高等机器学习

生成模型

刘 畅

微软亚洲研究院

Microsoft

清華大學 Tsinghua University

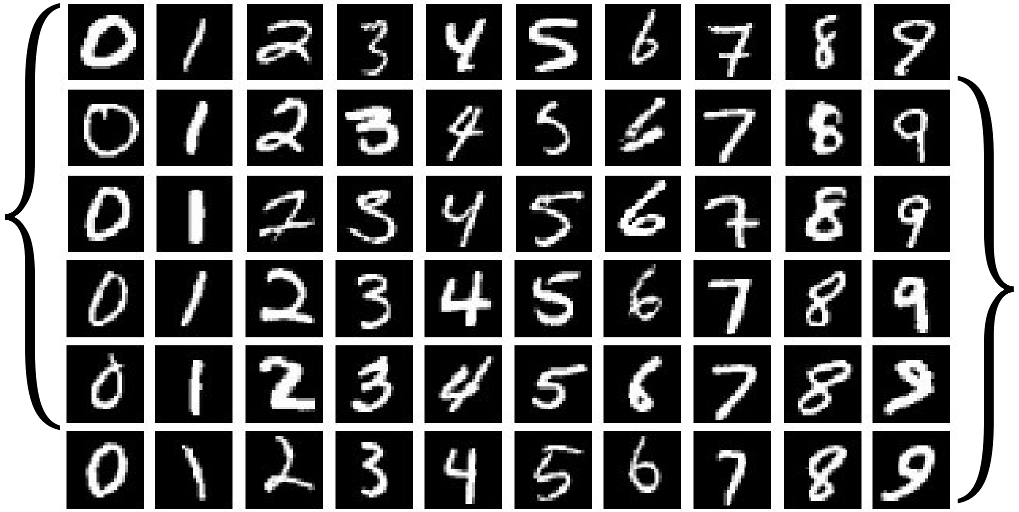
Outline

- Generative Models: Overview
- Plain Generative Models
  - Autoregressive Models
- Latent Variable Models
  - Deterministic Generative Models
    - Generative Adversarial Nets
    - Flow-Based Generative Models
  - Bayesian Generative Models
    - Bayesian Inference (variational inference, MCMC)
    - Bayesian Networks
      - Topic Models (LDA, LightLDA, sLDA)
      - Deep Bayesian Models (VAE)
    - Markov Random Fields (Boltzmann machines, deep energy-based models)

Generative Model: Overview

- Generative Models:
  - Models that describe the generating process of all observations.
  - Technically, they specify $p(x)$ (unsupervised) or $p(x, y)$ (supervised) in principle, either explicitly or implicitly.

$$\{x^{(n)}\} =$$

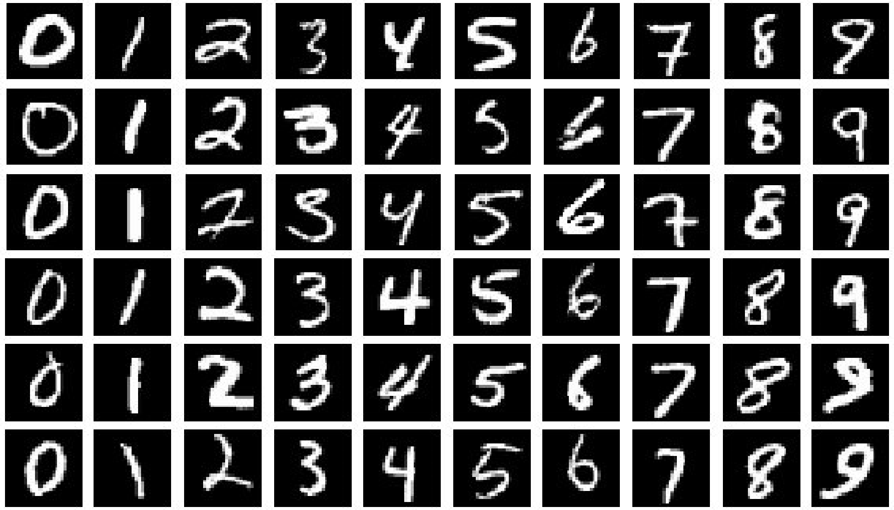


$$\sim p(x)$$

Generative Model: Overview

- Generative Models:
  - Models that describe the generating process of all observations.
  - Technically, they specify $p(x)$ (unsupervised) or $p(x, y)$ (supervised) in principle, either explicitly or implicitly.

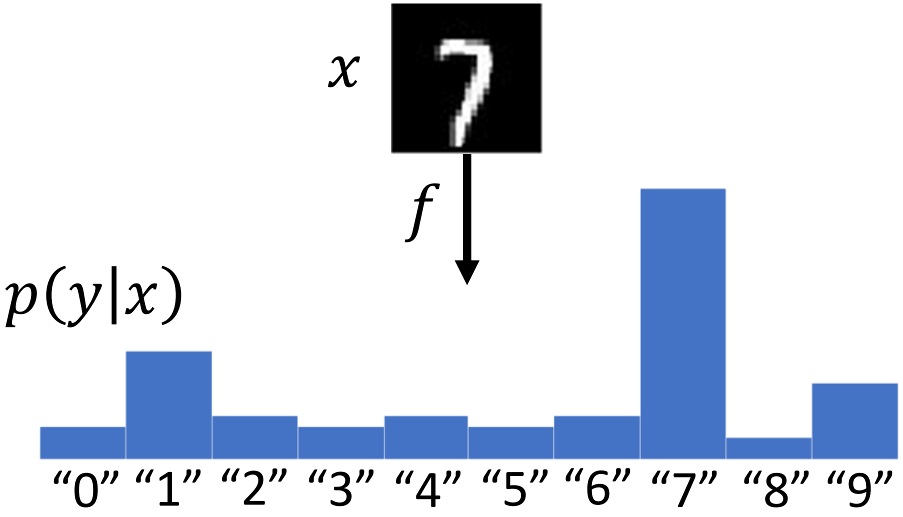
$$\{x^{(n)}, y^{(n)}\} = \left\{ \begin{array}{l} y^{(n)} = \text{“0” “1” “2” “3” “4” “5” “6” “7” “8” “9”} \\ x^{(n)} = \end{array} \right\} \sim p(x, y)$$



Generative Model: Overview

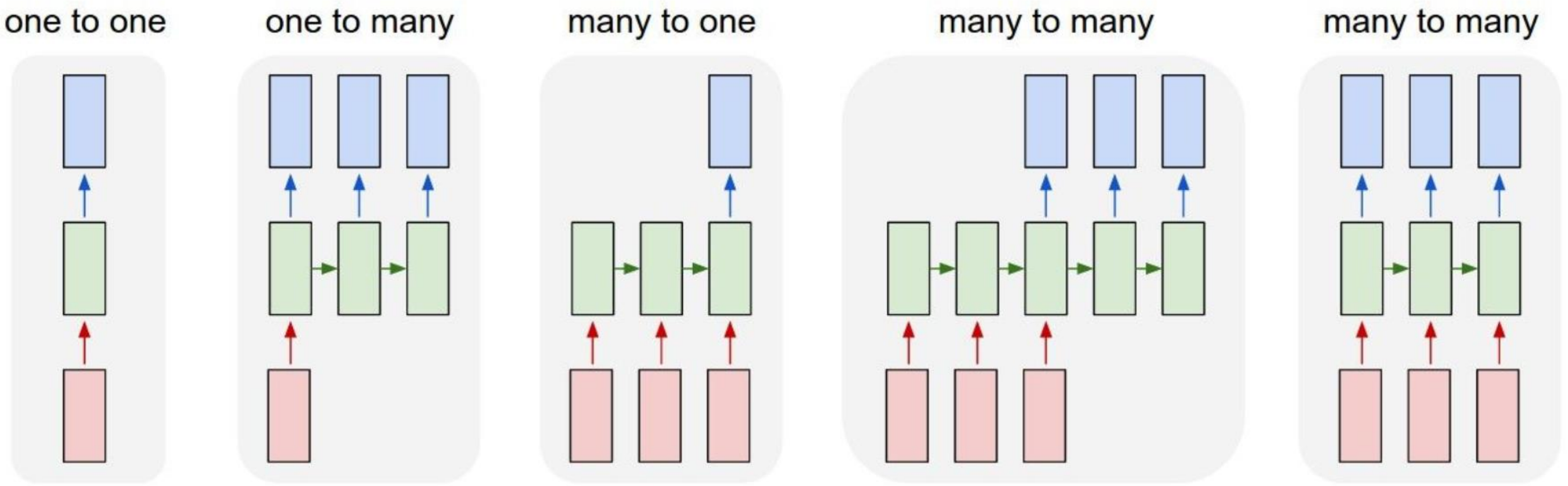
- Non-Generative Models:

Discriminative models (e.g., feedforward neural networks): only $p(y|x)$ is available.



Recurrent neural networks:

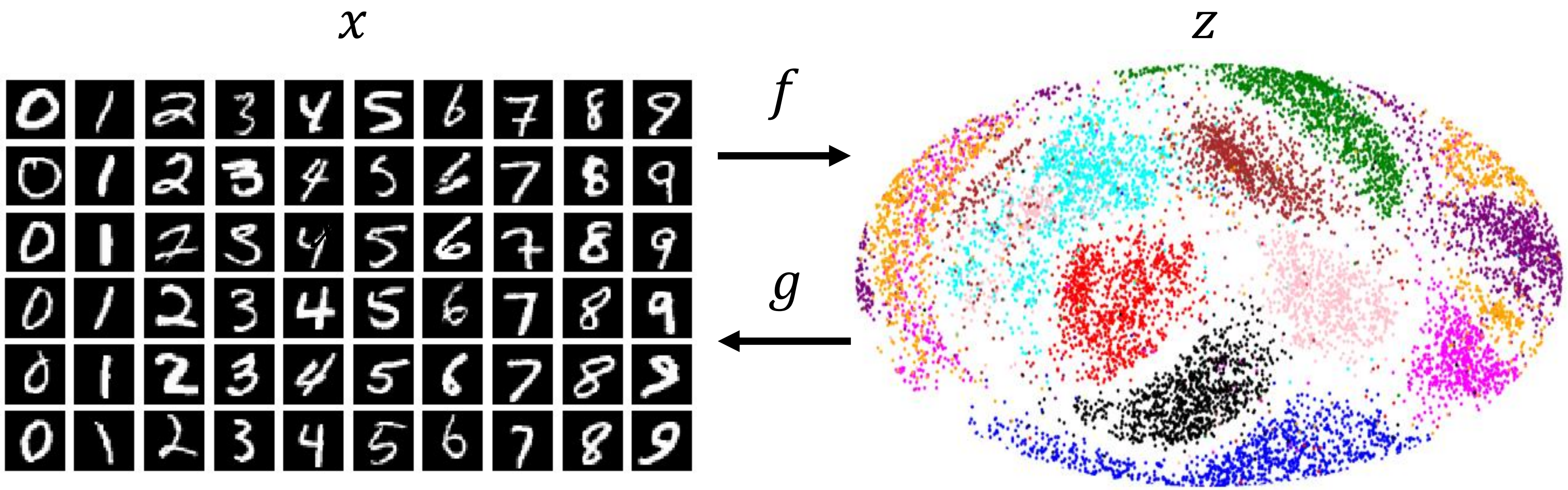
only $p($🟦$|$🟥$)$ is available.



Generative Model: Overview

- Non-Generative Models:

Autoencoders:
$p(x)$ unavailable.

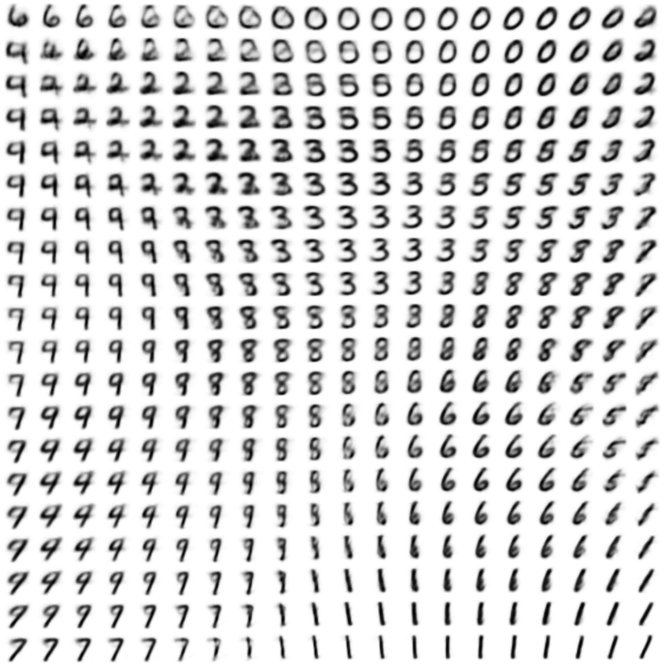


(Img: [DFD+18])

Generative Model: Overview

- What can generative models do:

  1. Generate new data.



Generation $p(x)$ [KW14]

Conditional Generation $p(x|y)$ [LWZZ18]

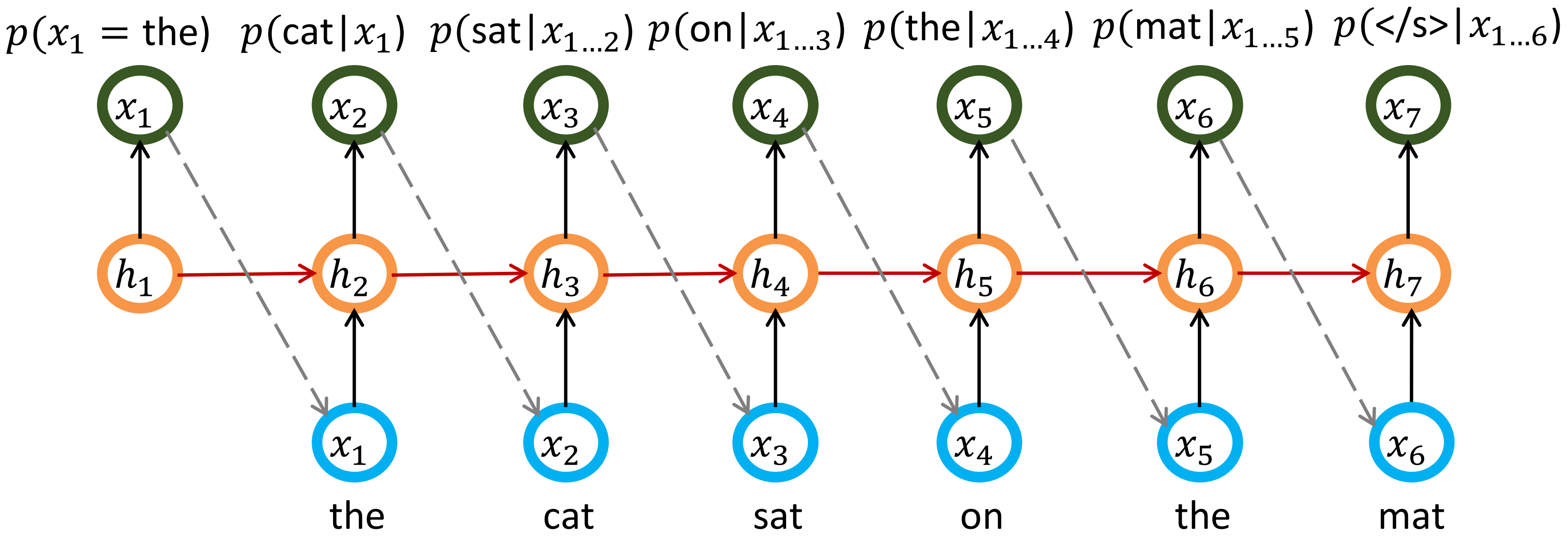
Missing Value Imputation (Completion) $p(x_{\text{hidden}}|x_{\text{observed}})$ [OKK16]

Generative Model: Overview

- What can generative models do:

1. Generate new data.

“*the cat sat on the mat*”$\sim p(x)$: Language Model.

