\, d\boldsymbol{x}$$

...no longer guaranteed to be non-negative.

Recap: Entropy and KL Divergence in the Continuous Case

- Let $\mathbb{P}$ be a probability density over $\mathcal{X}$. The differential entropy of $\mathbb{P}$ is

$$H(\mathbb{P}) = -\int_{\mathcal{X}} \mathbb{P}(\boldsymbol{x}) \log \mathbb{P}(\boldsymbol{x})\, d\boldsymbol{x} = \mathbb{E}_{\mathbb{P}(\boldsymbol{x})}\big[-\log \mathbb{P}(\boldsymbol{x})\big]$$

...no longer guaranteed to be non-negative.

Recap: Entropy and KL Divergence in the Continuous Case

- Let $\mathbb{P}$ be a probability density over $\mathcal{X}$. The differential entropy of $\mathbb{P}$ is

$$H(\mathbb{P}) = -\int_{\mathcal{X}} \mathbb{P}(\boldsymbol{x}) \log \mathbb{P}(\boldsymbol{x})\, d\boldsymbol{x} = \mathbb{E}_{\mathbb{P}(\boldsymbol{x})}\big[-\log \mathbb{P}(\boldsymbol{x})\big]$$

  ...no longer guaranteed to be non-negative.

- Let $\mathbb{P}$ and $\mathbb{Q}$ be two probability densities over $\mathcal{X}$.

$$\begin{aligned} KL(\mathbb{P}\|\mathbb{Q}) &= \int_{\mathcal{X}} \mathbb{P}(\boldsymbol{x}) \log \frac{\mathbb{P}(\boldsymbol{x})}{\mathbb{Q}(\boldsymbol{x})} d\boldsymbol{x} \\ &= \mathbb{E}_{\mathbb{P}(\boldsymbol{x})}\big[-\log \mathbb{Q}(\boldsymbol{x})\big] + \mathbb{E}_{\mathbb{P}(\boldsymbol{x})}\big[\log \mathbb{P}(\boldsymbol{x})\big] \\ &= \mathbb{E}_{\mathbb{P}(\boldsymbol{x})}\big[-\log \mathbb{Q}(\boldsymbol{x})\big] - H(\mathbb{P}) \end{aligned}$$

Recap: Entropy and KL Divergence in the Continuous Case

- Let $\mathbb{P}$ be a probability density over $\mathcal{X}$. The differential entropy of $\mathbb{P}$ is

$$H(\mathbb{P}) = -\int_{\mathcal{X}} \mathbb{P}(\boldsymbol{x}) \log \mathbb{P}(\boldsymbol{x})\, d\boldsymbol{x} = \mathbb{E}_{\mathbb{P}(\boldsymbol{x})}\big[-\log \mathbb{P}(\boldsymbol{x})\big]$$

  ...no longer guaranteed to be non-negative.

- Let $\mathbb{P}$ and $\mathbb{Q}$ be two probability densities over $\mathcal{X}$.

$$\begin{aligned} KL(\mathbb{P}\|\mathbb{Q}) &= \int_{\mathcal{X}} \mathbb{P}(\boldsymbol{x}) \log \frac{\mathbb{P}(\boldsymbol{x})}{\mathbb{Q}(\boldsymbol{x})} d\boldsymbol{x} \\ &= \mathbb{E}_{\mathbb{P}(\boldsymbol{x})}\big[-\log \mathbb{Q}(\boldsymbol{x})\big] + \mathbb{E}_{\mathbb{P}(\boldsymbol{x})}\big[\log \mathbb{P}(\boldsymbol{x})\big] \\ &= \mathbb{E}_{\mathbb{P}(\boldsymbol{x})}\big[-\log \mathbb{Q}(\boldsymbol{x})\big] - H(\mathbb{P}) \end{aligned}$$

- Also, always non-negative: $KL(\mathbb{P}\|\mathbb{Q}) \geq 0$

Recap: Entropy and KL Divergence in the Continuous Case

- Let $\mathbb{P}$ be a probability density over $\mathcal{X}$. The differential entropy of $\mathbb{P}$ is

$$H(\mathbb{P}) = -\int_{\mathcal{X}} \mathbb{P}(\boldsymbol{x}) \log \mathbb{P}(\boldsymbol{x})\, d\boldsymbol{x} = \mathbb{E}_{\mathbb{P}(\boldsymbol{x})}\big[-\log \mathbb{P}(\boldsymbol{x})\big]$$

  ...no longer guaranteed to be non-negative.

- Let $\mathbb{P}$ and $\mathbb{Q}$ be two probability densities over $\mathcal{X}$.

$$\begin{aligned} KL(\mathbb{P}\|\mathbb{Q}) &= \int_{\mathcal{X}} \mathbb{P}(\boldsymbol{x}) \log \frac{\mathbb{P}(\boldsymbol{x})}{\mathbb{Q}(\boldsymbol{x})} d\boldsymbol{x} \\ &= \mathbb{E}_{\mathbb{P}(\boldsymbol{x})}\big[-\log \mathbb{Q}(\boldsymbol{x})\big] + \mathbb{E}_{\mathbb{P}(\boldsymbol{x})}\big[\log \mathbb{P}(\boldsymbol{x})\big] \\ &= \mathbb{E}_{\mathbb{P}(\boldsymbol{x})}\big[-\log \mathbb{Q}(\boldsymbol{x})\big] - H(\mathbb{P}) \end{aligned}$$

- Also, always non-negative: $KL(\mathbb{P}\|\mathbb{Q}) \geq 0$
- $KL(\mathbb{P}\|\mathbb{Q}) = 0$ iff $\mathbb{P}(\boldsymbol{x}) = \mathbb{Q}(\boldsymbol{x})$, almost everywhere

Recap: Entropy and KL Divergence in the Continuous Case

- Let $\mathbb{P}$ be a probability density over $\mathcal{X}$. The differential entropy of $\mathbb{P}$ is

$$H(\mathbb{P}) = -\int_{\mathcal{X}} \mathbb{P}(\boldsymbol{x}) \log \mathbb{P}(\boldsymbol{x})\, d\boldsymbol{x} = \mathbb{E}_{\mathbb{P}(\boldsymbol{x})}\big[-\log \mathbb{P}(\boldsymbol{x})\big]$$

  ...no longer guaranteed to be non-negative.

- Let $\mathbb{P}$ and $\mathbb{Q}$ be two probability densities over $\mathcal{X}$.

$$\begin{aligned} KL(\mathbb{P}\|\mathbb{Q}) &= \int_{\mathcal{X}} \mathbb{P}(\boldsymbol{x}) \log \frac{\mathbb{P}(\boldsymbol{x})}{\mathbb{Q}(\boldsymbol{x})} d\boldsymbol{x} \\ &= \mathbb{E}_{\mathbb{P}(\boldsymbol{x})}\big[-\log \mathbb{Q}(\boldsymbol{x})\big] + \mathbb{E}_{\mathbb{P}(\boldsymbol{x})}\big[\log \mathbb{P}(\boldsymbol{x})\big] \\ &= \mathbb{E}_{\mathbb{P}(\boldsymbol{x})}\big[-\log \mathbb{Q}(\boldsymbol{x})\big] - H(\mathbb{P}) \end{aligned}$$

- Also, always non-negative: $KL(\mathbb{P}\|\mathbb{Q}) \geq 0$
- $KL(\mathbb{P}\|\mathbb{Q}) = 0$ iff $\mathbb{P}(\boldsymbol{x}) = \mathbb{Q}(\boldsymbol{x})$, almost everywhere
- Intuition: $KL(\mathbb{P}\|\mathbb{Q})$ still measure how different $\mathbb{Q}$ is from $\mathbb{P}$

Evidence Lower Bound (ELBO)

- ELBO is a central concept in variational inference.
- True posterior and evidence: $\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h} \mid \boldsymbol{x})$ and $\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x})$

Evidence Lower Bound (ELBO)

- ELBO is a central concept in variational inference.
- True posterior and evidence: $\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h} \mid \boldsymbol{x})$ and $\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x})$
- For any distribution $\mathbb{Q}(\boldsymbol{h})$,

$$\begin{aligned} 0 &\geq -KL(\mathbb{Q}(\boldsymbol{h}) \| \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h} \mid \boldsymbol{x})) \\ &= \mathbb{E}_{\mathbb{Q}(\boldsymbol{h})}[\log \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h} \mid \boldsymbol{x})] - \overbrace{\mathbb{E}_{\mathbb{Q}(\boldsymbol{h})}[\log \mathbb{Q}(\boldsymbol{h})]}^{H(\mathbb{Q})} \end{aligned}$$

Evidence Lower Bound (ELBO)

- ELBO is a central concept in variational inference.
- True posterior and evidence: $\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h} \mid \boldsymbol{x})$ and $\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x})$
- For any distribution $\mathbb{Q}(\boldsymbol{h})$,

$$
\begin{aligned}
0 \quad &\geq \quad -KL(\mathbb{Q}(\boldsymbol{h}) \| \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h} \mid \boldsymbol{x})) \\
&= \quad \mathbb{E}_{\mathbb{Q}(\boldsymbol{h})}[\log \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h} \mid \boldsymbol{x})] - \overbrace{\mathbb{E}_{\mathbb{Q}(\boldsymbol{h})}[\log \mathbb{Q}(\boldsymbol{h})]}^{H(\mathbb{Q})} \\
&\overset{(a)}{=} \quad \underbrace{\mathbb{E}_{\mathbb{Q}(\boldsymbol{h})}[\log \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{h})] - \mathbb{E}_{\mathbb{Q}(\boldsymbol{h})}[\log \mathbb{Q}(\boldsymbol{h})]}_{\text{ELBO}(\mathbb{Q})} - \log \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x}).
\end{aligned}
$$

where (a) uses Bayes law: $\log \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h} \mid \boldsymbol{x}) = \log \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{h}) - \log \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x})$

Evidence Lower Bound (ELBO)

- ELBO is a central concept in variational inference.
- True posterior and evidence: $\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h} \mid \boldsymbol{x})$ and $\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x})$
- For any distribution $\mathbb{Q}(\boldsymbol{h})$,

$$
\begin{aligned}
0 \quad &\geq \quad -KL(\mathbb{Q}(\boldsymbol{h}) \| \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h} \mid \boldsymbol{x})) \\
&= \quad \mathbb{E}_{\mathbb{Q}(\boldsymbol{h})}[\log \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h} \mid \boldsymbol{x})] - \overbrace{\mathbb{E}_{\mathbb{Q}(\boldsymbol{h})}[\log \mathbb{Q}(\boldsymbol{h})]}^{H(\mathbb{Q})} \\
&\overset{(a)}{=} \quad \underbrace{\mathbb{E}_{\mathbb{Q}(\boldsymbol{h})}[\log \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{h})] - \mathbb{E}_{\mathbb{Q}(\boldsymbol{h})}[\log \mathbb{Q}(\boldsymbol{h})]}_{\text{ELBO}(\mathbb{Q})} - \log \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x}).
\end{aligned}
$$

  where (a) uses Bayes law: $\log \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h} \mid \boldsymbol{x}) = \log \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{h}) - \log \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x})$

- Moving $\log \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x})$ to the l.h.s.,

$$\log \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x}) \geq \text{ELBO}(\mathbb{Q})$$

Variational Inference

- Evidence lower bound (ELBO):

$$\log \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x}) = \text{ELBO}(\mathbb{Q}) + KL(\mathbb{Q}(\boldsymbol{h}) \| \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h} \mid \boldsymbol{x})) \geq \text{ELBO}(\mathbb{Q}).$$

Variational Inference

- Evidence lower bound (ELBO):

$$\log \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x}) = \text{ELBO}(\mathbb{Q}) + KL(\mathbb{Q}(\boldsymbol{h}) \| \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h} \mid \boldsymbol{x})) \geq \text{ELBO}(\mathbb{Q}).$$

- Equality achieved for $\mathbb{Q}(\boldsymbol{h}) = \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h} \mid \boldsymbol{x})$, but intractable in general

Variational Inference

- Evidence lower bound (ELBO):

$$\log \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x}) = \text{ELBO}(\mathbb{Q}) + KL(\mathbb{Q}(\boldsymbol{h}) \| \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h} \mid \boldsymbol{x})) \geq \text{ELBO}(\mathbb{Q}).$$

- Equality achieved for $\mathbb{Q}(\boldsymbol{h}) = \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h} \mid \boldsymbol{x})$, but intractable in general
- Key idea:
  1. constrain $\mathbb{Q}(\boldsymbol{h})$ to a chosen tractable family;
  2. look for the $\mathbb{Q}(\boldsymbol{h})$ in this family that maximizes the ELBO.

Variational Inference

- Evidence lower bound (ELBO):

$$\log \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x}) = \text{ELBO}(\mathbb{Q}) + KL(\mathbb{Q}(\boldsymbol{h}) \| \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h} \mid \boldsymbol{x})) \geq \text{ELBO}(\mathbb{Q}).$$

- Equality achieved for $\mathbb{Q}(\boldsymbol{h}) = \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h} \mid \boldsymbol{x})$, but intractable in general
- Key idea:
  1. constrain $\mathbb{Q}(\boldsymbol{h})$ to a chosen tractable family;
  2. look for the $\mathbb{Q}(\boldsymbol{h})$ in this family that maximizes the ELBO.
- Since $\log \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x})$ fixed,

$$\text{maximizing ELBO}(\mathbb{Q}) \quad \Leftrightarrow \quad \text{minimizing} \quad KL(\mathbb{Q}(\boldsymbol{h}) \| \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h} \mid \boldsymbol{x}))$$

Variational Inference

- Evidence lower bound (ELBO):

$$\log \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x}) = \text{ELBO}(\mathbb{Q}) + KL(\mathbb{Q}(\boldsymbol{h}) \| \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h} \mid \boldsymbol{x})) \geq \text{ELBO}(\mathbb{Q}).$$

- Equality achieved for $\mathbb{Q}(\boldsymbol{h}) = \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h} \mid \boldsymbol{x})$, but intractable in general
- Key idea:
  1. constrain $\mathbb{Q}(\boldsymbol{h})$ to a chosen tractable family;
  2. look for the $\mathbb{Q}(\boldsymbol{h})$ in this family that maximizes the ELBO.
- Since $\log \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x})$ fixed,

$$\text{maximizing ELBO}(\mathbb{Q}) \quad \Leftrightarrow \quad \text{minimizing} \quad KL(\mathbb{Q}(\boldsymbol{h}) \| \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h} \mid \boldsymbol{x}))$$

- Old roots in calculus of variations (Euler, Lagrange, ...)

