Deep Generative Models

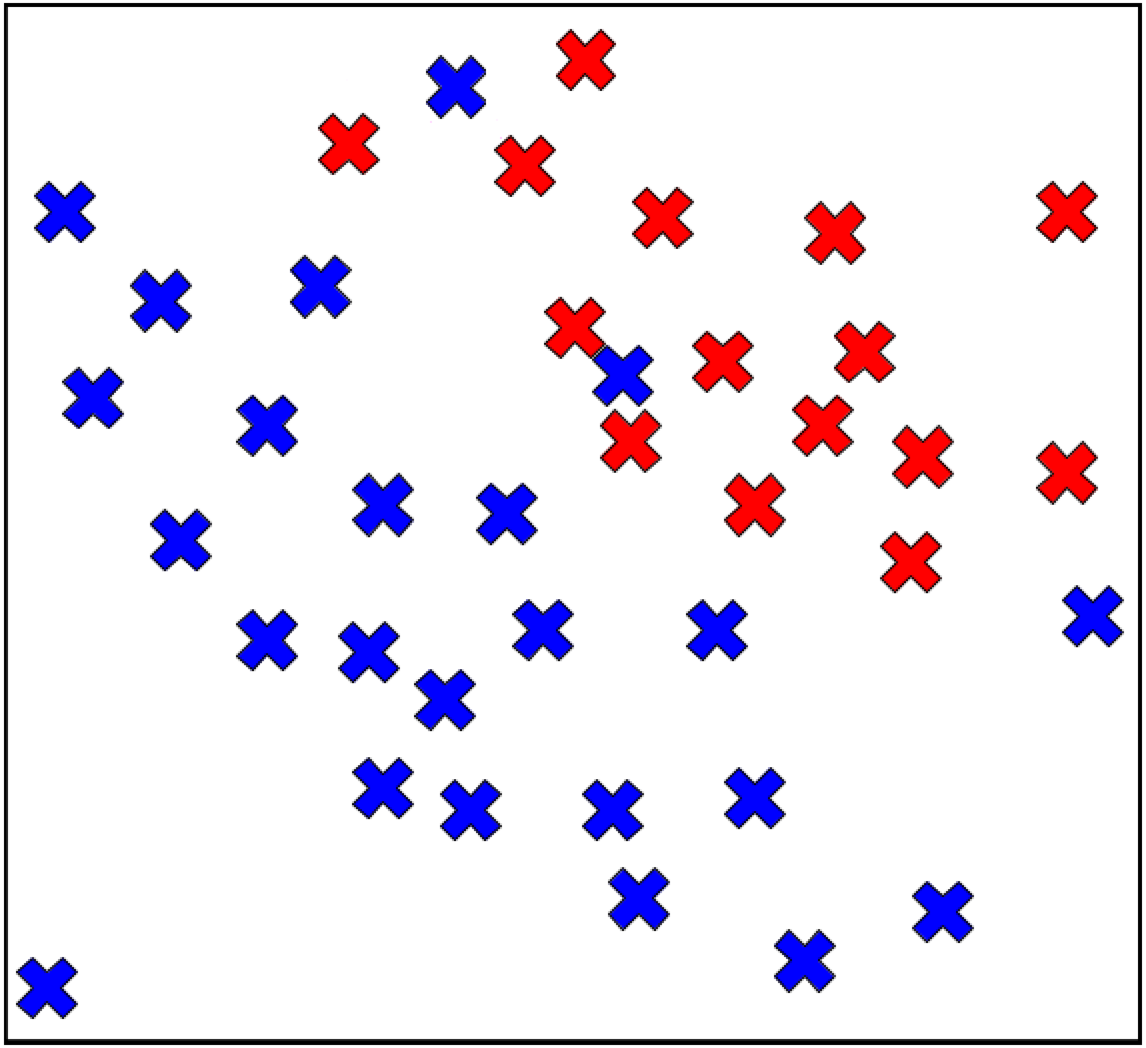
Jakub M. Tomczak
AMLAB, UvA

2017/11/29

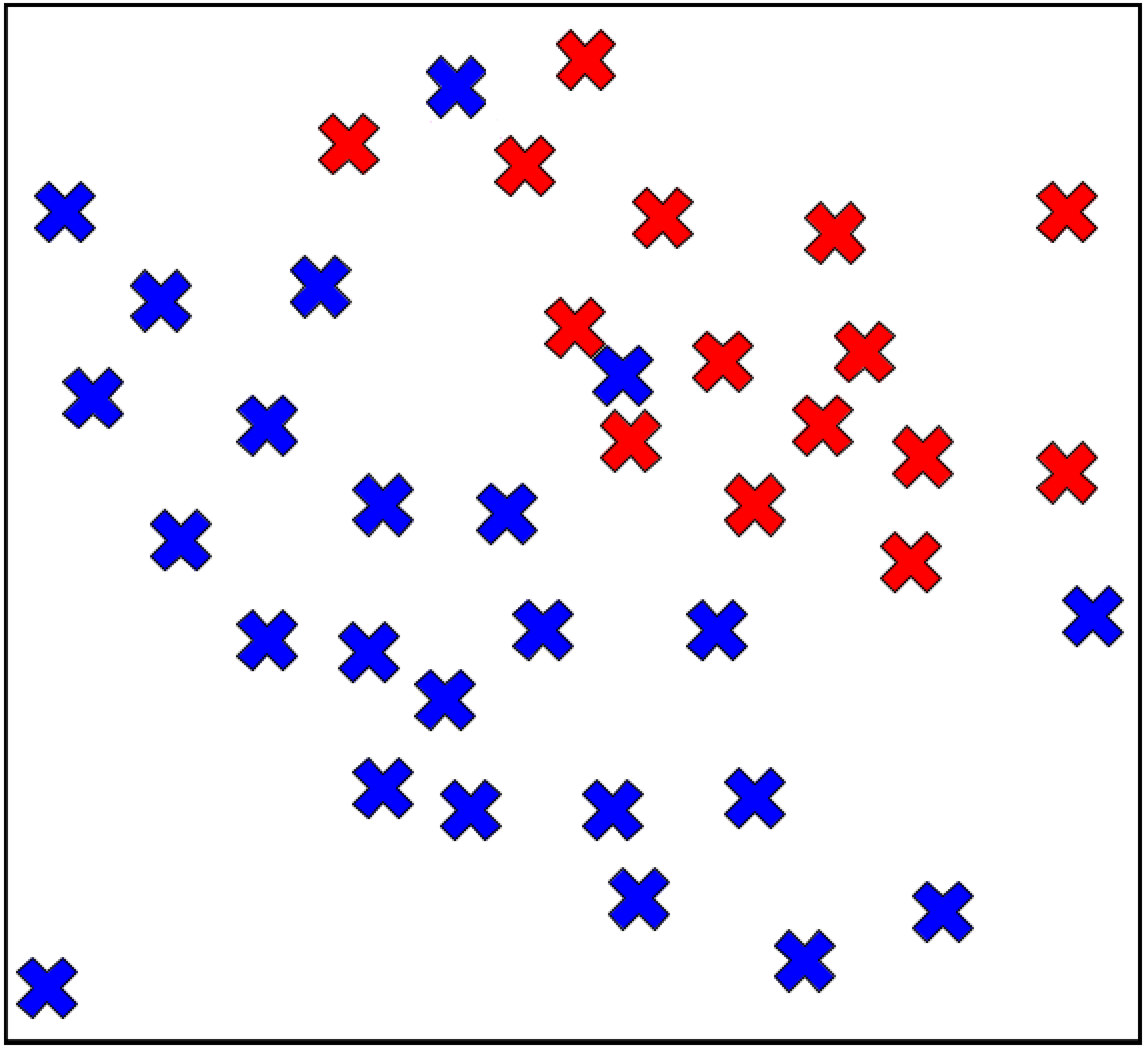
UNIVERSITY OF AMSTERDAM

Do we need generative modeling?

Do we need generative modeling?

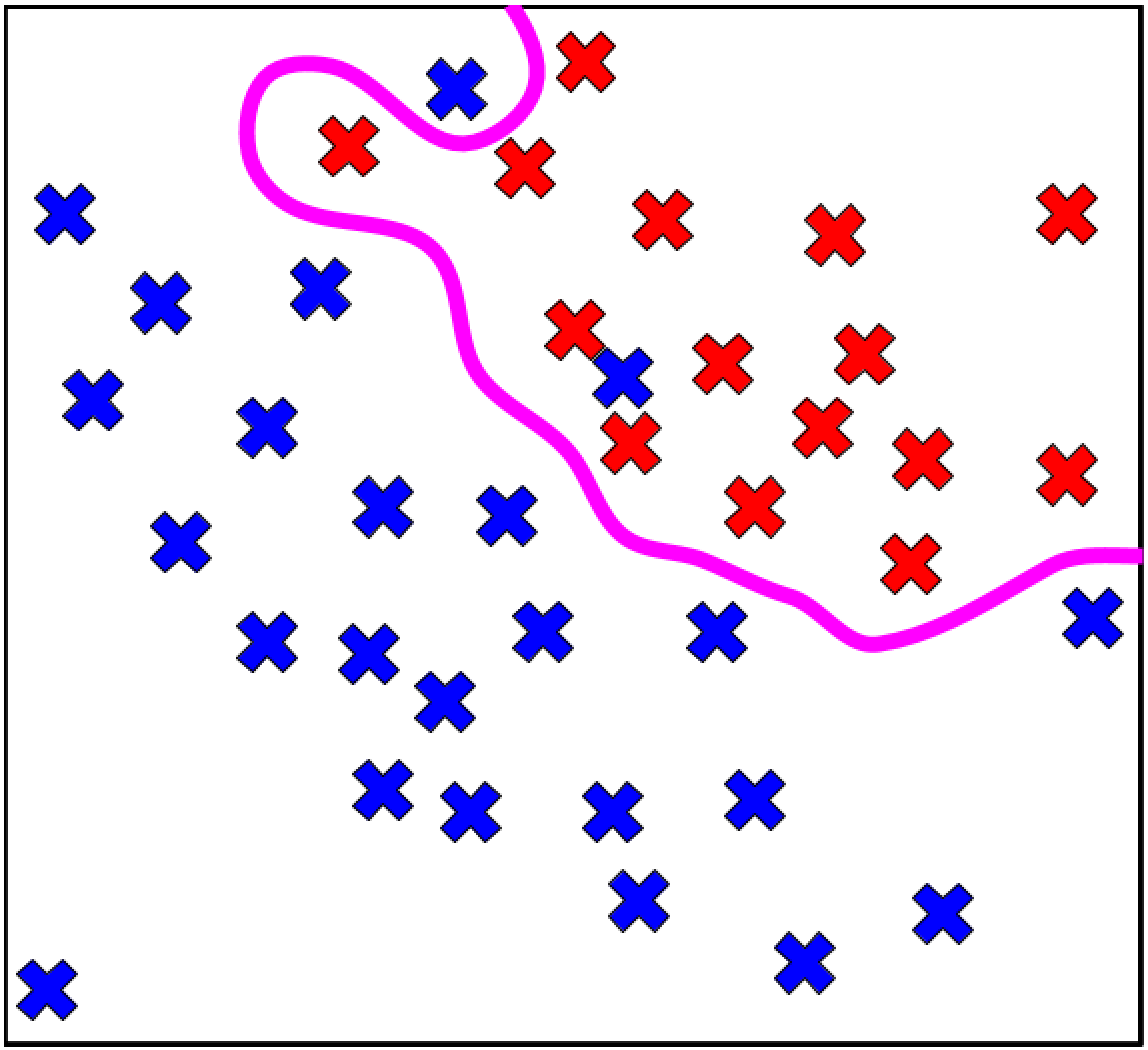


Do we need generative modeling?



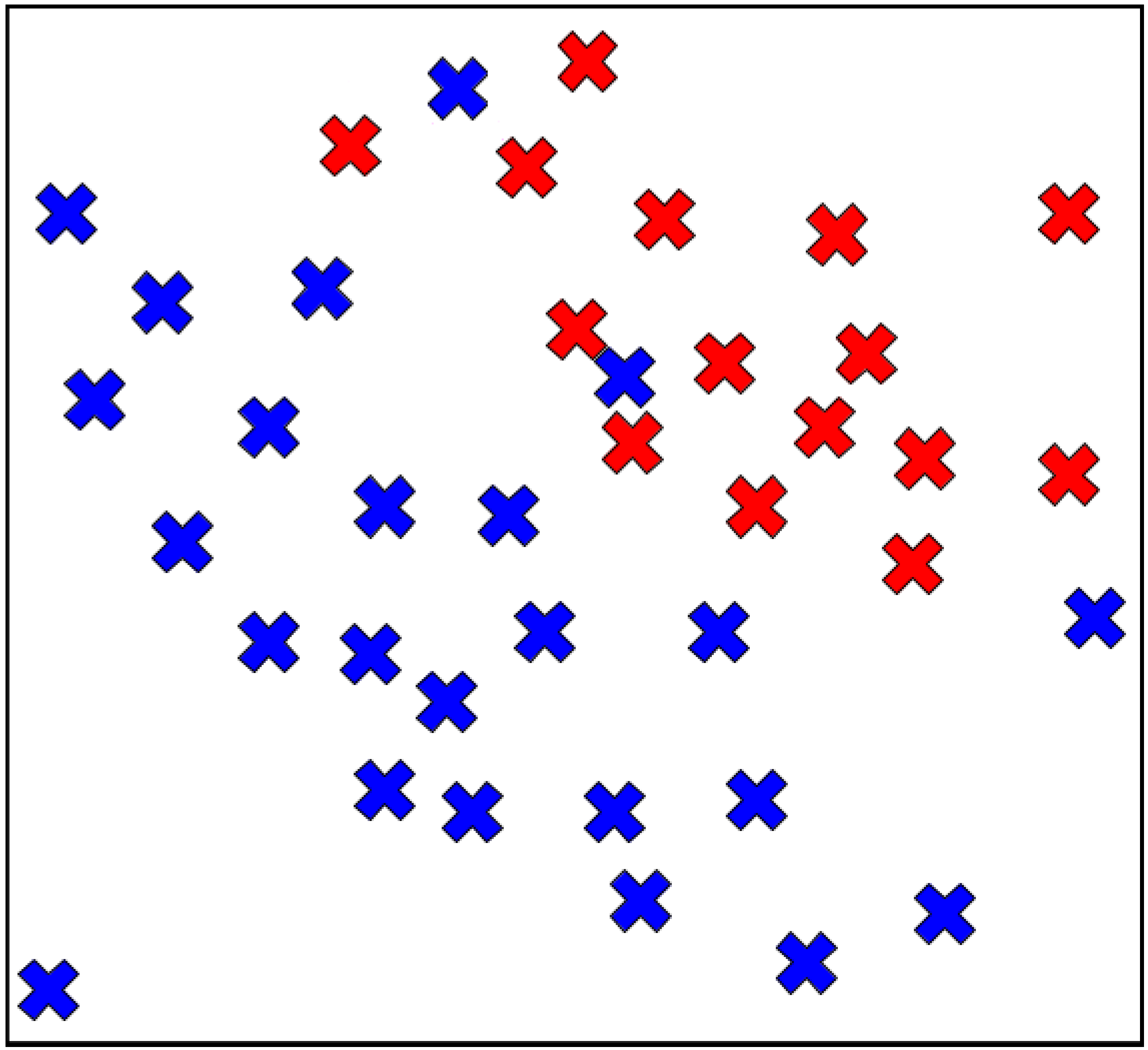
$$p_\theta(y|x)$$

Do we need generative modeling?



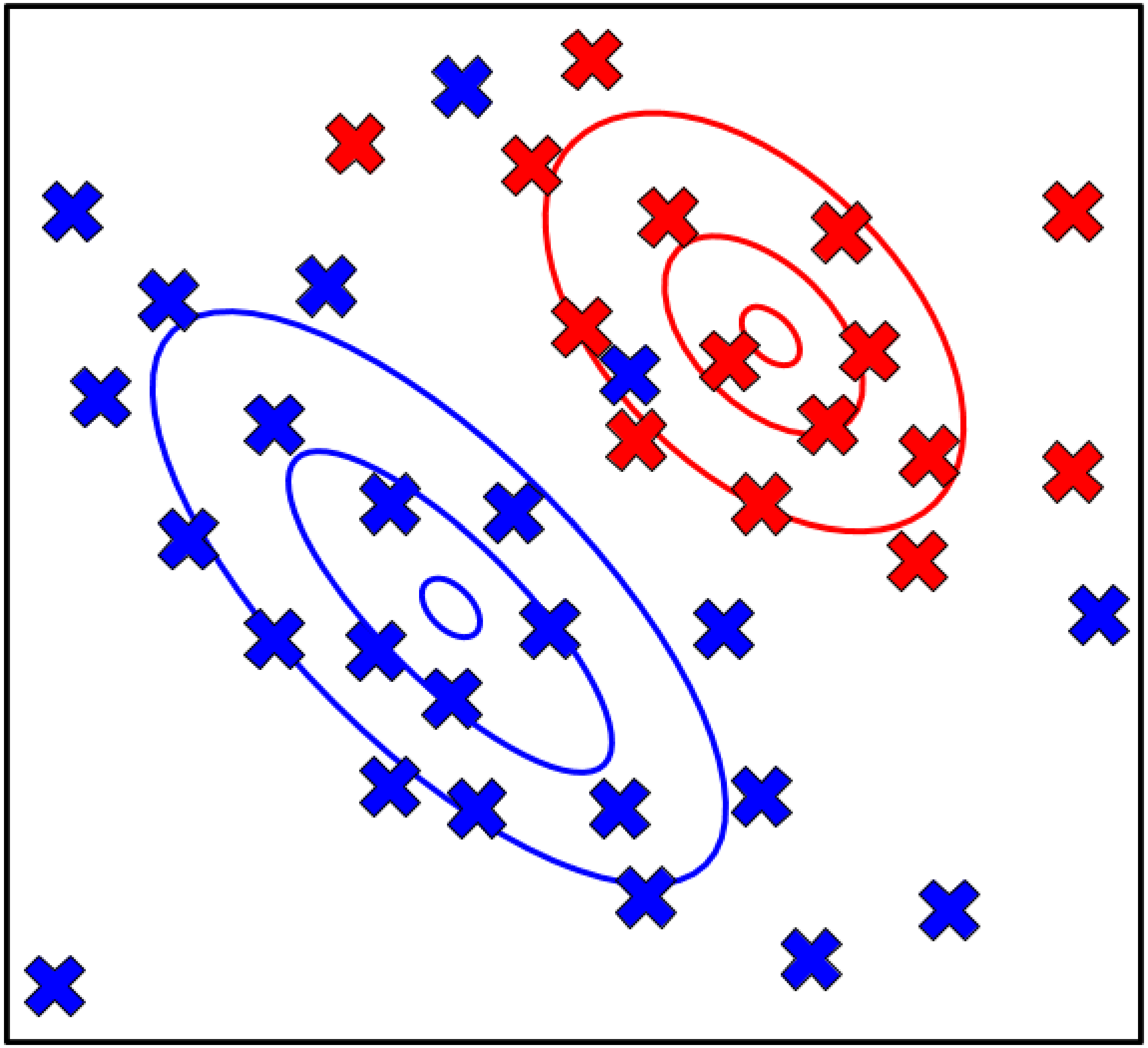
$p_\theta(y|x)$

Do we need generative modeling?