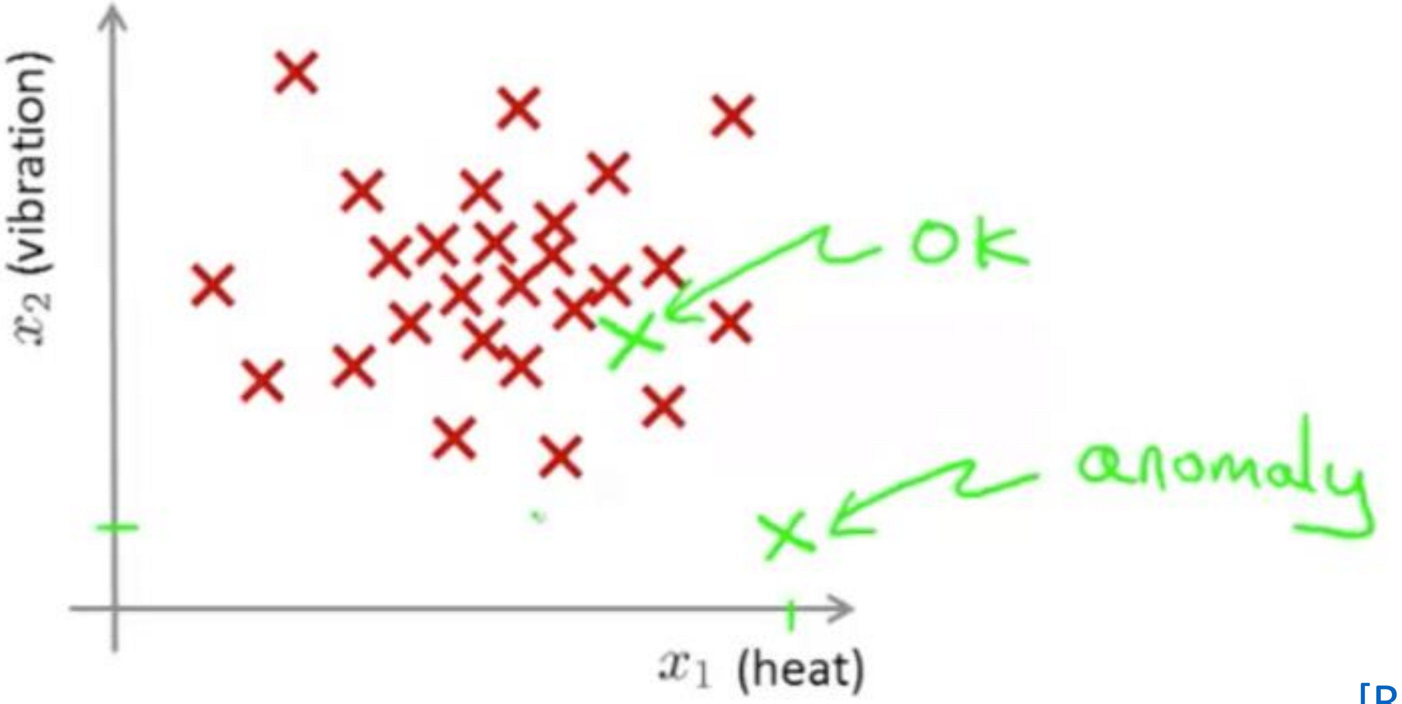


Generative Model: Overview

- What can generative models do:

  2. Density estimation $p(x)$.

Anomaly Detection:



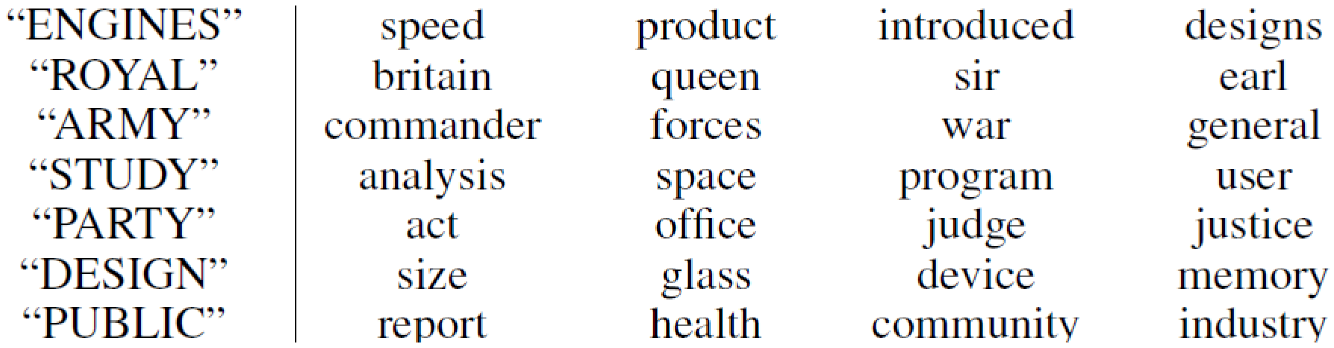
[Ritchie Ng]

Generative Model: Overview

- What can generative models do:

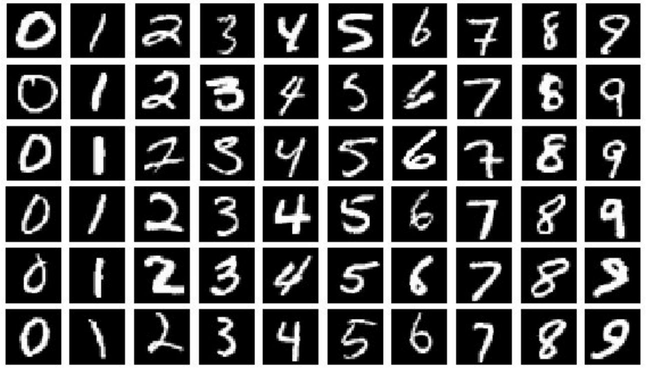
3. Draw semantic or concise representation of data x (via latent variable z).

x (documents)

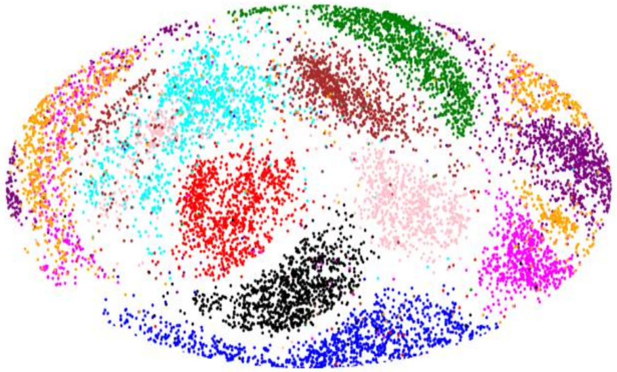


"ENGINES"	speed	product	introduced	designs
"ROYAL"	britain	queen	sir	earl
"ARMY"	commander	forces	war	general
"STUDY"	analysis	space	program	user
"PARTY"	act	office	judge	justice
"DESIGN"	size	glass	device	memory
"PUBLIC"	report	health	community	industry

z (topics) [PT13]



x (image)

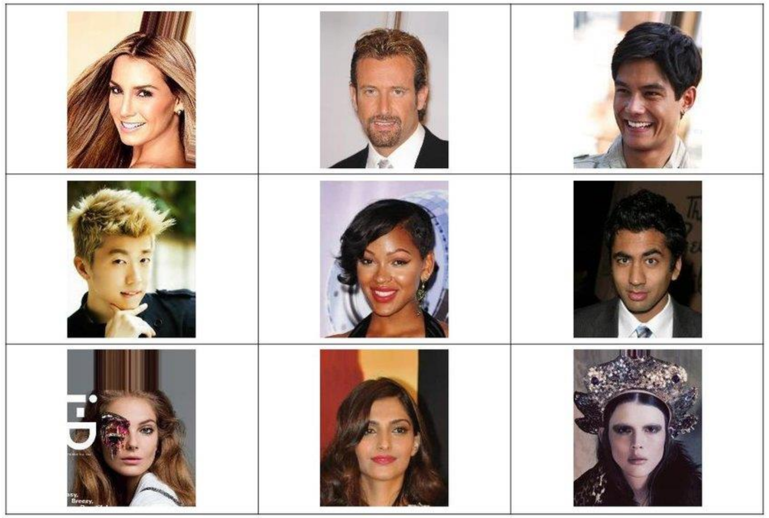


z (semantic regions) [DFD+18]

Generative Model: Overview

- What can generative models do:

3. Draw semantic or concise representation of data x (via latent variable z).



x (image)

(a) Smiling

(b) Pale Skin

(c) Blond Hair

(d) Narrow Eyes

(e) Young

(f) Male

z (semantic regions) [KD18]

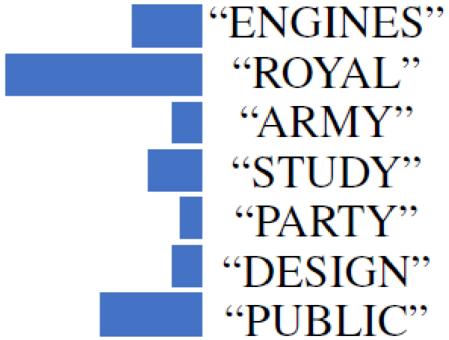
Generative Model: Overview

- What can generative models do:

3. Draw semantic or concise representation of data x (via latent variable z).

Dimensionality Reduction:

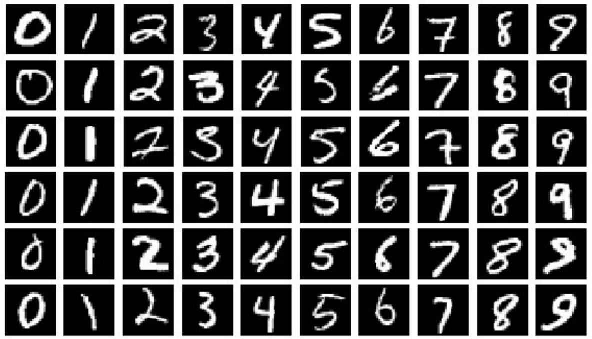
$x \in \mathbb{R}^{\#\text{vocabulary}}$



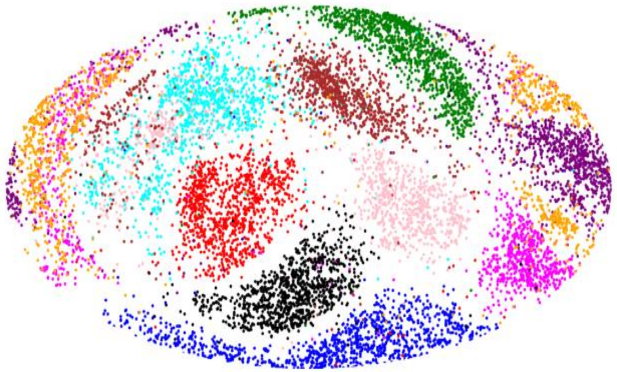
speed	product	introduced
britain	queen	sir
commander	forces	war
analysis	space	program
act	office	judge
size	glass	device
report	health	community

Topic proportion
$z \in \mathbb{R}^{\#\text{topic}}$

[PT13]



$x \in \mathbb{R}^{28\times 28}$

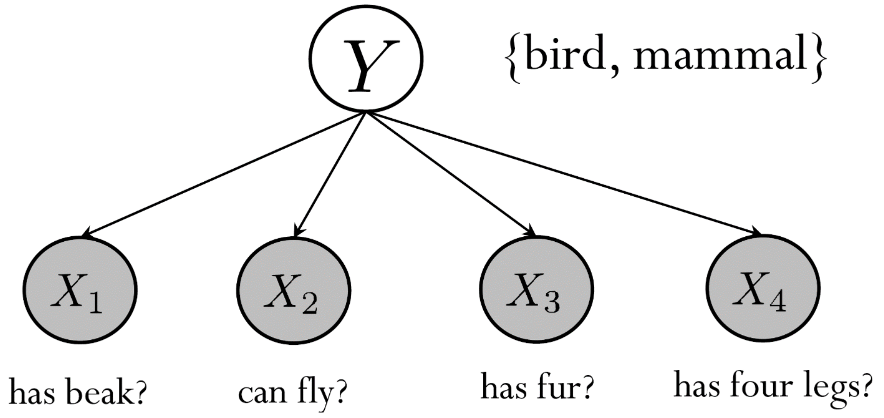


$z \in \mathbb{R}^{20}$ [DFD+18]

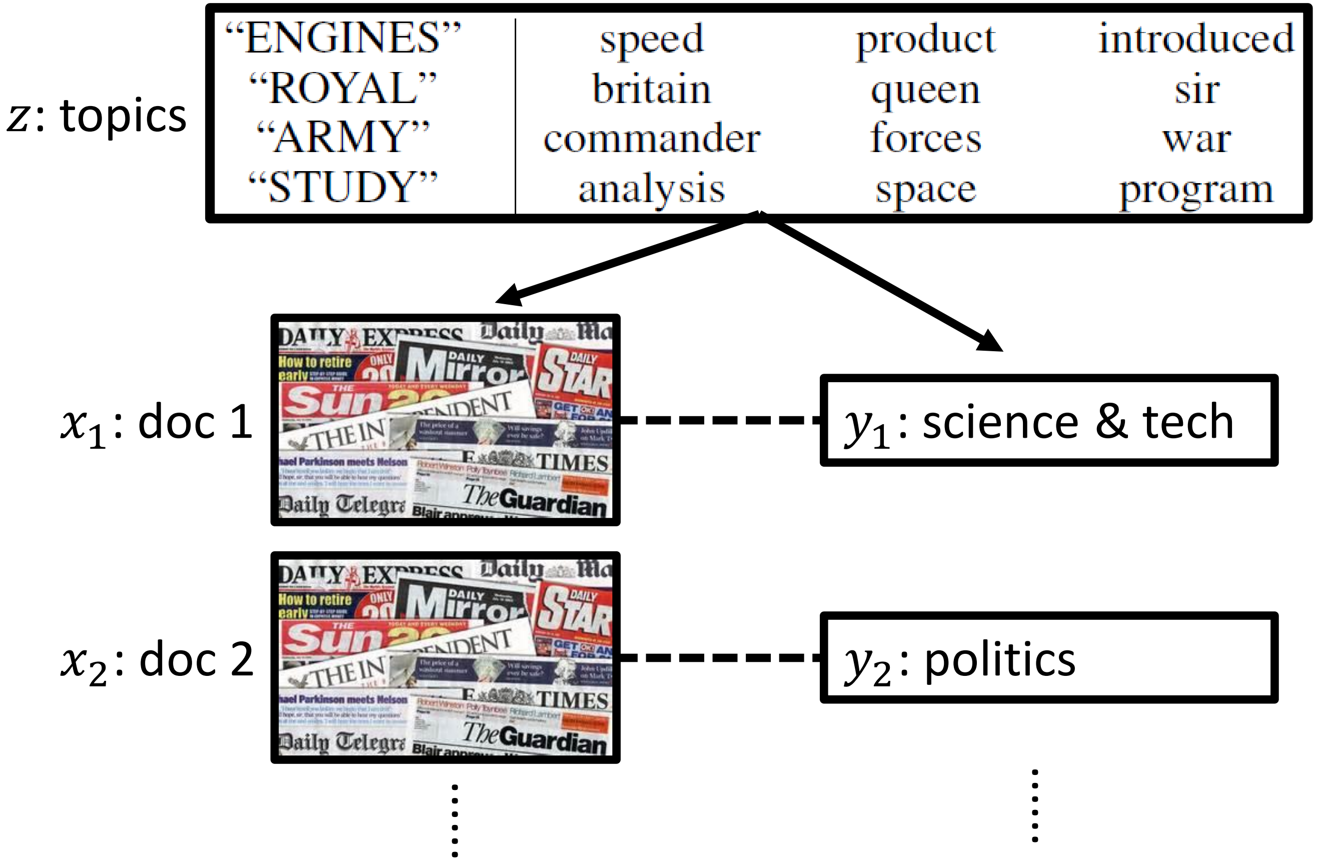
Generative Model: Overview

- What can generative models do:

4. Supervised Learning: $\arg\max_{y^*} p\left(y^* \middle| x^*, \left\{\left(x^{(n)}, y^{(n)}\right)\right\}\right)$.



[Naive Bayes]



"ENGINES"	speed	product	introduced
"ROYAL"	britain	queen	sir
"ARMY"	commander	forces	war
"STUDY"	analysis	space	program

Supervised LDA [MB08]

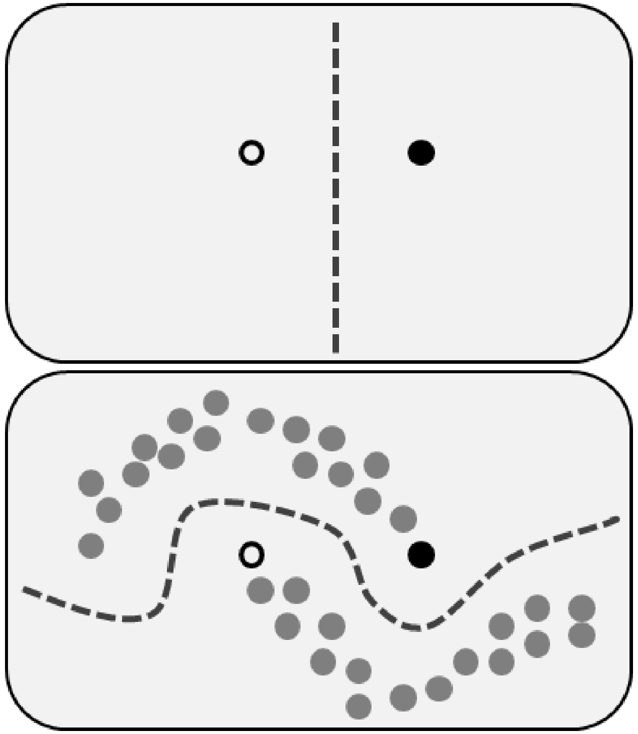
Generative Model: Overview

- What can generative models do:

4. Supervised Learning: $\arg\max_{y^*} p\left(y^* \middle| x^*, \left\{\left(x^{(n)}, y^{(n)}\right)\right\}, \left\{x^{(n)}\right\}\right)$.

Semi-Supervised Learning:

Unlabeled data $\left\{x^{(n)}\right\}$ can be utilized to learn a better $p(x, y)$.



Generative Model: Benefits

"What I cannot create, I do not understand." *—Richard Feynman*

- Natural for generation.
- For representation learning: responsible and faithful knowledge of the data.
- For supervised learning: can leverage unlabeled data.
- For supervised learning: more data-efficient.