Evidence Lower Bound

- The ELBO can be written in different ways:

$$\begin{aligned}
\text{ELBO}(\mathbb{Q}) &= \mathbb{E}_{\mathbb{Q}(\boldsymbol{h})}[\log \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{h})] - \mathbb{E}_{\mathbb{Q}(\boldsymbol{h})}[\log \mathbb{Q}(\boldsymbol{h})] \\
&= \mathbb{E}_{\mathbb{Q}(\boldsymbol{h})}[\log \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{h})] - \mathbb{E}_{\mathbb{Q}(\boldsymbol{h})}\left[\log \frac{\mathbb{Q}(\boldsymbol{h})}{\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h})}\right] \\
&= \mathbb{E}_{\mathbb{Q}(\boldsymbol{h})}[\log \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{h})] - KL(\mathbb{Q}(\boldsymbol{h}) \| \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h})).
\end{aligned}$$

Evidence Lower Bound

- The ELBO can be written in different ways:

$$\begin{aligned}
\text{ELBO}(\mathbb{Q}) &= \mathbb{E}_{\mathbb{Q}(\boldsymbol{h})}[\log \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{h})] - \mathbb{E}_{\mathbb{Q}(\boldsymbol{h})}[\log \mathbb{Q}(\boldsymbol{h})] \\
&= \mathbb{E}_{\mathbb{Q}(\boldsymbol{h})}[\log \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{h})] - \mathbb{E}_{\mathbb{Q}(\boldsymbol{h})}\left[\log \frac{\mathbb{Q}(\boldsymbol{h})}{\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h})}\right] \\
&= \mathbb{E}_{\mathbb{Q}(\boldsymbol{h})}[\log \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{h})] - KL(\mathbb{Q}(\boldsymbol{h}) \| \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h})).
\end{aligned}$$

- Which values of $\boldsymbol{h}$ is $\mathbb{Q}(\boldsymbol{h})$ encouraged to place its mass on?
  - First term: expected likelihood encourages placing mass on latent variables $\boldsymbol{h}$ that explain the observed data $\boldsymbol{x}$.
  - Second term: negative KLD between $\mathbb{Q}(\boldsymbol{h})$ and the prior encourages staying close to the prior.

Evidence Lower Bound

- The ELBO can be written in different ways:

$$\begin{aligned}
\text{ELBO}(\mathbb{Q}) &= \mathbb{E}_{\mathbb{Q}(\boldsymbol{h})}[\log \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{h})] - \mathbb{E}_{\mathbb{Q}(\boldsymbol{h})}[\log \mathbb{Q}(\boldsymbol{h})] \\
&= \mathbb{E}_{\mathbb{Q}(\boldsymbol{h})}[\log \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{h})] - \mathbb{E}_{\mathbb{Q}(\boldsymbol{h})}\left[\log \frac{\mathbb{Q}(\boldsymbol{h})}{\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h})}\right] \\
&= \mathbb{E}_{\mathbb{Q}(\boldsymbol{h})}[\log \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{h})] - KL(\mathbb{Q}(\boldsymbol{h}) \| \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h})).
\end{aligned}$$

- Which values of $\boldsymbol{h}$ is $\mathbb{Q}(\boldsymbol{h})$ encouraged to place its mass on?
  - First term: expected likelihood encourages placing mass on latent variables $\boldsymbol{h}$ that explain the observed data $\boldsymbol{x}$.
  - Second term: negative KLD between $\mathbb{Q}(\boldsymbol{h})$ and the prior encourages staying close to the prior.
- ELBO mirrors the Bayesian trade-off between likelihood and prior.

Mean Field Approximation

- Which tractable family to use for $\mathbb{Q}(\boldsymbol{h})$?

Mean Field Approximation

- Which tractable family to use for $\mathbb{Q}(\boldsymbol{h})$?

- Mean field approximation (MFA):

$$\mathbb{Q}(\boldsymbol{h}) = \prod_i \mathbb{Q}(h_i).$$

  i.e., model the h_i are independent.

Mean Field Approximation

- Which tractable family to use for $\mathbb{Q}(\boldsymbol{h})$?

- Mean field approximation (MFA):

$$\mathbb{Q}(\boldsymbol{h}) = \prod_i \mathbb{Q}(h_i).$$

  i.e., model the h_i are independent.

- Structured mean field: imposes a graphical model on $\mathbb{Q}$ capturing (some) interactions among the h_i, but still tractable. (Wainwright and Jordan, 2008).

Amortized Variational Inference

- As seen so far, the variational distribution $\mathbb{Q}(\boldsymbol{h})$ has to be optimized for every example $\boldsymbol{x}$.

Amortized Variational Inference

- As seen so far, the variational distribution $\mathbb{Q}(\boldsymbol{h})$ has to be optimized for every example $\boldsymbol{x}$.

- This can be expensive: requires several gradient/coordinate ascent iterations per example.

Amortized Variational Inference

- As seen so far, the variational distribution $\mathbb{Q}(\boldsymbol{h})$ has to be optimized for every example $\boldsymbol{x}$.

- This can be expensive: requires several gradient/coordinate ascent iterations per example.

- Alternative: amortized variational inference!

- Key idea: instead of optimizing $\mathbb{Q}(\boldsymbol{h})$ for every example, use an encoder with shared parameters $\boldsymbol{\phi}$ and define $\mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h} \mid \boldsymbol{x})$.

  For each example:

  - make a forward pass on the encoder to obtain $\mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h} \mid \boldsymbol{x})$
  - backpropagate through the encoder to update $\boldsymbol{\phi}$.

Example: Multivariate Bernoulli with Continuous Latent Variables (Kingma and Welling, 2013)

- Prior: standard Gaussian $\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h}) = \mathcal{N}(\boldsymbol{h}; 0, \boldsymbol{I})$

Example: Multivariate Bernoulli with Continuous Latent Variables (Kingma and Welling, 2013)

- Prior: standard Gaussian $\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h}) = \mathcal{N}(\boldsymbol{h}; 0, \boldsymbol{I})$
- Conditional: multivariate Bernoulli

$$\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{h}) = \prod_{i=1}^{D} \sigma(f_i(\boldsymbol{h}; \boldsymbol{\theta}))^{x_i} (1 - \sigma(f_i(\boldsymbol{h}; \boldsymbol{\theta})))^{1-x_i},$$

where $\boldsymbol{f}(\boldsymbol{h}; \boldsymbol{\theta})$ is an MLP, with parameters $\boldsymbol{\theta}$ and input $\boldsymbol{h}$

Example: Multivariate Bernoulli with Continuous Latent Variables (Kingma and Welling, 2013)

- Prior: standard Gaussian $\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h}) = \mathcal{N}(\boldsymbol{h}; 0, \boldsymbol{I})$
- Conditional: multivariate Bernoulli

$$\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{h}) = \prod_{i=1}^{D} \sigma(f_i(\boldsymbol{h}; \boldsymbol{\theta}))^{x_i} (1 - \sigma(f_i(\boldsymbol{h}; \boldsymbol{\theta})))^{1-x_i},$$

  where $\boldsymbol{f}(\boldsymbol{h}; \boldsymbol{\theta})$ is an MLP, with parameters $\boldsymbol{\theta}$ and input $\boldsymbol{h}$

- True posterior $\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h} \mid \boldsymbol{x})$: intractable

Example: Multivariate Bernoulli with Continuous Latent Variables (Kingma and Welling, 2013)

- Prior: standard Gaussian $\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h}) = \mathcal{N}(\boldsymbol{h}; 0, \boldsymbol{I})$
- Conditional: multivariate Bernoulli

$$\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{h}) = \prod_{i=1}^{D} \sigma(f_i(\boldsymbol{h}; \boldsymbol{\theta}))^{x_i} (1 - \sigma(f_i(\boldsymbol{h}; \boldsymbol{\theta})))^{1-x_i},$$

  where $\boldsymbol{f}(\boldsymbol{h}; \boldsymbol{\theta})$ is an MLP, with parameters $\boldsymbol{\theta}$ and input $\boldsymbol{h}$

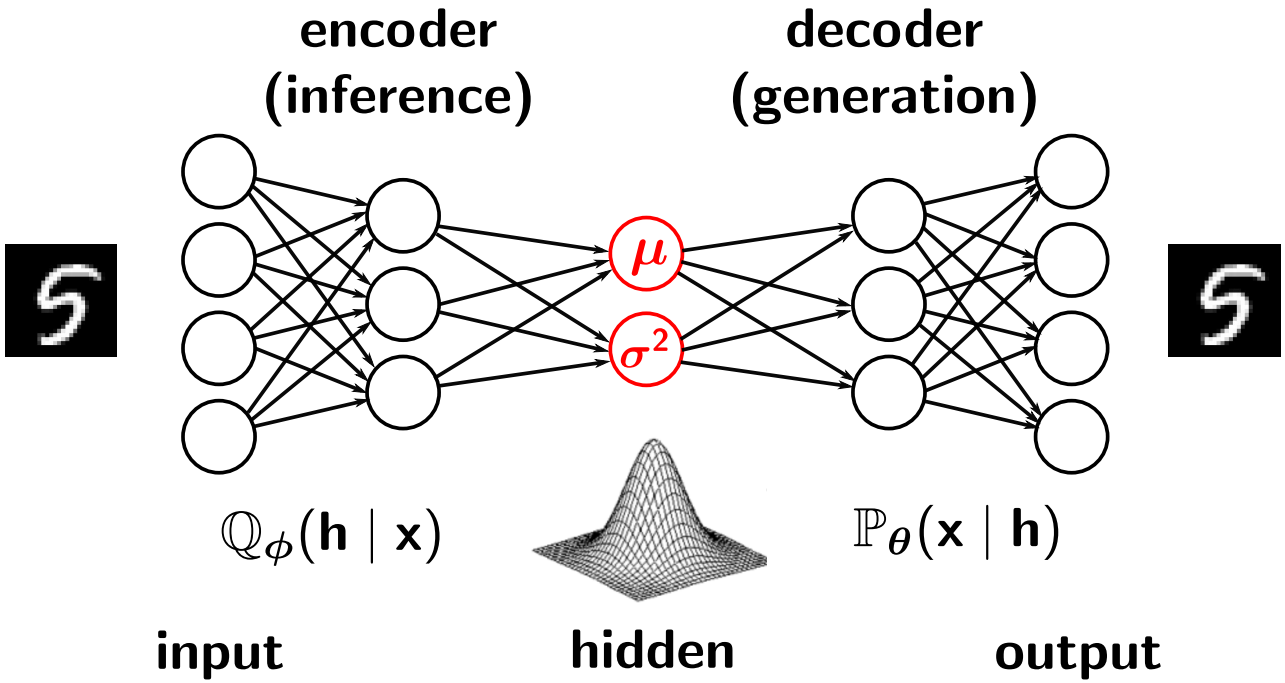
- True posterior $\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h} \mid \boldsymbol{x})$: intractable
- Approximate the posterior with a variational distribution

$$\mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h} \mid \boldsymbol{x}) = \mathcal{N}(\boldsymbol{h}; \boldsymbol{\mu}(\boldsymbol{x}; \boldsymbol{\phi}), \boldsymbol{\sigma}^2(\boldsymbol{x}; \boldsymbol{\phi}))$$

  where $\boldsymbol{\mu}(\boldsymbol{x}; \boldsymbol{\phi})$ and $\boldsymbol{\sigma}^2(\boldsymbol{x}; \boldsymbol{\phi})$ are (learned) MLPs

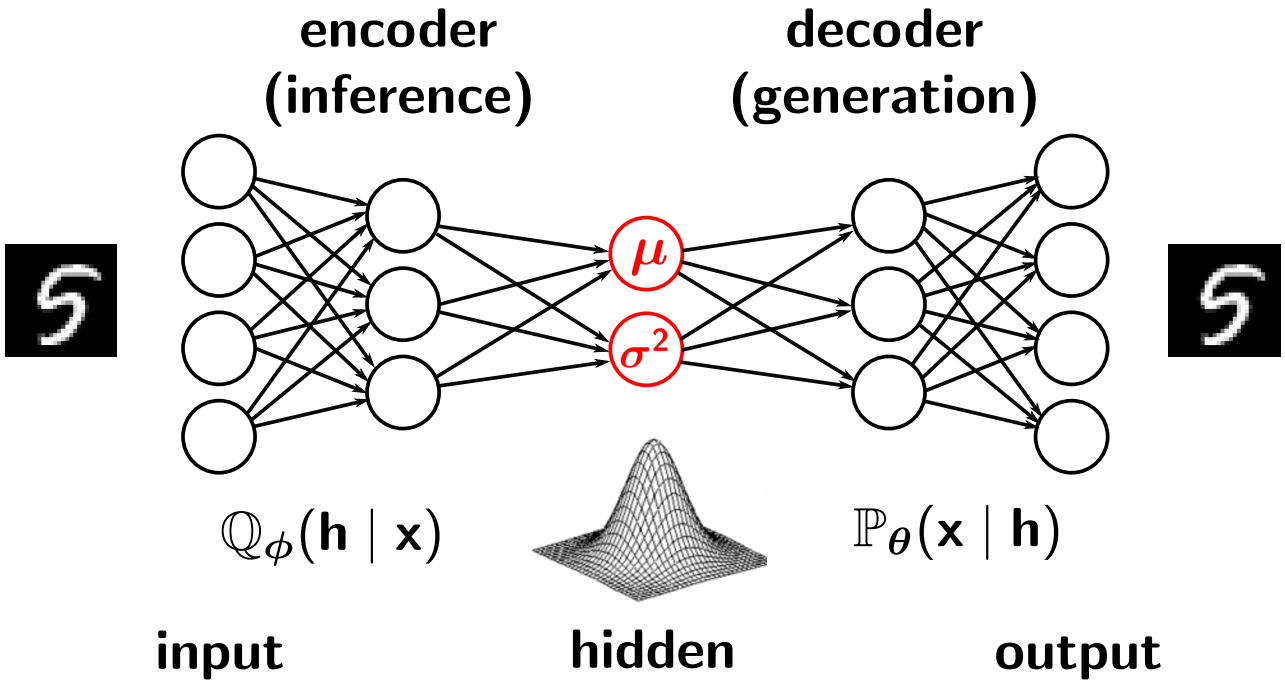
Example: Multivariate Bernoulli with Continuous Latent Variables (Kingma and Welling, 2013)

- This leads to variational auto-encoders:



Example: Multivariate Bernoulli with Continuous Latent Variables (Kingma and Welling, 2013)

- This leads to variational auto-encoders:



- ... we will come back to this!