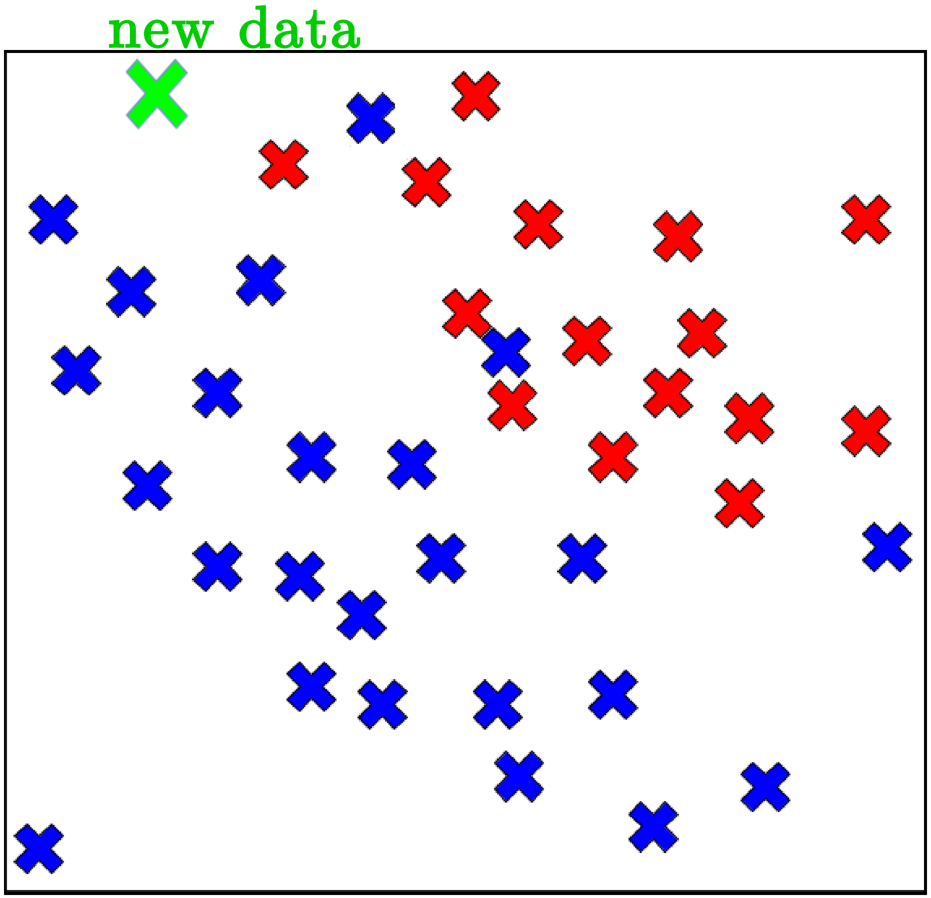
$$p_\theta(x, y) = p_\theta(y|x)\ p_\theta(x)$$

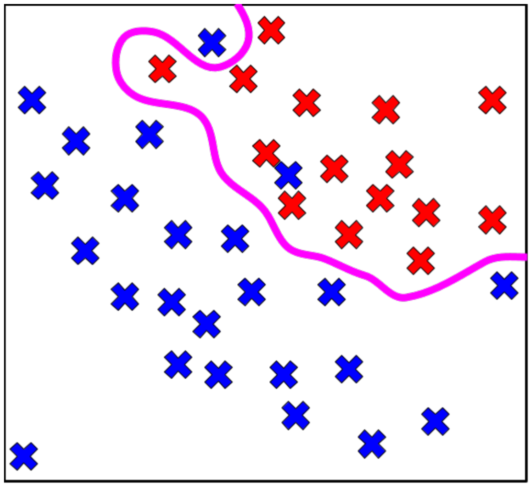
Do we need generative modeling?



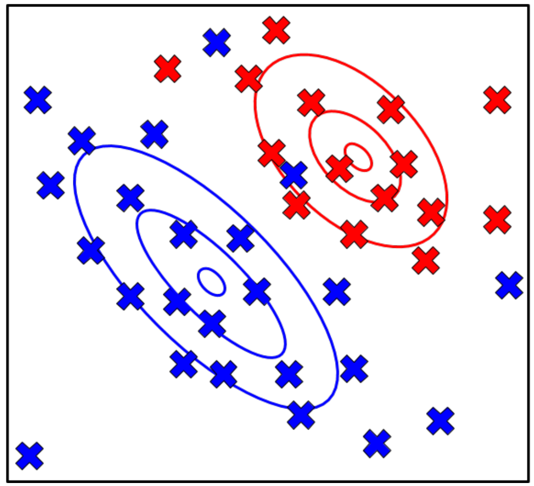
$$p_\theta(x, y) = p_\theta(y|x)\ p_\theta(x)$$

Do we need generative modeling?



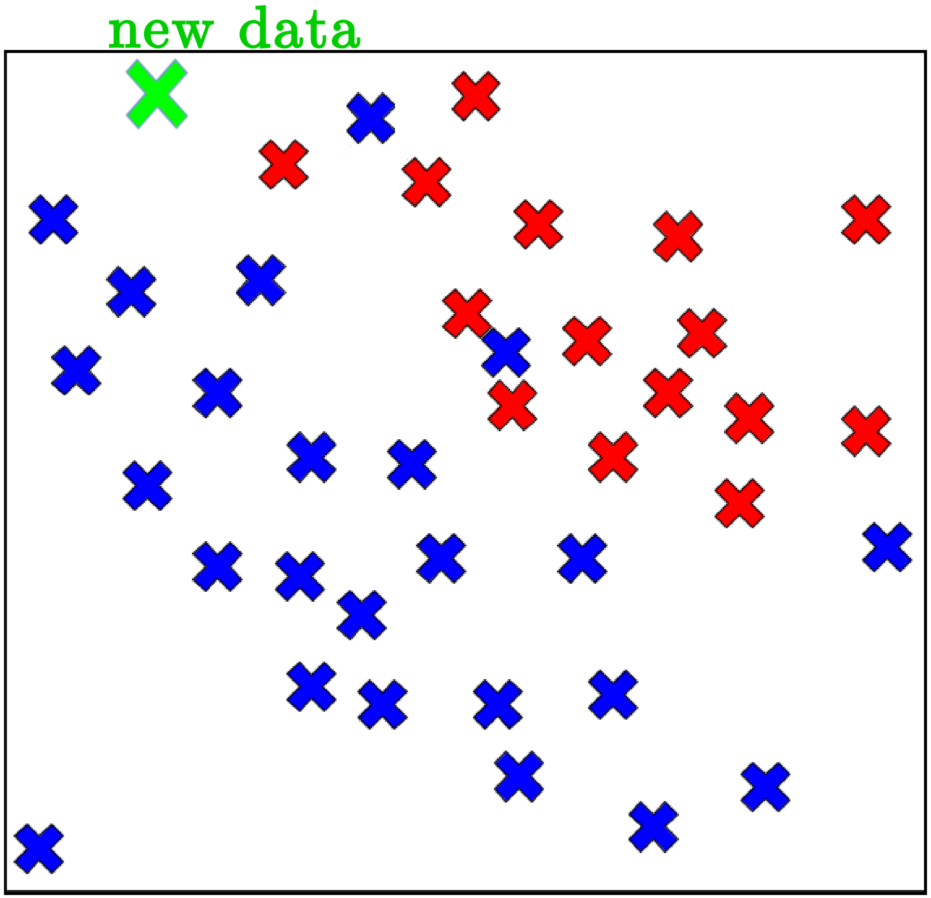


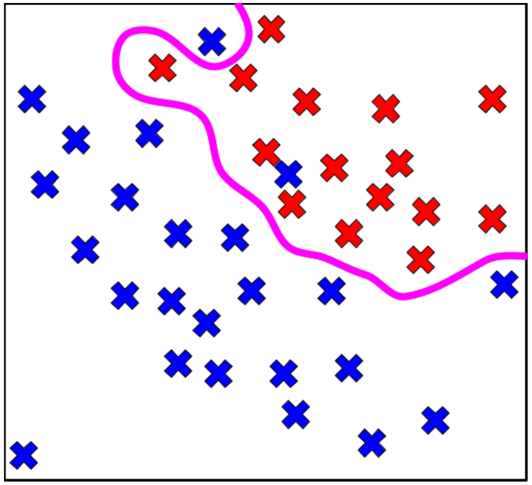
$p_\theta(y|x)$



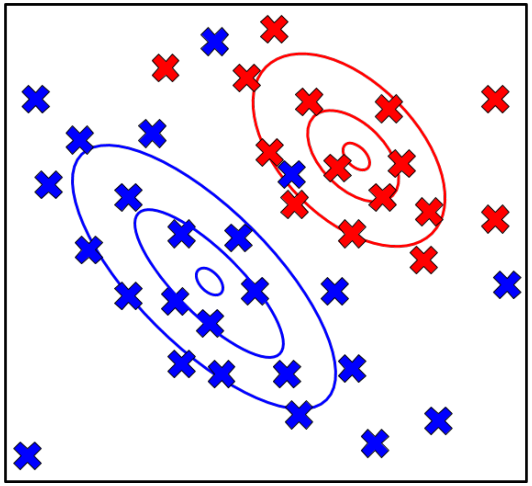
$p_\theta(x,y)$

Do we need generative modeling?





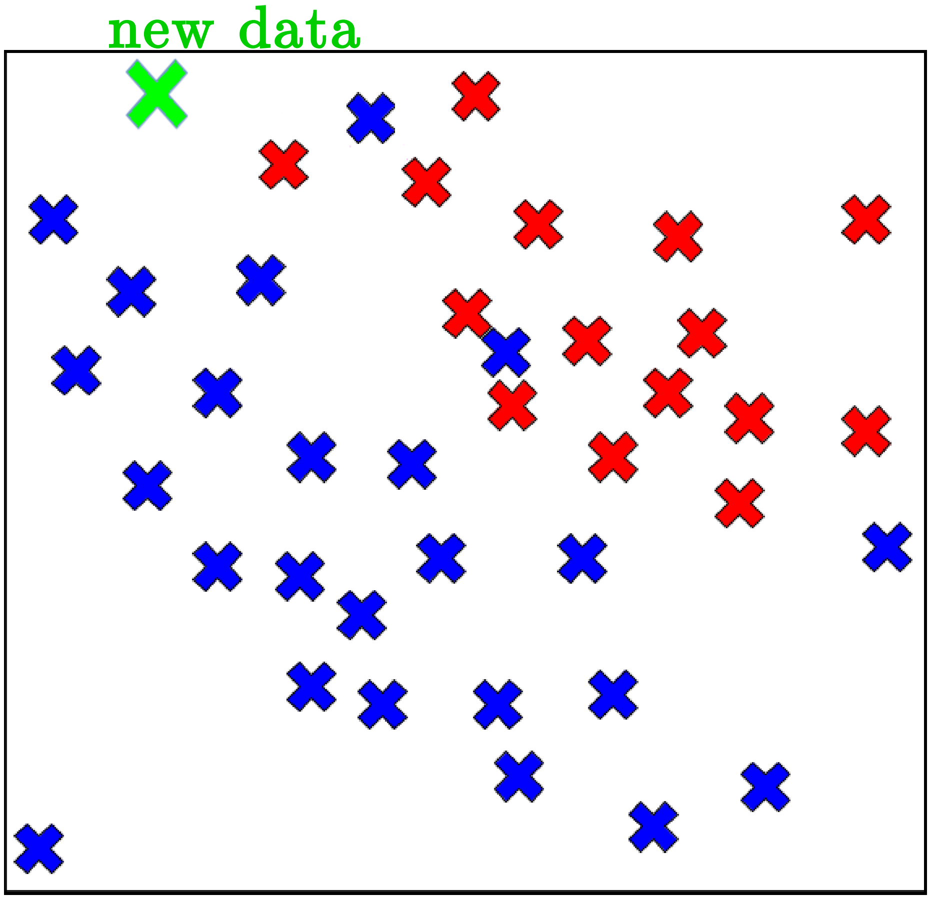
$p_\theta(y|x)$

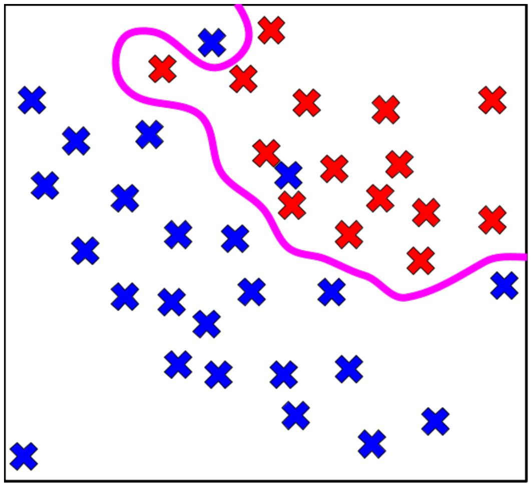


$p_\theta(x, y)$

High probability
of the **blue label**.
=
Highly probable
decision!

Do we need generative modeling?



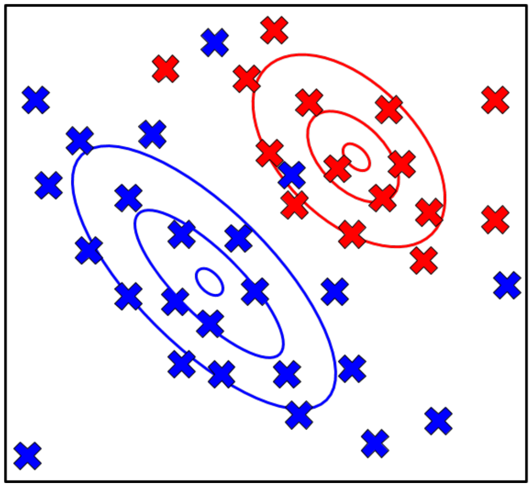


$p_\theta(y|x)$

High probability of the **blue label**.

$=$

Highly probable decision!



$p_\theta(x, y)$

High probability of the **blue label**.

x

Low probability of the **object**.

$=$

Uncertain decision!

Generative Modeling

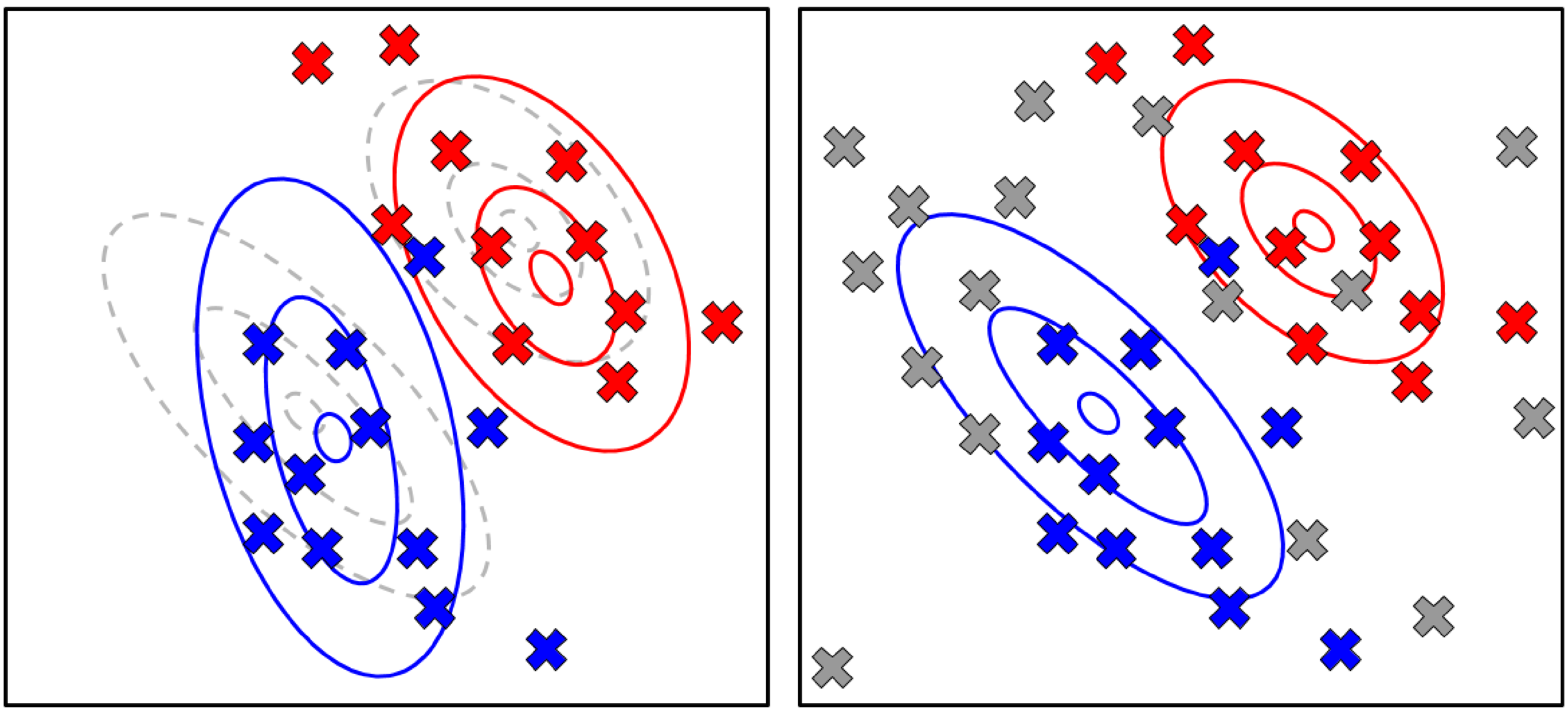
- Providing decision is not enough. *How to evaluate uncertainty? Distribution of y is only a part of the story.*

- Generalization problem. *Without knowing the distribution of* $\mathbf{x}$ *how we can generalize to new data?*

- Understanding the problem is crucial ("**What I cannot create, I do not understand**", Richard P. Feynman). *Properly modeling data is essential to make better decisions.*

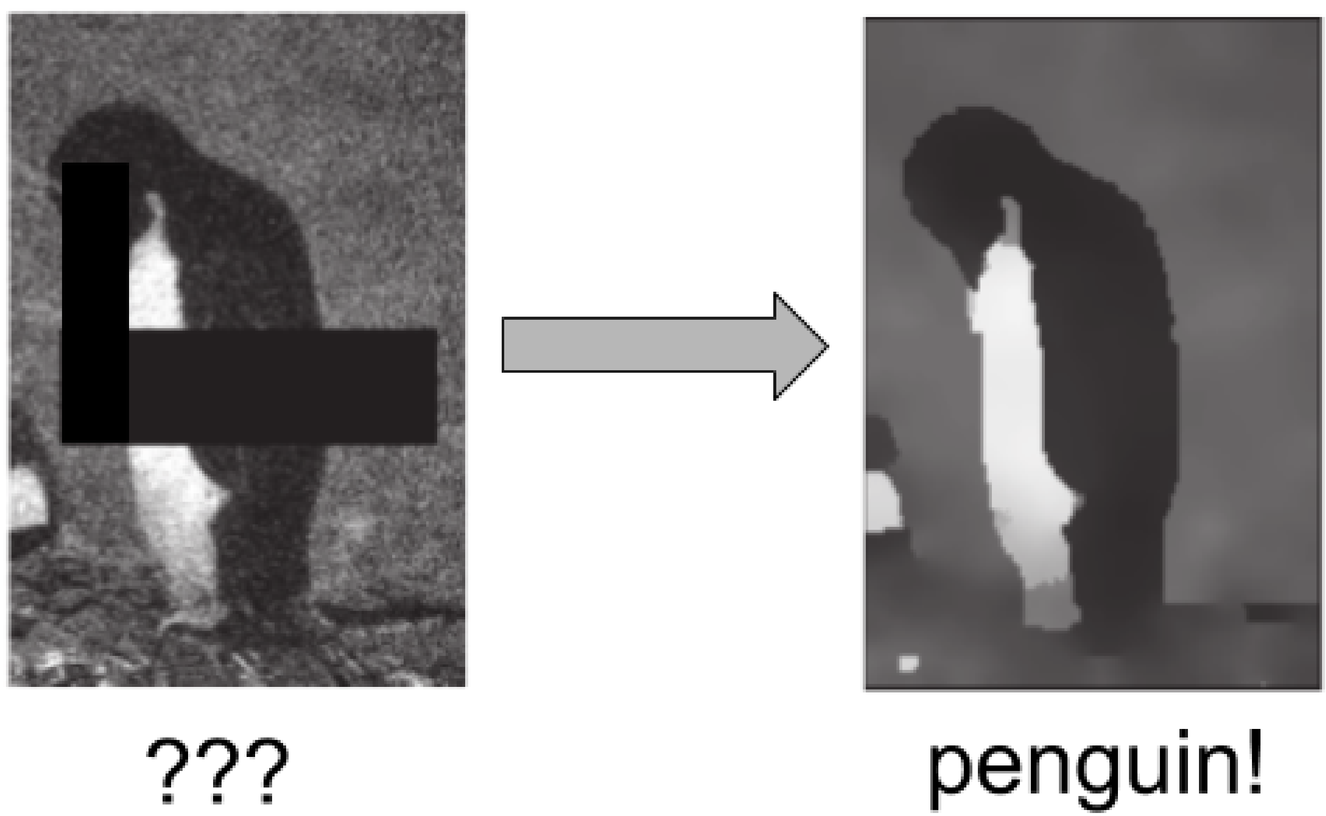
Generative Modeling

- Semi-supervised learning.
  Use unlabeled data to train a better classifier.