For logistic regression (discriminative) and naive Bayes (generative) [NJ01],

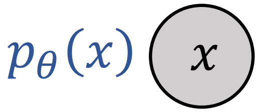
$$\epsilon_{\mathrm{Dis},N} \leq \epsilon_{\mathrm{Dis},\infty} + O\left(\sqrt{\frac{d}{N}}\right)$$

$$\epsilon_{\mathrm{Gen},N} \leq \epsilon_{\mathrm{Gen},\infty} + O\left(\sqrt{\frac{\log d}{N}}\right)$$

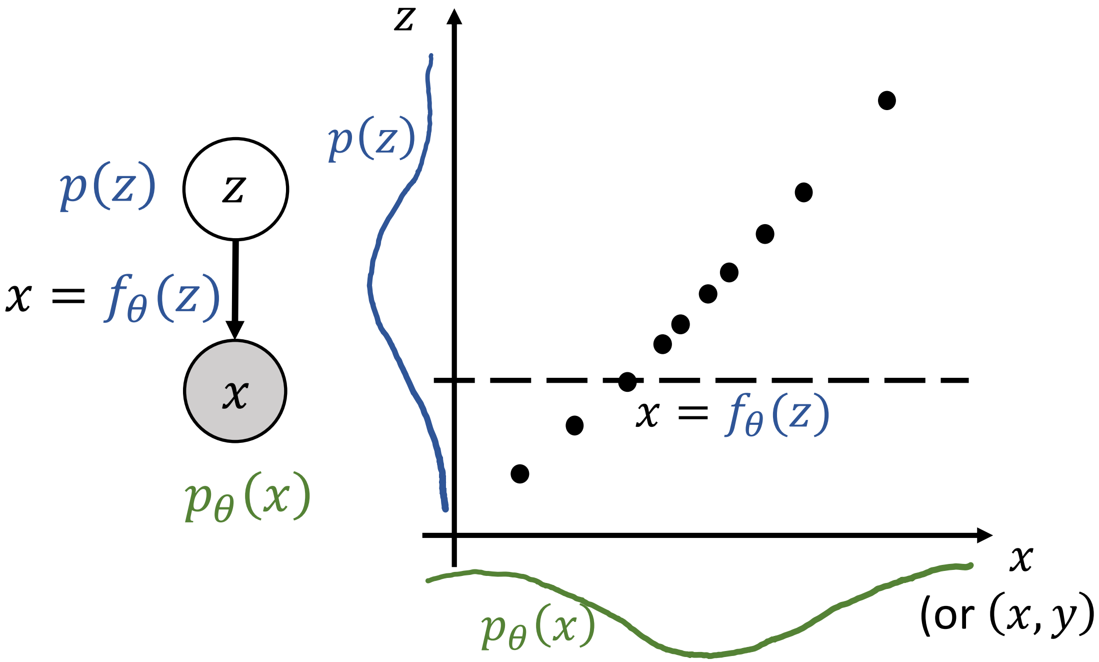
d: data dimension.
N: data size.

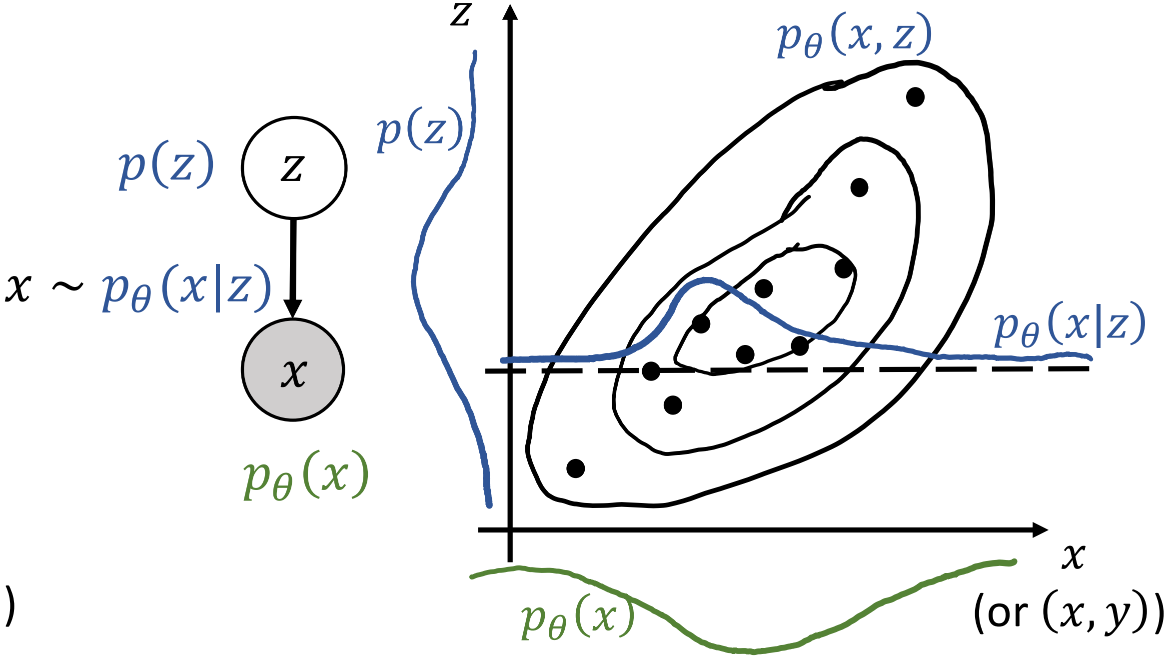
Generative Model: Taxonomy

- Plain Generative Models: Directly model $p(x)$; no latent variable.



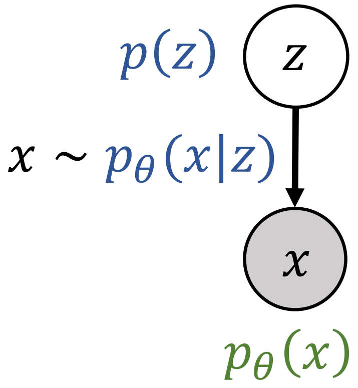
- Latent Variable Models:
  - Deterministic Generative Models: Dependency between x and z is *deterministic*: $x = f_\theta(z)$.
  - Bayesian Generative Models: Dependency between x and z is *probabilistic*: $(x, z) \sim p_\theta(x, z)$.

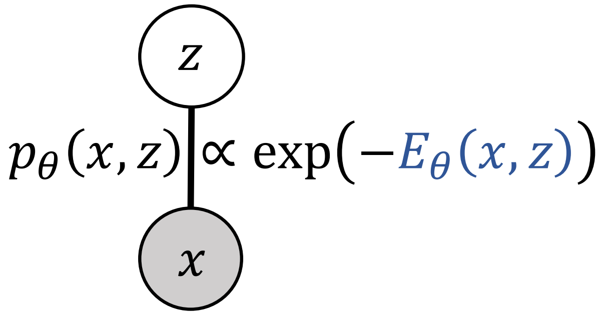




Generative Model: Taxonomy

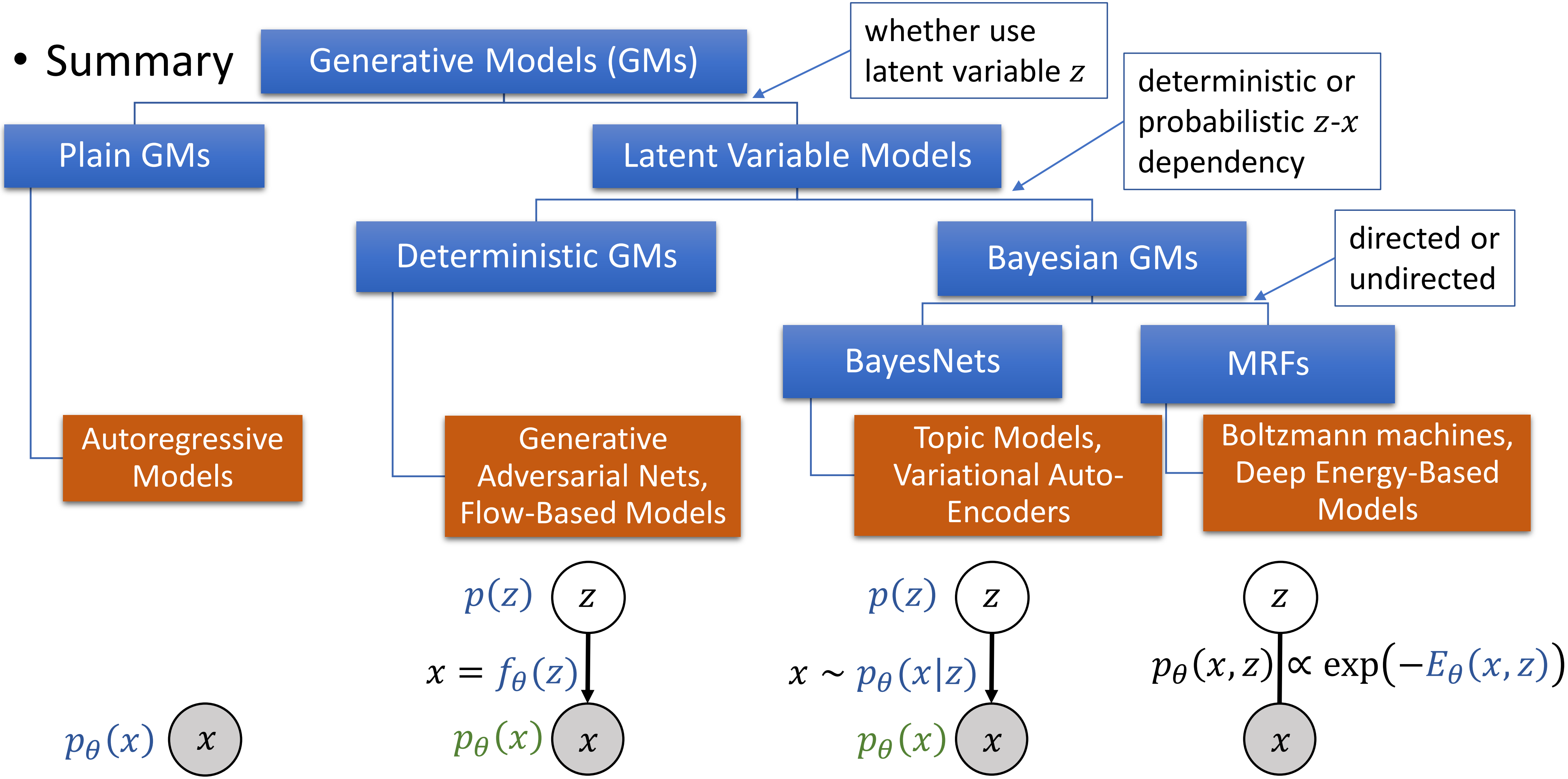
- Latent Variable Models
  - Bayesian Generative Models
    - Bayesian Network (BayesNet): $p(x,z)$ specified by $p(z)$ and $p(x|z)$.
      - Synonyms: Causal Networks, *Directed* Graphical Model
    - Markov Random Field (MRF): $p(x,z)$ specified by an Energy function $E_\theta(x,z)$: $p_\theta(x,z) \propto \exp\big(-E_\theta(x,z)\big)$.
      - Synonyms: Energy-Based Model, *Undirected* Graphical Model





Generative Model: Taxonomy

- Summary



Generative Models (GMs)

whether use latent variable z

Plain GMs

Latent Variable Models

deterministic or probabilistic z-x dependency

Deterministic GMs

Bayesian GMs

directed or undirected

BayesNets

MRFs

Autoregressive Models

Generative Adversarial Nets, Flow-Based Models

Topic Models, Variational Auto-Encoders

Boltzmann machines, Deep Energy-Based Models

$p_\theta(x)$

$p(z)$

$x = f_\theta(z)$

$p_\theta(x)$

$p(z)$

$x \sim p_\theta(x|z)$

$p_\theta(x)$

$p_\theta(x,z) \propto \exp(-E_\theta(x,z))$

Outline

- Generative Models: Overview
- **Plain Generative Models**
  - **Autoregressive Models**
- Latent Variable Models
  - Deterministic Generative Models
    - Generative Adversarial Nets
    - Flow-Based Generative Models
  - Bayesian Generative Models
    - Bayesian Inference (variational inference, MCMC)
    - Bayesian Networks
      - Topic Models (LDA, LightLDA, sLDA)
      - Deep Bayesian Models (VAE)
    - Markov Random Fields (Boltzmann machines, deep energy-based models)

Plain Generative Models

- Directly model $p(x)$; no latent variable involved.
- Easy to learn (no normalization constant issue) and use (generation).
- Learning: Maximum Likelihood Estimation (MLE).

$$\theta^* = \arg\max_\theta \mathbb{E}_{\hat{p}(x)}[\log p_\theta(x)] = \arg\min_\theta \mathrm{KL}(\hat{p}, p_\theta)$$

$$\approx \arg\max_\theta \frac{1}{N} \sum_{n=1}^{N} \log p_\theta(x^{(n)}).$$

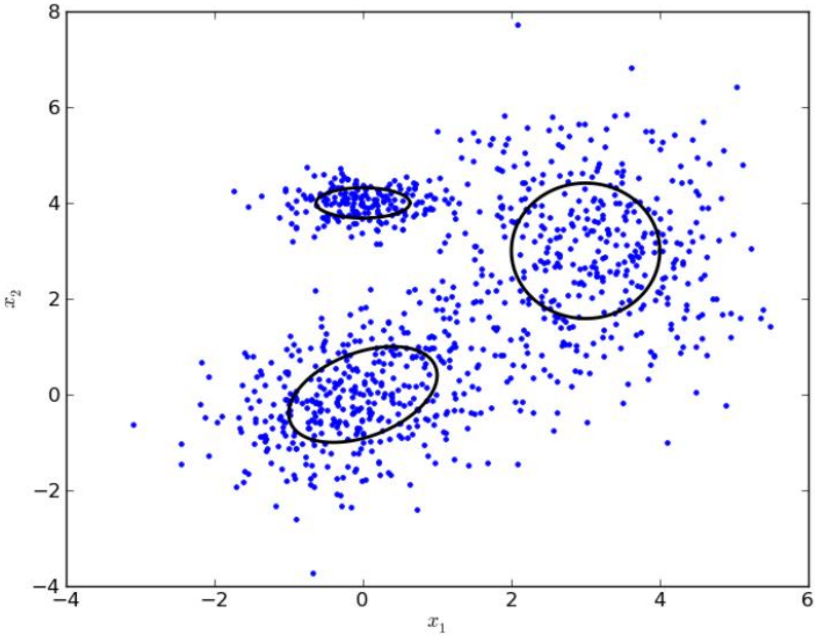
Kullback-Leibler divergence

$\mathrm{KL}(\hat{p}, p_\theta) \coloneqq \mathbb{E}_{\hat{p}(x)}[\log(\hat{p}/p_\theta)]$

- First example: Gaussian Mixture Model

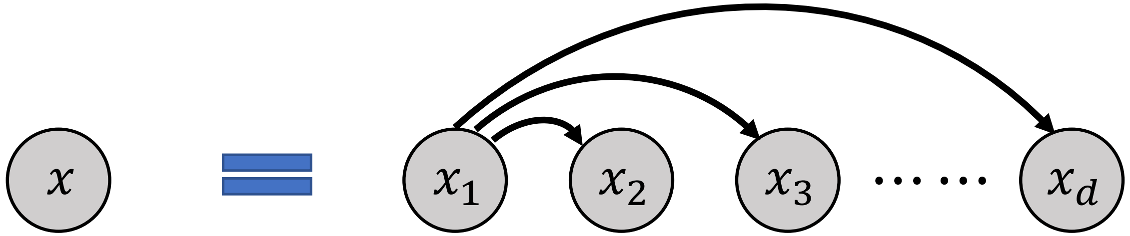
$$p_\theta(x) = \sum_{k=1}^{K} \alpha_k \mathcal{N}(x|\mu_k, \Sigma_k),$$

$$\theta = (\alpha, \mu, \Sigma).$$



Plain Generative Models

- Autoregressive Model:



$p(x_1, x_2, \dots, x_d)$ $p(x_1)\ p(x_2|x_1)\ p(x_3|x_1, x_2) \cdots p(x_d|x_{<d})$

Model $p(x)$ by each conditional $p(x_i|x_{<i})$ (i indices components).

  - Full dependency can be restored.
  - Conditionals are easier to model.
  - Easy learning (MLE).
  - Easy generation:

$$x \sim p(x) \Longleftrightarrow x_1 \sim p(x_1), x_2 \sim p(x_2|x_1), \dots, x_d \sim p(x_d|x_1, \dots, x_{d-1}).$$

But non-parallelizable.