Outline

Parameter Gradients

- Recall that:

$$\text{ELBO}(\boldsymbol{\phi}; \boldsymbol{\theta}) = \mathbb{E}_{\mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h}|\boldsymbol{x})}[\log \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{h}) - \log \mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h} \mid \boldsymbol{x})].$$

Parameter Gradients

- Recall that:

$$\text{ELBO}(\boldsymbol{\phi}; \boldsymbol{\theta}) = \mathbb{E}_{\mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h}|\boldsymbol{x})}[\log \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{h}) - \log \mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h} \mid \boldsymbol{x})].$$

- We need to compute gradients with respect to $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$.

Parameter Gradients

- Recall that:

$$\text{ELBO}(\boldsymbol{\phi}; \boldsymbol{\theta}) = \mathbb{E}_{\mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h}|\boldsymbol{x})}[\log \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{h}) - \log \mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h} \mid \boldsymbol{x})].$$

- We need to compute gradients with respect to $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$.

- Gradient w.r.t. $\boldsymbol{\theta}$, parameters of the generation (decoder) network:

$$\nabla_{\boldsymbol{\theta}} \text{ELBO}(\boldsymbol{\phi}; \boldsymbol{\theta}) = \mathbb{E}_{\mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h}|\boldsymbol{x})}[\nabla_{\boldsymbol{\theta}} \log \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{h})]$$

  - Follows from linearity of the expectation.

  - This is simple and can be well approximated with Monte Carlo samples.

Parameter Gradients

- Recall that:

$$\text{ELBO}(\boldsymbol{\phi}; \boldsymbol{\theta}) = \mathbb{E}_{\mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h}|\boldsymbol{x})}[\log \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{h}) - \log \mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h} \mid \boldsymbol{x})].$$

Parameter Gradients

- Recall that:

$$\text{ELBO}(\boldsymbol{\phi}; \boldsymbol{\theta}) = \mathbb{E}_{\mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h}|\boldsymbol{x})}[\log \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{h}) - \log \mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h} \mid \boldsymbol{x})].$$

- Gradient w.r.t. $\boldsymbol{\phi}$, parameters of the inference network:

$$\begin{aligned} &\nabla_{\boldsymbol{\phi}} \text{ELBO}(\boldsymbol{\phi}; \boldsymbol{\theta}) \\ &= \mathbb{E}_{\mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h}|\boldsymbol{x})}[\underbrace{(\log \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{h}) - \log \mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h} \mid \boldsymbol{x}))}_{\text{“reward” } R_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{h})} \nabla_{\boldsymbol{\phi}} \log \mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h} \mid \boldsymbol{x})]. \end{aligned}$$

(derivation in the next slide)

Parameter Gradients

- Recall that:

$$\text{ELBO}(\boldsymbol{\phi}; \boldsymbol{\theta}) = \mathbb{E}_{\mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h}|\boldsymbol{x})}[\log \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{h}) - \log \mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h} \mid \boldsymbol{x})].$$

- Gradient w.r.t. $\boldsymbol{\phi}$, parameters of the inference network:

$$\begin{aligned} &\nabla_{\boldsymbol{\phi}} \text{ELBO}(\boldsymbol{\phi}; \boldsymbol{\theta}) \\ &= \mathbb{E}_{\mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h}|\boldsymbol{x})}[\underbrace{(\log \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{h}) - \log \mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h} \mid \boldsymbol{x}))}_{\text{“reward” } R_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{h})} \nabla_{\boldsymbol{\phi}} \log \mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h} \mid \boldsymbol{x})]. \end{aligned}$$

(derivation in the next slide)

  - High variance of Monte Carlo estimates due to the left part!
  - Requires variance reduction techniques.

Derivation of the Inference Network Gradient

$$
\begin{aligned}
&\nabla_{\boldsymbol{\phi}} \text{ELBO}(\boldsymbol{\phi}; \boldsymbol{\theta}) \\
&= \nabla_{\boldsymbol{\phi}} \mathbb{E}_{\mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h}|\boldsymbol{x})} \Big[\log \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{h}) - \log \mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h} \mid \boldsymbol{x}) \Big] \\
&= \nabla_{\boldsymbol{\phi}} \sum_{\boldsymbol{h}} \mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h} \mid \boldsymbol{x}) \log \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{h}) - \nabla_{\boldsymbol{\phi}} \sum_{\boldsymbol{h}} \mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h} \mid \boldsymbol{x}) \log \mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h} \mid \boldsymbol{x}) \\
&= \sum_{\boldsymbol{h}} \log \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{h}) \nabla_{\boldsymbol{\phi}} \mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h} \mid \boldsymbol{x}) - \sum_{\boldsymbol{h}} (1 + \log \mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h} \mid \boldsymbol{x})) \nabla_{\boldsymbol{\phi}} \mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h} \mid \boldsymbol{x}) \\
&= \sum_{\boldsymbol{h}} \Big[\log \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{h}) - \log \mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h} \mid \boldsymbol{x}) \Big] \nabla_{\boldsymbol{\phi}} \mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h} \mid \boldsymbol{x}) \\
&= \mathbb{E}_{\mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h}|\boldsymbol{x})} \Big[(\log \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{h}) - \log \mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h} \mid \boldsymbol{x})) \, \nabla_{\boldsymbol{\phi}} \log \mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h} \mid \boldsymbol{x}) \Big],
\end{aligned}
$$

where we used the facts:

$$
\sum_{\boldsymbol{h}} \nabla_{\boldsymbol{\phi}} \mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h} \mid \boldsymbol{x}) = \nabla_{\boldsymbol{\phi}} \sum_{\boldsymbol{h}} \mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h} \mid \boldsymbol{x}) = \nabla_{\boldsymbol{\phi}} 1 = 0.
$$

$$
\nabla_{\boldsymbol{\phi}} \mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h} \mid \boldsymbol{x}) = \mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h} \mid \boldsymbol{x}) \nabla_{\boldsymbol{\phi}} \log \mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h} \mid \boldsymbol{x}).
$$

- Summing up, the difficulty is the gradient w.r.t. ϕ, the parameters of the inference (encoder) network

- ... the Monte Carlo approximation has large variance.

- Summing up, the difficulty is the gradient w.r.t. ϕ, the parameters of the inference (encoder) network
- ... the Monte Carlo approximation has large variance.
- Is there a better strategy?

- Summing up, the difficulty is the gradient w.r.t. ϕ, the parameters of the inference (encoder) network
- ... the Monte Carlo approximation has large variance.
- Is there a better strategy?
- Yes: the reparameterization trick.

Reparameterization Trick (Kingma and Welling, 2013)

- How to draw samples from $\mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h} \mid \boldsymbol{x})$?

Reparameterization Trick (Kingma and Welling, 2013)

- How to draw samples from $\mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h} \mid \boldsymbol{x})$?

- Trick:

  - Use an auxiliary random variable $\boldsymbol{\epsilon}$ with fixed distribution $\mathbb{P}(\boldsymbol{\epsilon})$

  - Sample $\boldsymbol{\epsilon} \sim \mathbb{P}(\boldsymbol{\epsilon})$, and obtain $\boldsymbol{h}$ as a deterministic function of $\boldsymbol{\epsilon}$ and $\boldsymbol{x}$

$$\boldsymbol{h} = \boldsymbol{g}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \boldsymbol{x})$$

Reparameterization Trick (Kingma and Welling, 2013)

- How to draw samples from $\mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h} \mid \boldsymbol{x})$?

- Trick:

  - Use an auxiliary random variable $\boldsymbol{\epsilon}$ with fixed distribution $\mathbb{P}(\boldsymbol{\epsilon})$

  - Sample $\boldsymbol{\epsilon} \sim \mathbb{P}(\boldsymbol{\epsilon})$, and obtain $\boldsymbol{h}$ as a deterministic function of $\boldsymbol{\epsilon}$ and $\boldsymbol{x}$

$$\boldsymbol{h} = \boldsymbol{g}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \boldsymbol{x})$$

- Consequently, for any function f,

$$\mathbb{E}_{\mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h}|\boldsymbol{x})}[f(\boldsymbol{h})] \approx \frac{1}{N} \sum_{i=1}^{N} f(\boldsymbol{g}_{\boldsymbol{\phi}}(\boldsymbol{x}, \boldsymbol{\epsilon}^{(i)}))$$

Reparameterization Trick (Kingma and Welling, 2013)

- How to draw samples from $\mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h} \mid \boldsymbol{x})$?

- Trick:

  - Use an auxiliary random variable $\boldsymbol{\epsilon}$ with fixed distribution $\mathbb{P}(\boldsymbol{\epsilon})$

  - Sample $\boldsymbol{\epsilon} \sim \mathbb{P}(\boldsymbol{\epsilon})$, and obtain $\boldsymbol{h}$ as a deterministic function of $\boldsymbol{\epsilon}$ and $\boldsymbol{x}$

$$\boldsymbol{h} = \boldsymbol{g}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \boldsymbol{x})$$

- Consequently, for any function f,

$$\mathbb{E}_{\mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h}|\boldsymbol{x})}[f(\boldsymbol{h})] \approx \frac{1}{N} \sum_{i=1}^{N} f(\boldsymbol{g}_{\boldsymbol{\phi}}(\boldsymbol{x}, \boldsymbol{\epsilon}^{(i)}))$$

- $\nabla_{\boldsymbol{\phi}}$ can be estimated with regular backpropagation over $\boldsymbol{g}_{\boldsymbol{\phi}}$.

Reparameterization Trick

- This is possible in many cases for continuous latent variables:
  - exponential
  - Gaussian
  - location-scale families
  - log-normal
  - etc...

Reparameterization Trick

- This is possible in many cases for continuous latent variables:
  - exponential
  - Gaussian
  - location-scale families
  - log-normal
  - etc...
- For discrete latent variables, it is still possible via the Gumbel-softmax trick (not covered in the course).

Example: Gaussian

1. Sample d-variate standard Gaussian $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon}; 0, \boldsymbol{I})$

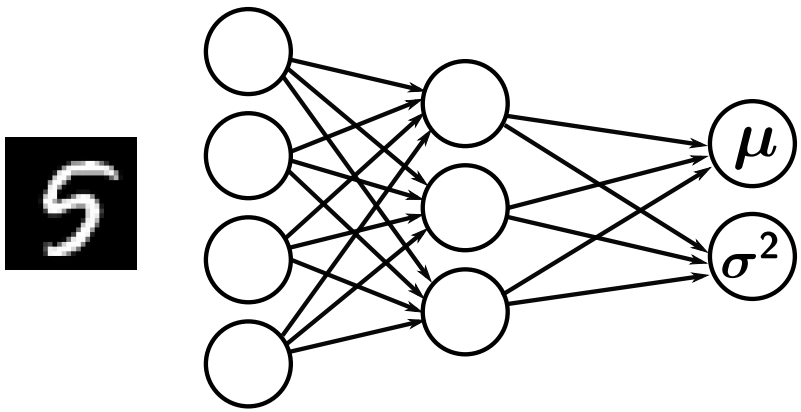
2. Use inference network $\boldsymbol{g}_\phi$ with input $\boldsymbol{x}$ to output:

   - mean $\boldsymbol{\mu}(\boldsymbol{x})$
   - component-wise variance $\boldsymbol{\sigma}^2(\boldsymbol{x}) = \text{diag}[\sigma_1^2(\boldsymbol{x}), ..., \sigma_d^2(\boldsymbol{x})]$

3. Set $\boldsymbol{h} = \boldsymbol{\mu}(\boldsymbol{x}) + \boldsymbol{\epsilon} \odot \sqrt{\sigma^2(\boldsymbol{x})}$.