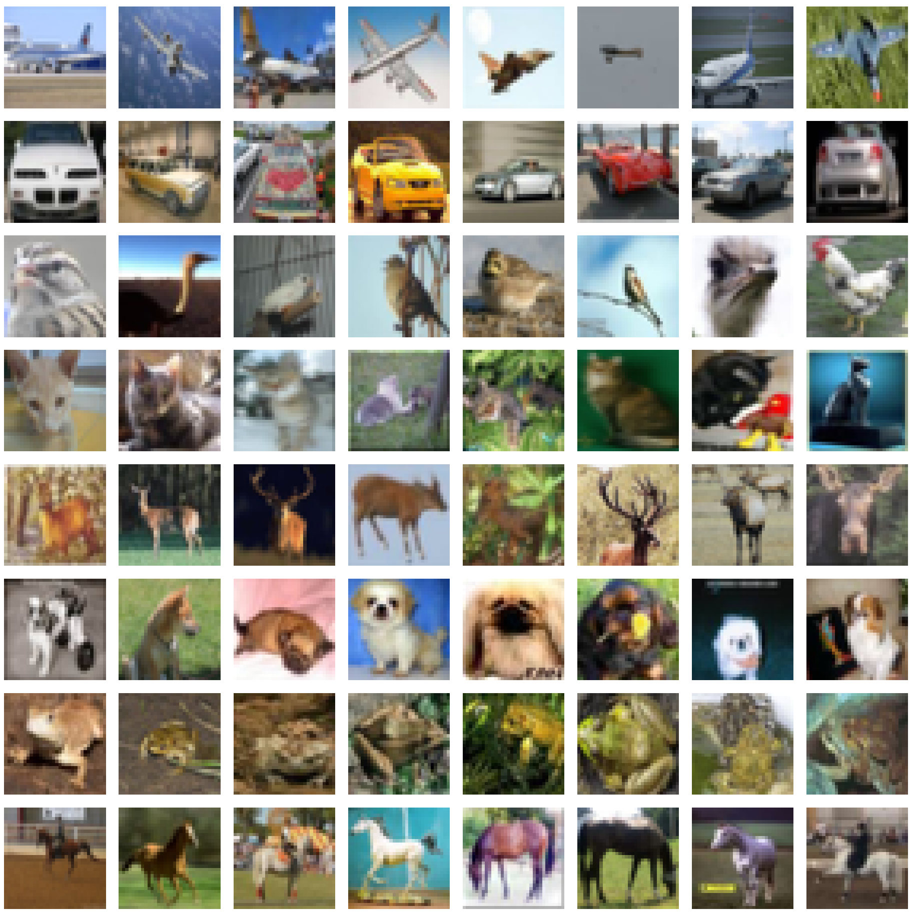
Generative Modeling

- Handling missing or distorted data.
  Reconstruct and/or denoise data.

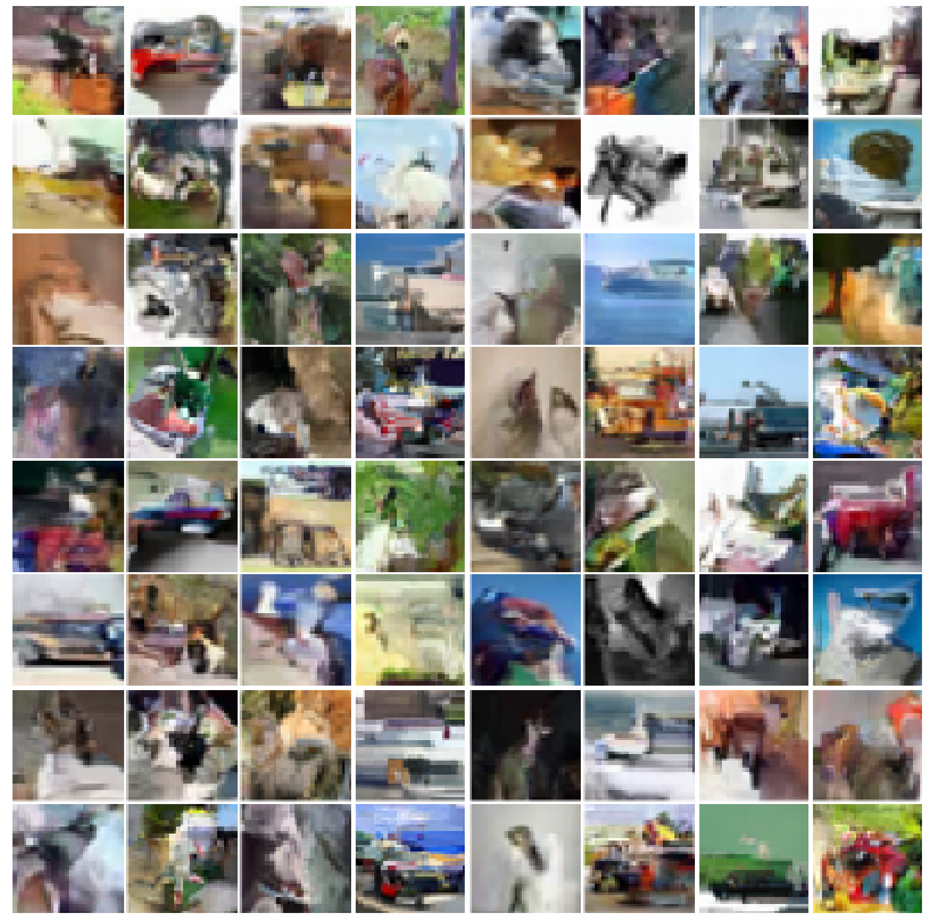


Generative Modeling

Image generation



Real



Generated

CHEN, Xi, et al. Variational lossy autoencoder. arXiv preprint arXiv:1611.02731, 2016.

Generative Modeling

Sequence generation

he had been unable to conceal the fact that there was a logical explanation for his inability to alter the fact that they were supposed to be on the other side of the house .
with a variety of pots strewn scattered across the vast expanse of the high ceiling , a vase of colorful flowers adorned the tops of the rose petals littered the floor and littered the floor .
atop the circular dais perched atop the gleaming marble columns began to emerge from atop the stone dais, perched atop the dais .

Generated

BOWMAN, Samuel R., et al. Generating sentences from a continuous space. arXiv preprint arXiv:1511.06349, 2015.

How to formulate a generative model?

Modeling in a high-dimensional space is difficult.

How to formulate a generative model?

Modeling in a high-dimensional space is difficult.

How to formulate a generative model?

Modeling in a high-dimensional space is difficult.

$\rightarrow$ modeling all dependencies among pixels.

$$p(x) = \prod_{d=1}^{c} \psi_c(x_c)$$

How to formulate a generative model?

Modeling in a high-dimensional space is difficult.

$\rightarrow$ modeling all dependencies among pixels.

$$p(x) = \prod_{d=1}^{c} \psi_c(x_c)$$

very inefficient!

How to formulate a generative model?

Modeling in a high-dimensional space is difficult.

$\rightarrow$ modeling all dependencies among pixels.

$$p(x) = \prod_{d=1}^{c} \psi_c(x_c)$$

very inefficient!

A possible **solution**? $\rightarrow$ **Models with latent variables**

Boltzmann distribution

A joint distribution of random variables:

$$p(\mathbf{x}) = \frac{1}{Z} \exp\{-E(\mathbf{x})\}$$

Boltzmann distribution

A joint distribution of random variables:

$$p(\mathbf{x}) = \frac{1}{Z} \exp\{-E(\mathbf{x})\}$$

Energy function

Boltzmann distribution

A joint distribution of random variables:

$$p(\mathbf{x}) = \frac{1}{Z}\exp\{-E(\mathbf{x})\}$$

Partition function

Energy function

$$Z = \sum_{\mathbf{x}} \exp\{-E(\mathbf{x})\}$$

Boltzmann distribution

A joint distribution of random variables:

$$p(\mathbf{x}) = \frac{1}{Z}\exp\{-E(\mathbf{x})\}$$

It is called the **Boltzmann** (or **Gibbs**) **distribution.**

Boltzmann distribution

A joint distribution of random variables:

$$p(\mathbf{x}) = \frac{1}{Z}\exp\{-E(\mathbf{x})\}$$

Advantages:

– Joint distribution is described by $E(\mathbf{x})$.
– Well-studied distribution in statistical physics.

Boltzmann distribution

A joint distribution of random variables:

$$p(\mathbf{x}) = \frac{1}{Z} \exp\{-E(\mathbf{x})\}$$

Advantages:

– Joint distribution is described by $E(\mathbf{x})$.
– Well-studied distribution in statistical physics.

Drawbacks:

– Calculation of partition function requires enumeration of all $\mathbf{x}$, *e.g.*, if $\mathbf{x}$'s are binary $\rightarrow 2^n$ combinations.

Boltzmann Machines

Let us consider the following energy function:

$$E(\mathbf{x}) = -\mathbf{x}^\top \mathbf{W}\mathbf{x} - \mathbf{b}^\top \mathbf{x}$$

where $\mathbf{x}$ are binary.

Boltzmann Machines

Let us consider the following energy function:

$$E(\mathbf{x}) = -\mathbf{x}^\top \mathbf{W}\mathbf{x} - \mathbf{b}^\top \mathbf{x}$$

where $\mathbf{x}$ are binary.

first-order term (models "*prior*")

Boltzmann Machines

Let us consider the following energy function:

$$E(\mathbf{x}) = -\mathbf{x}^\top \mathbf{W}\mathbf{x} - \mathbf{b}^\top \mathbf{x}$$

where $\mathbf{x}$ are binary.

second-order term
(models correlations)

first-order term
(models "*prior*")

Boltzmann Machines

Let us consider the following energy function:

$$E(\mathbf{x}) = -\mathbf{x}^\top \mathbf{W} \mathbf{x} - \mathbf{b}^\top \mathbf{x}$$

where $\mathbf{x}$ are binary.

second-order term
(models correlations)

first-order term
(models "*prior*")

Very complicated...

Can we do better?

Boltzmann Machines

Let us consider the following energy function:

$$E(\mathbf{x}) = -\mathbf{x}^\top \mathbf{W}\mathbf{x} - \mathbf{b}^\top \mathbf{x}$$

where $\mathbf{x}$ are binary.

second-order term
(models correlations)

first-order term
(models "*prior*")

Very complicated...

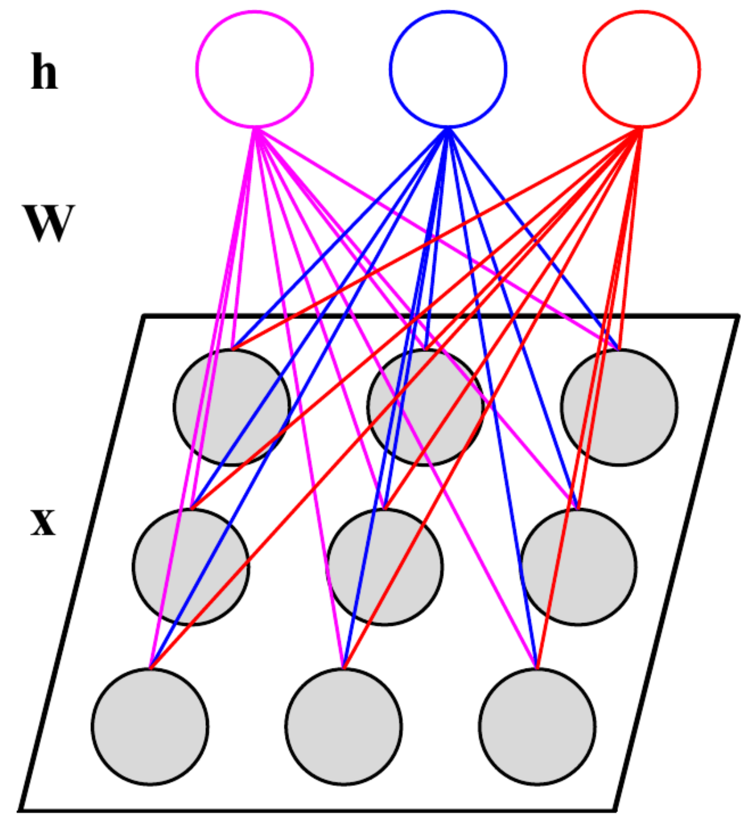
Can we do better? Latent variables!

Restricted Boltzmann Machines

Let us consider the following energy function:

$$E(\mathbf{x}, \mathbf{h}, \theta) = -\mathbf{x}^\top \mathbf{W}\mathbf{h} - \mathbf{b}^\top \mathbf{x} - \mathbf{c}^\top \mathbf{h}$$

where $\mathbf{x}$ and $\mathbf{h}$ are binary.

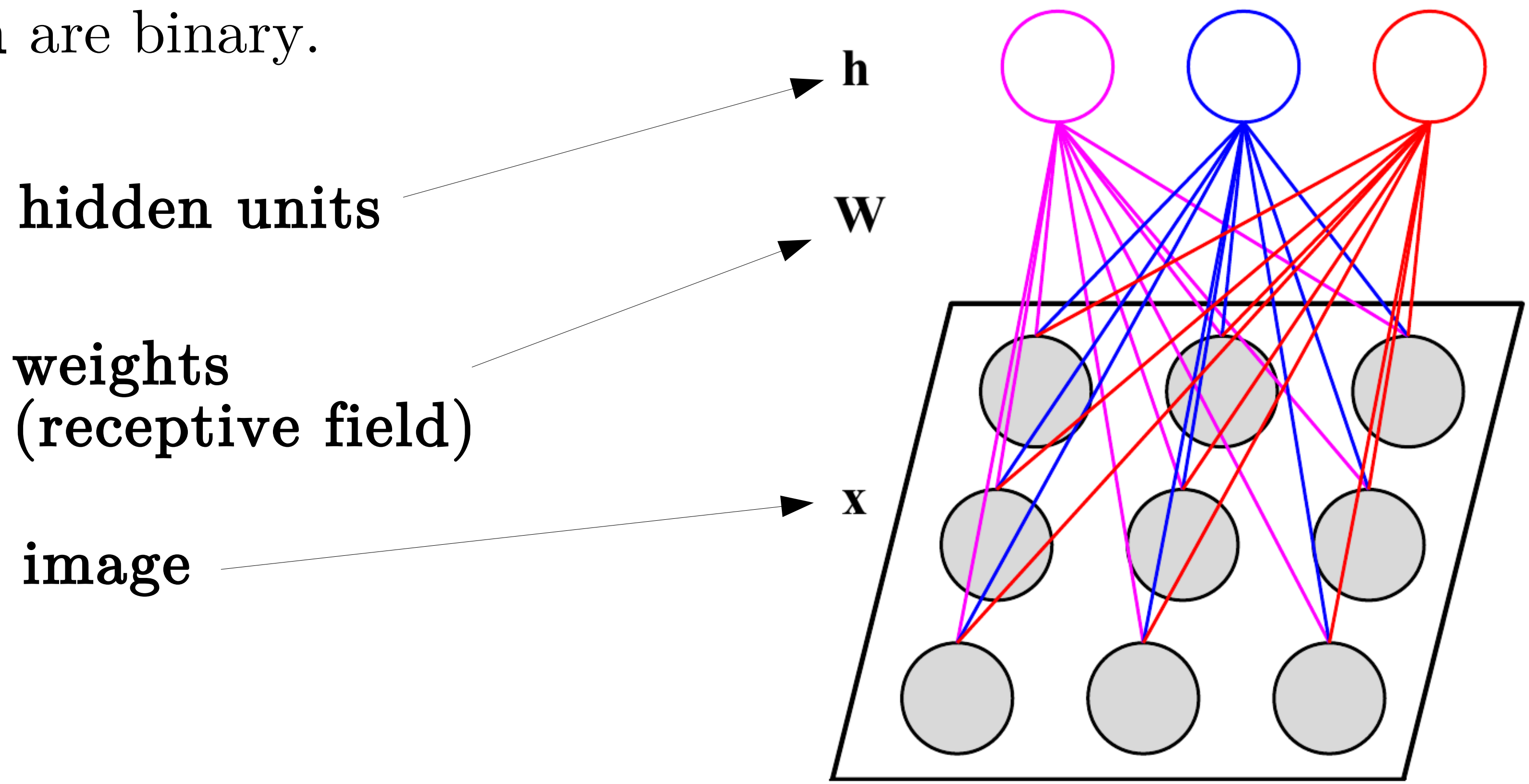


Restricted Boltzmann Machines

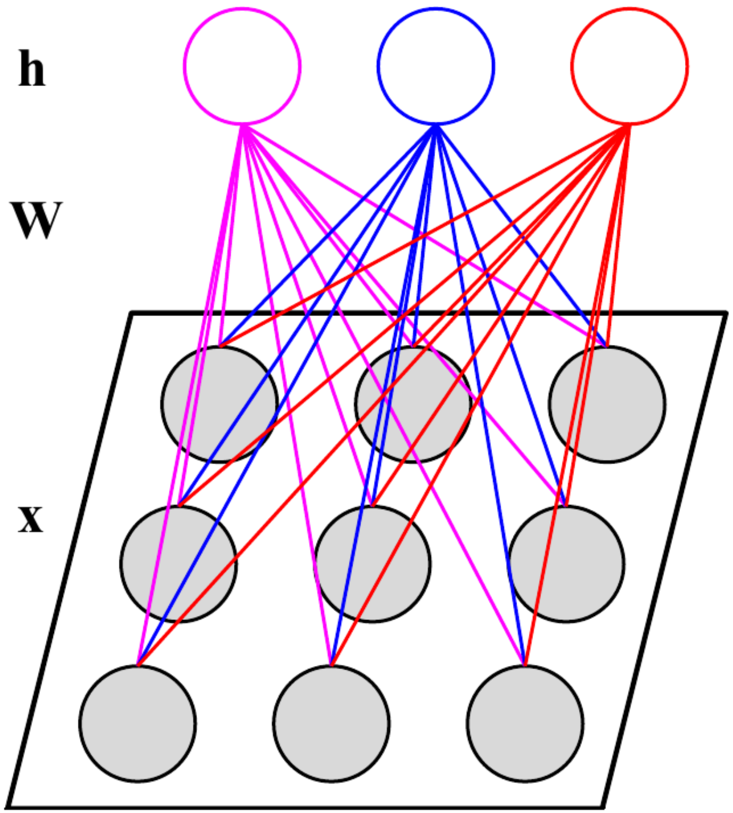
Let us consider the following energy function:

$$E(\mathbf{x}, \mathbf{h}, \theta) = -\mathbf{x}^\top \mathbf{W}\mathbf{h} - \mathbf{b}^\top \mathbf{x} - \mathbf{c}^\top \mathbf{h}$$

where $\mathbf{x}$ and $\mathbf{h}$ are binary.