Autoregressive Models

Sigmoid function

$$\sigma(r) = \frac{1}{1+e^{-r}}$$

- Fully Visible Sigmoid Belief Network [Fre98]

$$p(x_i|x_{<i}) = \mathrm{Bern}\left(x_i \middle| \sigma\left(\textstyle\sum_{j<i} W_{ij} x_j\right)\right)$$

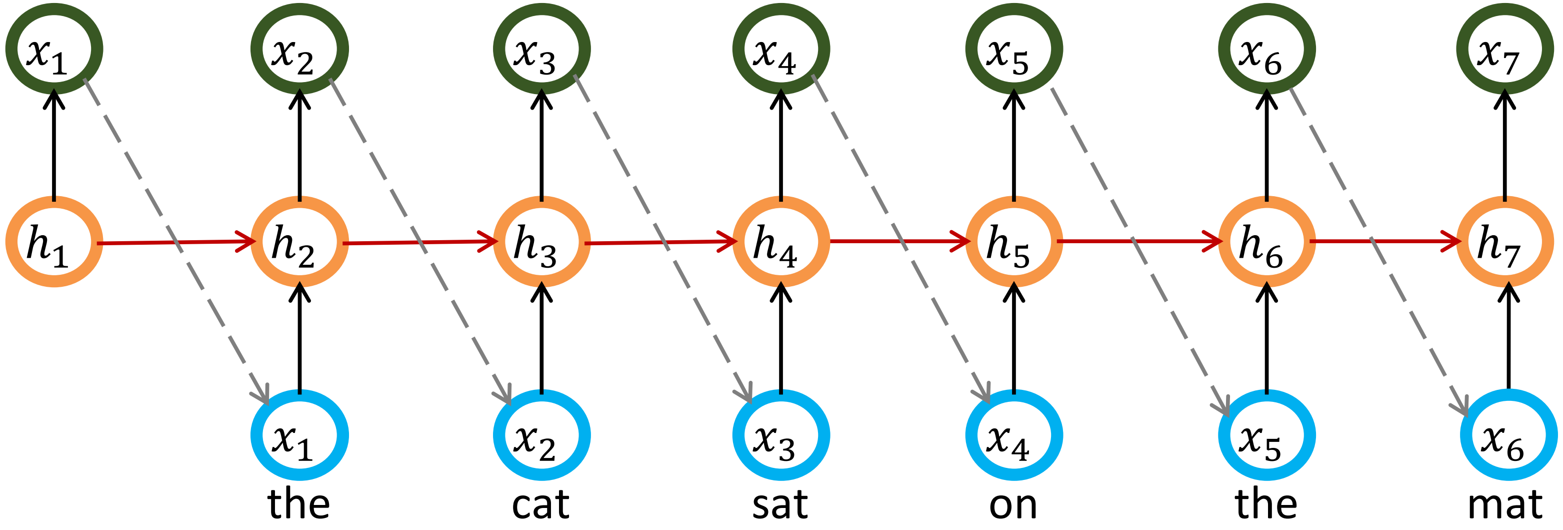
- Neural Autoregressive Distribution Estimator [LM11]

$$p(x_i|x_{<i}) = \mathrm{Bern}\left(x_i \middle| \sigma\left(V_{i,:}\,\sigma\left(W_{:,<i}\,x_{<i} + a\right) + b_i\right)\right)$$

- A typical language model:

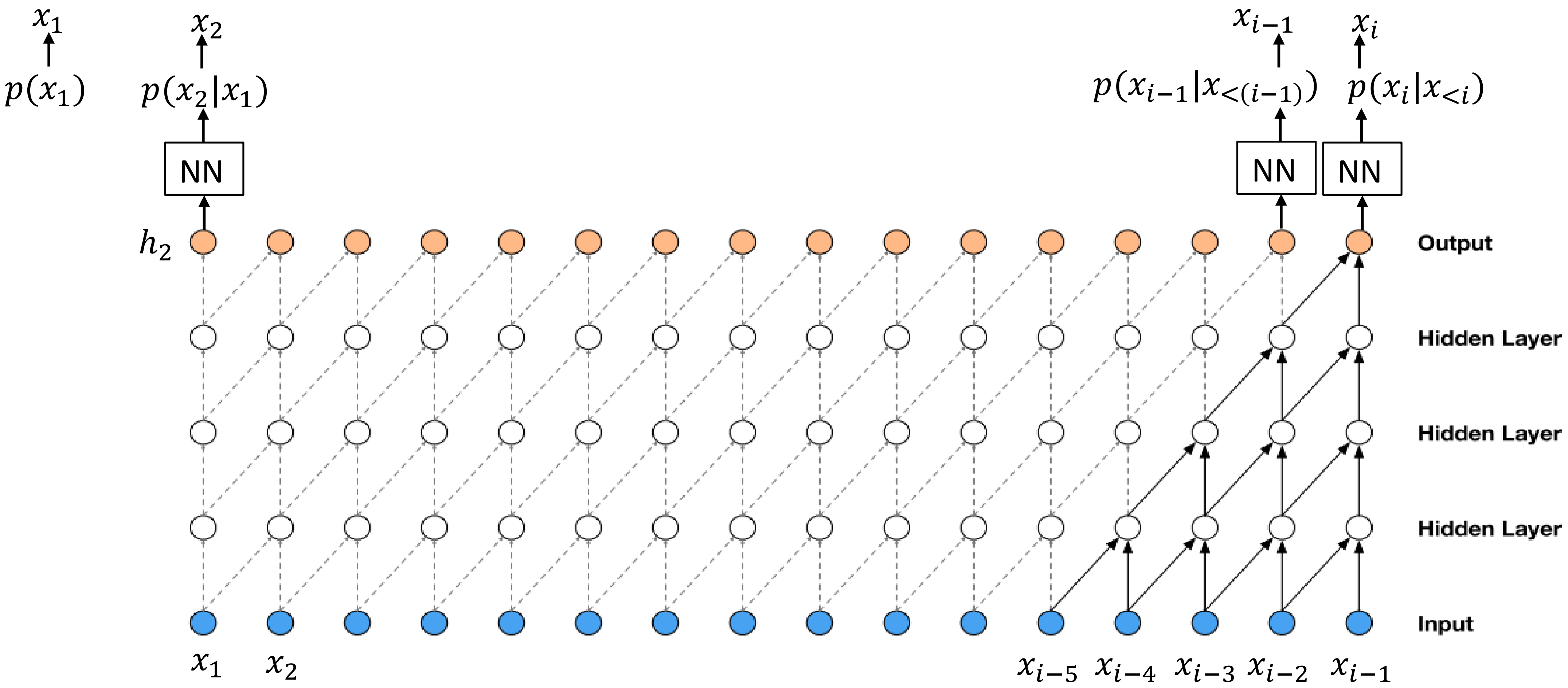
$$p(\textit{“the cat sat on the mat”}) = p(x)$$

$$= p(x_1 = \text{the})\ p(\text{cat}|x_1)\ p(\text{sat}|x_{1\ldots2})\ p(\text{on}|x_{1\ldots3})\ p(\text{the}|x_{1\ldots4})\ p(\text{mat}|x_{1\ldots5})\ p(\text{</s>}|x_{1\ldots6})$$



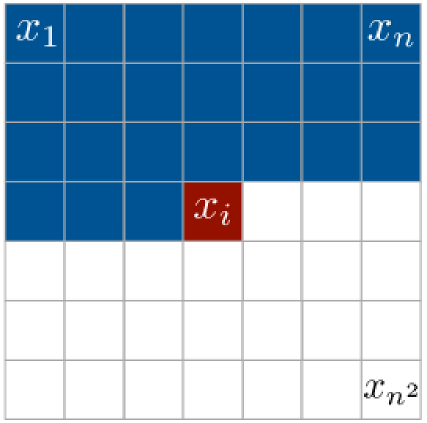
Autoregressive Models

- WaveNet [ODZ+16]
  - Construct $p(x_i|x_{<i})$ via Causal Convolution



Autoregressive Models

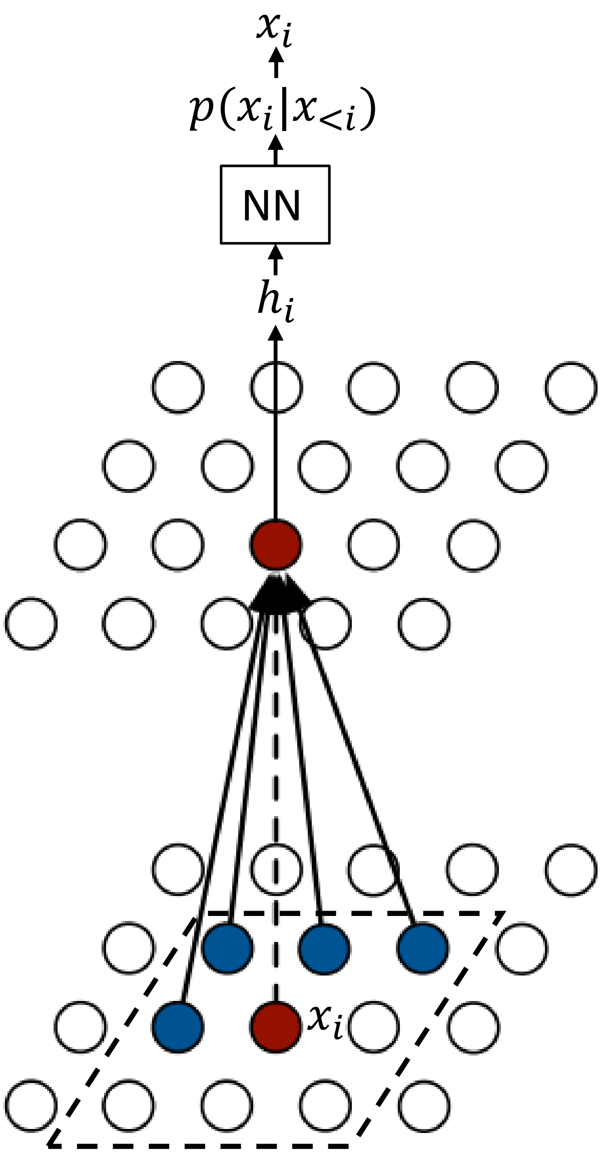
- PixelCNN & PixelRNN [OKK16]
  - Autoregressive structure of an image:



  - PixelCNN: model conditional distributions via (masked) convolution:

$$h_i = K * x_{<i},$$
$$p(x_i|x_{<i}) = \mathrm{NN}(h_i).$$

    - Bounded receptive field.
    - Likelihood evaluation: parallel

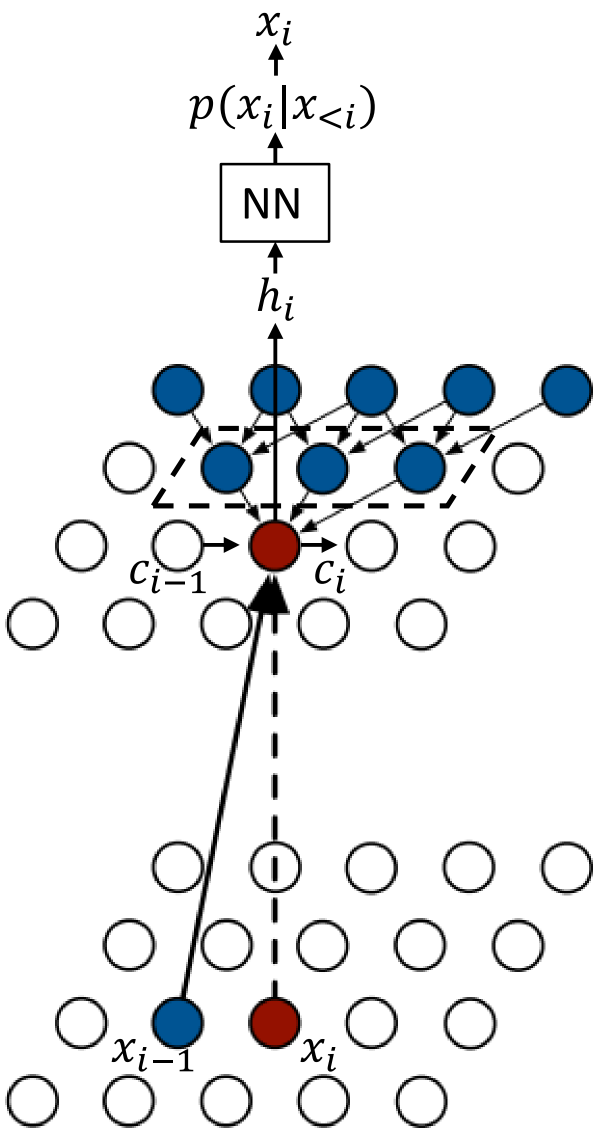


Autoregressive Models

- PixelCNN & PixelRNN [OKK16]
  - PixelRNN: model conditional distributions via recurrent connection:

$$[h_i, c_i] = \text{LSTM}\left(\overbrace{K * h_{(\lfloor i/n \rfloor n - n):\lfloor i/n \rfloor n}}^{\text{1D convolution}}, c_{i-1}, x_{i-1}\right),$$
$$p(x_i | x_{<i}) = \text{NN}(h_i).$$

    - Unbounded receptive field.
    - Likelihood evaluation (in-row): parallel
      Likelihood evaluation (inter-row): sequential



Autoregressive Models

- PixelCNN & PixelRNN [OKK16]

Image Generation

Image Completion

Outline

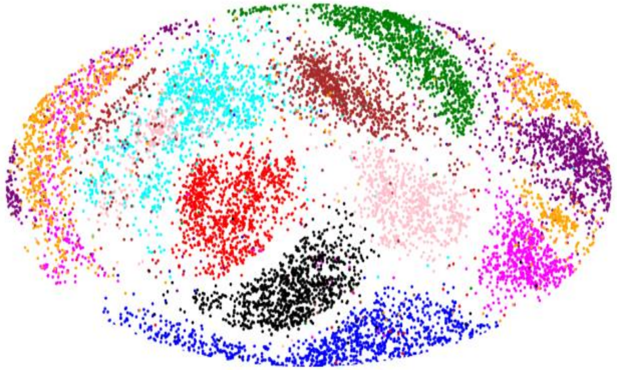
- Generative Models: Overview
- Plain Generative Models
    - Autoregressive Models
- **Latent Variable Models**
    - Deterministic Generative Models
        - Generative Adversarial Nets
        - Flow-Based Generative Models
    - Bayesian Generative Models
        - Bayesian Inference (variational inference, MCMC)
        - Bayesian Networks
            - Topic Models (LDA, LightLDA, sLDA)
            - Deep Bayesian Models (VAE)
        - Markov Random Fields (Boltzmann machines, deep energy-based models)

Latent Variable Models

- Latent Variable:
  - Abstract knowledge of data; enables various tasks.

Knowledge Discovery

"ENGINES"	speed	product	introduced
"ROYAL"	britain	queen	sir
"ARMY"	commander	forces	war
"STUDY"	analysis	space	program
"PARTY"	act	office	judge
"DESIGN"	size	glass	device
"PUBLIC"	report	health	community



Dimensionality Reduction

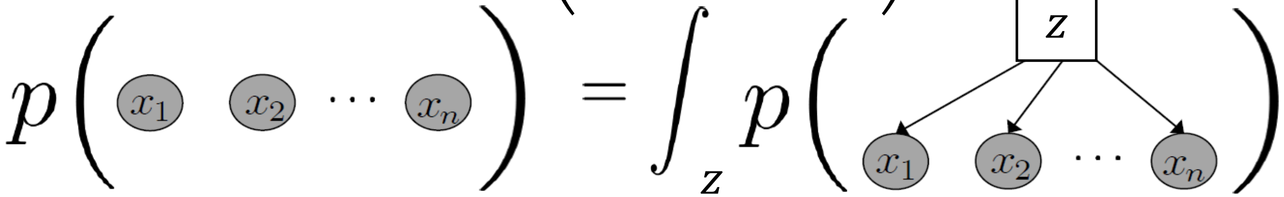
Manipulated Generation

Latent Variable Models

- Latent Variable:
  - Compact representation of dependency.

    De Finetti's Theorem (1955): if $(x_1, x_2, \dots)$ are *infinitely exchangeable*, then ∃ r.v. z and $p(\cdot\,|z)$ s.t. $\forall n$,

$$p(x_1, \dots, x_n) = \int \left(\prod_{i=1}^{n} p(x_i|z) \right) p(z)\,\mathrm{d}z\,.$$



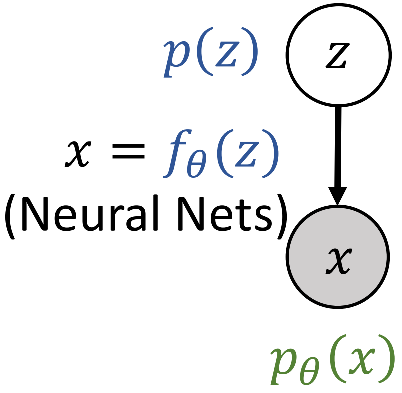
Infinite exchangeability:

For all n and permutation σ, $p(x_1, \dots, x_n) = p\big(x_{\sigma(1)}, \dots, x_{\sigma(n)}\big)$.

Outline

- Generative Models: Overview
- Plain Generative Models
  - Autoregressive Models
- **Latent Variable Models**
  - **Deterministic Generative Models**
    - **Generative Adversarial Nets**
    - Flow-Based Generative Models
  - Bayesian Generative Models
    - Bayesian Inference (variational inference, MCMC)
    - Bayesian Networks
      - Topic Models (LDA, LightLDA, sLDA)
      - Deep Bayesian Models (VAE)
    - Markov Random Fields (Boltzmann machines, deep energy-based models)

Generative Adversarial Nets



- Deterministic $f_\theta: z \mapsto x$, modeled by a neural network.

\+ Flexible modeling ability.

\+ Good generation performance.

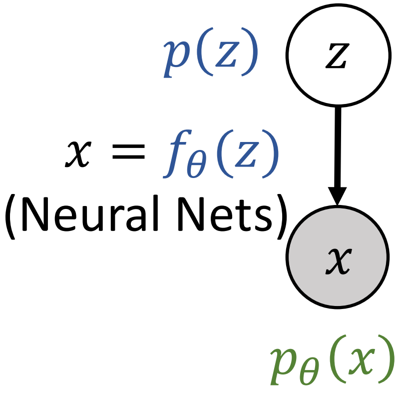
\- Hard to infer z of a data point x.

\- Unavailable density function $p_\theta(x)$.