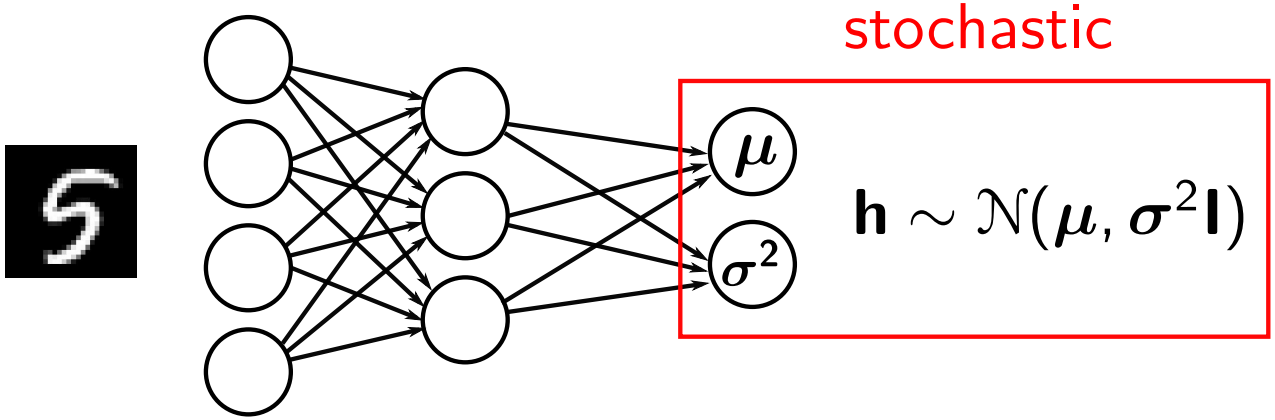
4. Thus, $\boldsymbol{h} \mid \boldsymbol{x} \sim \mathcal{N}\big(\boldsymbol{\mu}(\boldsymbol{x}), \boldsymbol{\sigma}^2(\boldsymbol{x})\big)$

Reparameterization Trick

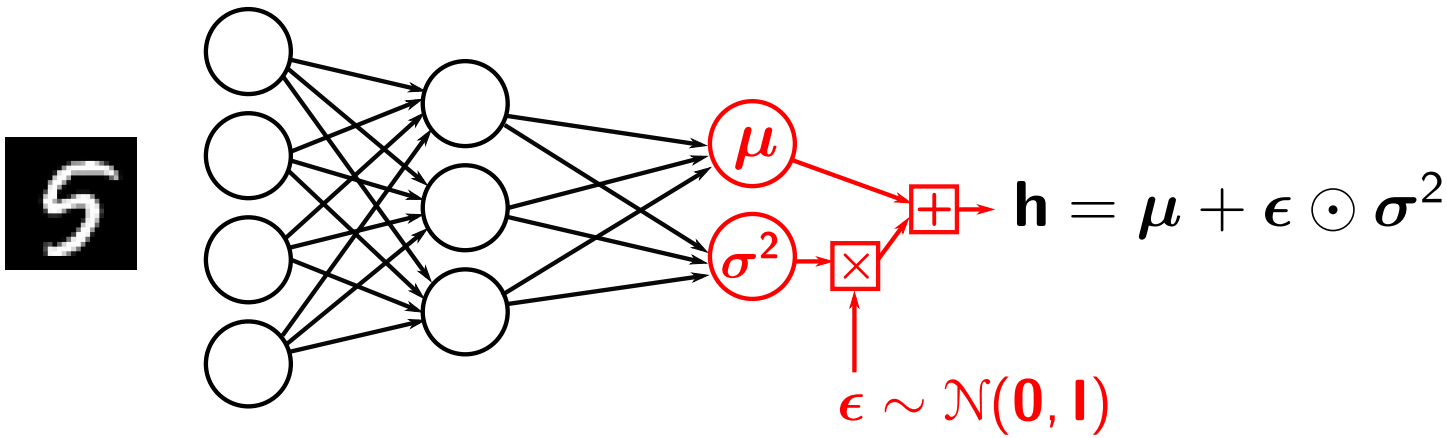


$$\mathbf{h} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2 \mathbf{I})$$

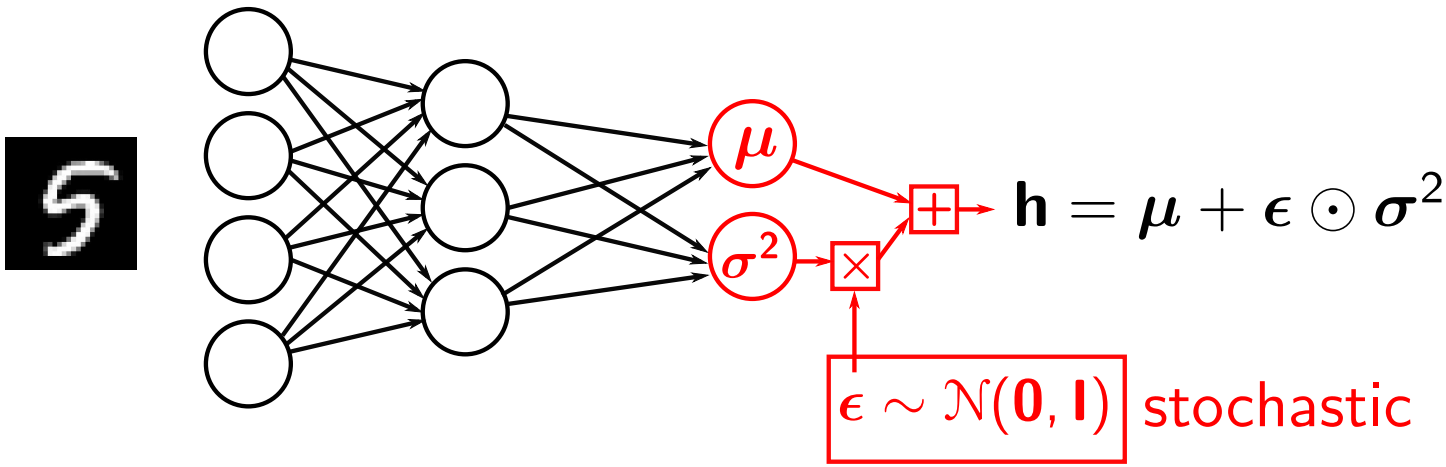
Reparameterization Trick



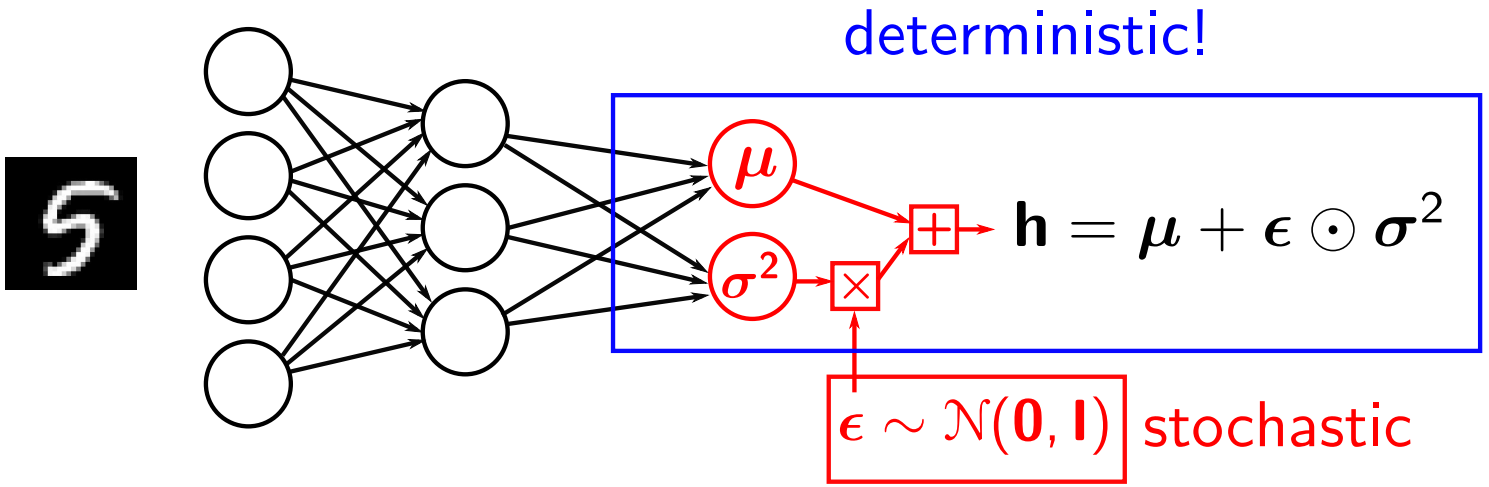
Reparameterization Trick



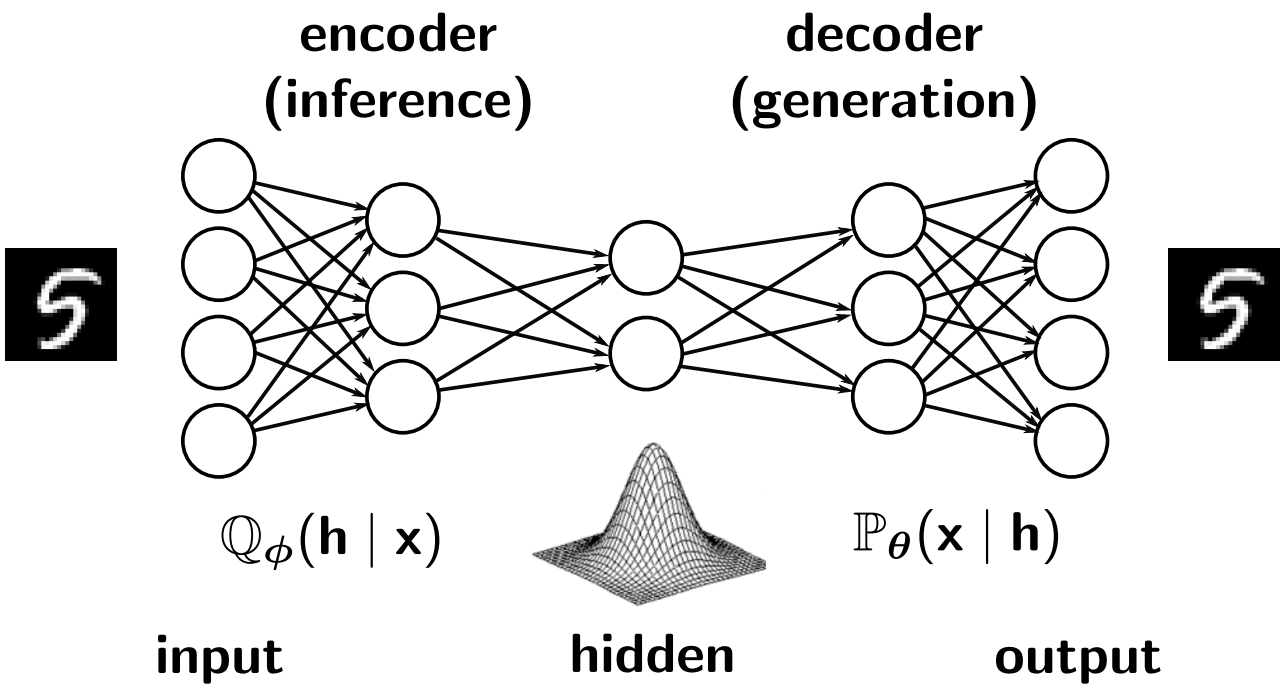
Reparameterization Trick



Reparameterization Trick

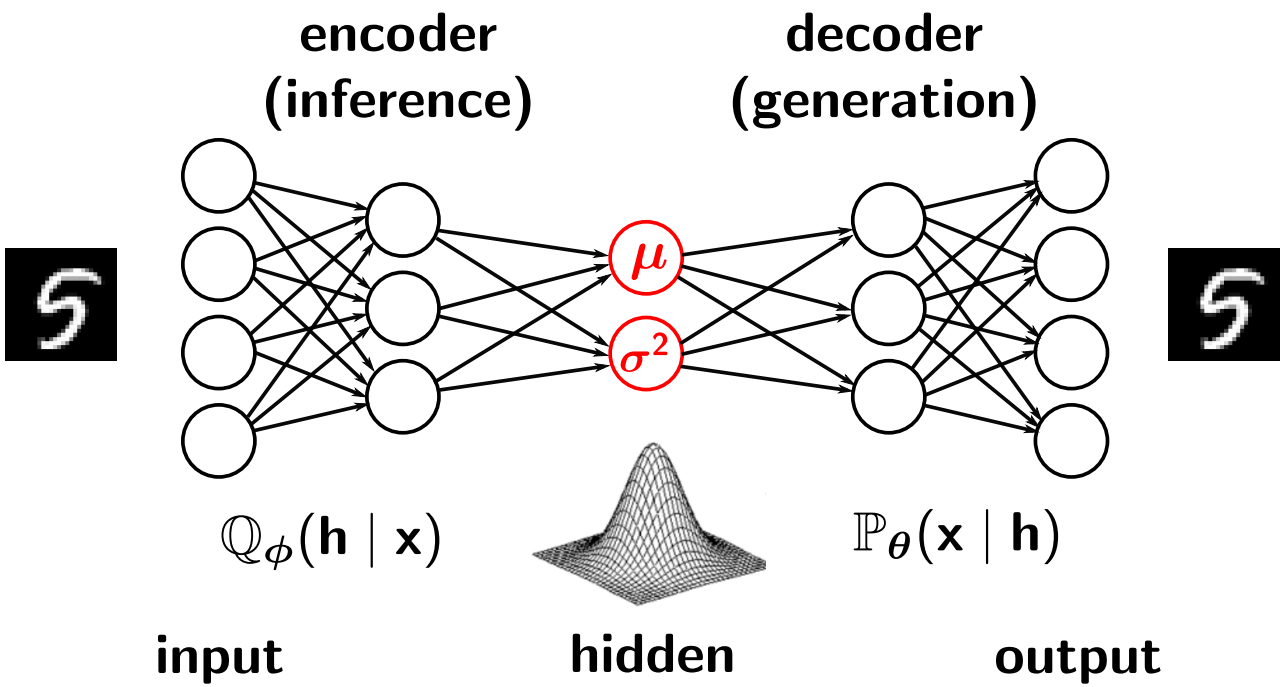


Variational Auto-Encoders (Kingma and Welling, 2013)