Restricted Boltzmann Machines



Conditional dependencies:

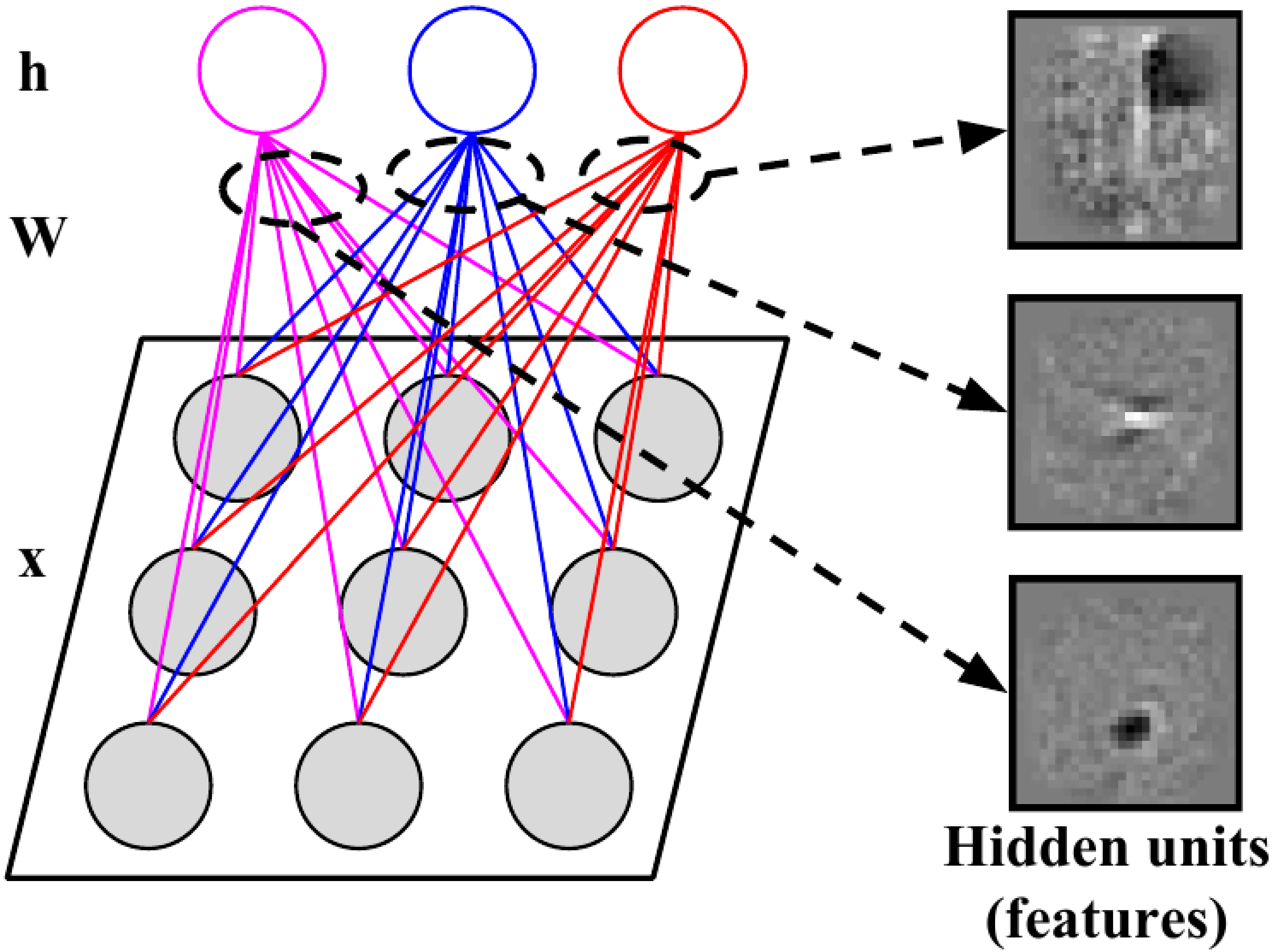
$$p(\mathbf{h}|\mathbf{x},\theta) = \prod_j p(h_j|\mathbf{x},\theta)$$

$$p(\mathbf{x}|\mathbf{h},\theta) = \prod_i p(x_i|\mathbf{h},\theta)$$

where

$$p(h_j = 1|\mathbf{x},\theta) = \frac{1}{1+\exp(-(\mathbf{W}_{\cdot j}^\top)\mathbf{x} - c_j)}$$

$$p(x_i = 1|\mathbf{h},\theta) = \frac{1}{1+\exp(-\mathbf{W}_{\cdot i}\mathbf{h} - b_j)}$$



$$P(h_j = 1|\mathbf{x}, \theta) = \frac{1}{1 - \exp(-(\mathbf{W}_{\cdot j})^\top \mathbf{x} - c_j)}$$

RBM: Training

We train RBMs using the **log-likelihood function**:

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{n=1}^{N} \log p(\mathbf{x}_n | \theta)$$

where

$$\begin{aligned} p(\mathbf{x}_n | \theta) &= \frac{1}{Z} \exp(-FE(\mathbf{x}_n)) \\ &= \frac{1}{Z} \exp\left(\mathbf{b}^\top \mathbf{x}_n + \sum_j \log(1 + \exp\{(\mathbf{W}_{\cdot j})^\top \mathbf{x}_n + c_j\})\right) \end{aligned}$$

RBM: Training

We train RBMs using the **log-likelihood function**:

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{n=1}^{N} \log p(\mathbf{x}_n|\theta)$$

where

Free energy

$$\begin{aligned} p(\mathbf{x}_n|\theta) &= \frac{1}{Z} \exp(-FE(\mathbf{x}_n)) \\ &= \frac{1}{Z} \exp\left(\mathbf{b}^\top \mathbf{x}_n + \sum_j \underbrace{\log(1 + \exp\{(\mathbf{W}_{\cdot j})^\top \mathbf{x}_n + c_j\})}_{\textbf{softplus}}\right) \end{aligned}$$

RBM: Training

Let us calculate gradient of the LL wrt parameters:

$$
\begin{aligned}
\frac{\partial \log p(\mathbf{x}_n|\theta)}{\partial \theta} &= -\frac{\partial}{\partial \theta} FE(\mathbf{x}_n) - \frac{\partial}{\partial \theta} \log Z \\
&= -\sum_{\mathbf{h}} p(\mathbf{h}|\mathbf{x}, \theta) \frac{\partial}{\partial \theta} E(\mathbf{x}_n, \mathbf{h}|\theta) + \sum_{\widetilde{\mathbf{x}}, \mathbf{h}} p(\widetilde{\mathbf{x}}, \mathbf{h}|\theta) \frac{\partial}{\partial \theta} E(\widetilde{\mathbf{x}}, \mathbf{h}|\theta)
\end{aligned}
$$

RBM: Training

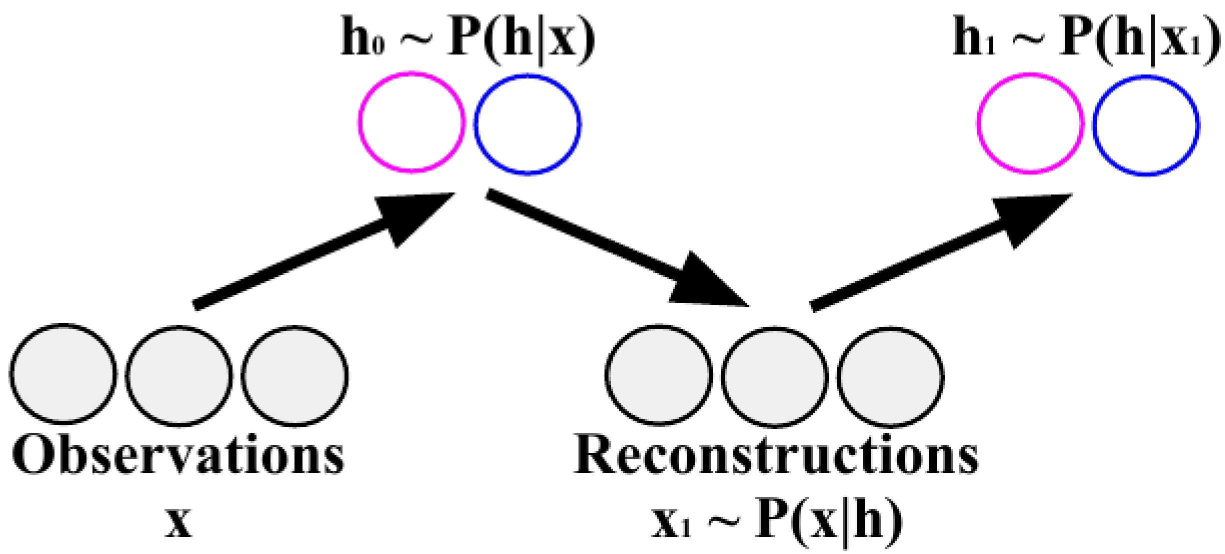
Let us calculate gradient of the LL wrt parameters:

$$\frac{\partial \log p(\mathbf{x}_n|\theta)}{\partial \theta} = -\frac{\partial}{\partial \theta} FE(\mathbf{x}_n) - \frac{\partial}{\partial \theta} \log Z$$

$$= \underbrace{-\sum_{\mathbf{h}} p(\mathbf{h}|\mathbf{x}, \theta) \frac{\partial}{\partial \theta} E(\mathbf{x}_n, \mathbf{h}|\theta)}_{\text{easy}} + \underbrace{\sum_{\widetilde{\mathbf{x}}, \mathbf{h}} p(\widetilde{\mathbf{x}}, \mathbf{h}|\theta) \frac{\partial}{\partial \theta} E(\widetilde{\mathbf{x}}, \mathbf{h}|\theta)}_{\text{difficult}}$$

It requires application of an approximate inference (*e.g.*, MCMC).

RBM: Contrastive Divergence

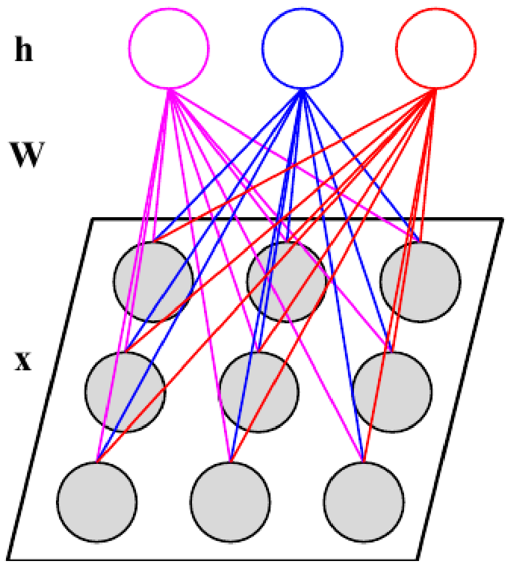


Apply k steps of **Gibbs sampler** and approximate the gradient as follows:

$$\frac{\partial \log p(\mathbf{x}_n|\theta)}{\partial \theta} = -\frac{\partial}{\partial \theta} E(\mathbf{x}_n, \mathbf{h}_0|\theta) + \frac{\partial}{\partial \theta} E(\mathbf{x}_k, \mathbf{h}_k|\theta)$$

Hinton, G. E. (2002), *Training products of experts by minimizing contrastive divergence*, Neural Computation, 14(8), 1771-1800

RBM with Gaussian inputs



- **Observables**: $\mathbf{x} \in \mathbb{R}^D$
- **Hiddens**: $\mathbf{h} \in \{0,1\}^M$
- Good for modelling **natural images**.

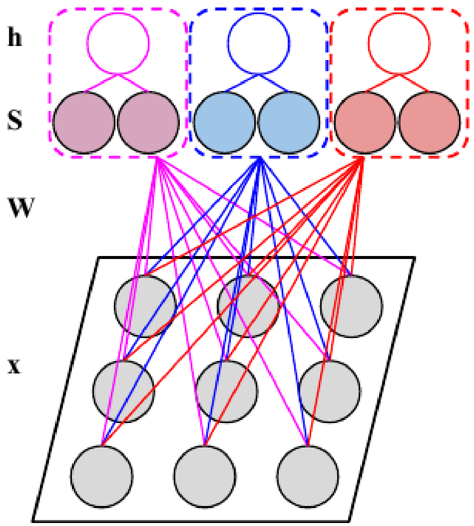
Energy for a joint configuration $(\mathbf{x}, \mathbf{h})$:

$$E(\mathbf{x}, \mathbf{h}|\theta) = \sum_i \sum_j W_{ij} h_j \frac{x_i}{\sigma_i} + \sum_i \frac{(x_i - b_i)}{2\sigma_i^2} + \sum_j c_j h_j$$

Then:

$$P(\mathbf{x}|\mathbf{h}, \theta) = \prod_i \mathcal{N}(b_i + \sum_j W_{ij} h_j, \sigma_i^2)$$

Subspace RBM



- **Observables**: $\mathbf{x} \in \{0,1\}^D$, **hidden** variables: $\mathbf{h} \in \{0,1\}^M$, $\mathbf{S} \in \{0,1\}^{M \times K}$
- Aim: **features invariant to transformations**.

Energy for a joint configuration $(\mathbf{x}, \mathbf{h}, \mathbf{S})$:

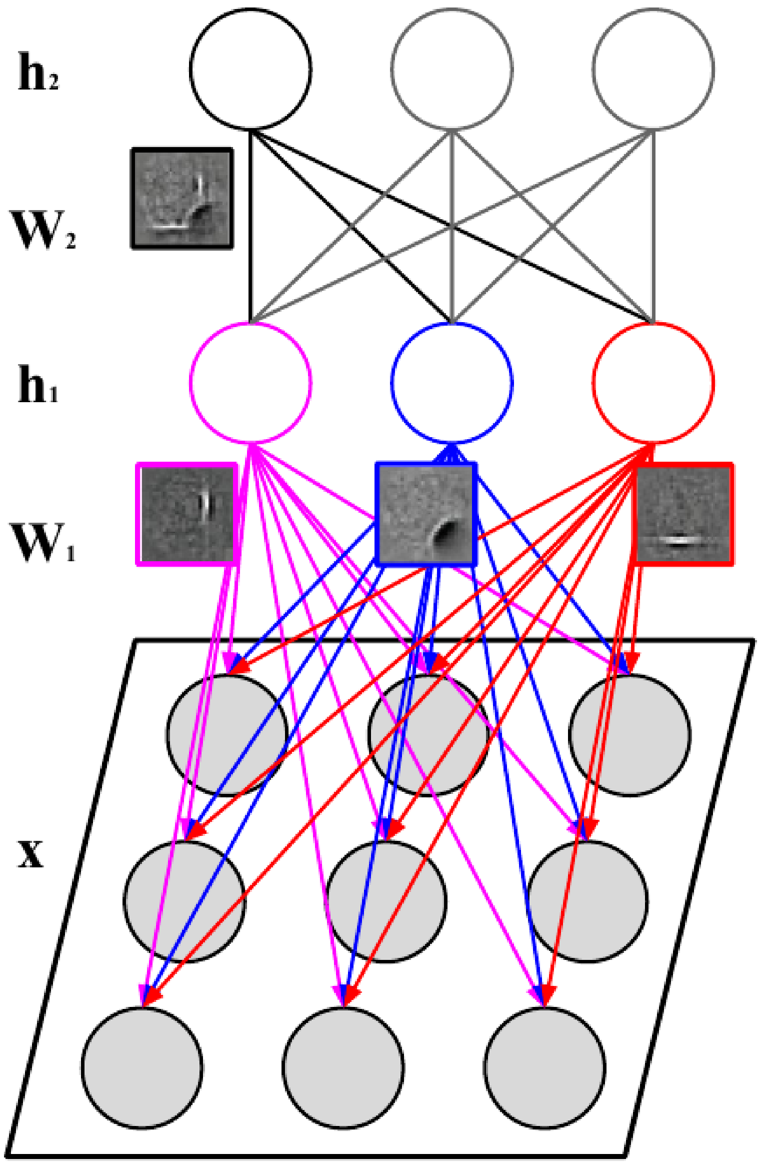
$$E(\mathbf{x}, \mathbf{h}, \mathbf{S}|\theta) = \sum_{i,j,k} W_{ijk} x_i h_j S_{jk} + \mathbf{b}^\top \mathbf{x} + \mathbf{c}^\top \mathbf{h} + \sum_j h_j \sum_k A_{jk} S_{jk}$$

Then:

$$P(\mathbf{x}) \propto \exp(\mathbf{b}^\top \mathbf{x}) \prod_j \left[2^K + \exp(c_j) \prod_k \left(1 + \exp\Big(\sum_i W_{ijk} x_i + A_{jk} \Big) \right) \right]$$

Tomczak, J. M., & Gonczarek, A. (2017). *Learning invariant features using Subspace Restricted Boltzmann Machine*. Neural Processing Letters, 45(1), 173-182.

Deep Belief Network



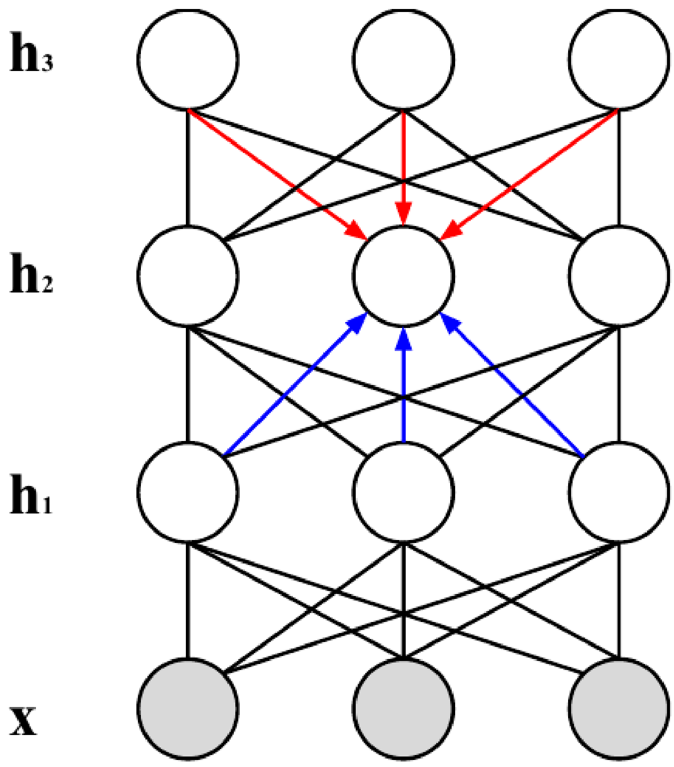
- RBM (or its extension) can be treated as a **building block**.
- Stacking RBMs leads to a **deep belief network**:

$$P(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2) = \underbrace{P(\mathbf{x}|\mathbf{h}_1)}_{\text{from RBM}} \underbrace{P(\mathbf{h}_1, \mathbf{h}_2)}_{\text{RBM}}$$

Deep Boltzmann Machines

Energy function:

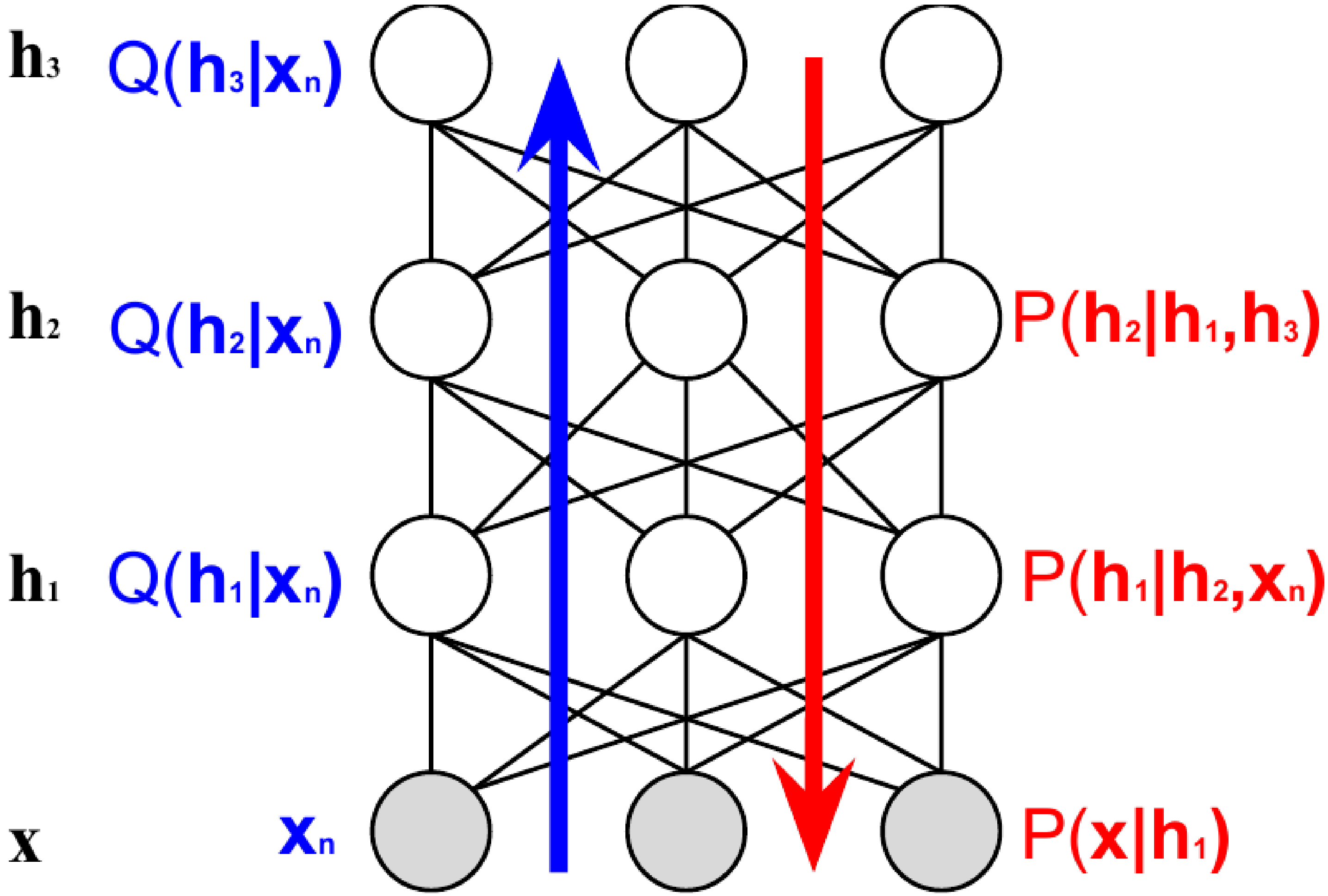
$$E(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2|\theta) = \mathbf{x}^\top \mathbf{W}_1 \mathbf{h}_1 + \mathbf{h}_1^\top \mathbf{W}_2 \mathbf{h}_2 + \mathbf{h}_2^\top \mathbf{W}_3 \mathbf{h}_3$$