\- Mode-collapse.

- Learning: $\min_\theta \mathrm{discr}\big(\hat{p}(x), p_\theta(x)\big)$.
  - discr. = $\mathrm{KL}(\hat{p}, p_\theta) \Longrightarrow$ MLE: $\max_\theta \mathbb{E}_{\hat{p}}[\log p_\theta]$, but $p_\theta(x)$ is unavailable!
  - discr. = Jensen-Shannon divergence [GPM+14].
  - discr. = Wasserstein distance [ACB17].

* Generative Adversarial Nets



$p(z)$

z

$x = f_\theta(z)$

(Neural Nets)

x

$p_\theta(x)$

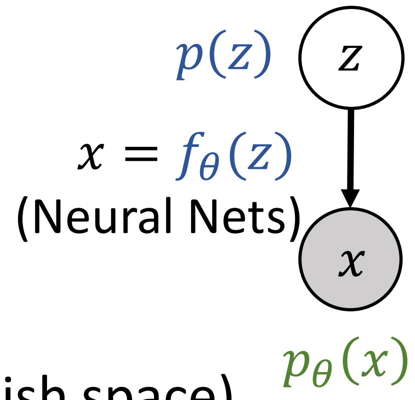
- Learning: $\min_\theta \mathrm{discr}\big(\hat{p}(x), p_\theta(x)\big)$.
  - GAN [GPM+14]: discr. = Jensen-Shannon divergence.

$$\mathrm{JS}(\hat{p}, p_\theta) := \frac{1}{2}\left(\mathrm{KL}\left(\hat{p}, \frac{p_\theta + \hat{p}}{2}\right) + \mathrm{KL}\left(p_\theta, \frac{p_\theta + \hat{p}}{2}\right)\right)$$

$$= \frac{1}{2}\max_{T(\cdot)} \mathbb{E}_{\hat{p}(x)}\left[\log \sigma\big(T(x)\big)\right] + \underbrace{\mathbb{E}_{p_\theta(x)}\left[\log\Big(1 - \sigma\big(T(x)\big)\Big)\right]}_{=\mathbb{E}_{p(z)}\left[\log\left(1-\sigma\left(T\left(f_\theta(z)\right)\right)\right)\right]} + \log 2\,.$$

  - $\sigma\big(T(x)\big)$ is the discriminator; T implemented as a neural network.
  - Expectations can be estimated by samples.

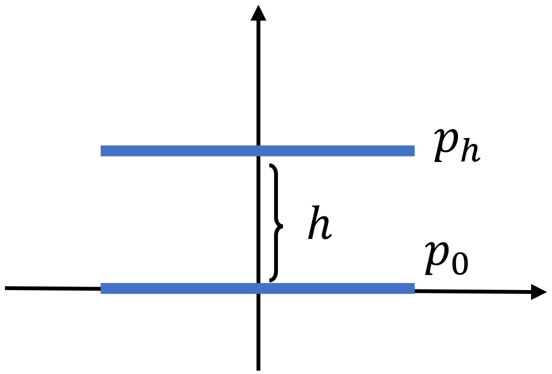
* Generative Adversarial Nets

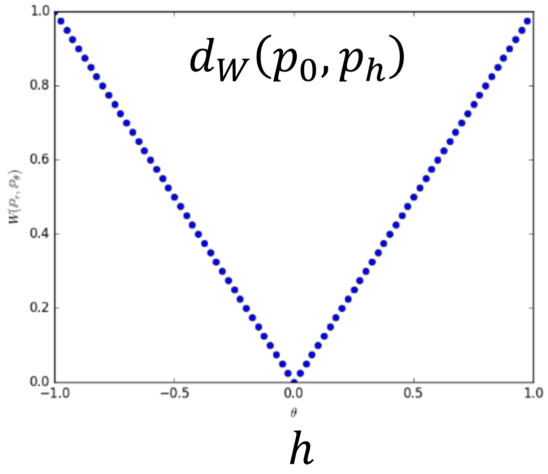


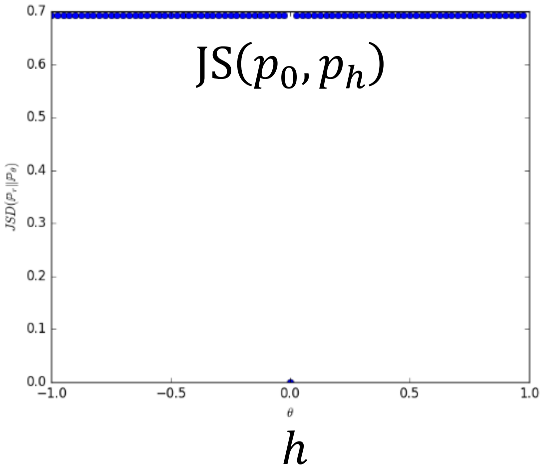
- Learning: $\min_\theta \mathrm{discr}\big(\hat{p}(x), p_\theta(x)\big)$.
  - WGAN [ACB17]: discr. = Wasserstein distance:

$$d_W(\hat{p}, p_\theta) = \inf_{\gamma \in \Gamma(\hat{p}, p_\theta)} \mathbb{E}_{\gamma(x,y)}[c(x,y)]$$
$$= \sup_{\phi \in \mathrm{Lip}_1} \mathbb{E}_{\hat{p}}[\phi] - \mathbb{E}_{p_\theta}[\phi].$$ (if c is a distance in a Polish space)

    - Choose ϕ as a neural network with parameter clipping.
    - Benefit: d_W has more alleviative reaction to distribution difference than JS.



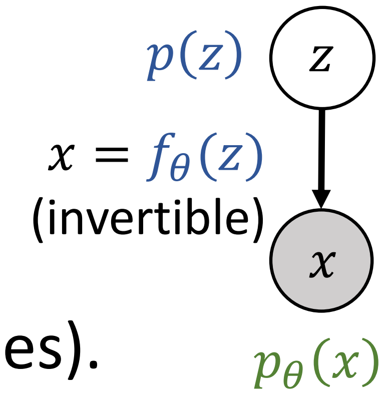




Outline

- Generative Models: Overview
- Plain Generative Models
  - Autoregressive Models
- **Latent Variable Models**
  - **Deterministic Generative Models**
    - Generative Adversarial Nets
    - **Flow-Based Generative Models**
  - Bayesian Generative Models
    - Bayesian Inference (variational inference, MCMC)
    - Bayesian Networks
      - Topic Models (LDA, LightLDA, sLDA)
      - Deep Bayesian Models (VAE)
    - Markov Random Fields (Boltzmann machines, deep energy-based models)

Flow-Based Generative Models



$p(z)$

z

$x = f_\theta(z)$

(invertible)

x

$p_\theta(x)$

- Deterministic and invertible $f_\theta: z \mapsto x$.

\+ Available density function!

$$p_\theta(x) = p\left(z = f_\theta^{-1}(x)\right)\left|\frac{\partial f_\theta^{-1}}{\partial x}\right| \quad \text{(rule of change of variables).}$$

Jacobian Determinant

\+ Easy inference: $z = f_\theta^{-1}(x)$.

\- Redundant representation: dim. z = dim. x.

\- Restricted f_θ: deliberative design; either f_θ or f_θ^{-1} computes costly.

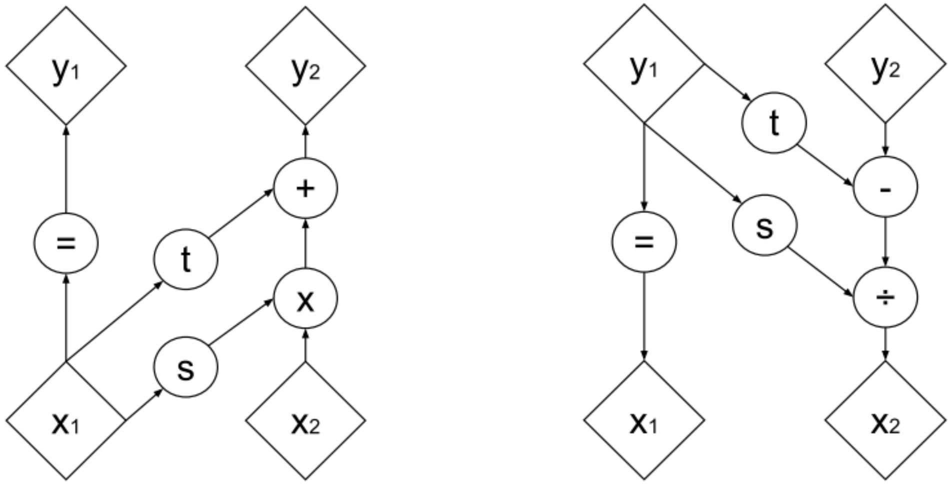
- Learning: $\min_\theta \mathrm{KL}\big(\hat{p}(x), p_\theta(x)\big) \Longrightarrow$ MLE: $\max_\theta \mathbb{E}_{\hat{p}(x)}[\log p_\theta(x)]$.
- NICE [DKB15], RealNVP [DSB17], MAF [PPM17], GLOW [KD18].

* Flow-Based Generative Models

- RealNVP [DSB17]
  - Building block: coupling: $y = g(x)$,

$$\begin{cases} y_{1:d} &= x_{1:d} \\ y_{d+1:D} &= x_{d+1:D} \odot \exp\big(s(x_{1:d})\big) + t(x_{1:d}) \end{cases}$$

$$\Leftrightarrow \begin{cases} x_{1:d} &= y_{1:d} \\ x_{d+1:D} &= \big(y_{d+1:D} - t(y_{1:d})\big) \odot \exp\big(-s(y_{1:d})\big), \end{cases}$$

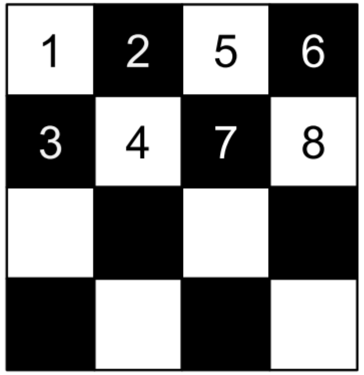


(a) Forward propagation (b) Inverse propagation

where s and t: $\mathbb{R}^{D-d} \rightarrow \mathbb{R}^{D-d}$ are general functions for scale and translation.

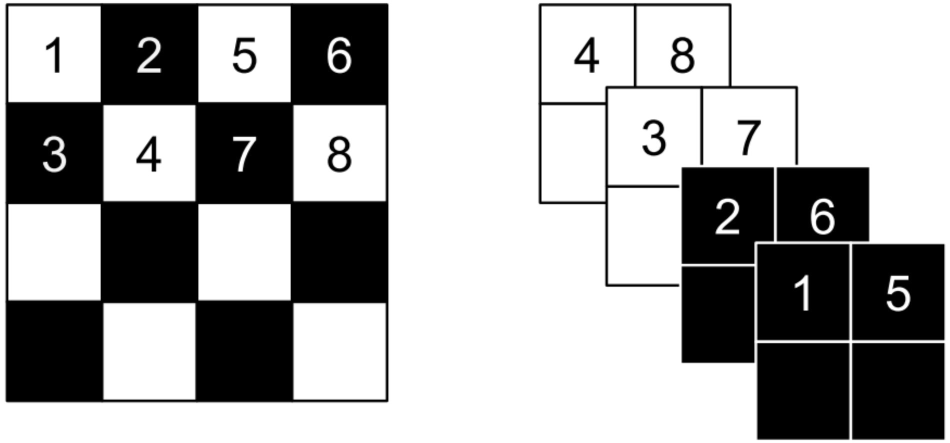
  - Jacobian Determinant: $\left|\frac{\partial g}{\partial x}\right| = \exp\left(\sum_{j=1}^{D-d} s_j(x_{1:d})\right)$.
  - Partitioning x using a binary mask b:

$$y = b \odot x + (1 - b) \odot \Big(x \odot \exp\big(s(b \odot x)\big) + t(b \odot x)\Big).$$



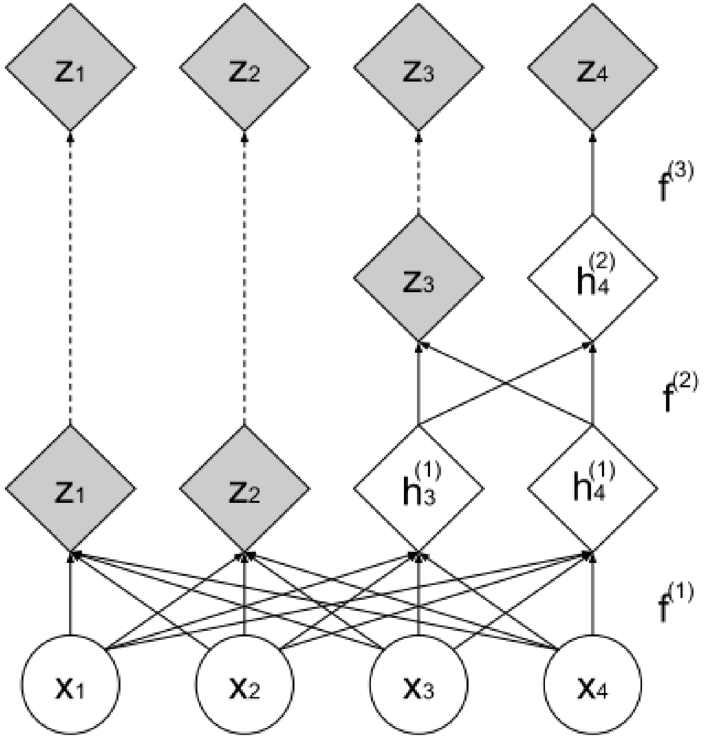
* Flow-Based Generative Models

- RealNVP [DSB17]
  - Building block: squeezing: from $s \times s \times c$ to $\frac{s}{2} \times \frac{s}{2} \times 4c$:



  - Combining with a multi-scale architecture:

$$\begin{aligned} h^{(0)} &= x \\ (z^{(i+1)}, h^{(i+1)}) &= f^{(i+1)}(h^{(i)}) \\ z^{(L)} &= f^{(L)}(h^{(L-1)}) \\ z &= (z^{(1)}, \ldots, z^{(L)}). \end{aligned}$$



where each f follows a “coupling-squeezing-coupling” architecture.

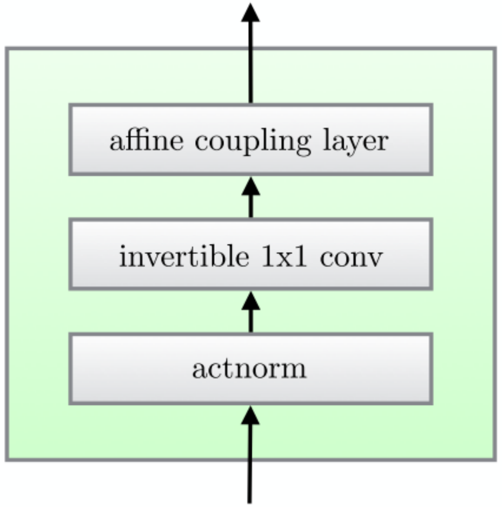
* Flow-Based Generative Models

- RealNVP [DSB17]

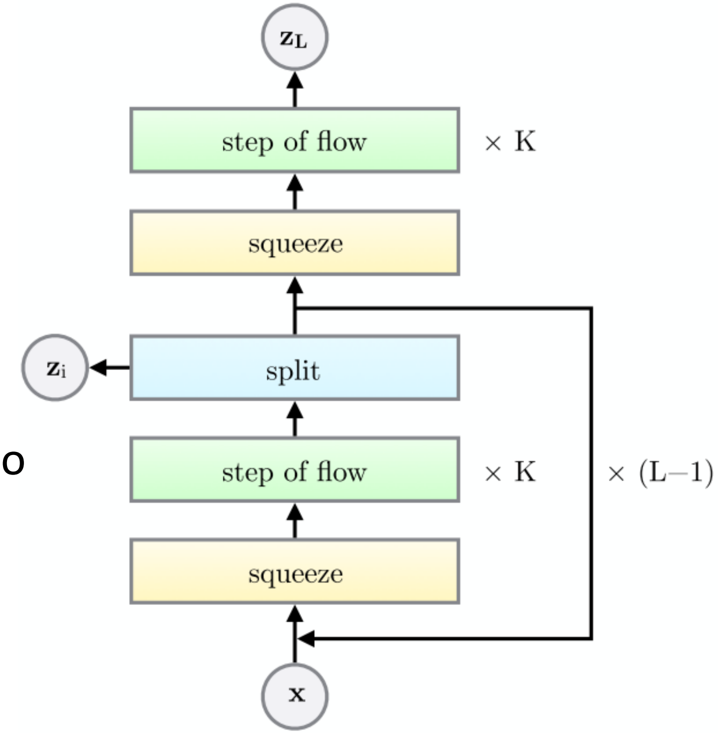
Flow-Based Generative Models

- GLOW [KD18]