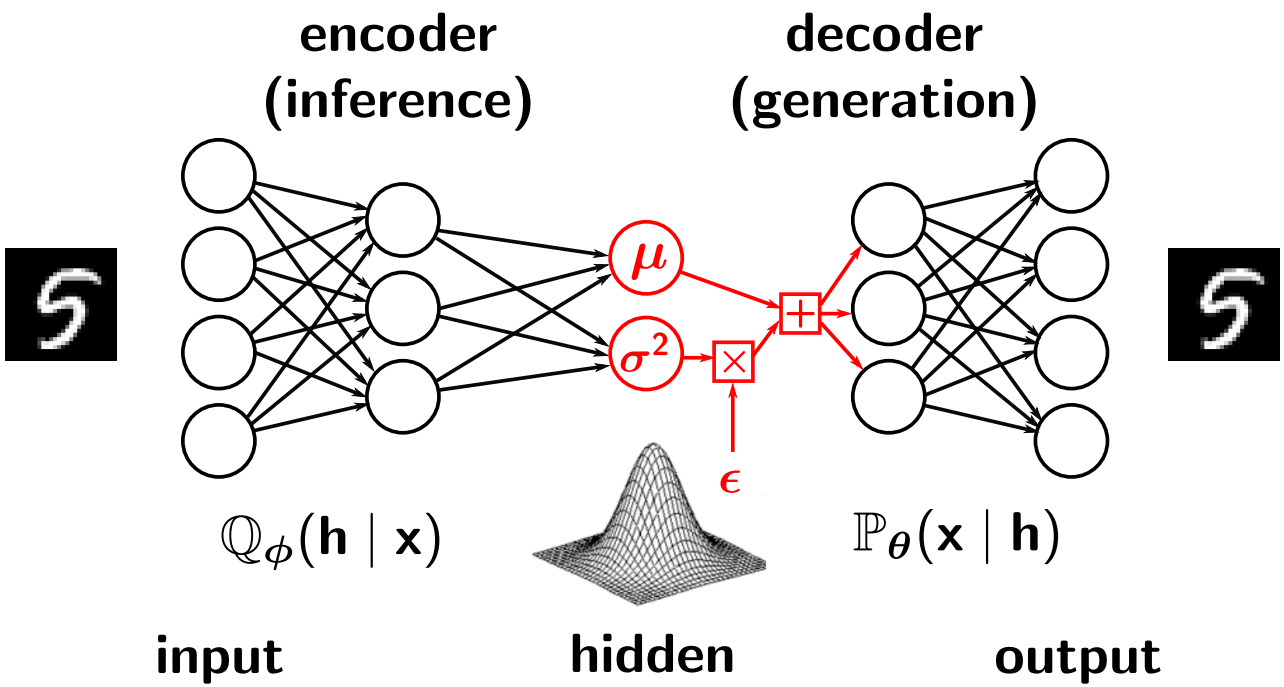
- Decoder computes $\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h})$ and $\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{h})$
- Encoder computes $\mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h} \mid \boldsymbol{x}) = \mathcal{N}(\boldsymbol{h}; \boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x}), \boldsymbol{\sigma}^2_{\boldsymbol{\phi}}(\boldsymbol{x}))$
- Loss function: ELBO.

Variational Auto-Encoders (Kingma and Welling, 2013)



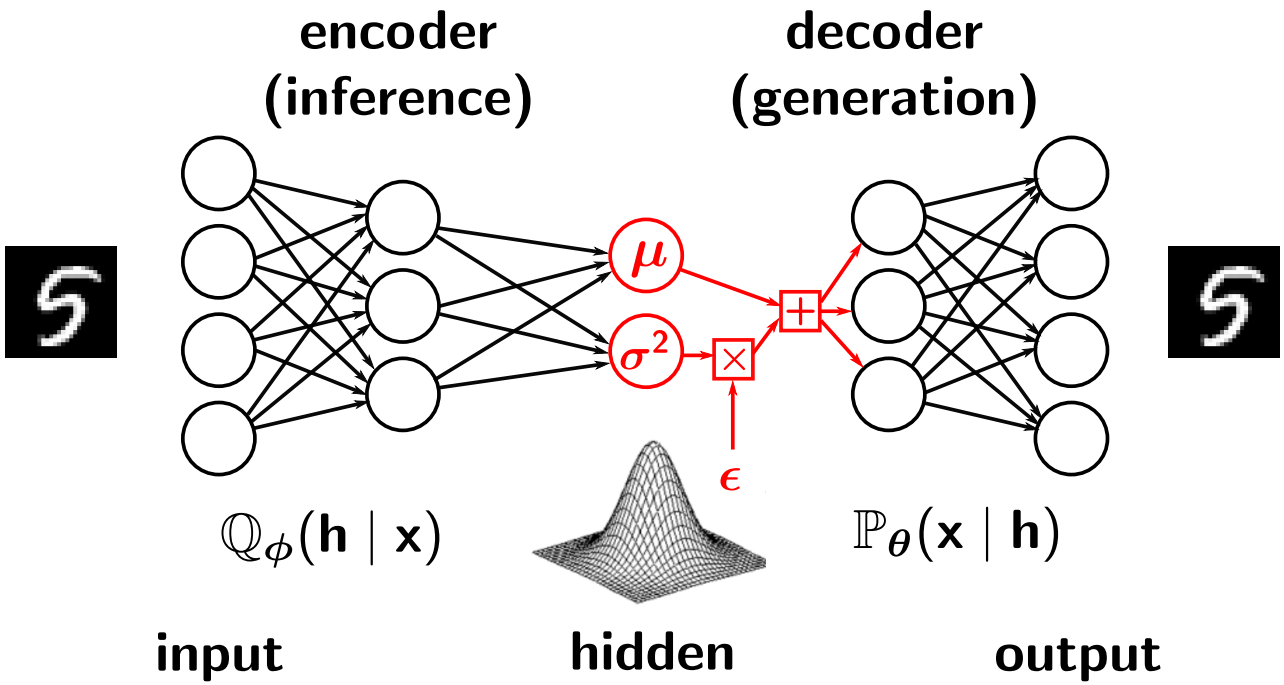
- Decoder computes $\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h})$ and $\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{h})$
- Encoder computes $\mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h} \mid \boldsymbol{x}) = \mathcal{N}(\boldsymbol{h}; \boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x}), \boldsymbol{\sigma}^2_{\boldsymbol{\phi}}(\boldsymbol{x}))$
- Loss function: ELBO.

Variational Auto-Encoders (Kingma and Welling, 2013)

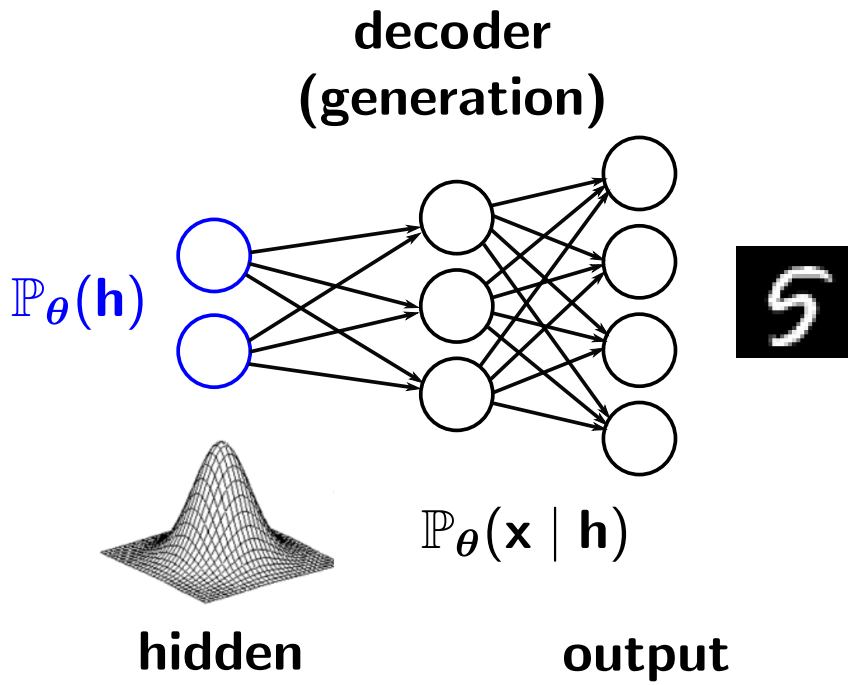


- Decoder computes $\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h})$ and $\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{h})$
- Encoder computes $\mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h} \mid \boldsymbol{x}) = \mathcal{N}(\boldsymbol{h}; \boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x}), \boldsymbol{\sigma}^2_{\boldsymbol{\phi}}(\boldsymbol{x}))$
- Loss function: ELBO.

Summing Up: VAEs at Training Time



Summing Up: VAEs at Test Time



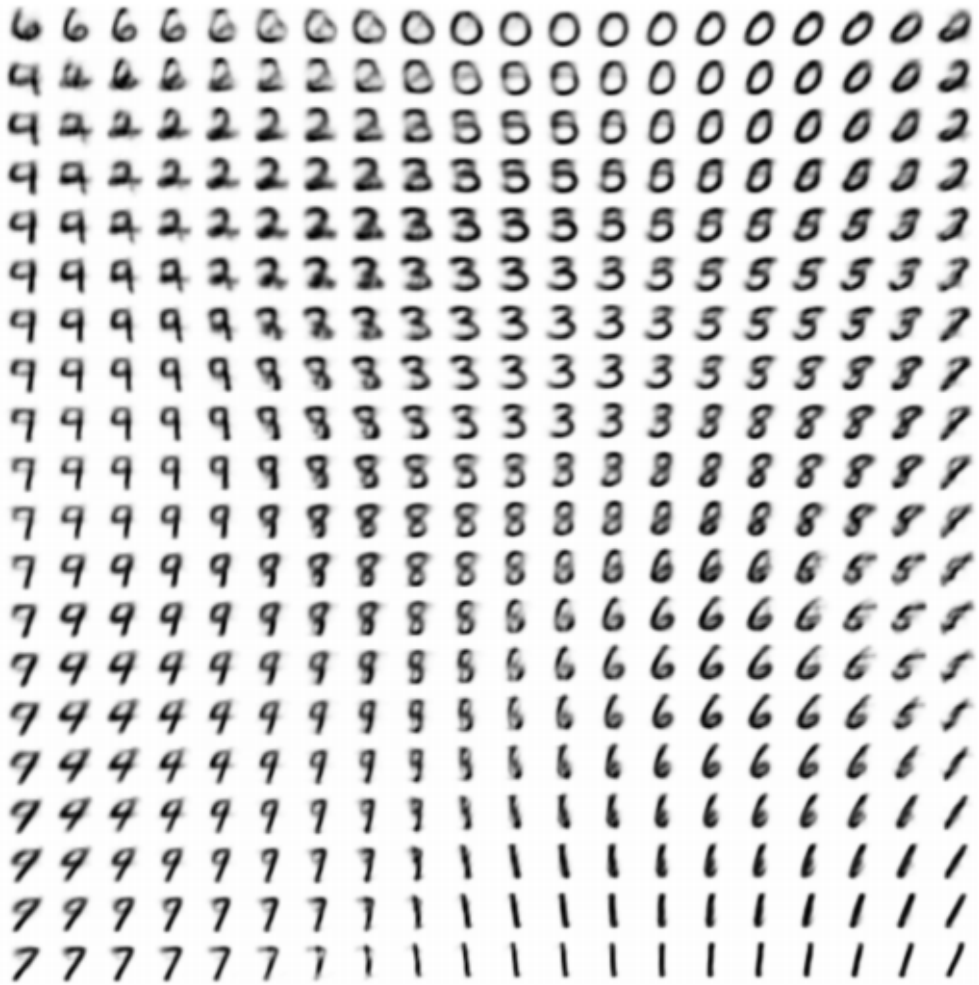
What is the Latent Variable Representing?

- Nice property of VAE: simultaneously training a parametric encoder in combination with a generator network forces the model to learn a predictable coordinate system that the encoder can capture.
- This makes it an excellent manifold learning algorithm.
- Example: the algorithm discovered two independent factors of variation present in images of faces: angle of rotation and emotional expression.

What is the Latent Variable Representing?

From Kingma and Welling (2013).

What is the Latent Variable Representing?



(b) Learned MNIST manifold

From Kingma and Welling (2013).

Issues with VAEs

- Posterior collapse: if the generative part is strong, the model may learn to ignore the latent variables:

$$\begin{aligned} \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{h}) &\approx \mathbb{P}(\boldsymbol{x}) \\ \mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h} \mid \boldsymbol{x}) &\approx \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h}). \end{aligned}$$

Issues with VAEs

- Posterior collapse: if the generative part is strong, the model may learn to ignore the latent variables:

$$\begin{aligned} \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{h}) &\approx \mathbb{P}(\boldsymbol{x}) \\ \mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h} \mid \boldsymbol{x}) &\approx \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h}). \end{aligned}$$

- Can be mitigated with a few tricks:
  - Decrease/anneal the weight of $KL(\mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h} \mid \boldsymbol{x}) \| \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h}))$ in the ELBO
  - Use auxiliary losses
  - Combine stochastic and amortized inference.