- Joint distribution – **Boltzmann (Gibbs) distribution**.
- All connections are **undirected**.
- Two types of relationships: (**top-down** and **bottom-up**):

$$P(h_2^{(k)} = 1|\mathbf{h}_1, \mathbf{h}_3) = \mathrm{sigm}\Big(\sum_j W_1^{(jk)} h_1^{(j)} + \sum_l W_3^{(kl)} h_3^{(l)}\Big)$$

Training DBM

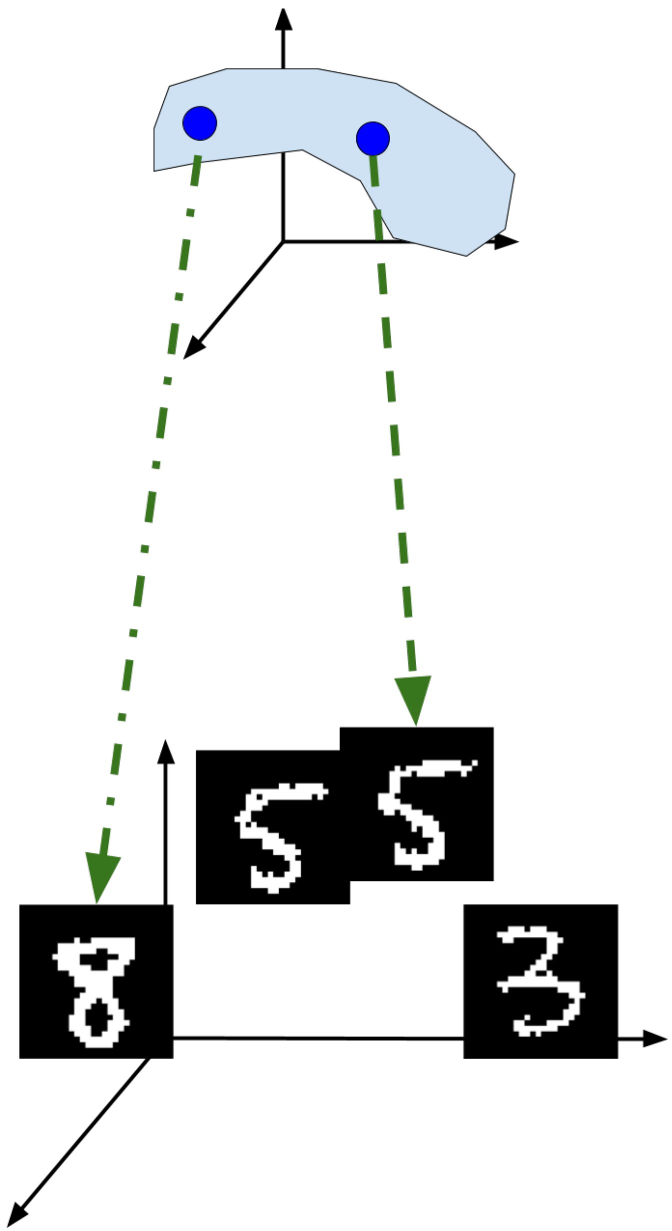


Calculating gradients is **intractable**, thus, **variational methods** (**mean-field**) and **sampling methods** are utilized.

Latent Variable Models

- Latent variable model:

$$p(\mathbf{x}) = \int p_\theta(\mathbf{x}|\mathbf{z})\; p_\lambda(\mathbf{z})\; \mathrm{d}\mathbf{z}$$



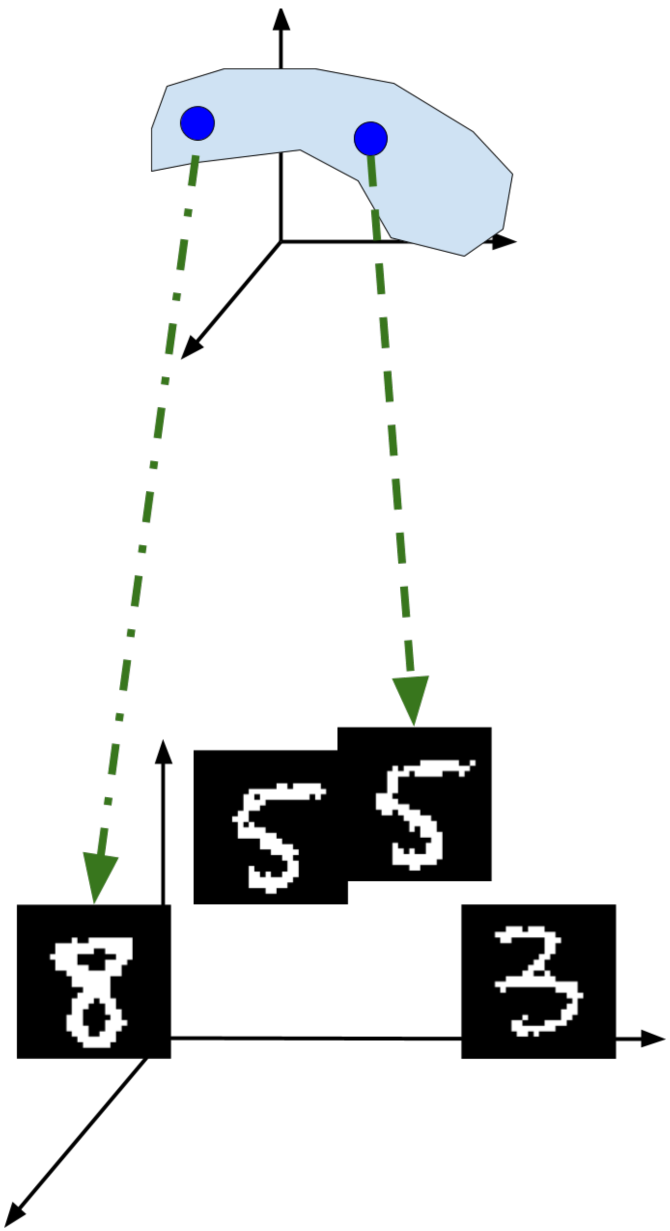
Latent Variable Models

- **Latent variable model**:

$$p(\mathbf{x}) = \int p_\theta(\mathbf{x}|\mathbf{z})\; p_\lambda(\mathbf{z})\; \mathrm{d}\mathbf{z}$$

First sample $\mathbf{z}$.

Second, sample $\mathbf{x}$ for given $\mathbf{z}$.



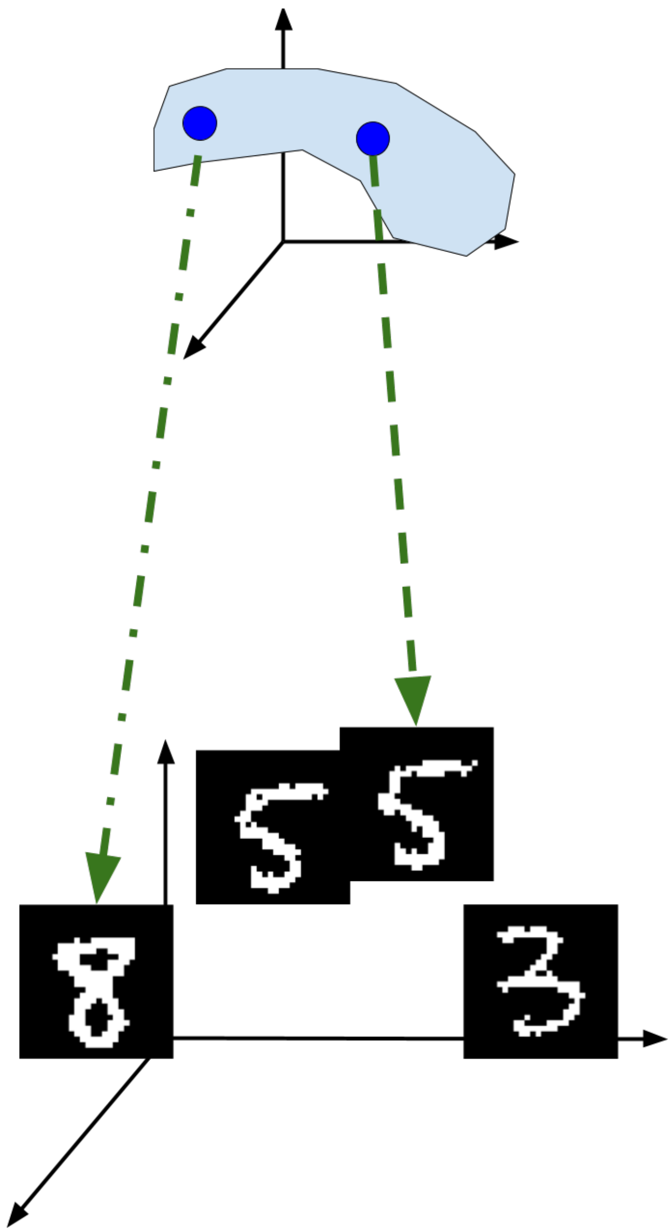
Latent Variable Models

- **Latent variable model**:

$$p(\mathbf{x}) = \int \underbrace{p_\theta(\mathbf{x}|\mathbf{z})}_{} \, p_\lambda(\mathbf{z}) \, \mathrm{d}\mathbf{z}$$

First sample $\mathbf{z}$.

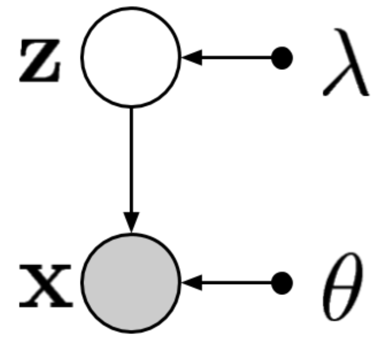
Second, sample $\mathbf{x}$ for given $\mathbf{z}$.



Latent Variable Models

- Latent variable model:

$$p(\mathbf{x}) = \int p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})\ \mathrm{d}\mathbf{z}$$

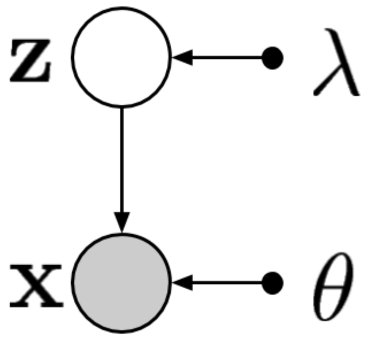


- If $p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{W}\mathbf{z} + \mathbf{b}, \Psi)$ and $p_\lambda(\mathbf{z}) = \mathcal{N}(\mu_0, \Sigma_0)$, then $\rightarrow$ **Factor Analysis**.

- What if we take a **non-linear transformation** of z? $\rightarrow$ ***an infinite mixture of Gaussians.***

Latent Variable Models

- Latent variable model:



$$p(\mathbf{x}) = \int p_\theta(\mathbf{x}|\mathbf{z})\; p_\lambda(\mathbf{z})\; \mathrm{d}\mathbf{z}$$

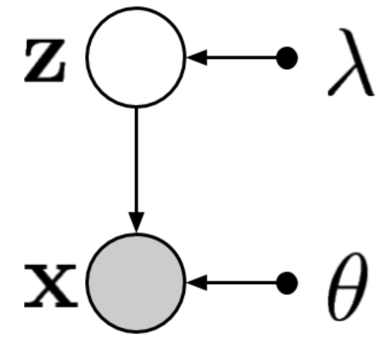
- If $p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{W}\mathbf{z} + \mathbf{b}, \Psi)$ and $p_\lambda(\mathbf{z}) = \mathcal{N}(\mu_0, \Sigma_0)$, then → **Factor Analysis**.

Convenient but limiting!

- What if we take a **non-linear transformation** of z? → *an infinite mixture of Gaussians.*

Latent Variable Models

- Latent variable model:



$$p(\mathbf{x}) = \int p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})\ \mathrm{d}\mathbf{z}$$

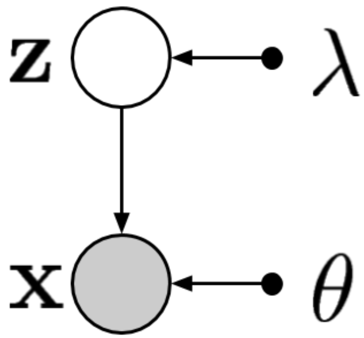
- If $p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{Wz}+\mathbf{b},\Psi)$ and $p_\lambda(\mathbf{z}) = \mathcal{N}(\mu_0,\Sigma_0)$, then $\rightarrow$ **Factor Analysis**.

- What if we take a **non-linear transformation** of z? $\rightarrow$ *an infinite mixture of Gaussians.*

Latent Variable Models

- **Latent variable model**:

$$p(\mathbf{x}) = \int p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})\ \mathrm{d}\mathbf{z}$$

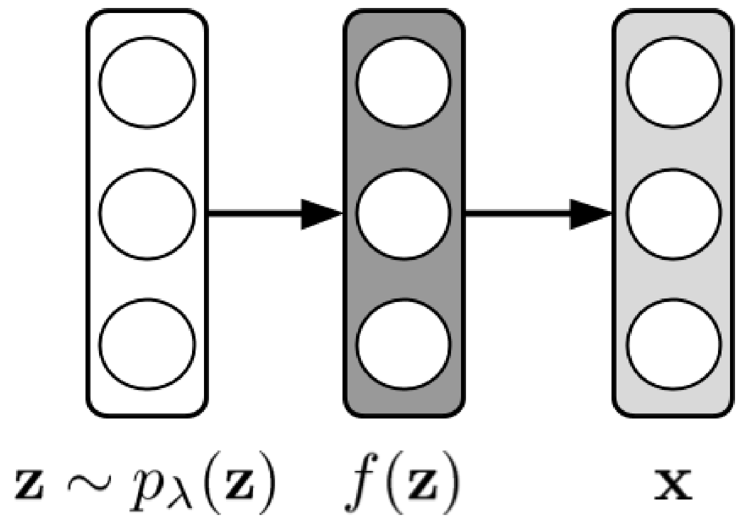


- If $p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{W}\mathbf{z} + \mathbf{b}, \Psi)$ and $p_\lambda(\mathbf{z}) = \mathcal{N}(\mu_0, \Sigma_0)$, then $\rightarrow$ **Factor Analysis**.

- What if we take a **non-linear transformation** of z? $\rightarrow$ ***an infinite mixture of Gaussians.***

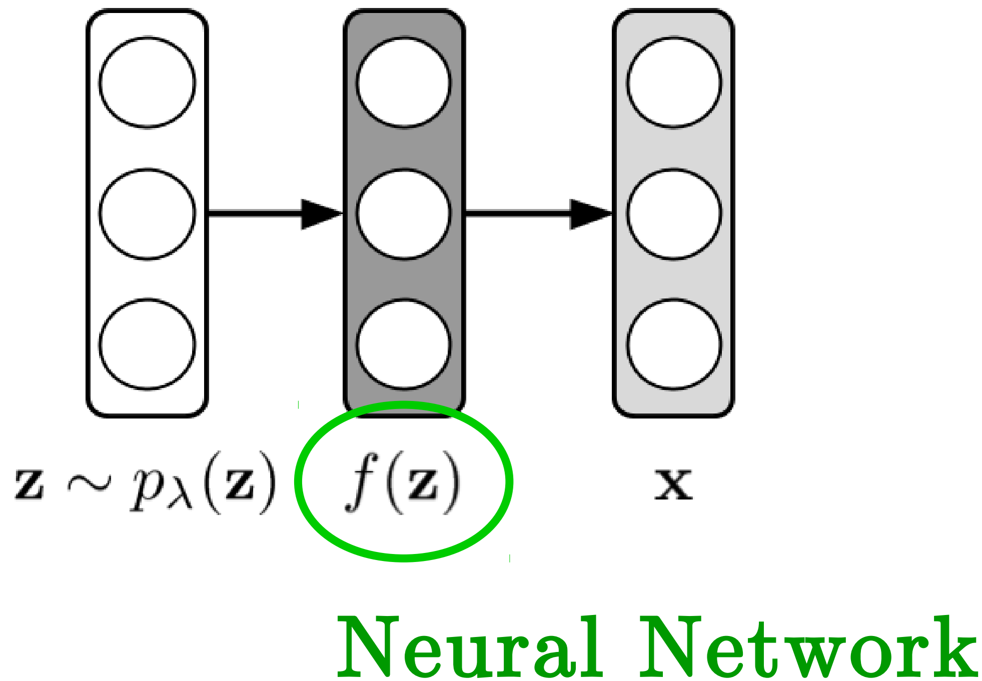
Neural network

Deep Generative Models (DGM): Density Network



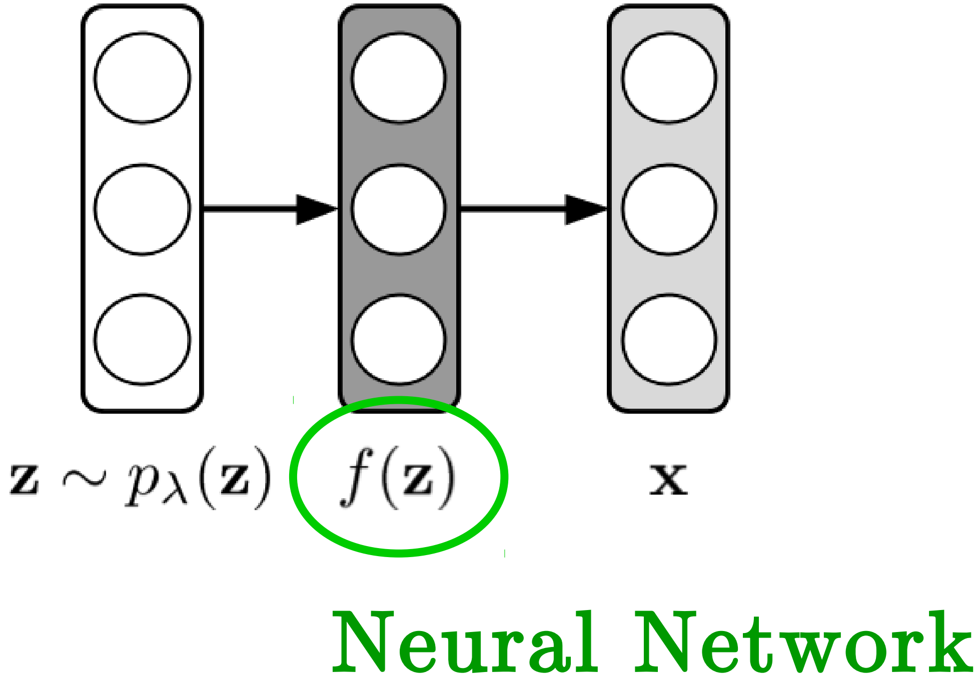
MacKay, D. J., & Gibbs, M. N. (1999). *Density networks*. Statistics and neural networks: advances at the interface. Oxford University Press, Oxford, 129-144.