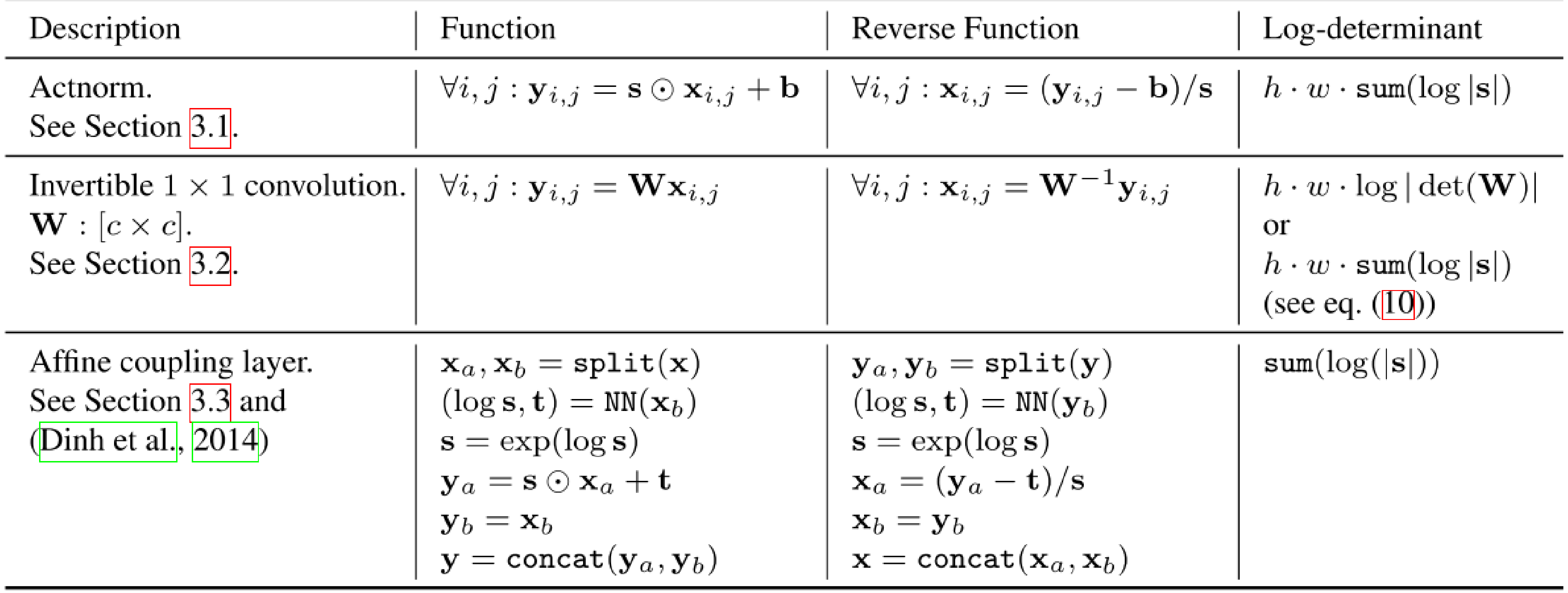
One step of f_θ



Combination of the steps to form f_θ



Component Details



Description	Function	Reverse Function	Log-determinant
Actnorm. See Section 3.1.	$\forall i,j : \mathbf{y}_{i,j} = \mathbf{s} \odot \mathbf{x}_{i,j} + \mathbf{b}$	$\forall i,j : \mathbf{x}_{i,j} = (\mathbf{y}_{i,j} - \mathbf{b})/\mathbf{s}$	$h \cdot w \cdot \texttt{sum}(\log \lvert\mathbf{s}\rvert)$
Invertible 1 × 1 convolution. $\mathbf{W} : [c \times c]$. See Section 3.2.	$\forall i,j : \mathbf{y}_{i,j} = \mathbf{W}\mathbf{x}_{i,j}$	$\forall i,j : \mathbf{x}_{i,j} = \mathbf{W}^{-1}\mathbf{y}_{i,j}$	$h \cdot w \cdot \log \lvert\det(\mathbf{W})\rvert$ or $h \cdot w \cdot \texttt{sum}(\log \lvert\mathbf{s}\rvert)$ (see eq. (10))
Affine coupling layer. See Section 3.3 and (Dinh et al., 2014)	$\mathbf{x}_a, \mathbf{x}_b = \texttt{split}(\mathbf{x})$ $(\log \mathbf{s}, \mathbf{t}) = \texttt{NN}(\mathbf{x}_b)$ $\mathbf{s} = \exp(\log \mathbf{s})$ $\mathbf{y}_a = \mathbf{s} \odot \mathbf{x}_a + \mathbf{t}$ $\mathbf{y}_b = \mathbf{x}_b$ $\mathbf{y} = \texttt{concat}(\mathbf{y}_a, \mathbf{y}_b)$	$\mathbf{y}_a, \mathbf{y}_b = \texttt{split}(\mathbf{y})$ $(\log \mathbf{s}, \mathbf{t}) = \texttt{NN}(\mathbf{y}_b)$ $\mathbf{s} = \exp(\log \mathbf{s})$ $\mathbf{x}_a = (\mathbf{y}_a - \mathbf{t})/\mathbf{s}$ $\mathbf{x}_b = \mathbf{y}_b$ $\mathbf{x} = \texttt{concat}(\mathbf{x}_a, \mathbf{x}_b)$	$\texttt{sum}(\log(\lvert\mathbf{s}\rvert))$

Flow-Based Generative Models

- GLOW [KD18]

Generation Results (Interpolation)

Generation Results (Manipulation; each semantic direction = $\bar{z}_{\text{pos}} - \bar{z}_{\text{neg}}$)

(a) Smiling

(b) Pale Skin

(c) Blond Hair

(d) Narrow Eyes

(e) Young

(f) Male

Outline

- Generative Models: Overview
- Plain Generative Models
    - Autoregressive Models
- **Latent Variable Models**
    - Deterministic Generative Models
        - Generative Adversarial Nets
        - Flow-Based Generative Models
    - **Bayesian Generative Models**
        - Bayesian Inference (variational inference, MCMC)
        - Bayesian Networks
            - Topic Models (LDA, LightLDA, sLDA)
            - Deep Bayesian Models (VAE)
        - Markov Random Fields (Boltzmann machines, deep energy-based models)

Bayesian Generative Models: Overview

Bayesian Networks

- Model structure (*Bayesian Modeling*):
  - *Prior* $p(z)$: *initial* belief of z.
  - *Likelihood* $p(x|z)$: dependence of x on z.
- Learning (*Model Selection*): MLE.

$$\theta^* = \arg\max_{\theta} \mathbb{E}_{\hat{p}(x)}[\log p_\theta(x)],$$

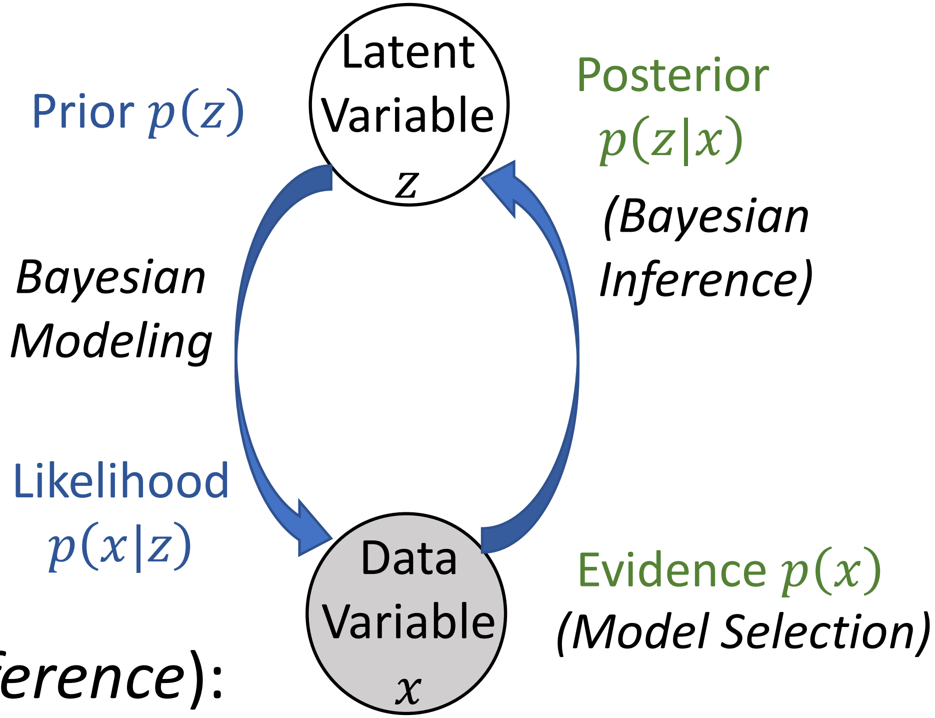
Evidence $p(x) = \int p(z,x) \, \mathrm{d}z$.

- Feature/representation learning (*Bayesian Inference*):

Posterior $p(z|x) = \frac{p(z,x)}{p(x)} = \frac{p(z)p(x|z)}{\int p(z,x)\,\mathrm{d}z}$ (Bayes' rule)

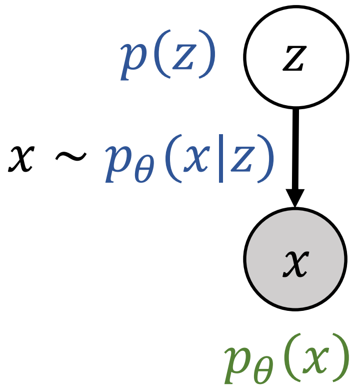
represents the *updated* information that observation x conveys to z.

- Generation/prediction: $z_{\text{new}} \sim p(z|x), x_{\text{new}} \sim p(x|z_{\text{new}})$.

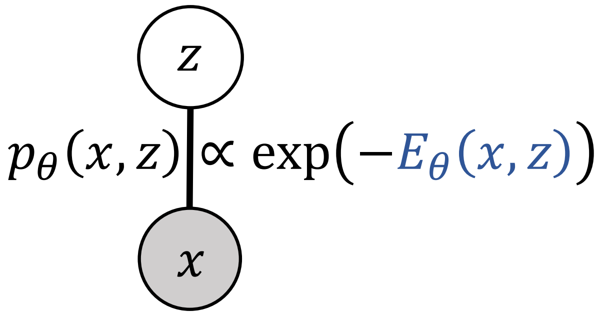


Bayesian Generative Models: Overview

- Dependency between x and z is *probabilistic*: $(x, z) \sim p_\theta(x, z)$.
  - Bayesian Network (BayesNet): $p(x, z)$ specified by $p(z)$ and $p(x|z)$.
    - Synonyms: Causal Networks, *Directed* Graphical Model.
    - Directional/Causal belief encoded: x is generated/caused by z, not the other way.



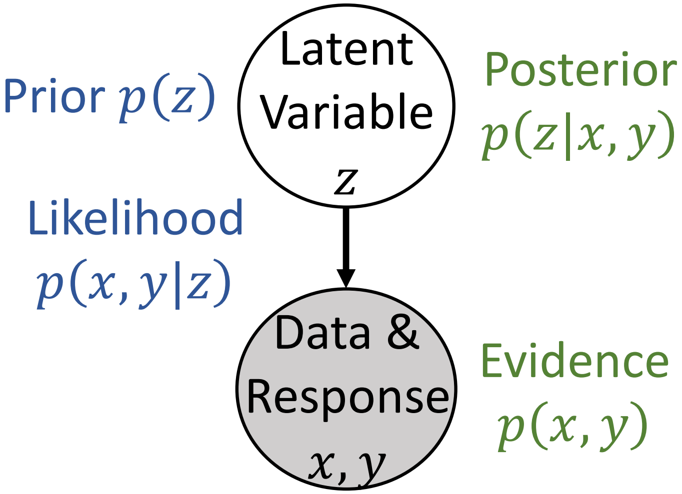
  - Markov Random Field (MRF): $p(x, z)$ specified by an Energy function $E_\theta(x, z)$: $p_\theta(x, z) \propto \exp\big(-E_\theta(x, z)\big)$.
    - Synonyms: Energy-Based Model, *Undirected* Graphical Model.
    - Modeling the symmetric correlation.
    - Harder learning and generation.



* Bayesian Generative Models: Overview

Not all Bayesian models are generative:

Generative Bayesian Models



Prior $p(z)$

Latent Variable z

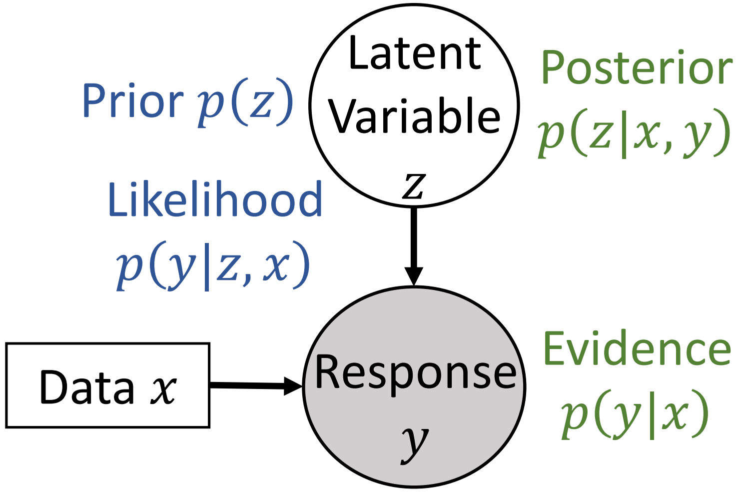
Posterior $p(z|x,y)$

Likelihood $p(x,y|z)$

Data & Response x,y

Evidence $p(x,y)$

Non-Generative Bayesian Models



Prior $p(z)$

Latent Variable z

Posterior $p(z|x,y)$

Likelihood $p(y|z,x)$

Data x

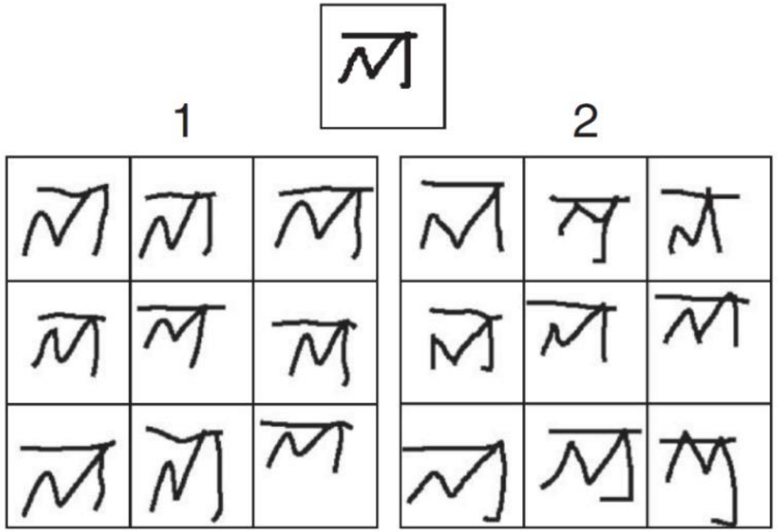
Response y

Evidence $p(y|x)$

	Generative	**Non-generative**
Supervised	Naive Bayes, supervised LDA	Bayesian Logistic Regression, Bayesian Neural Networks
Unsupervised	BayesNets (LDA, VAE), MRFs (BM, RBM, DBM)	(invalid task)

Bayesian Generative Models: Benefits

- Robust to small data and adversarial attack.

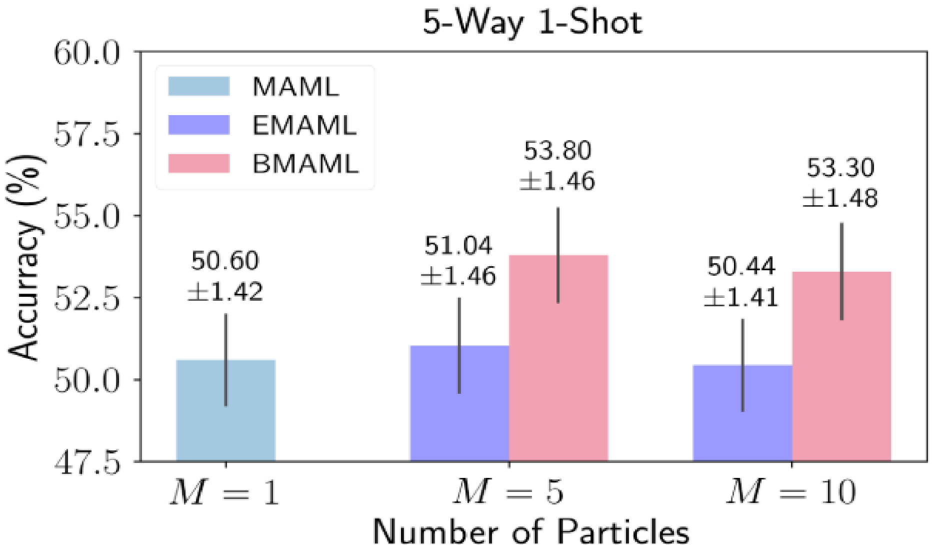


One-shot generation [LST15]

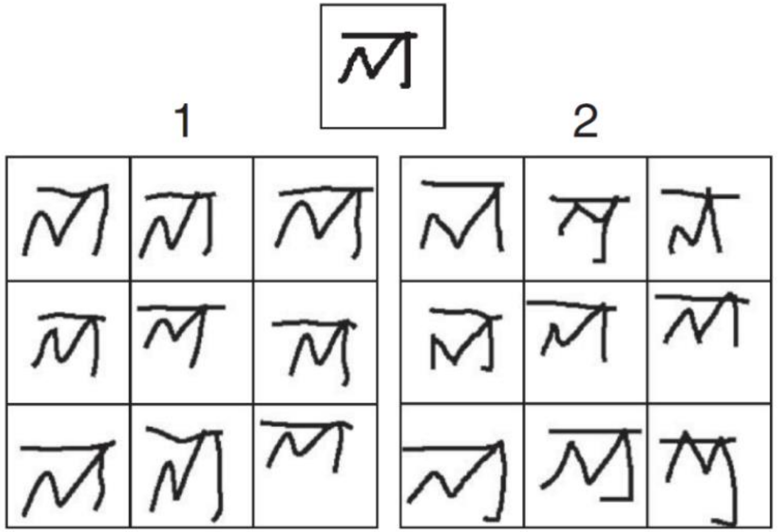
- Stable training process
- Principled and natural inference $p(z|x)$ via Bayes’ rule

* Bayesian Generative Models: Benefits

- Robust to small data and adversarial attack.