Issues with VAEs

- Posterior collapse: if the generative part is strong, the model may learn to ignore the latent variables:

$$\begin{aligned} \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{h}) &\approx \mathbb{P}(\boldsymbol{x}) \\ \mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h} \mid \boldsymbol{x}) &\approx \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h}). \end{aligned}$$

- Can be mitigated with a few tricks:
  - Decrease/anneal the weight of $KL(\mathbb{Q}_{\boldsymbol{\phi}}(\boldsymbol{h} \mid \boldsymbol{x}) \| \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h}))$ in the ELBO
  - Use auxiliary losses
  - Combine stochastic and amortized inference.
- In general, reporting both reconstruction loss and the KL term is needed to assess if the model makes use of the latent variables.

Examples of Deep Generative Models

- Auto-Regressive Networks ✓
- Restricted Boltzmann Machines ✓
- Deep Belief Networks
- Deep Boltzmann Machines
- Gaussian-Bernoulli RBMs
- Convolutional Boltzmann Machines
- Sigmoid Belief Nets
- Variational Auto-Encoders ✓
- Generative Adversarial Networks
- Denoising diffusion models

Examples of Deep Generative Models

- Auto-Regressive Networks ✓
- Restricted Boltzmann Machines ✓
- Deep Belief Networks
- Deep Boltzmann Machines
- Gaussian-Bernoulli RBMs
- Convolutional Boltzmann Machines
- Sigmoid Belief Nets
- Variational Auto-Encoders ✓
- Generative Adversarial Networks
- Denoising diffusion models

Outline

Why Maximum Likelihood?

- All models discussed aim to maximize the likelihood (evidence) $\mathbb{P}(\boldsymbol{x})$
- In fact, since this is intractable, they maximize a lower bound (ELBO)
- But if we want to build a generator, is this really the best criterion?
- Maximum likelihood tends to produce blurry images

Generative Adversarial Networks (GANs) (Goodfellow et al., 2014)

- **Key idea:**
  - keep the generation network $G = \{\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h}), \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{h})\}$
  - drop the inference network: use a discriminator network $D : \mathcal{X} \to \{0, 1\}$

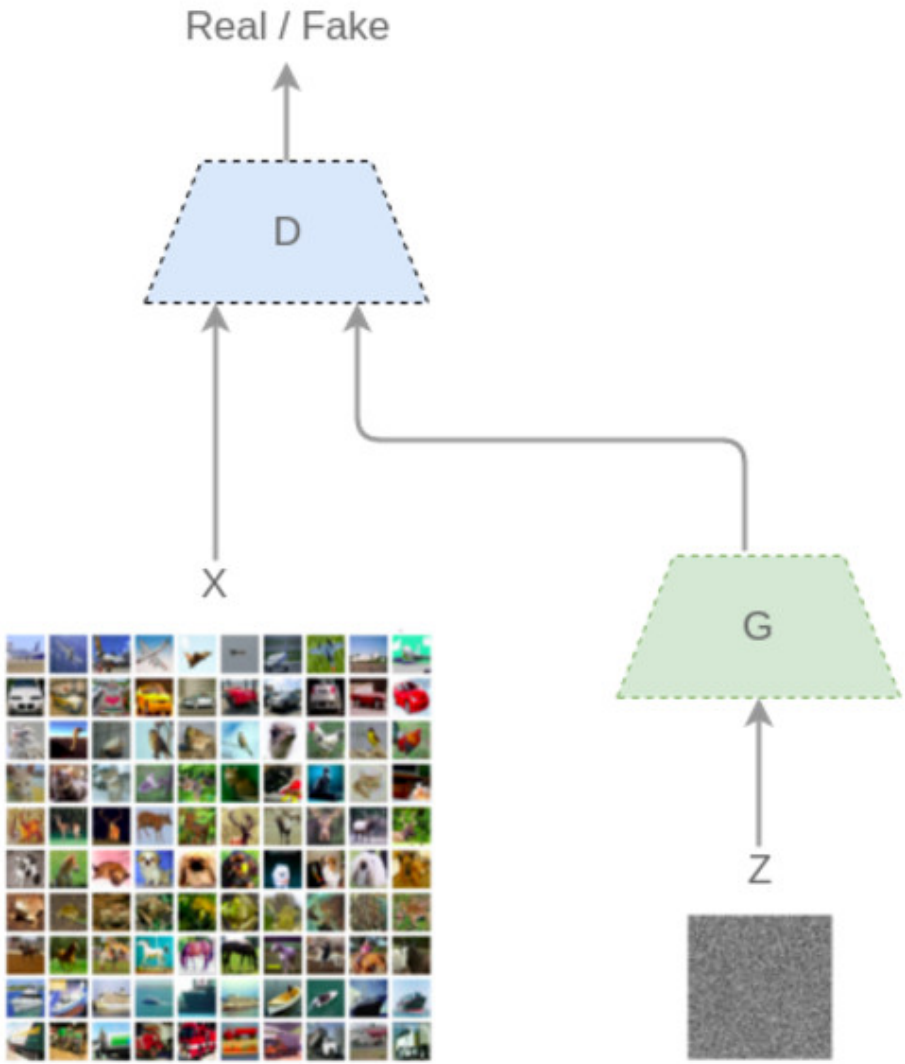
Generative Adversarial Networks (GANs) (Goodfellow et al., 2014)

- **Key idea:**
  - keep the generation network $G = \{\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h}), \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{h})\}$
  - drop the inference network: use a discriminator network $D : \mathcal{X} \to \{0, 1\}$
- Formulate the learning problem as a game between two players:
  - the generator's job is to generate data that looks real
  - the discriminator's job is to distinguish between real data and fake data generated by the generator

Generative Adversarial Networks (GANs) (Goodfellow et al., 2014)

- **Key idea:**
  - keep the generation network $G = \{\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h}), \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{h})\}$
  - drop the inference network: use a discriminator network $D : \mathcal{X} \to \{0, 1\}$
- Formulate the learning problem as a game between two players:
  - the generator's job is to generate data that looks real
  - the discriminator's job is to distinguish between real data and fake data generated by the generator
- Sort of like a Turing test: distinguish artificial from real.

Generative Adversarial Networks (GANs) (Goodfellow et al., 2014)



Minimax Game

- Saddle point problem:

$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbb{P}_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h})}[\log(1 - D(G(\boldsymbol{h})))].$$

Minimax Game

- Saddle point problem:

$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbb{P}_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h})}[\log(1 - D(G(\boldsymbol{h})))].$$

- The optimal discriminator (intractable to compute) is:

$$D^\star(\boldsymbol{x}) = \frac{\mathbb{P}_{\text{data}}(\boldsymbol{x})}{\mathbb{P}_{\text{data}}(\boldsymbol{x}) + \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x})}, \qquad \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x}) = \int \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{h}) \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h}).$$

Minimax Game

- Saddle point problem:

$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbb{P}_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h})}[\log(1 - D(G(\boldsymbol{h})))].$$

- The optimal discriminator (intractable to compute) is:

$$D^\star(\boldsymbol{x}) = \frac{\mathbb{P}_{\text{data}}(\boldsymbol{x})}{\mathbb{P}_{\text{data}}(\boldsymbol{x}) + \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x})}, \qquad \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x}) = \int \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{h}) \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h}).$$

- The optimal generator $G^\star$ minimizes the Jensen-Shannon divergence between $\mathbb{P}_{\text{data}}(\boldsymbol{x})$ and $\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x})$:

$$JS(\mathbb{P}_{\text{data}}(\boldsymbol{x}), \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x})) = \frac{1}{2} KL\left(\mathbb{P}_{\text{data}}(\boldsymbol{x}) \| \bar{\mathbb{P}}(\boldsymbol{x})\right) + \frac{1}{2} KL\left(\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x}) \| \bar{\mathbb{P}}(\boldsymbol{x})\right),$$

where $\bar{\mathbb{P}}(\boldsymbol{x}) = \frac{\mathbb{P}_{\text{data}}(\boldsymbol{x}) + \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x})}{2}$.

Wasserstein GANs (WGANs)

- Find the saddle point ($\min_G \max_D$) of

$$V(D, G) \quad = \quad \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_{\text{data}}(\boldsymbol{x})}[D(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{h} \sim \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{h})}[D(G(\boldsymbol{h}))].$$

- This is related to the Wasserstein distance (also called Earth mover's distance).

- A technical condition is that ∇D is bounded; in practice this is ensured with gradient clipping.

- This improves stability and mitigates the mode collapse problem.

Training GANs

- Use stochastic gradient descent! Alternate between:
  - Stochastic gradients updates of the generator parameters $\boldsymbol{\theta}$
  - Stochastic gradients updates of the discriminator D.

Training GANs

- Use stochastic gradient descent! Alternate between:
  - Stochastic gradients updates of the generator parameters $\boldsymbol{\theta}$
  - Stochastic gradients updates of the discriminator D.
- Several variants and schedules have been proposed.

Training GANs

- Use stochastic gradient descent! Alternate between:
  - Stochastic gradients updates of the generator parameters $\boldsymbol{\theta}$
  - Stochastic gradients updates of the discriminator D.
- Several variants and schedules have been proposed.
- Caveats:
  - no convergence guarantees
  - optimization in GANs is often difficult

Images Generated by GANs

(https://tryolabs.com/blog/2016/12/06/major-advancements-deep-learning-2016/)

Remarkable Improvement from 2014 to 2017

Ian Goodfellow
@goodfellow_ian

Follow

4 years of GAN progress (source: eff.org/files/2018/02/ …)

2014

2015

2016

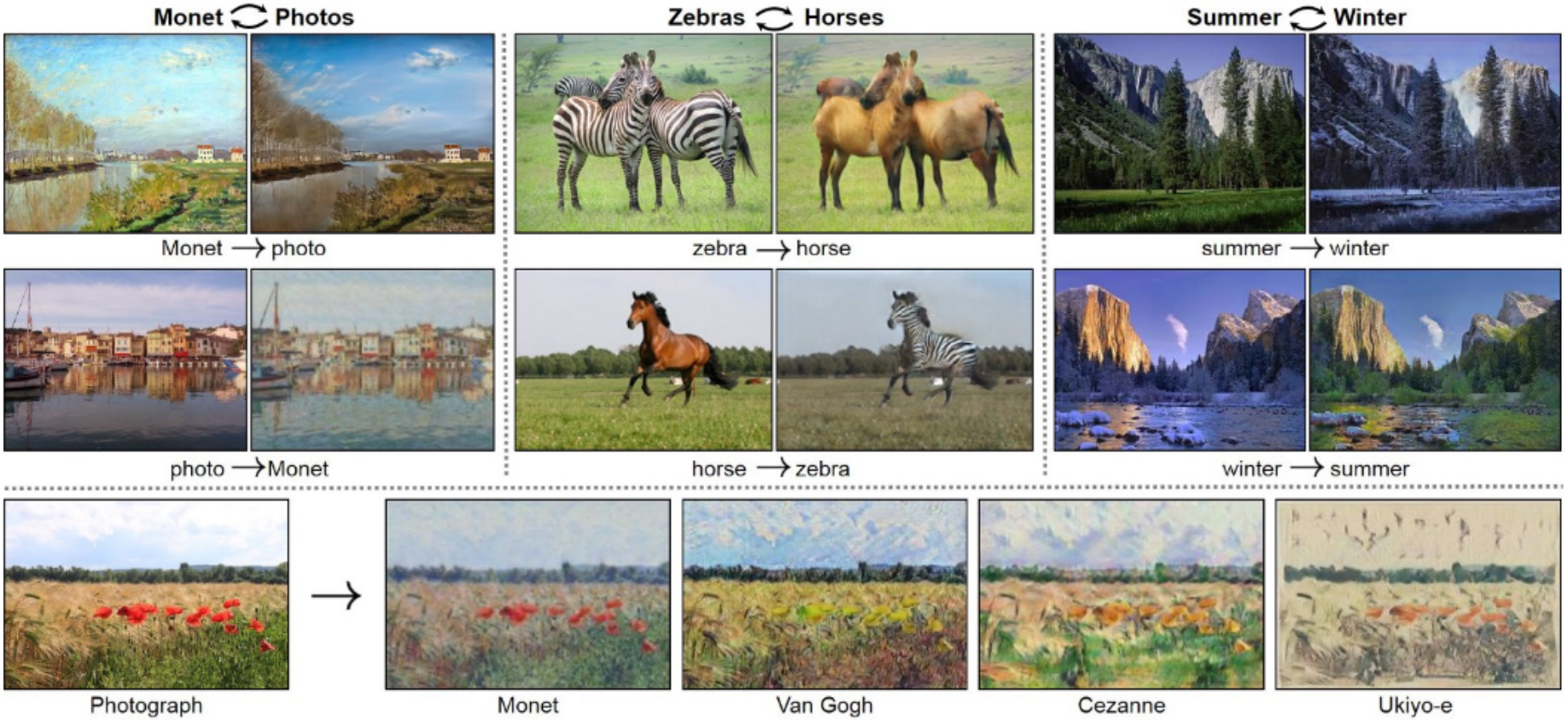
2017

7:26 PM - 2 Mar 2018

Some Extensions of GANs

- Augmenting GANs with an inference network (Dumoulin et al., 2016; Donahue et al., 2016)
- Domain adversarial training for domain adaptation (Ganin et al., 2016)
- Conditional GANs and semi-supervised GANs (Salimans et al., 2016)
- CycleGAN (Zhu et al., 2017): "translate" images from a source domain $\mathcal{X}$ to a target domain $\mathcal{Y}$ without paired examples. Use two generators $G : \mathcal{Y} \rightarrow \mathcal{X}$ and $F : \mathcal{X} \rightarrow \mathcal{Y}$ and introduce a **cycle consistency loss** to push $F(G(\boldsymbol{y})) \approx \boldsymbol{y}$ and $G(F(\boldsymbol{x})) \approx \boldsymbol{x}$.