DGM: Density Network



MacKay, D. J., & Gibbs, M. N. (1999). *Density networks*. Statistics and neural networks: advances at the interface. Oxford University Press, Oxford, 129-144.

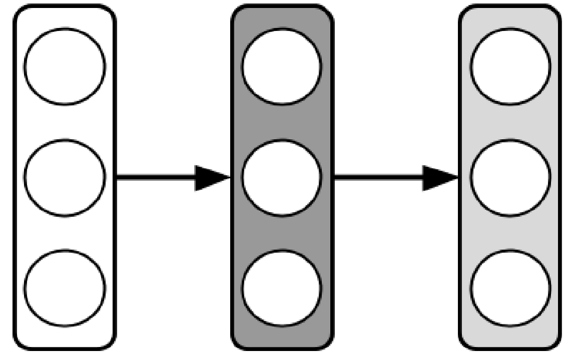
DGM: Density Network



How to train this model?!

MacKay, D. J., & Gibbs, M. N. (1999). *Density networks*. Statistics and neural networks: advances at the interface. Oxford University Press, Oxford, 129-144.

DGM: Density Network



- MC approximation:

$$\log p(\mathbf{x}) = \log \int p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})\ \mathrm{d}\mathbf{z}$$

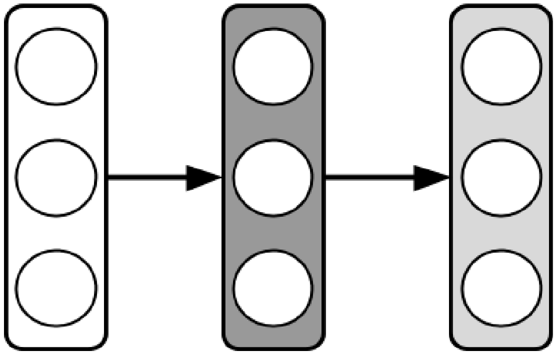
$$\approx \log \frac{1}{S} \sum_{s=1}^{S} \exp\Big(\log p_\theta(\mathbf{x}|\mathbf{z}_s)\Big)$$

where:

$$\mathbf{z}_s \sim p_\lambda(\mathbf{z})$$

MacKay, D. J., & Gibbs, M. N. (1999). *Density networks*. Statistics and neural networks: advances at the interface. Oxford University Press, Oxford, 129-144.

DGM: Density Network



- MC approximation:

$$\log p(\mathbf{x}) = \log \int p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})\ \mathrm{d}\mathbf{z}$$

$$\approx \log \frac{1}{S} \sum_{s=1}^{S} \exp\Big(\log p_\theta(\mathbf{x}|\mathbf{z}_s)\Big)$$

where:

$$\mathbf{z}_s \sim p_\lambda(\mathbf{z})$$

Sample $\mathbf{z}$ many times, apply log-sum-exp trick and maximize log-likelihood.

MacKay, D. J., & Gibbs, M. N. (1999). *Density networks*. Statistics and neural networks: advances at the interface. Oxford University Press, Oxford, 129-144.

DGM: Density Network

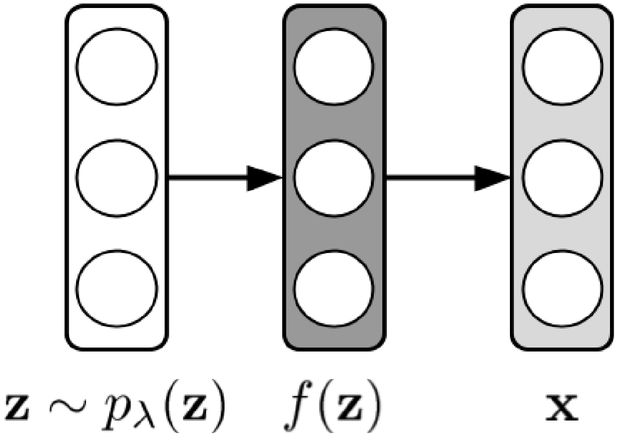
- MC approximation:

$$\log p(\mathbf{x}) = \log \int p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})\ \mathrm{d}\mathbf{z}$$

$$\approx \log \frac{1}{S} \sum_{s=1}^{S} \exp\Big(\log p_\theta(\mathbf{x}|\mathbf{z}_s)\Big)$$

where:

$$\mathbf{z}_s \sim p_\lambda(\mathbf{z})$$



Sample $\mathbf{z}$ many times, apply log-sum-exp trick and maximize log-likelihood.

It scales badly in high dimensional cases!

MacKay, D. J., & Gibbs, M. N. (1999). *Density networks*. Statistics and neural networks: advances at the interface. Oxford University Press, Oxford, 129-144.

DGM: Density Network

PROS

Log-**likelihood** approach

Easy **sampling**

Training using **gradient-based methods**

CONS

Requires **explicit** models

Fails in **high dim. cases**

MacKay, D. J., & Gibbs, M. N. (1999). *Density networks*. Statistics and neural networks: advances at the interface. Oxford University Press, Oxford, 129-144.

DGM: Density Network

PROS

Log-**likelihood** approach

Easy **sampling**

Training using **gradient-based methods**

CONS

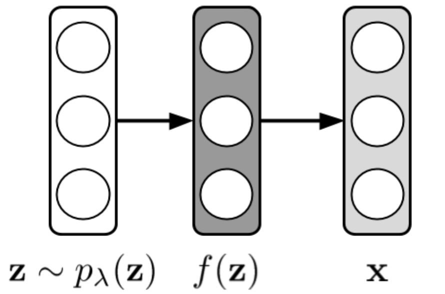
Requires **explicit** models

Fails in **high dim. cases**

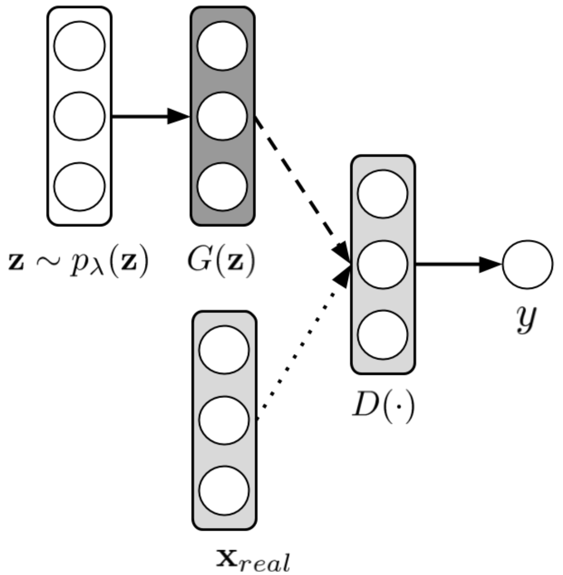
Can we do better?

MacKay, D. J., & Gibbs, M. N. (1999). *Density networks*. Statistics and neural networks: advances at the interface. Oxford University Press, Oxford, 129-144.

DGM: so far we have

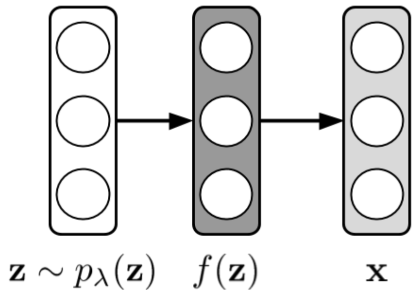


Density Network

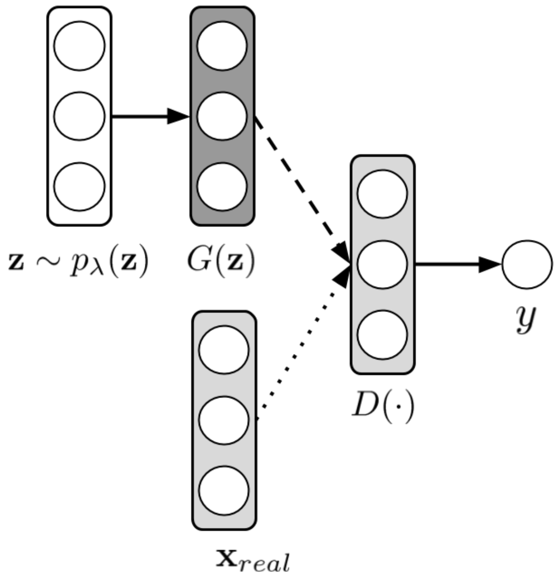


Generative Adversarial Net

DGM: so far we have



Density Network

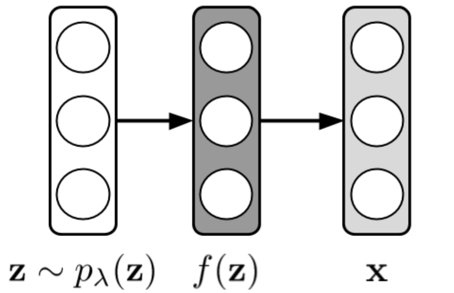


Generative Adversarial Net

Works only for low dim. cases...

Inefficient training...

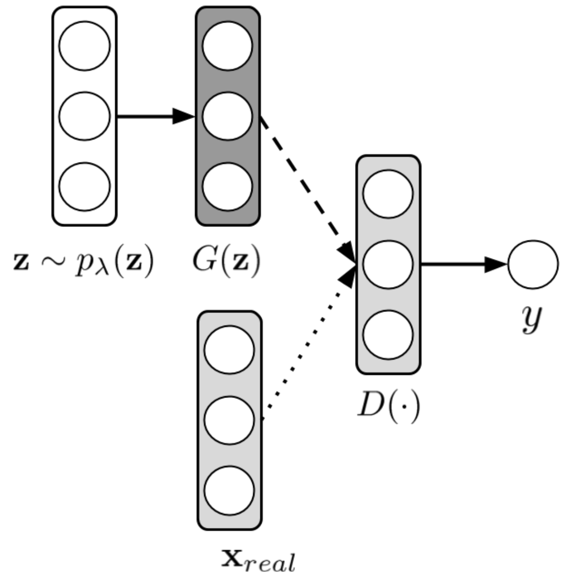
DGM: so far we have



Density Network

Works only for low dim. cases...

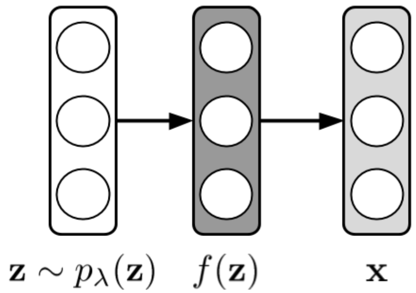
Inefficient training...



Generative Adversarial Net

Works for high dim. cases!

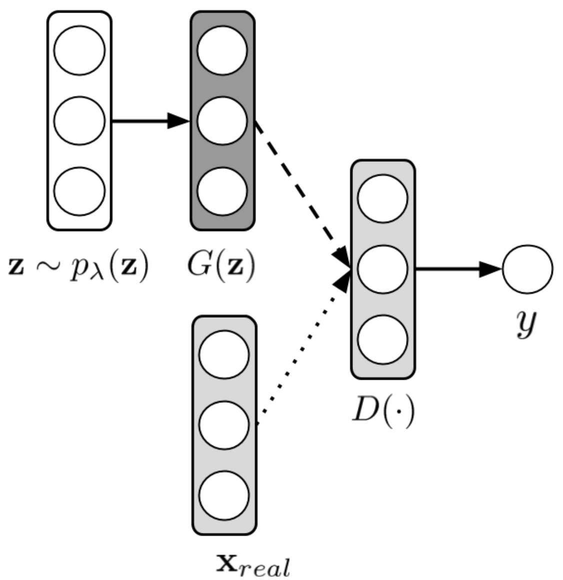
DGM: so far we have



Density Network

Works only for low dim. cases...

Inefficient training...



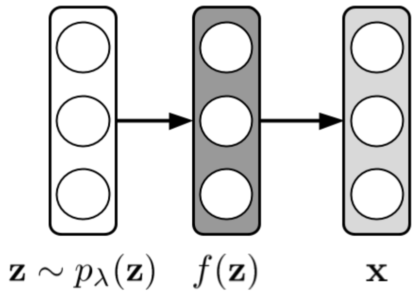
Generative Adversarial Net

Works for high dim. cases!

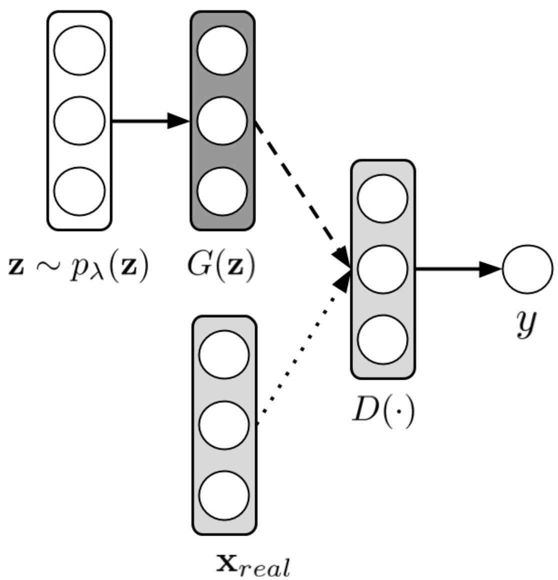
Doesn't train a distribution...

Unstable training...

DGM: so far we have



Density Network

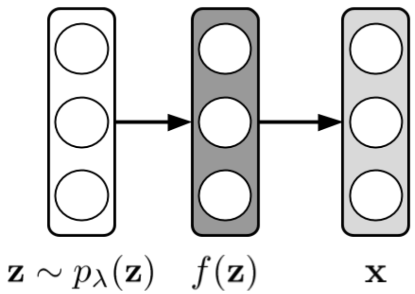


Generative Adversarial Net

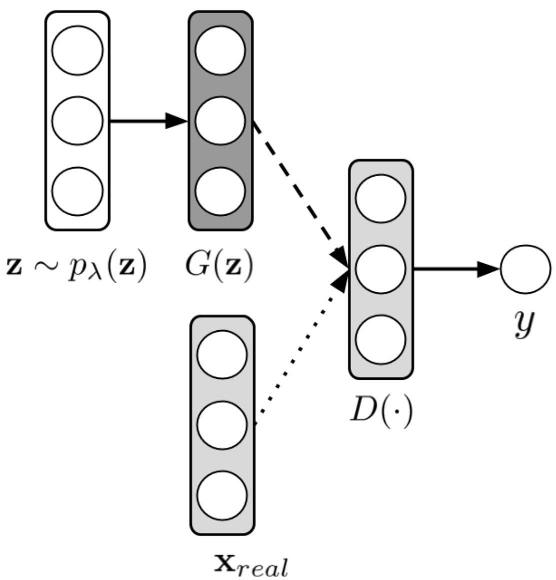
QUESTION

Can we stick to the log-likelihood approach but with a simple training procedure?

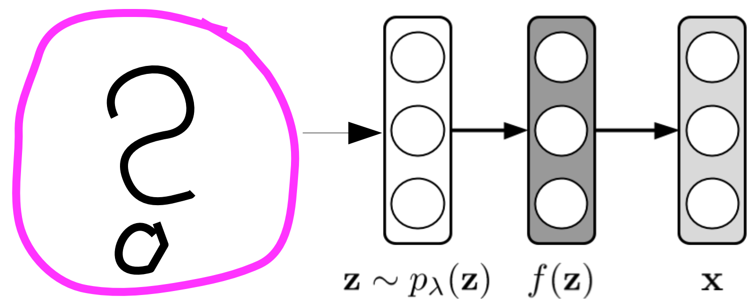
DGM: so far we have



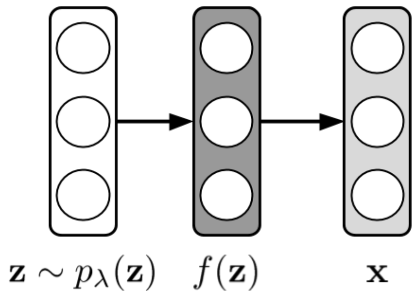
Density Network



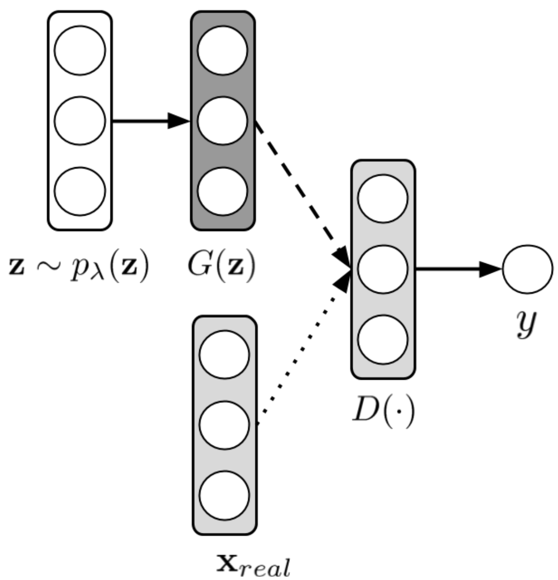
Generative Adversarial Net



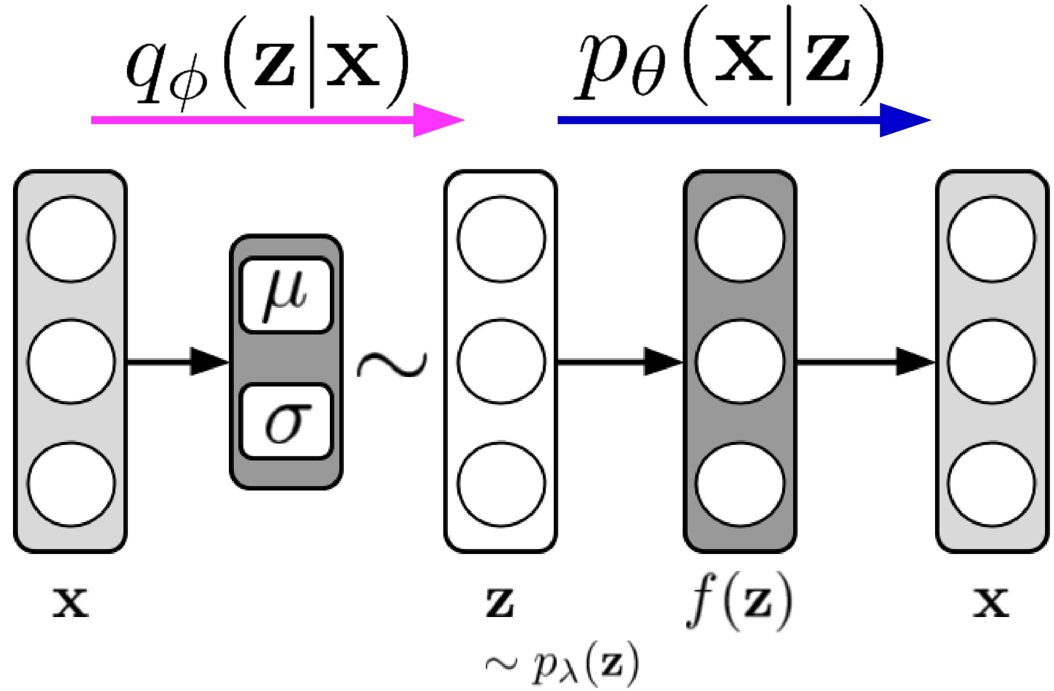
DGM: Variational Auto-Encoder



Density Network



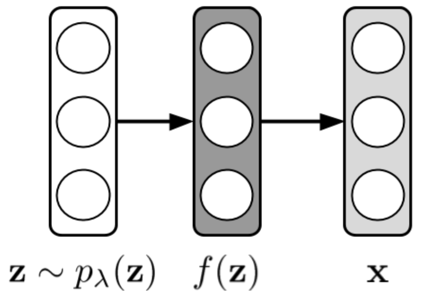
Generative Adversarial Net



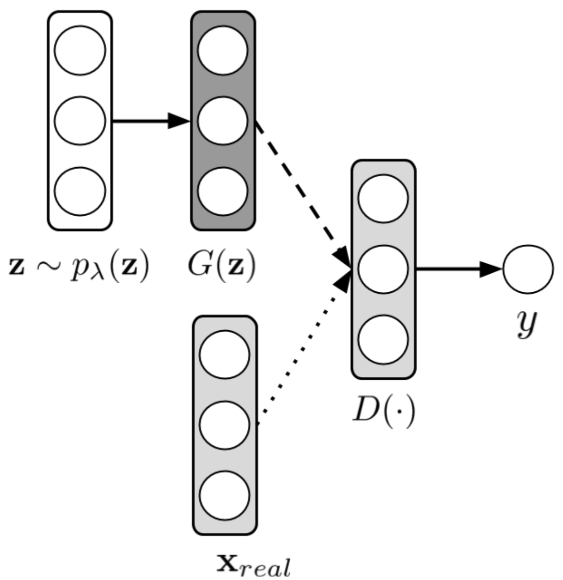
Variational Auto-Encoder

Kingma, D. P., & Welling, M. (2013). Auto-encoding Variational Bayes. arXiv preprint arXiv:1312.6114.