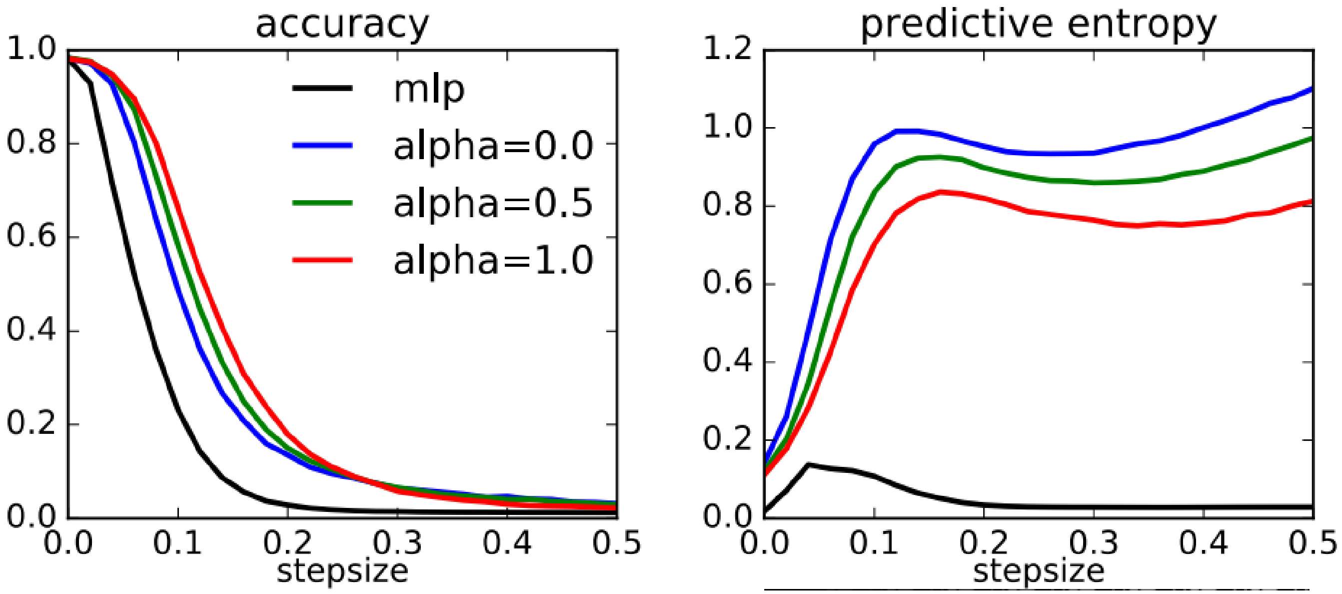
Meta-learning [KYD+18]



One-shot generation [LST15]

* Bayesian Generative Models: Benefits

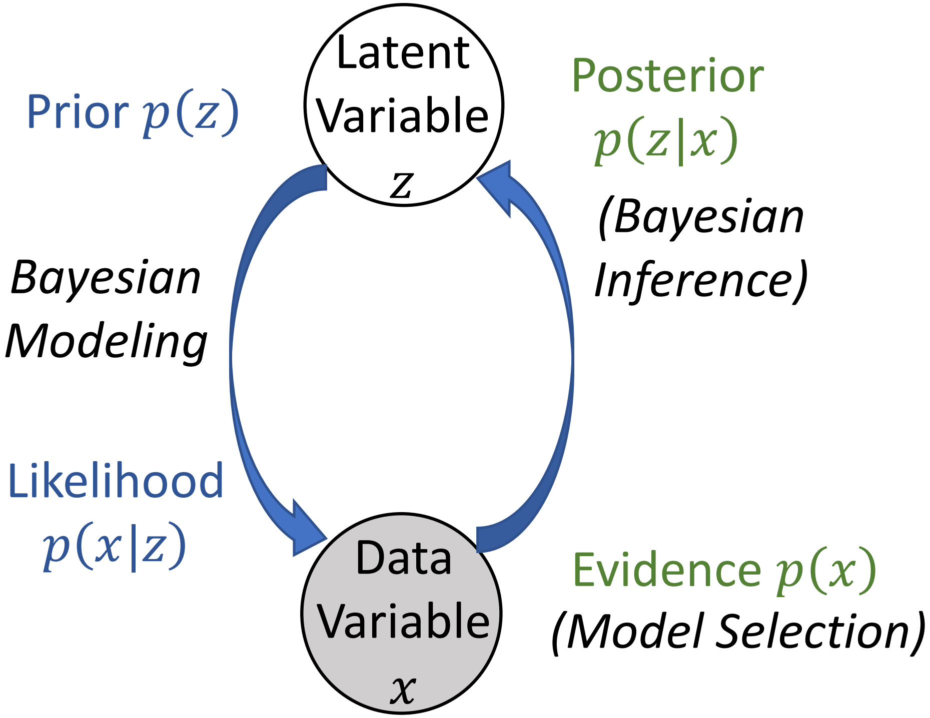
- Robust to small data and adversarial attack.



Adversarial robustness [LG17]
(non-generative case)

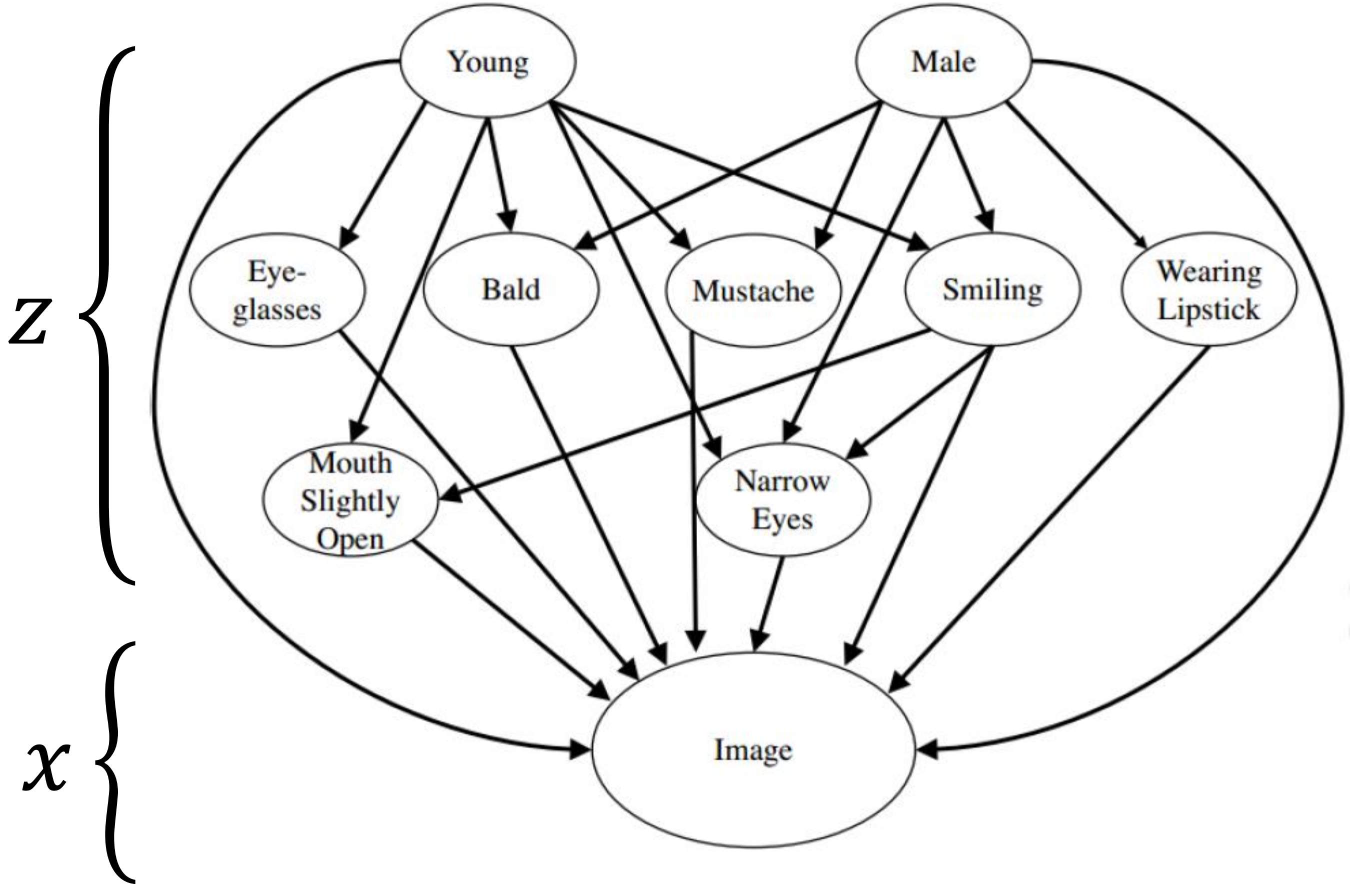
* Bayesian Generative Models: Benefits

- Stable training process
- Principled and natural inference $p(z|x)$ via Bayes' rule



Bayesian Generative Models: Benefits

- Natural to incorporate prior knowledge



Bald

Mustache

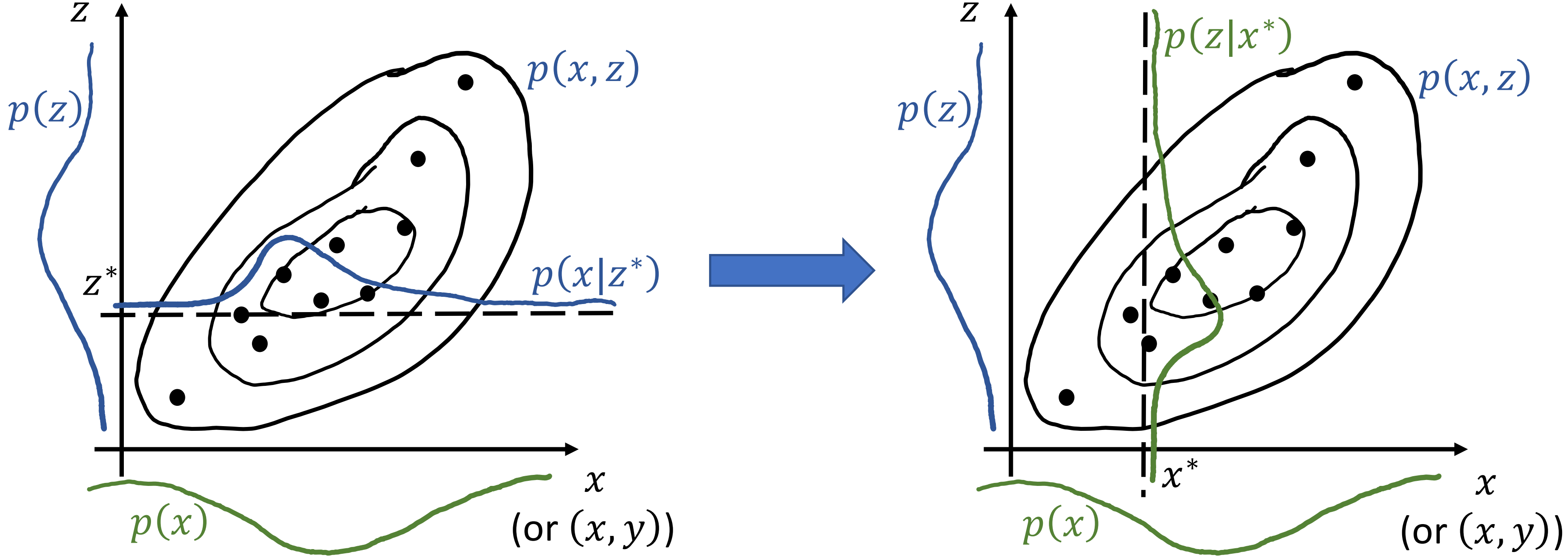
[KSDV18]

Outline

- Generative Models: Overview
- Plain Generative Models
  - Autoregressive Models
- **Latent Variable Models**
  - Deterministic Generative Models
    - Generative Adversarial Nets
    - Flow-Based Generative Models
  - **Bayesian Generative Models**
    - **Bayesian Inference** (variational inference, MCMC)
    - Bayesian Networks
      - Topic Models (LDA, LightLDA, sLDA)
      - Deep Bayesian Models (VAE)
    - Markov Random Fields (Boltzmann machines, deep energy-based models)

Bayesian Inference

Estimate the posterior $p(z|x)$.



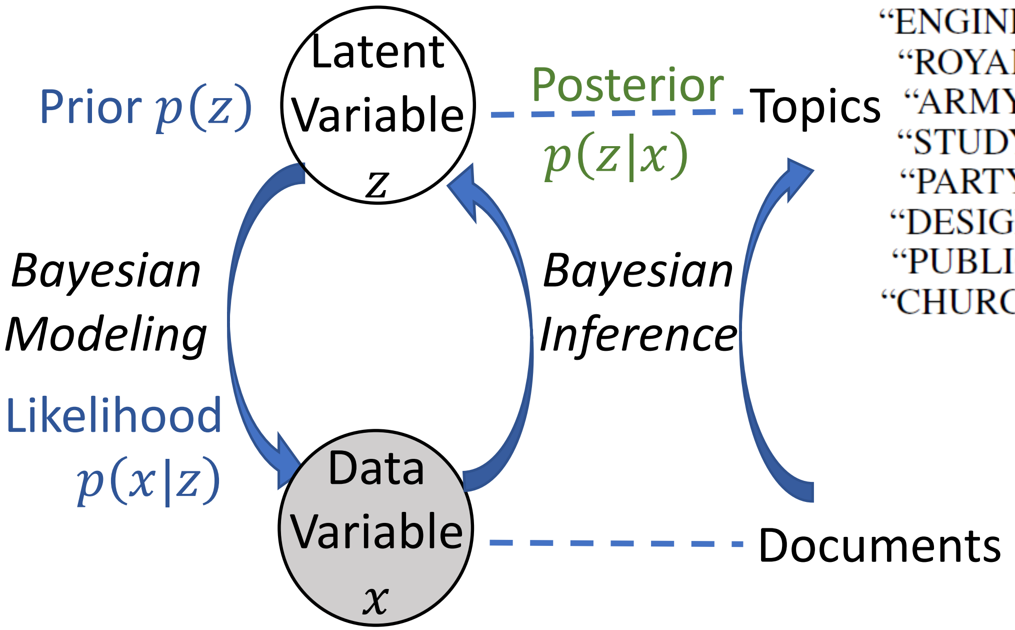
Bayesian Modeling

Bayesian Inference

Bayesian Inference

Estimate the posterior $p(z|x)$.

- Extract knowledge/representation from data.



Prior $p(z)$

Latent Variable z

Posterior $p(z|x)$

Topics

Bayesian Modeling

Bayesian Inference

Likelihood $p(x|z)$

Data Variable x

Documents

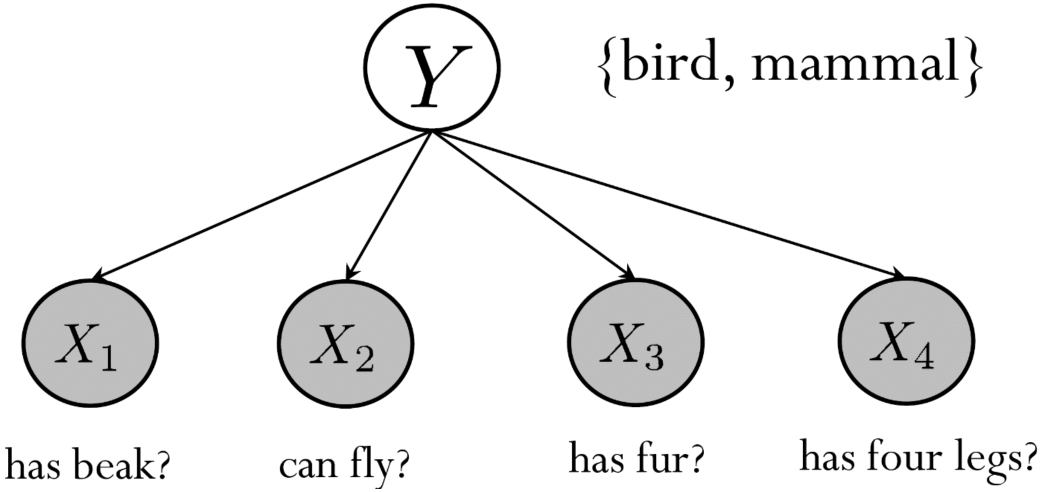
“ENGINES”	speed	product	introduced	designs	fuel
“ROYAL”	britain	queen	sir	earl	died
“ARMY”	commander	forces	war	general	military
“STUDY”	analysis	space	program	user	research
“PARTY”	act	office	judge	justice	legal
“DESIGN”	size	glass	device	memory	engine
“PUBLIC”	report	health	community	industry	conference
“CHURCH”	prayers	communion	religious	faith	historical

* Bayesian Inference

Estimate the posterior $p(z|x)$.