Image-to-Image Translation with CycleGAN (Zhu et al., 2017)



(https://junyanz.github.io/CycleGAN)

Failure Cases

(https://junyanz.github.io/CycleGAN)

Examples of Deep Generative Models

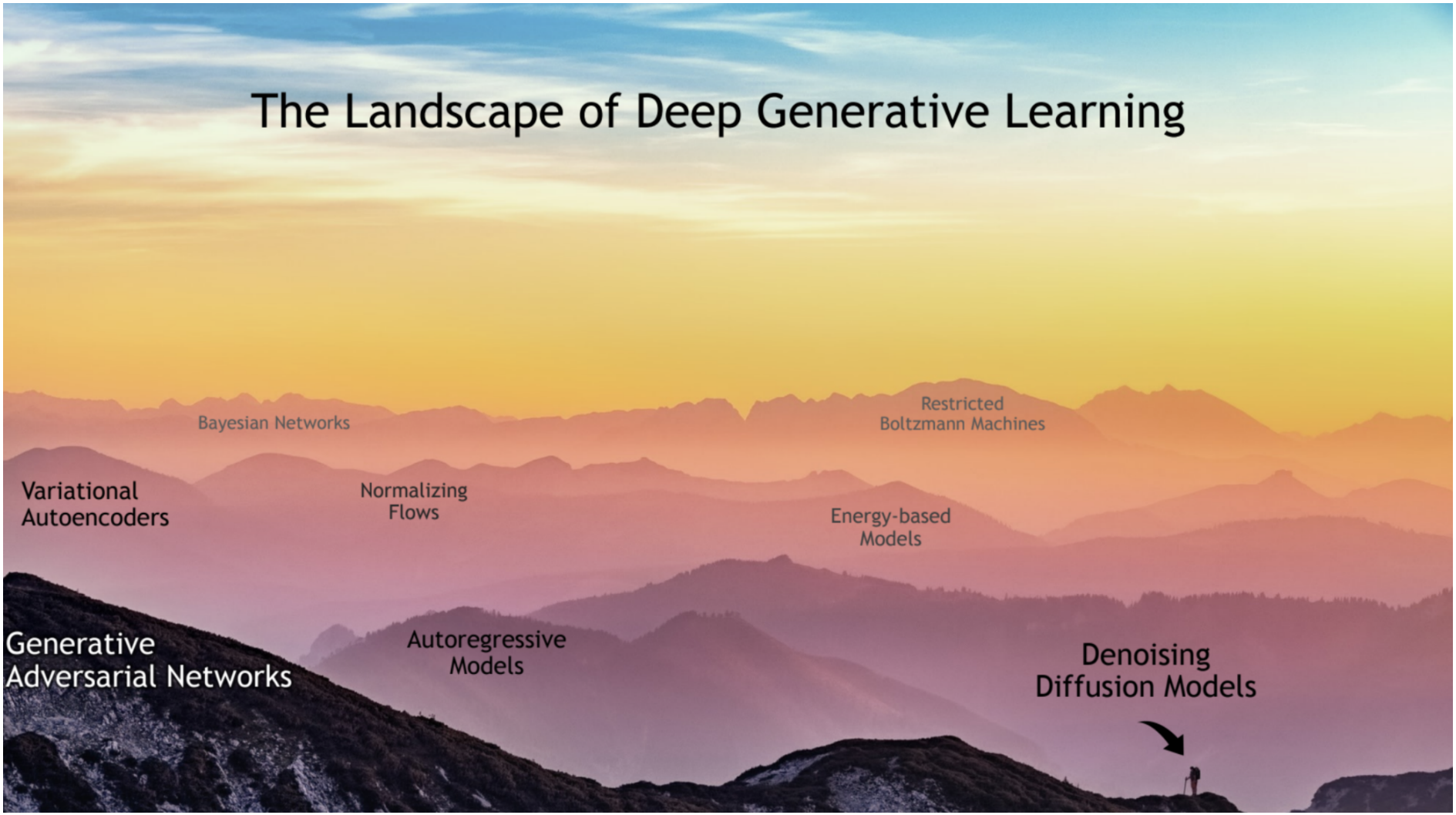
- Auto-Regressive Networks ✓
- Restricted Boltzmann Machines ✓
- Deep Belief Networks
- Deep Boltzmann Machines
- Gaussian-Bernoulli RBMs
- Convolutional Boltzmann Machines
- Sigmoid Belief Nets
- Variational Auto-Encoders ✓
- Generative Adversarial Networks ✓
- Denoising diffusion models

Examples of Deep Generative Models

- Auto-Regressive Networks ✓
- Restricted Boltzmann Machines ✓
- Deep Belief Networks
- Deep Boltzmann Machines
- Gaussian-Bernoulli RBMs
- Convolutional Boltzmann Machines
- Sigmoid Belief Nets
- Variational Auto-Encoders ✓
- Generative Adversarial Networks ✓
- Denoising diffusion models

Outline

Denoising Diffusion Models

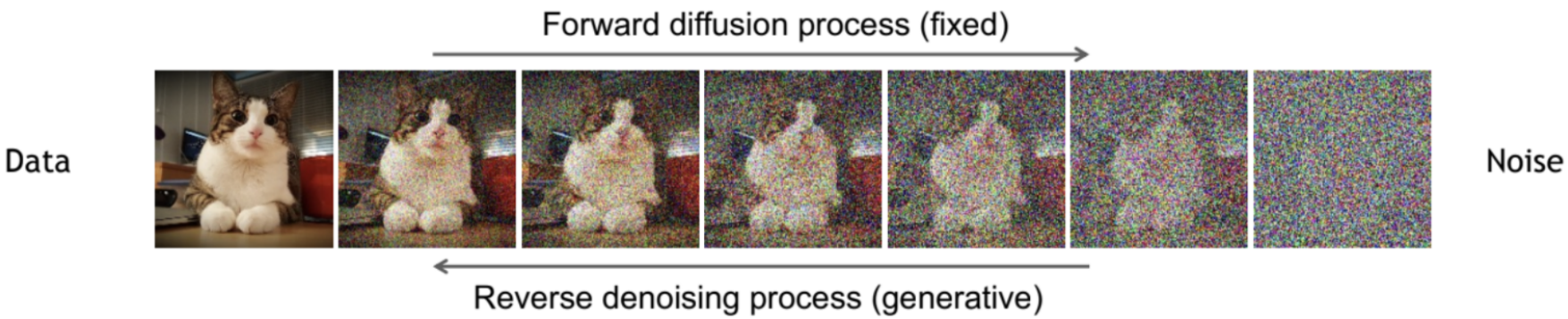


This and the next 10 slides: Diffusion Models Tutorial, CVPR2023, J. Song, C. Meng, A. Vahdat. `https://cvpr2023-tutorial-diffusion-models.github.io/`

Denoising Diffusion Models: Basic Idea

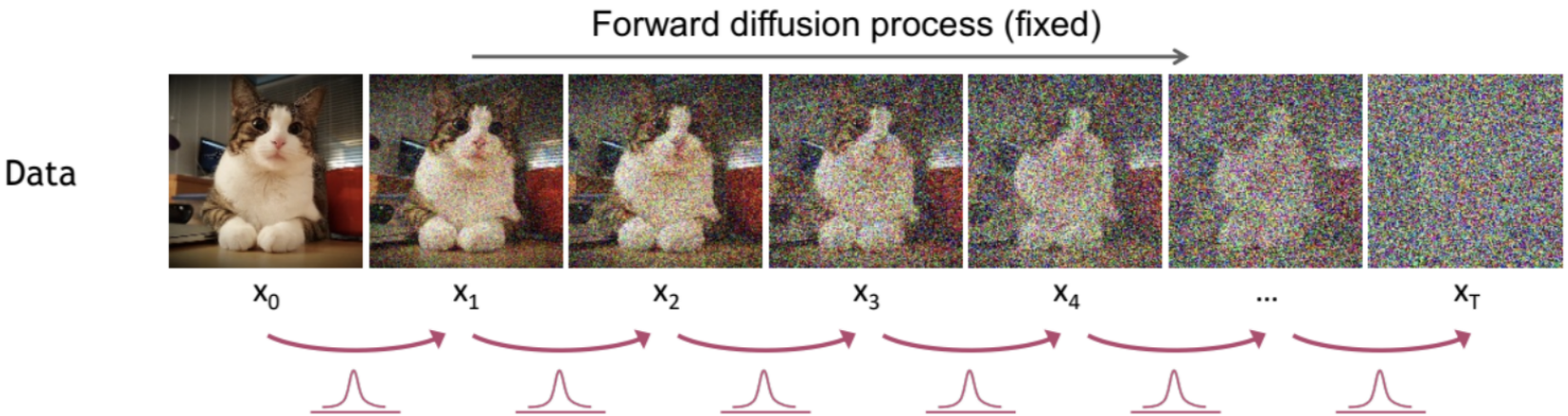
Denoising diffusion models consist of two processes:

- Forward diffusion process that gradually adds noise to input
- Reverse denoising process that learns to generate data by denoising



Forward Diffusion

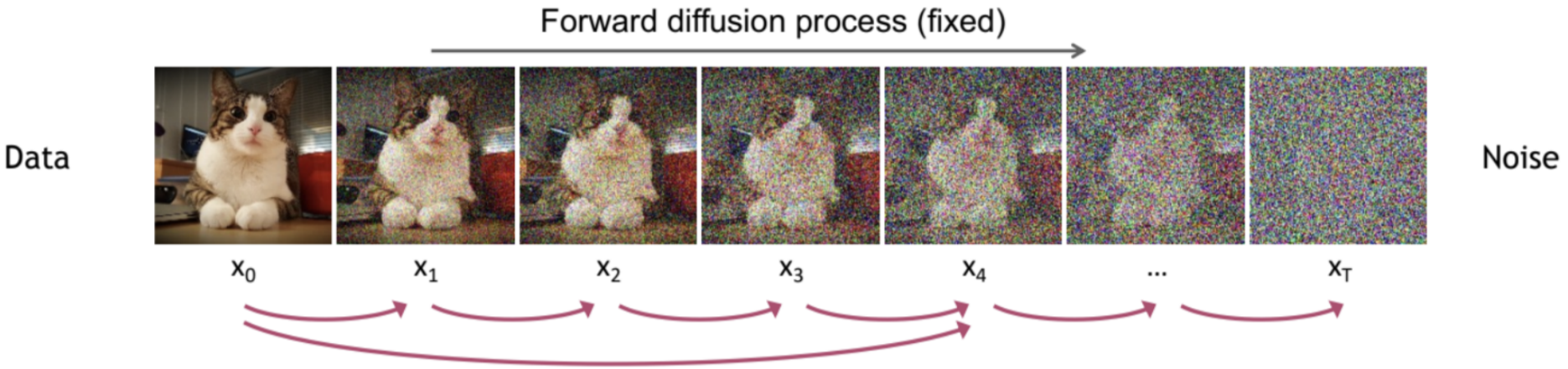
The formal definition of the forward process in T steps:



$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x_t}; \sqrt{1-\beta_t}\mathbf{x_{t-1}}, \beta_t\mathbf{I}) \quad \Rightarrow \quad q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) \qquad \text{(joint)}$$

14

Diffusion Kernel



Define $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$ ➡ $q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}))$ (Diffusion Kernel)

For sampling: $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{(1-\bar{\alpha}_t)}\,\epsilon$ where $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

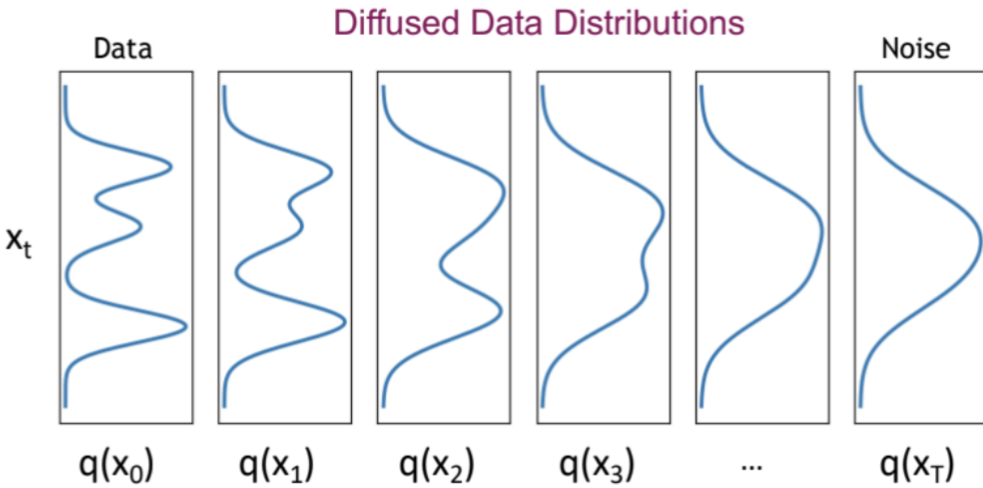
β_t values schedule (i.e., the noise schedule) is designed such that $\bar{\alpha}_T \to 0$ and $q(\mathbf{x}_T|\mathbf{x}_0) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I}))$

Diffusions are Convolutions

So far, we discussed the diffusion kernel $q(\mathbf{x}_t|\mathbf{x}_0)$ but what about $q(\mathbf{x}_t)$?

$$\underbrace{q(\mathbf{x}_t)}_{\text{Diffused data dist.}} = \int \underbrace{q(\mathbf{x}_0, \mathbf{x}_t)}_{\text{Joint dist.}} d\mathbf{x}_0 = \int \underbrace{q(\mathbf{x}_0)}_{\text{Input data dist.}} \underbrace{q(\mathbf{x}_t|\mathbf{x}_0)}_{\text{Diffusion kernel}} d\mathbf{x}_0$$

The diffusion kernel is Gaussian convolution.