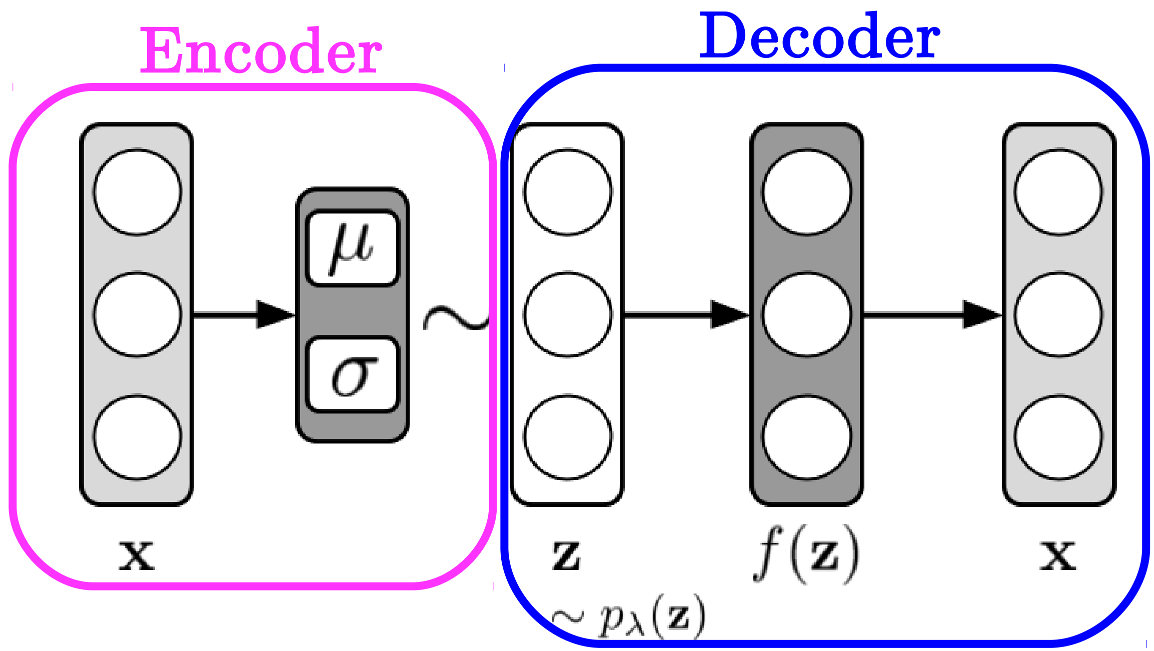
DGM: Variational Auto-Encoder



Density Network



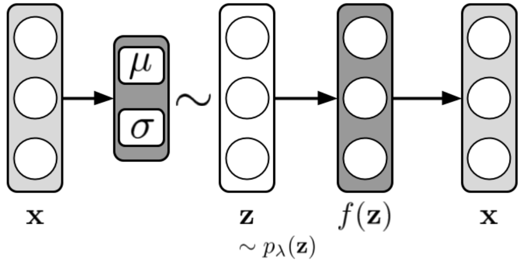
Generative Adversarial Net



Variational Auto-Encoder

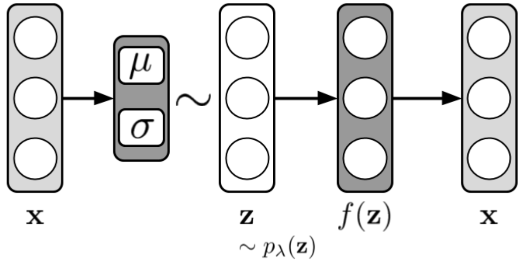
Kingma, D. P., & Welling, M. (2013). Auto-encoding Variational Bayes. arXiv preprint arXiv:1312.6114.

DGM: Variational Auto-Encoder



$$\begin{aligned}
\log p(\mathbf{x}) &= \log \int p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})\ \mathrm{d}\mathbf{z} \\
&= \log \int \frac{q_\phi(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z}|\mathbf{x})}\ p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})\ \mathrm{d}\mathbf{z} \\
&\geq \int q_\phi(\mathbf{z}|\mathbf{x})\ \log \frac{p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\ \mathrm{d}\mathbf{z} \\
&= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \mathrm{KL}[q_\phi(\mathbf{z}|\mathbf{x})||p_\lambda(\mathbf{z})]
\end{aligned}$$

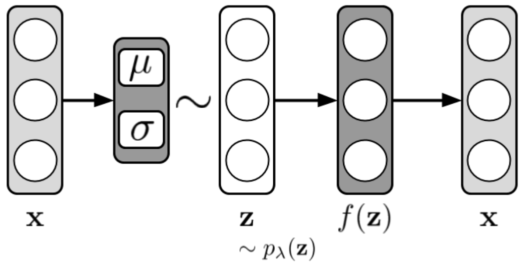
DGM: Variational Auto-Encoder



Variational posterior

$$
\begin{aligned}
\log p(\mathbf{x}) &= \log \int p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})\ \mathrm{d}\mathbf{z} \\
&= \log \int \frac{q_\phi(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z}|\mathbf{x})}\ p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})\ \mathrm{d}\mathbf{z} \\
&\geq \int q_\phi(\mathbf{z}|\mathbf{x})\ \log \frac{p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\ \mathrm{d}\mathbf{z} \\
&= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \mathrm{KL}[q_\phi(\mathbf{z}|\mathbf{x})||p_\lambda(\mathbf{z})]
\end{aligned}
$$

DGM: Variational Auto-Encoder



$$
\begin{aligned}
\log p(\mathbf{x}) &= \log \int p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})\ \mathrm{d}\mathbf{z} \\
&= \log \int \frac{q_\phi(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z}|\mathbf{x})}\ p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})\ \mathrm{d}\mathbf{z} \\
&\geq \int q_\phi(\mathbf{z}|\mathbf{x})\ \log \frac{p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\ \mathrm{d}\mathbf{z} \\
&= \underbrace{\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction error}} - \underbrace{\mathrm{KL}[q_\phi(\mathbf{z}|\mathbf{x})||p_\lambda(\mathbf{z})]}_{\text{Regularization}}
\end{aligned}
$$

DGM: Variational Auto-Encoder

Our objective it **the evidence lower bound.**

$$\log p(\mathbf{x}) \geq \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \mathrm{KL}[q_\phi(\mathbf{z}|\mathbf{x}) || p_\lambda(\mathbf{z})]$$

We can approximate it using **MC sample**.

$$\mathcal{L}(\mathbf{x}) \approx \frac{1}{S} \sum_{s=1}^{S} [\log p_\theta(\mathbf{x}|\mathbf{z}_s) - \log q_\phi(\mathbf{z}_s|\mathbf{x}) + \log p_\lambda(\mathbf{z}_s)]$$

DGM: Variational Auto-Encoder

Our objective it **the evidence lower bound.**

$$\log p(\mathbf{x}) \geq \mathbb{E}_{\mathbf{z}\sim q_{\phi}(\mathbf{z}|\mathbf{x})}[\log p_{\theta}(\mathbf{x}|\mathbf{z})] - \mathrm{KL}[q_{\phi}(\mathbf{z}|\mathbf{x})||p_{\lambda}(\mathbf{z})]$$

We can approximate it using **MC sample**.

$$\mathcal{L}(\mathbf{x}) \approx \frac{1}{S}\sum_{s=1}^{S}[\log p_{\theta}(\mathbf{x}|\mathbf{z}_s) - \log q_{\phi}(\mathbf{z}_s|\mathbf{x}) + \log p_{\lambda}(\mathbf{z}_s)]$$

How to properly calculate gradients (*i.e.*, train the model)?

DGM: Variational Auto-Encoder

$$\mathcal{L}(\mathbf{x}) \approx \frac{1}{S} \sum_{s=1}^{S} [\log p_{\theta}(\mathbf{x}|\mathbf{z}_s) - \log q_{\phi}(\mathbf{z}_s|\mathbf{x}) + \log p_{\lambda}(\mathbf{z}_s)]$$

PROBLEM: calculating gradient wrt parameters of the variational posterior (*i.e.*, sampling process).

Kingma, D. P., & Welling, M. (2013). Auto-encoding Variational Bayes. arXiv preprint arXiv:1312.6114.

DGM: Variational Auto-Encoder

$$\mathcal{L}(\mathbf{x}) \approx \frac{1}{S}\sum_{s=1}^{S}[\log p_\theta(\mathbf{x}|\mathbf{z}_s) - \log q_\phi(\mathbf{z}_s|\mathbf{x}) + \log p_\lambda(\mathbf{z}_s)]$$

PROBLEM: calculating gradient wrt parameters of the variational posterior (*i.e.*, sampling process).

SOLUTION: use a non-centered parameterization (a.k.a. ***reparameterization trick***).

$$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}|\mu, \sigma^2)$$

$$\mathbf{z}_s = \mu + \sigma \odot \epsilon, \quad \epsilon \sim \mathcal{N}(0, \mathbf{I})$$

Kingma, D. P., & Welling, M. (2013). Auto-encoding Variational Bayes. arXiv preprint arXiv:1312.6114.

DGM: Variational Auto-Encoder

$$\mathcal{L}(\mathbf{x}) \approx \frac{1}{S}\sum_{s=1}^{S}[\log p_\theta(\mathbf{x}|\mathbf{z}_s) - \log q_\phi(\mathbf{z}_s|\mathbf{x}) + \log p_\lambda(\mathbf{z}_s)]$$

PROBLEM: calculating gradient wrt parameters of the variational posterior (*i.e.*, sampling process).

SOLUTION: use a non-centered parameterization (a.k.a. ***reparameterization trick***).

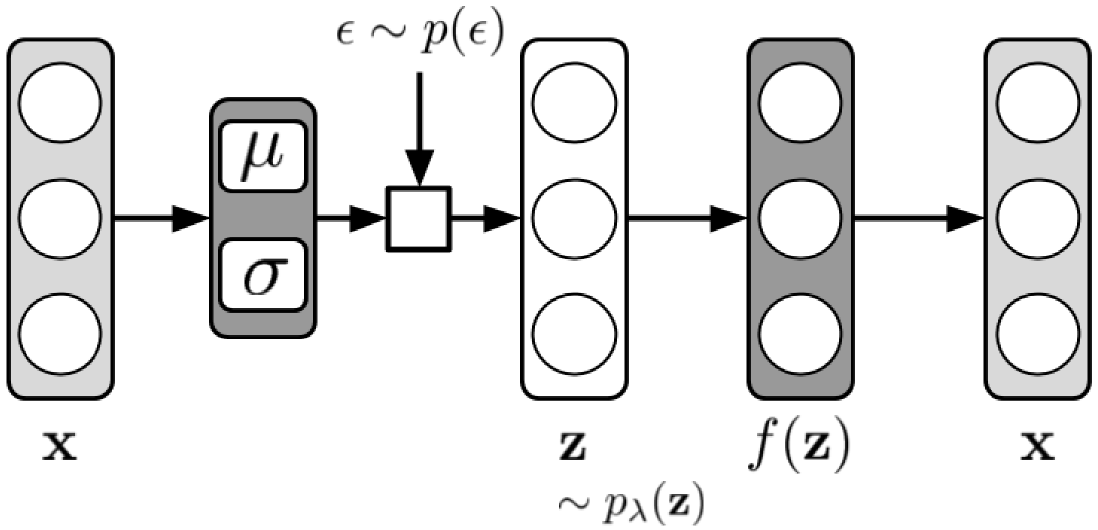
$$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}|\mu, \sigma^2)$$

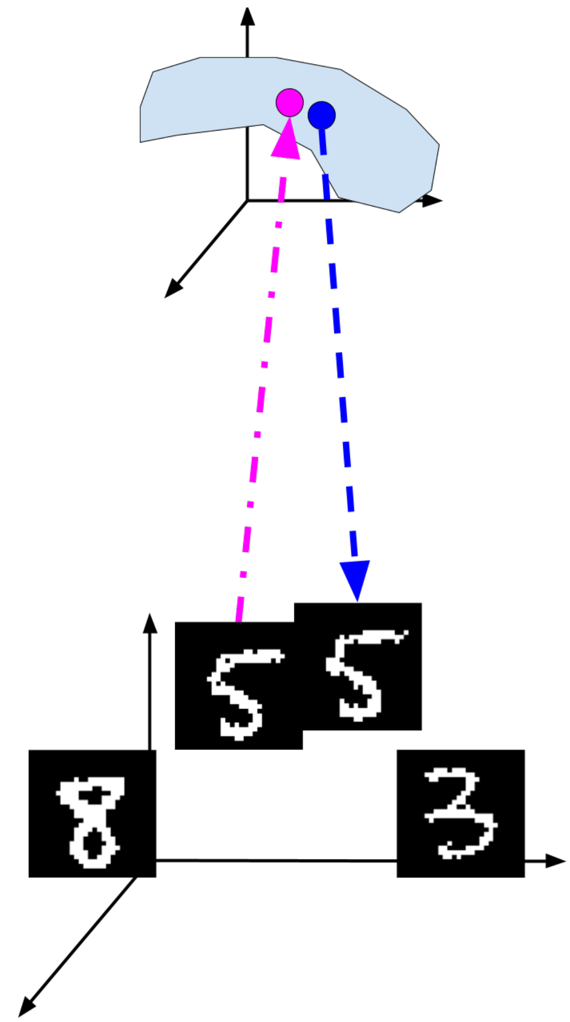
$$\mathbf{z}_s = \mu + \sigma \odot \epsilon, \quad \epsilon \sim \mathcal{N}(0, \mathbf{I})$$

Output of a neural network

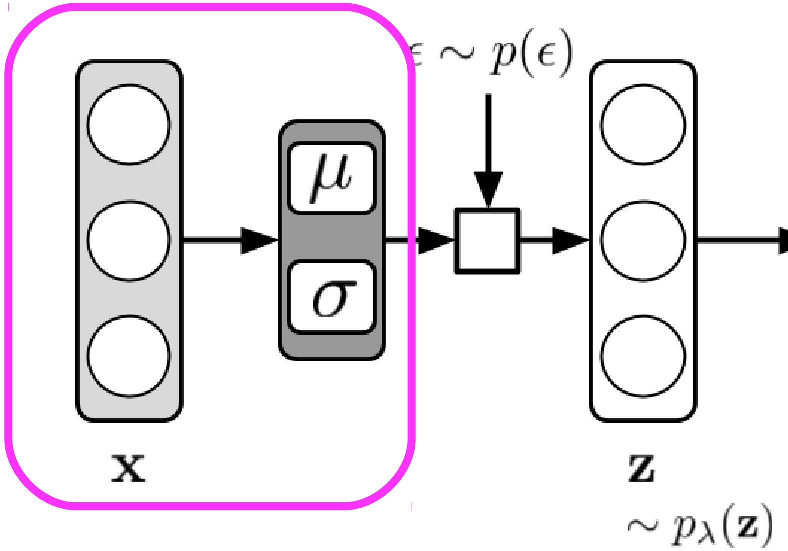
Kingma, D. P., & Welling, M. (2013). Auto-encoding Variational Bayes. arXiv preprint arXiv:1312.6114.

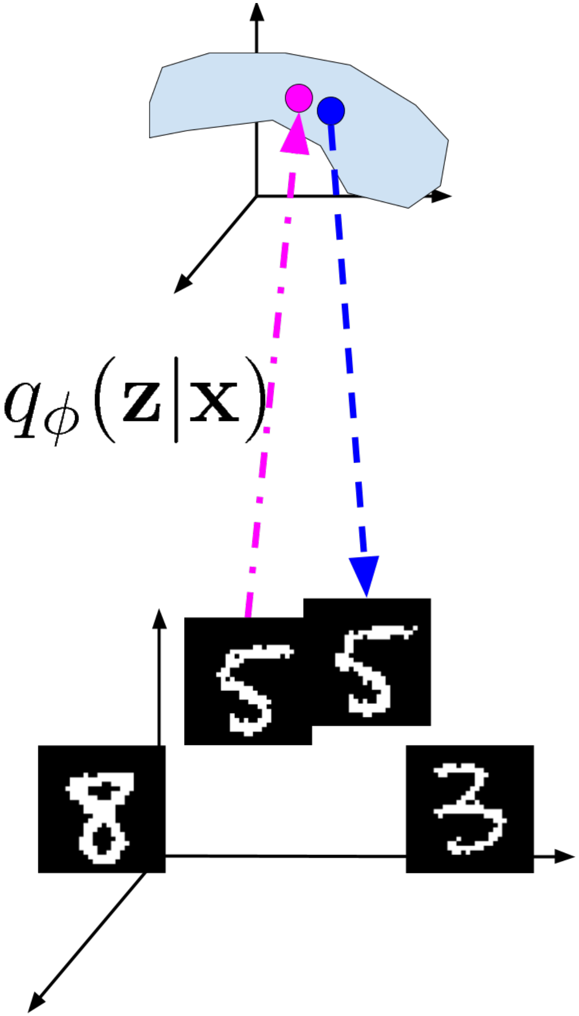
DGM: Variational Auto-Encoder





DGM: Variational Auto-Encoder

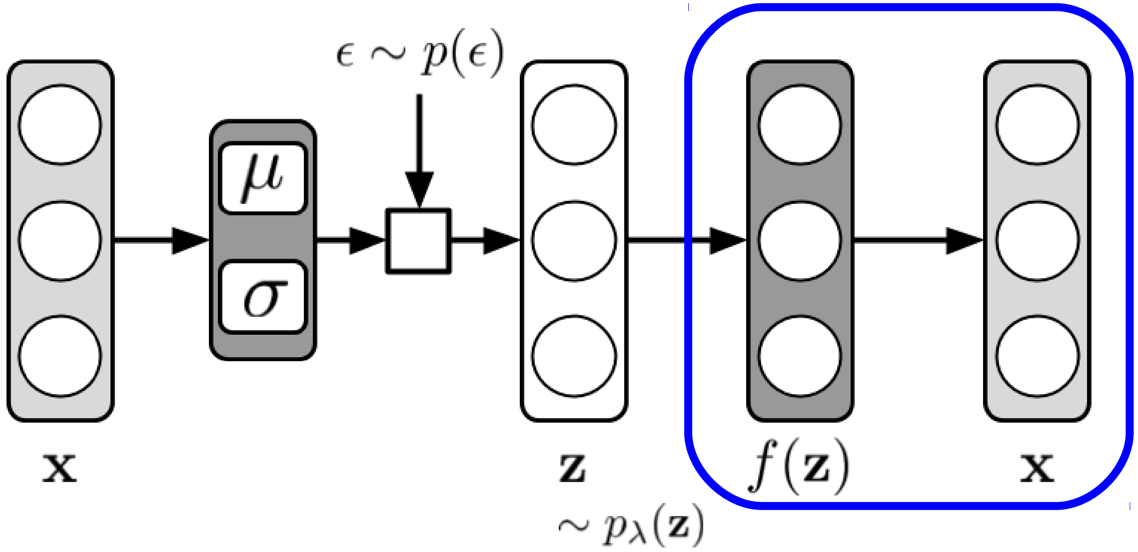


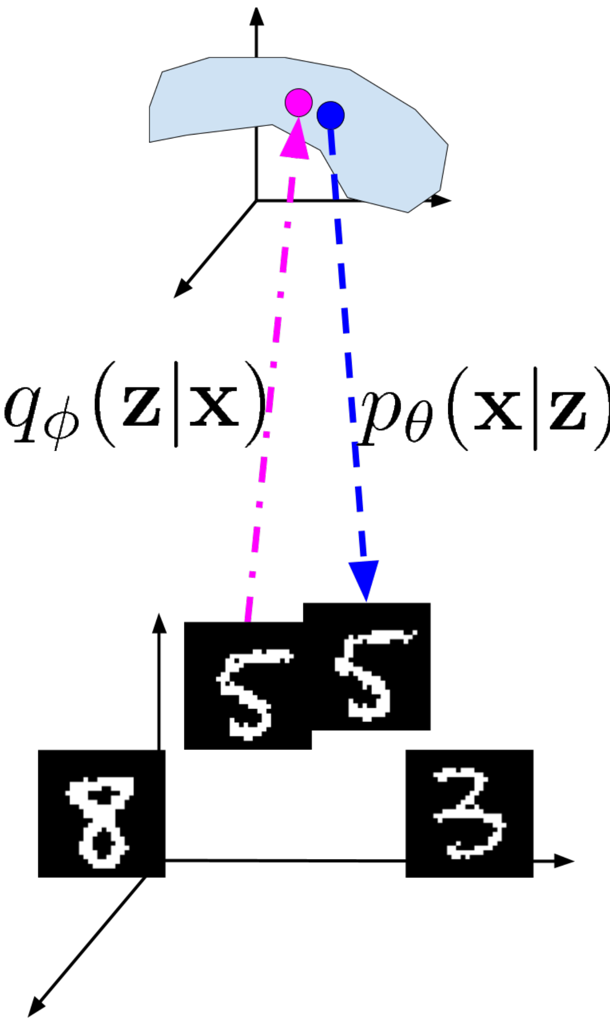


A deep neural net that outputs parameters of the variational posterior (encoder):

$$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}|\mu(\mathbf{x}), \mathrm{diag}\{\sigma^2(\mathbf{x})\})$$