We can sample $\mathbf{x}_t \sim q(\mathbf{x}_t)$ by first sampling $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ and then sampling $\mathbf{x}_t \sim q(\mathbf{x}_t|\mathbf{x}_0)$ (i.e., ancestral sampling).

16

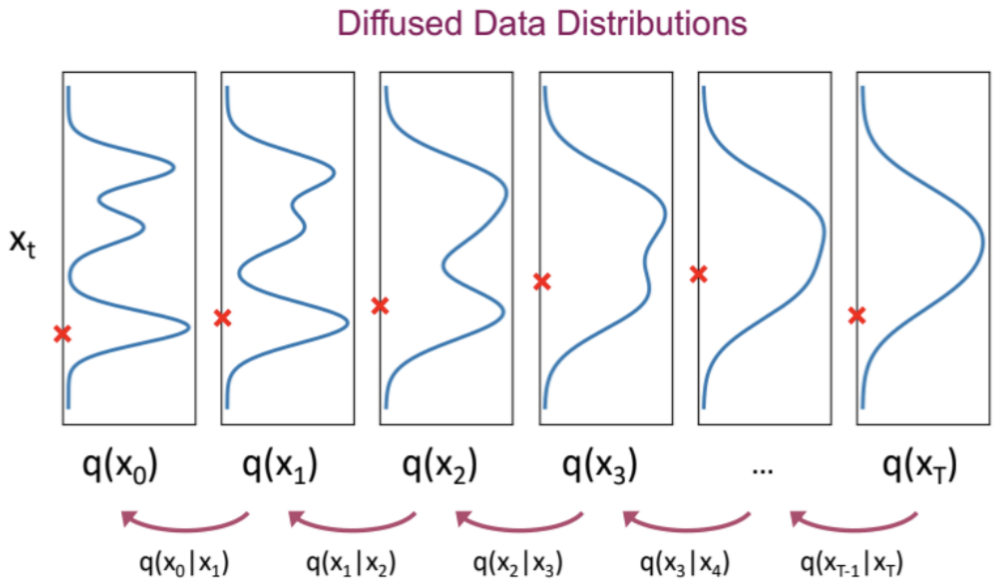
Generative Learning by Learning to Denoise

Recall, that the diffusion parameters are designed such that $q(\mathbf{x}_T) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I}))$

Generation:

Sample $\mathbf{x}_T \sim \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$

Iteratively sample $\mathbf{x}_{t-1} \sim \underbrace{q(\mathbf{x}_{t-1}|\mathbf{x}_t)}_{\text{True Denoising Dist.}}$

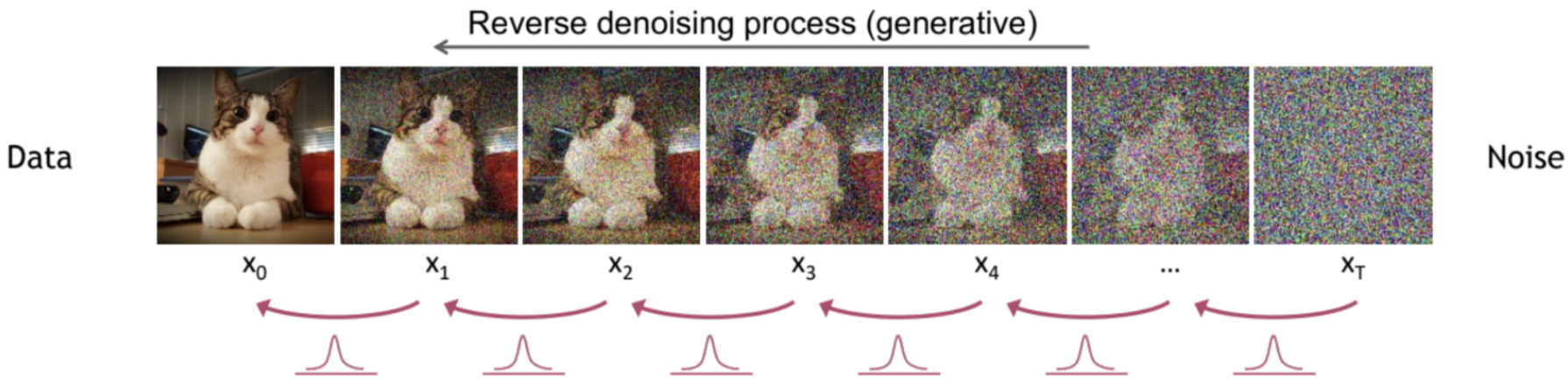


Can we approximate $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$? Yes, we can use a Normal distribution if β_t is small in each forward diffusion step.

18

Reverse Denoising Process

Formal definition of forward and reverse processes in T steps:



$$p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$$

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \underbrace{\mu_\theta(\mathbf{x}_t, t)}_{\text{Trainable network}}, \sigma_t^2\mathbf{I})$$

$$\Rightarrow \quad p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$$

Generative Learning by Learning to Denoise

For training, we can form variational upper bound that is commonly used for training variational autoencoders:

$$\mathbb{E}_{q(\mathbf{x}_0)}\left[-\log p_\theta(\mathbf{x}_0)\right] \leq \mathbb{E}_{q(\mathbf{x}_0)q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] =: L$$

Recall that $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{(1-\bar{\alpha}_t)}\,\epsilon$. Ho et al. NeurIPS 2020 parameterized the mean of denoising model via:

$$\mu_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{1-\beta_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\,\epsilon_\theta(\mathbf{x}_t, t)\right)$$

Using a few simple arithmetic operations, we can write down the variational objective as:

$$L = \mathbb{E}_{\mathbf{x}_0\sim q(\mathbf{x}_0), t\sim\mathcal{U}\{1,T\}, \epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left[\lambda_t||\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\epsilon, t)||^2\right]$$

Ho et al. NeurIPS 2020 observe that simply setting λ_t to 1 for all t works best in practice.

Algorithm 1 Training

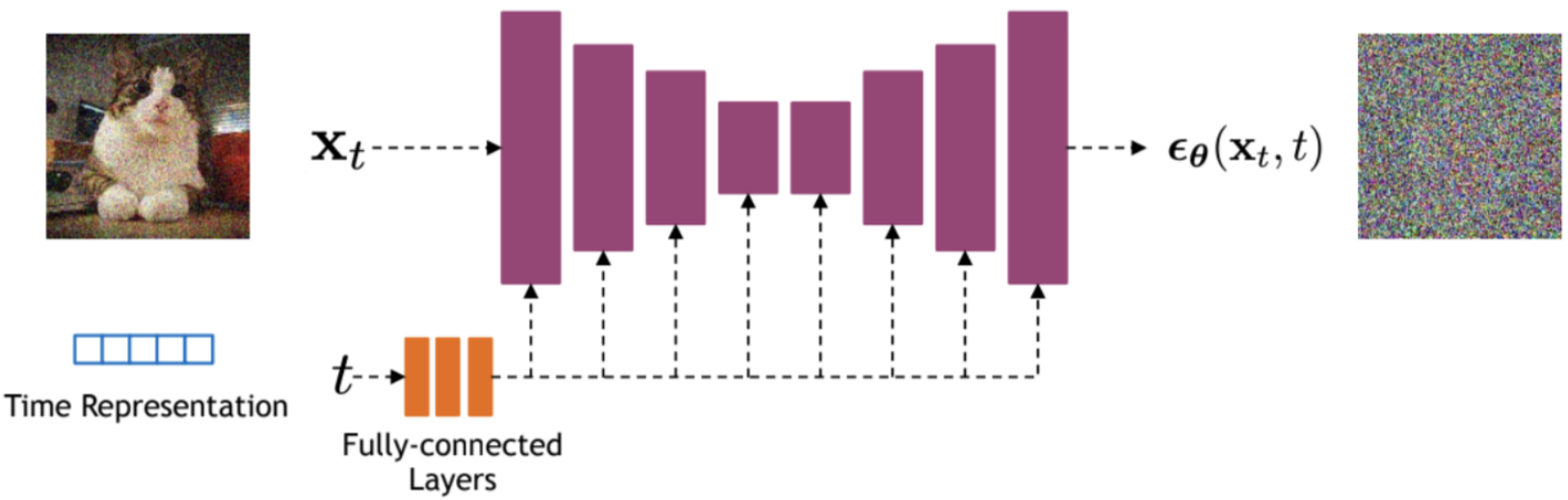
1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

Implementation of the Denoisers

Diffusion models often use U-Net architectures with ResNet blocks and self-attention layers to represent $\epsilon_\theta(\mathbf{x}_t, t)$



Time representation: sinusoidal positional embeddings or random Fourier features.

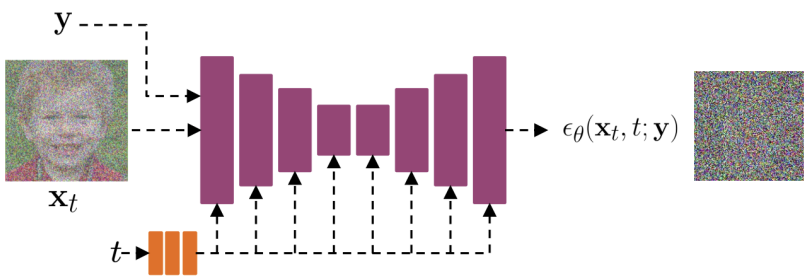
Conditional Training and Generation

Conditional sampling can be considered as training $p(\mathbf{x}|\mathbf{y})$ where $\mathbf{y}$ is the input conditioning (e.g., text) and $\mathbf{x}$ is generated output (e.g., image)

Train the score model for $\mathbf{x}$ conditioned on $\mathbf{y}$ using:

$$\mathbb{E}_{(\mathbf{x},\mathbf{y})\sim p_{\text{data}}(\mathbf{x},\mathbf{y})}\mathbb{E}_{\epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\mathbb{E}_{t\sim\mathcal{U}[0,T]}\ ||\epsilon_\theta(\mathbf{x}_t, t; \mathbf{y}) - \epsilon||_2^2$$

The conditional score is simply a U-Net with $\mathbf{x_t}$ and $\mathbf{y}$ together in the input.



Text-to-Image Generation Examples

DALL·E 2

"a propaganda poster depicting a cat dressed as french emperor napoleon holding a piece of cheese"

IMAGEN

"A photo of a raccoon wearing an astronaut helmet, looking out of the window at night."

TIME

← THE BEST INVENTIONS OF 2022

Artificial Imagination

OpenAI DALL-E 2

DALL·E

Outline

Conclusions

- Generative models are useful to model high-dimensional data.
- Latent-variable generative models are appealing since they are more compact.
- Often, computing evidence and posterior distributions is intractable.
- A common surrogate for maximum likelihood is the evidence lower bound (ELBO).
- Variational auto-encoders optimize the ELBO with amortized VI.
- Generative adversarial networks (GANs) are formulated as a game between a generator and a discriminator.
- Denoising diffusion models have impressive results and can be conditioned by text.
- Open problem (for VAE, GAN, and diffusion models): discrete data.

Thank you!

Questions?

References I

Ackley, D., Hinton, G., and Sejnowski, T. (1985). A learning algorithm for Boltzmann machines. *Cognitive science*, 9(1):147–169.

Donahue, J., Krähenbühl, P., and Darrell, T. (2016). Adversarial feature learning. *arXiv preprint arXiv:1605.09782*.

Dumoulin, V., Belghazi, I., Poole, B., Mastropietro, O., Lamb, A., Arjovsky, M., and Courville, A. (2016). Adversarially learned inference. *arXiv preprint arXiv:1606.00704*.

Finn, C., Christiano, P., Abbeel, P., and Levine, S. (2016). A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *arXiv preprint arXiv:1611.03852*.

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. (2016). Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). Deep Learning. Book in preparation for MIT Press.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680.

Jang, E., Gu, S., and Poole, B. (2016). Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Le Roux, N. and Bengio, Y. (2008). Representational power of restricted boltzmann machines and deep belief networks. *Neural computation*, 20(6):1631–1649.

Maddison, C. J., Mnih, A., and Teh, Y. W. (2016). The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*.

Metz, L., Poole, B., Pfau, D., and Sohl-Dickstein, J. (2016). Unrolled generative adversarial networks. *arXiv preprint arXiv:1611.02163*.

Oord, A. v. d., Kalchbrenner, N., and Kavukcuoglu, K. (2016). Pixel Recurrent Neural Networks. In *Proc. of the International Conference on Machine Learning*.

Pfau, D. and Vinyals, O. (2016). Connecting generative adversarial networks and actor-critic methods. *arXiv preprint arXiv:1610.01945*.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. (2016). Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2234–2242.

References II

Smolensky, P. (1986). Information processing in dynamical systems: Foundations of harmony theory. Technical report, DTIC Document.

Theis, L., Oord, A. v. d., and Bethge, M. (2015). A note on the evaluation of generative models. *arXiv preprint arXiv:1511.01844*.

Wainwright, M. and Jordan, M. (2008). *Graphical Models, Exponential Families, and Variational Inference.* Now Publishers.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint*.