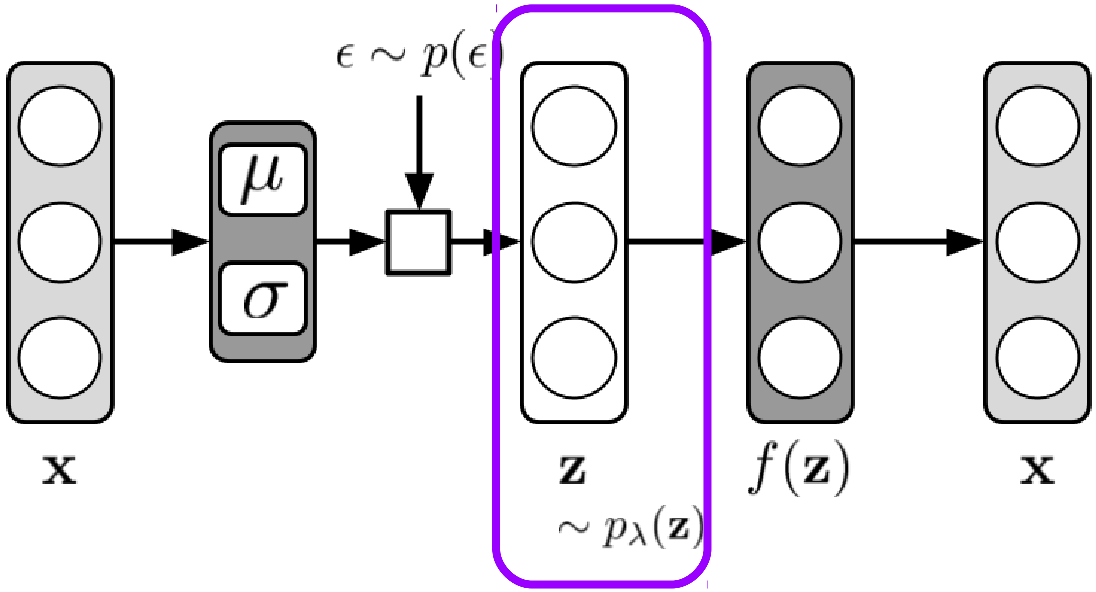
DGM: Variational Auto-Encoder

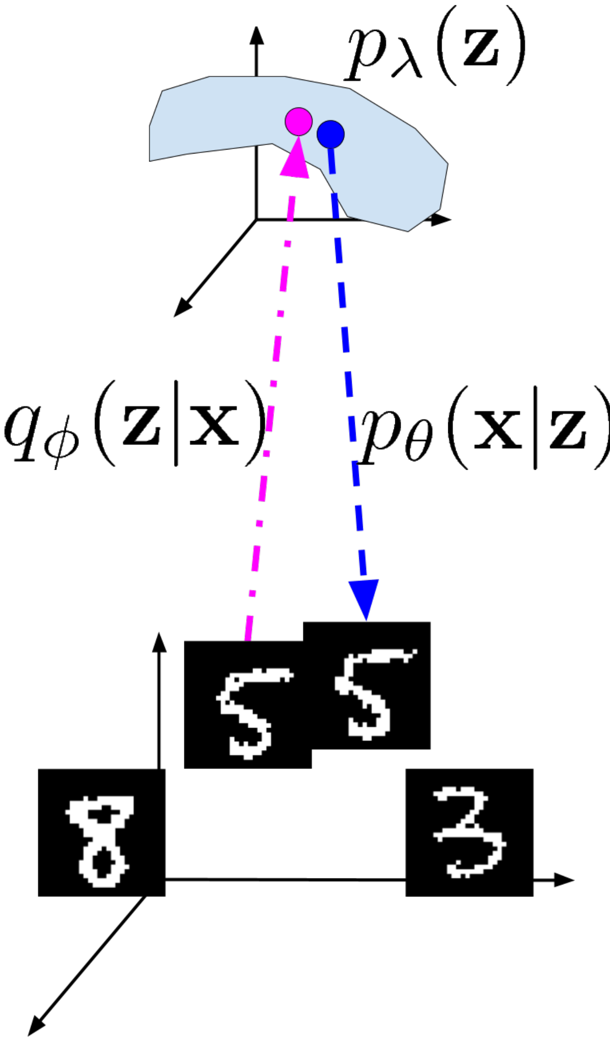




A **deep neural net** that outputs parameters of the generator (**decoder**), *e.g.*, a normal distribution or Bernoulli distribution.

DGM: Variational Auto-Encoder



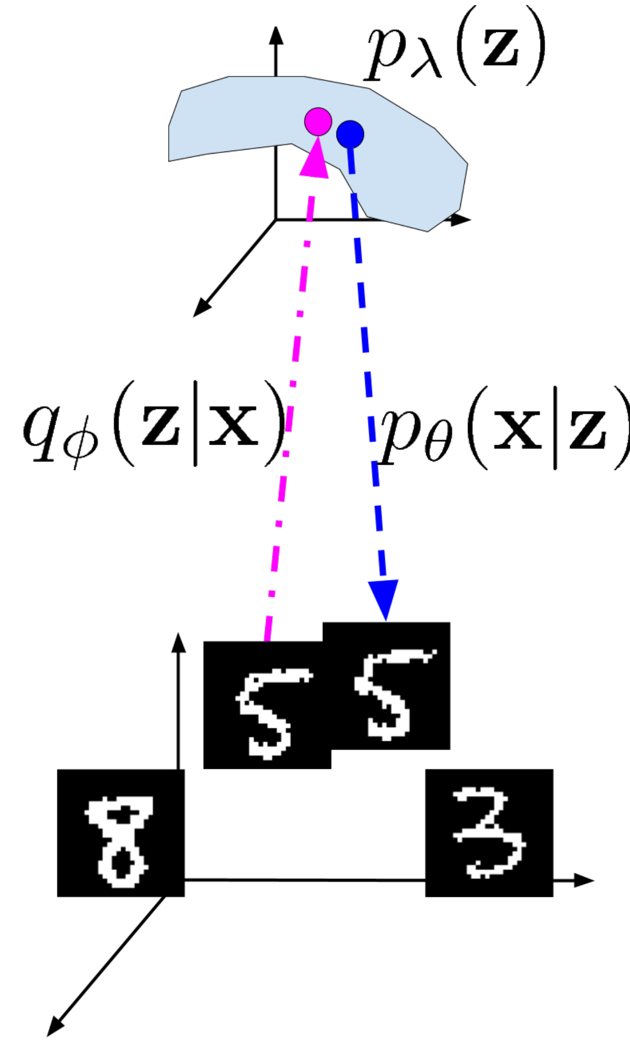


A prior that regularizes the encoder and takes part in the generative process.

$$p_\lambda(\mathbf{z}) = \mathcal{N}(\mathbf{z}|0, \mathbf{I})$$

DGM: Variational Auto-Encoder

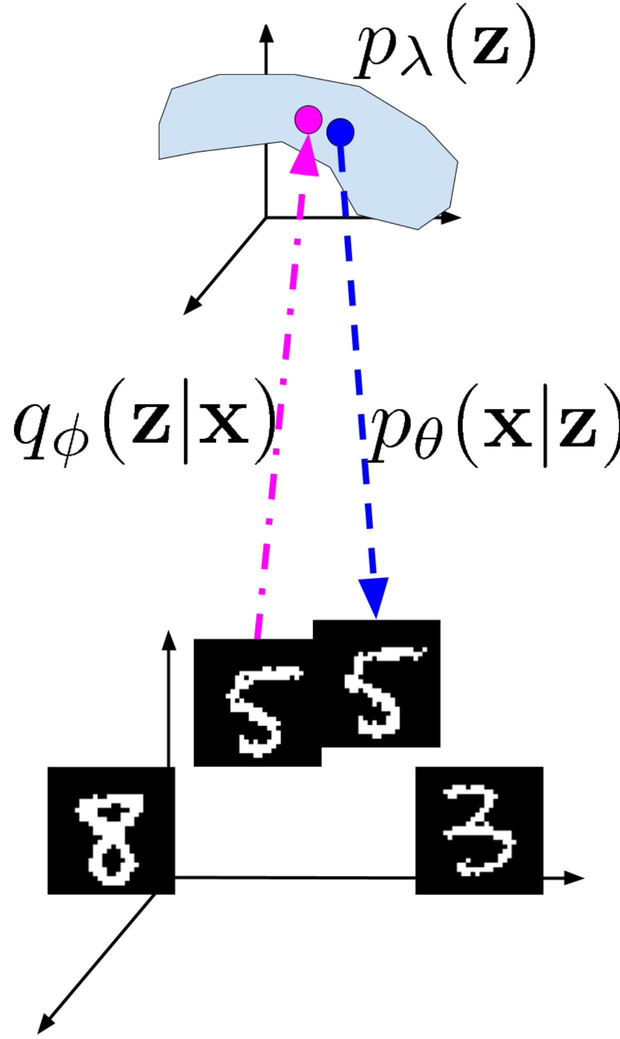
$$q_\phi(\mathbf{z}|\mathbf{x}) \propto p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})$$



DGM: Variational Auto-Encoder

$$q_\phi(\mathbf{z}|\mathbf{x}) \propto p_\theta(\mathbf{x}|\mathbf{z})\, p_\lambda(\mathbf{z})$$

Feedforward nets
Convolutional nets
PixelCNN
Gated PixelCNN

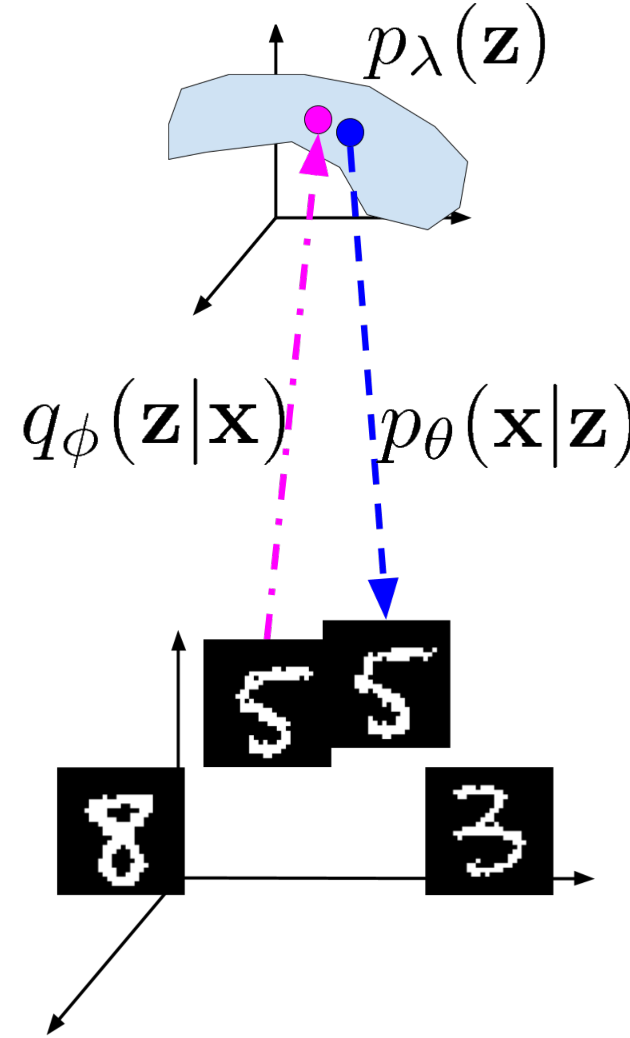


DGM: Variational Auto-Encoder

$$q_\phi(\mathbf{z}|\mathbf{x}) \propto p_\theta(\mathbf{x}|\mathbf{z})\, p_\lambda(\mathbf{z})$$

Normalizing flows
Volume-preserving flows
Gaussian processes
Stein Particle Descent
Operator VI

Feedforward nets
Convolutional nets
PixelCNN
Gated PixelCNN



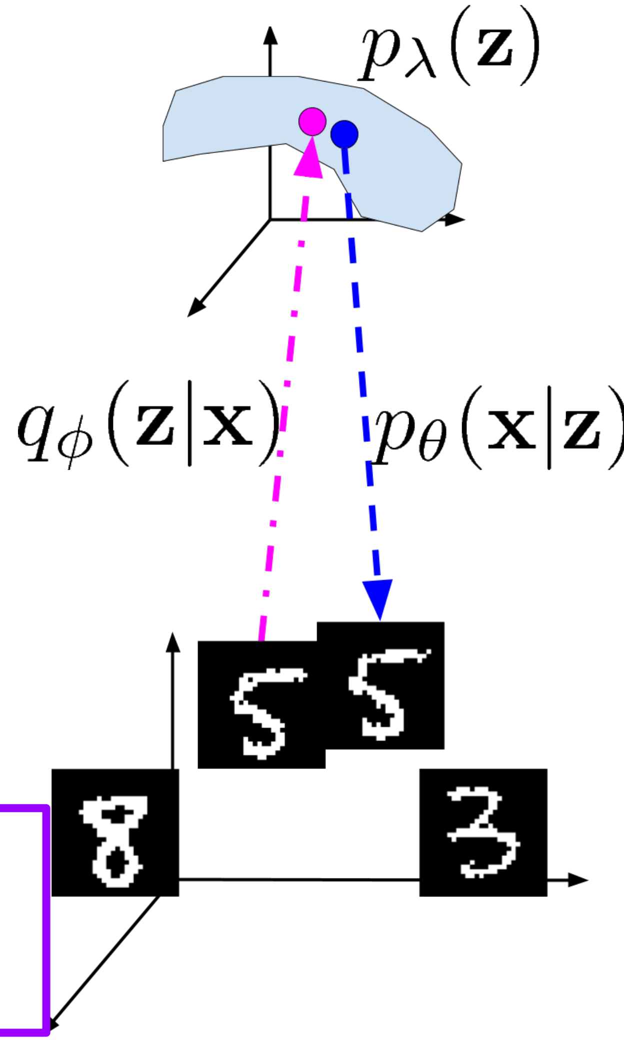
DGM: Variational Auto-Encoder

$$q_\phi(\mathbf{z}|\mathbf{x}) \propto p_\theta(\mathbf{x}|\mathbf{z})\, p_\lambda(\mathbf{z})$$

Normalizing flows
Volume-preserving flows
Gaussian processes
Stein Particle Descent
Operator VI

Feedforward nets
Convolutional nets
PixelCNN
Gated PixelCNN

Auto-regressive Prior
Objective Prior
Stick-Breaking Prior
VampPrior



DGM: Variational Auto-Encoder

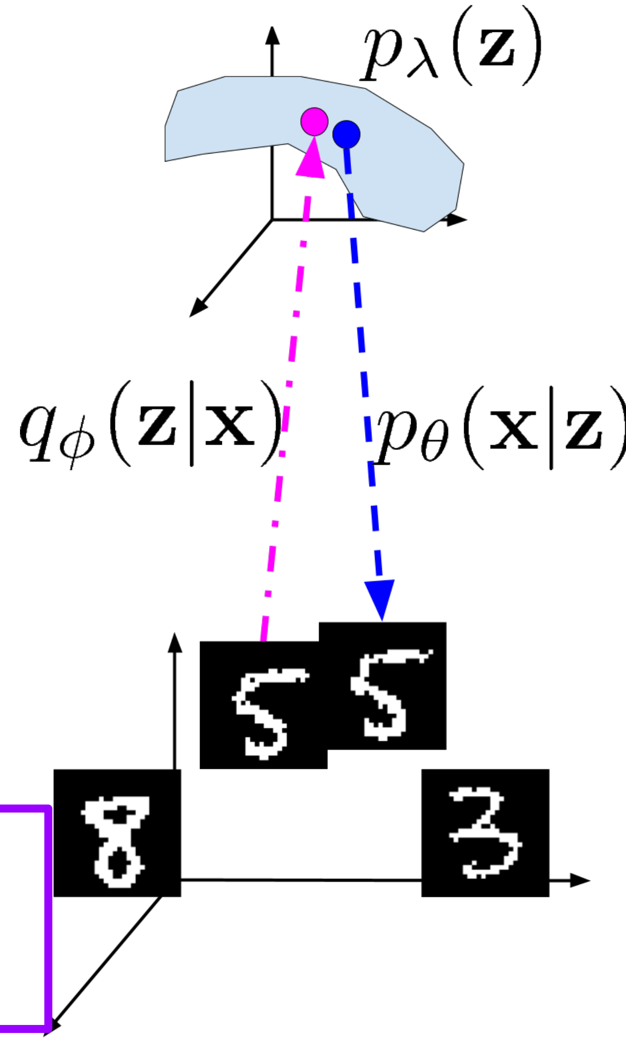
$$q_\phi(\mathbf{z}|\mathbf{x}) \propto p_\theta(\mathbf{x}|\mathbf{z})\, p_\lambda(\mathbf{z})$$

Normalizing flows
Volume-preserving flows
Gaussian processes
Stein Particle Descent
Operator VI

Feedforward nets
Convolutional nets
PixelCNN
Gated PixelCNN

Auto-regressive Prior
Objective Prior
Stick-Breaking Prior
VampPrior

Importance Weighted AE
Renyi Divergence
Stein Divergence



DGM: VAE

PROS

Log-likelihood framework

Easy **sampling**

Training using **gradient-based methods**

Stable training

Discovers **latent representation**

Could be easily combined with other probabilistic frameworks

CONS

Only **explicit** models

Produces **blurry images(?)**

In order to *make better* ***decisions***, we need a *better understanding* of ***reality***.

=

generative modeling

Web-page:
https://jmtomczak.github.io

Code on github:
https://github.com/jmtomczak

Contact:

J.M.Tomczak@uva.nl
jakubmkt@gmail.com

Part of the presented research was funded by the European Commission within the Marie Skłodowska-Curie Individual Fellowship (Grant No. 702666, ''*Deep learning and Bayesian inference for medical imaging*'').

RESEARCH & INNOVATION
Marie Skłodowska-Curie actions