- Extract knowledge/representation from data.

Naive Bayes: $z = y$.



$$p(y=0|x) = \frac{p(x|y=0)p(y=0)}{p(x|y=0)p(y=0) + p(x|y=1)p(y=1)}$$

$f(x) = \arg\max_y p(y|x)$ achieves the lowest error $\int p\big(y = (1 - f(x))|x\big)\, p(x)\, \mathrm{d}x$.

* Bayesian Inference

Estimate the posterior $p(z|x)$.

- Facilitate model learning: $\max_\theta \frac{1}{N}\sum_{n=1}^N \log p_\theta(x^{(n)})$.
  - $p_\theta(x) = \int p_\theta(x|z)p_\theta(z)\,\mathrm{d}z$ is *hard* to evaluate:
    - Closed-form integration is generally unavailable.
    - Numerical integration
      - Curse of dimensionality
      - Hard to optimize.
  - $\log p_\theta(x) = \log \mathbb{E}_{p(z)}[p_\theta(x|z)] \approx \log \frac{1}{N}\sum_n p_\theta\left(x\middle|z^{(n)}\right), \left\{z^{(n)}\right\} \sim p(z)$.
    - Hard for $p(z)$ to cover regions where $p_\theta(x|z)$ is large.
    - $\log \frac{1}{N}\sum_n p_\theta\left(x\middle|z^{(n)}\right)$ is biased:

      $\mathbb{E}\left[\log \frac{1}{N}\sum_n p_\theta\left(x\middle|z^{(n)}\right)\right] \leq \log \mathbb{E}\left[\frac{1}{N}\sum_n p_\theta\left(x\middle|z^{(n)}\right)\right] = \log p_\theta(x)$.

Bayesian Inference

$p_\theta(x) = \int p_\theta(x|z) p_\theta(z) \, \mathrm{d}z$
Hard to evaluate!

Estimate the posterior $p(z|x)$.

- Facilitate model learning: $\max_\theta \frac{1}{N} \sum_{n=1}^{N} \log p_\theta(x^{(n)})$.

  An effective and practical learning approach:

  - Introduce a *variational distribution* $q(z)$:

    $$\log p_\theta(x) = \mathcal{L}_\theta[q(z)] + \mathrm{KL}\big(q(z), p_\theta(z|x)\big),$$
    $$\mathcal{L}_\theta[q(z)] := \mathbb{E}_{q(z)}[\log p_\theta(z, x)] - \mathbb{E}_{q(z)}[\log q(z)].$$

    - $\mathcal{L}_\theta[q(z)] \leq \log p_\theta(x)$ ➔ Evidence Lower BOund (ELBO)!
    - $\mathcal{L}_\theta[q(z)]$ is easier to estimate.

  - (Variational) Expectation-Maximization Algorithm:

    (a) E-step: Let $\mathcal{L}_\theta[q(z)] \approx \log p_\theta(x), \forall\theta \Longleftrightarrow \overbrace{\min_{q \in \mathcal{Q}} \mathrm{KL}\big(q(z), p_\theta(z|x)\big)}^{\text{Bayesian Inference}}$;

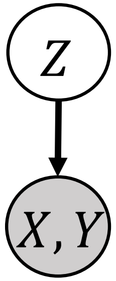
    (b) M-step: $\max_\theta \mathcal{L}_\theta[q(z)]$.

* Bayesian Inference

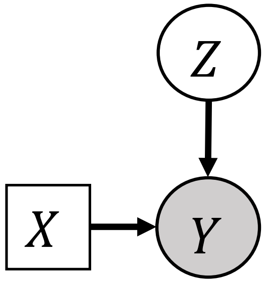
Estimate the posterior $p(z|x)$.

- For prediction:

$$p(y^*|x^*,x,y) = \begin{cases} \int p(y^*|z,x^*)p(z|x^*,x,y)\,\mathrm{d}z\,, \\ \int p(y^*|z,x^*)p(z|x,y)\,\mathrm{d}z\,. \end{cases}$$



(Generative)



(Non-Generative)

* Bayesian Inference

Estimate the posterior $p(z|x)$.

- It is a hard problem
  - Closed form of $p(z|x) \propto p(z)p(x|z)$ is generally intractable.
  - We care about *expectations* w.r.t $p(z|x)$ (prediction, computing ELBO).
    - So that even if we know the closed form (e.g., by numerical integration), downstream tasks are still hard.
    - So that the Maximum *a Posteriori* (MAP) estimate

$$\arg\max_z \log p\left(z \middle| \left\{x^{(n)}\right\}_{n=1}^{N}\right) = \arg\max_z \log p(z) + \sum_{n=1}^{N} \log p(x^{(n)}|z)$$

does not help much for Bayesian tasks.

Modeling Method	**Mathematical Problem**
Parametric Method	Optimization
Bayesian Method	Bayesian Inference

Bayesian Inference

- Variational inference (VI)

  Use a *tractable* variational distribution $q(z)$ to approximate $p(z|x)$:
  $$\min_{q \in \mathcal{Q}} \mathrm{KL}\big(q(z), p(z|x)\big) .$$
  Tractability: known density function, or samples are easy to draw.
  - Parametric VI: use a parameter ϕ to represent $q_\phi(z)$.
  - Particle-based VI: use a set of particles $\left\{z^{(i)}\right\}_{i=1}^{N}$ to represent $q(z)$.
- Monte Carlo (MC)
  - Draw samples from $p(z|x)$.
  - Typically by simulating a *Markov chain* (i.e., MCMC) to release requirements on $p(z|x)$.

Bayesian Inference: Variational Inference

$$\min_{q \in \mathcal{Q}} \mathrm{KL}\big(q(z), p(z|x)\big).$$

- Parametric variational inference: use a parameter ϕ to represent $q_\phi(z)$.

  But $\mathrm{KL}\left(q_\phi(z), p_\theta(z|x)\right)$ is hard to compute…

  Recall $\log p_\theta(x) = \mathcal{L}_\theta[q(z)] + \mathrm{KL}\big(q(z), p_\theta(z|x)\big)$,

  so $\min_\phi \mathrm{KL}\left(q_\phi(z), p(z|x)\right) \Leftrightarrow \max_\phi \mathcal{L}_\theta[q_\phi(z)]$.

  The ELBO $\mathcal{L}_\theta[q_\phi(z)] = \mathbb{E}_{q_\phi(z)}[\log p_\theta(z, x)] - \mathbb{E}_{q_\phi(z)}[\log q_\phi(z)]$ is easier to compute.

  - For model-specifically designed $q_\phi(z)$, $\mathrm{ELBO}(\theta, \phi)$ has closed form (e.g., [SJJ96] for SBN, [BNJ03] for LDA).

* Bayesian Inference: Variational Inference

$$\min_{q \in \mathcal{Q}} \mathrm{KL}\big(q(z), p(z|x)\big).$$

- Parametric variational inference: use a parameter ϕ to represent $q_\phi(z)$.
  - Information theory perspective of the ELBO: Bits-Back Coding [HV93].
    - Average coding length for communicating x after communicating its code z:
      $$\mathbb{E}_{q(z|x)}[-\log p(x|z)].$$
    - Average coding length for communicating z under the bits-back coding:
      $$\mathbb{E}_{q(z|x)}[-\log p(z)] - \mathbb{E}_{q(z|x)}[-\log q(z|x)].$$
      The second term: the receiver knowns the encoder $q(z|x)$ that the sender uses.
    - Average coding length for communicating x with the help of z:
      $$\mathbb{E}_{q(z|x)}[-\log p(x|z) - \log p(z) + \log q(z|x)].$$
      This coincides with the negative ELBO!

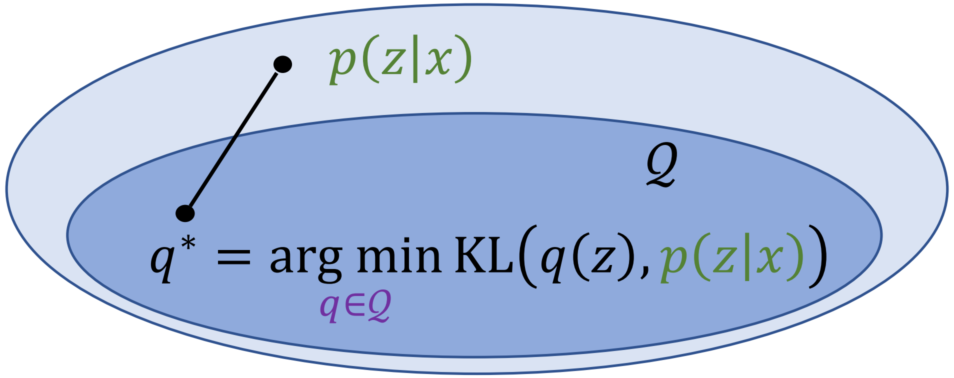
      Maximize ELBO = Minimize averaged coding length under the bits-back scheme.

Bayesian Inference: Variational Inference

$$\min_{q \in \mathcal{Q}} \mathrm{KL}\big(q(z), p(z|x)\big).$$

- Parametric variational inference: use a parameter ϕ to represent $q_\phi(z)$.

  Main Challenge:
  - $\mathcal{Q}$ should be as large/general/flexible as possible,
  - while enables practical optimization of the ELBO.



Bayesian Inference: Variational Inference

- Parametric variational inference: use a parameter ϕ to represent $q_\phi(z)$.

$$\max_{\phi}\left(\mathcal{L}_\theta[q_\phi(z)] = \mathbb{E}_{q_\phi(z)}[\log p_\theta(z,x)] - \mathbb{E}_{q_\phi(z)}[\log q_\phi(z)]\right).$$

  - Explicit variational inference: specify the form of the density function $q_\phi(z)$.
    - Model-specific $q_\phi(z)$: [SJJ96] for SBN, [BNJ03] for LDA.
    - [GHB12, HBWP13, RGB14]: model-agnostic $q_\phi(z)$ (e.g., mixture of Gaussians).
    - [RM15, KSJ+16]: define $q_\phi(z)$ by a flow-based generative model.
  - Implicit variational inference: define $q_\phi(z)$ by a GAN-like generative model.
    - More flexible but more difficult to optimize.
    - Density ratio estimation: [MNG17, SSZ18a].

$$\mathcal{L}_\theta[q_\phi(z)] = \mathbb{E}_{q_\phi(z)}[\log p_\theta(x|z)] - \mathbb{E}_{q_\phi(z)}\left[\log \frac{q_\phi(z)}{p(z)}\right].$$

    - Gradient Estimation $\nabla \log q_\phi(z)$: [VLBM08, LT18, SSZ18b].

* Bayesian Inference: Variational Inference

- Parametric variational inference: use a parameter ϕ to represent $q_\phi(z)$.

$$\max_{\phi} \left(\mathcal{L}_\theta[q_\phi(z)] = \mathbb{E}_{q_\phi(z)}[\log p_\theta(z,x)] - \mathbb{E}_{q_\phi(z)}\left[\log q_\phi(z)\right] \right).$$

  - Explicit variational inference: specify the form of the density function $q_\phi(z)$.

    To be applicable to any model (model-agnostic $q_\phi(z)$):

    - [GHB12]: mixture of Gaussian $q_\phi(z) = \frac{1}{N}\sum_{n=1}^{N} \mathcal{N}\left(z \middle| \mu_n, \sigma_n^2 I\right)$.

      Blue $= \frac{1}{N}\sum_{n=1}^{N} \mathbb{E}_{\mathcal{N}(\mu_n,\sigma_n^2 I)}[f(z)]$

      $\approx \frac{1}{N}\sum_{n=1}^{N} \mathbb{E}_{\mathcal{N}(\mu_n,\sigma_n^2 I)}[\mathrm{Taylor}_2(f,\mu_n)] = \frac{1}{N}\sum_{n=1}^{N} f(\mu_n) + \frac{\sigma_n^2}{2}\mathrm{tr}\left(\nabla^2 f(\mu_n)\right),$

      Red $\geq -\frac{1}{N}\sum_{n=1}^{N} \log \sum_{j=1}^{N} \mathcal{N}\left(\mu_n \middle| \mu_j, (\sigma_n^2 + \sigma_j^2) I\right) + \log N.$

    - [RGB14]: mean-field $q_\phi(z) = \prod_{d=1}^{D} q_{\phi_d}(z_d)$.
      - $\nabla_\theta \mathcal{L}_\theta[q_\phi] = \mathbb{E}_{q_\phi(z)}[\nabla_\theta \log p_\theta(z,x)]$.
      - $\nabla_\phi \mathcal{L}_\theta[q_\phi] = \mathbb{E}_{q_\phi(z)}\left[\left(\nabla_\phi \log q_\phi(z)\right)\left(\log p_\theta(z,x) - \log q_\phi(z)\right)\right]$

        (similar to REINFORCE [Wil92]) (with variance reduction).

* Bayesian Inference: Variational Inference

- Parametric variational inference: use a parameter ϕ to represent $q_\phi(z)$.

$$\max_{\phi} \left(\mathcal{L}_\theta\left[q_\phi(z)\right] = \mathbb{E}_{q_\phi(z)}[\log p_\theta(z,x)] - \mathbb{E}_{q_\phi(z)}\left[\log q_\phi(z)\right] \right).$$

  - Explicit variational inference: specify the form of the density function $q_\phi(z)$.

  To be more flexible and model-agnostic:

  - [RM15, KSJ+16]: define $q_\phi(z)$ by a generative model:

$$z \sim q_\phi(z) \Leftrightarrow z = g_\phi(\epsilon), \epsilon \sim q(\epsilon),$$

    where g_ϕ is invertible (flow model).

    Density function $q_\phi(z)$ is known!

$$q_\phi(z) = q\left(\epsilon = g_\phi^{-1}(z)\right) \left| \frac{\partial g_\phi^{-1}}{\partial z} \right|. \quad \text{(rule of change of variables)}$$

$$\mathcal{L}_\theta\left[q_\phi\right] = \mathbb{E}_{q(\epsilon)}\left[\log p_\theta(z,x) \Big|_{z=g_\phi(\epsilon)} - \log q_\phi(z) \Big|_{z=g_\phi(\epsilon)}\right].$$

* Bayesian Inference: Variational Inference

- Parametric variational inference: use a parameter ϕ to represent $q_\phi(z)$.

$$\max_{\phi}\left(\mathcal{L}_\theta[q_\phi(z)] = \mathbb{E}_{q_\phi(z)}[\log p_\theta(z,x)] - \mathbb{E}_{q_\phi(z)}[\log q_\phi(z)]\right).$$

  - Implicit variational inference: define $q_\phi(z)$ by a generative model:

$$z \sim q_\phi(z) \Leftrightarrow z = g_\phi(\epsilon), \epsilon \sim q(\epsilon),$$

  where g_ϕ is a general function.

    - More flexible than explicit VIs.
    - Samples are easy to draw, but density function $q_\phi(z)$ is unavailable.
    - $\mathcal{L}_\theta[q_\phi(z)] = \mathbb{E}_{q(\epsilon)}\left[\log p_\theta(x|z)\,|_{z=g_\phi(\epsilon)}\right] - \mathbb{E}_{q(\epsilon)}\left[\log \frac{q_\phi(z)}{p(z)}\,|_{z=g_\phi(\epsilon)}\right].$

  Key Problem:

    - Density Ratio Estimation $r(z) := \frac{q_\phi(z)}{p(z)}$.
    - Gradient Estimation $\nabla \log q(z)$.

* Bayesian Inference: Variational Inference

- Parametric variational inference: use a parameter ϕ to represent $q_\phi(z)$.

$$\max_{\phi} \left(\mathcal{L}_\theta\left[q_\phi(z)\right] = \mathbb{E}_{q_\phi(z)}[\log p_\theta(z, x)] - \mathbb{E}_{q_\phi(z)}\left[\log q_\phi(z)\right] \right).$$

  - Implicit variational inference

    Density Ratio Estimation:

    - [MNG17]: $\log r = \arg\max_{T} \mathbb{E}_{q_\phi(Z)}\left[\log \sigma\big(T(Z)\big)\right] + \mathbb{E}_{p(Z)} \left[\log\Big(1 - \sigma\big(T(Z)\big)\Big)\right]$.

      Also used in [MSJ+15, Hus17, TRB17].

    - [SSZ18a]:

$$r \approx \arg\min_{\hat{r}\in\mathcal{H}} \frac{1}{2} \mathbb{E}_p[(\hat{r} - r)^2] + \frac{\lambda}{2} \|\hat{r}\|_{\mathcal{H}}^2 \approx \frac{1}{\lambda N_q} \mathbf{1}^\top K_q - \frac{1}{\lambda N_p N_q} \mathbf{1}^\top K_{qp} \left(\frac{1}{N_p} K_{pp} + \lambda I \right)^{-1} K_p,$$

      where $K_p(z)_j = K\left(z_j^{(p)}, z\right)$, $\left(K_{qp}\right)_{ij} = K\left(z_i^{(q)}, z_j^{(p)}\right)$, $\left\{z_i^{(q)}\right\}_{i=1}^{N_q} \sim q_\phi(z)$, $\left\{z_j^{(p)}\right\}_{j=1}^{N_p} \sim p(z)$.

    Gradient Estimation:

    - [VLBM08, LT18, SSZ18b].

Bayesian Inference: Variational Inference

$$\min_{q \in \mathcal{Q}} \mathrm{KL}\big(q(z), p(z|x)\big).$$

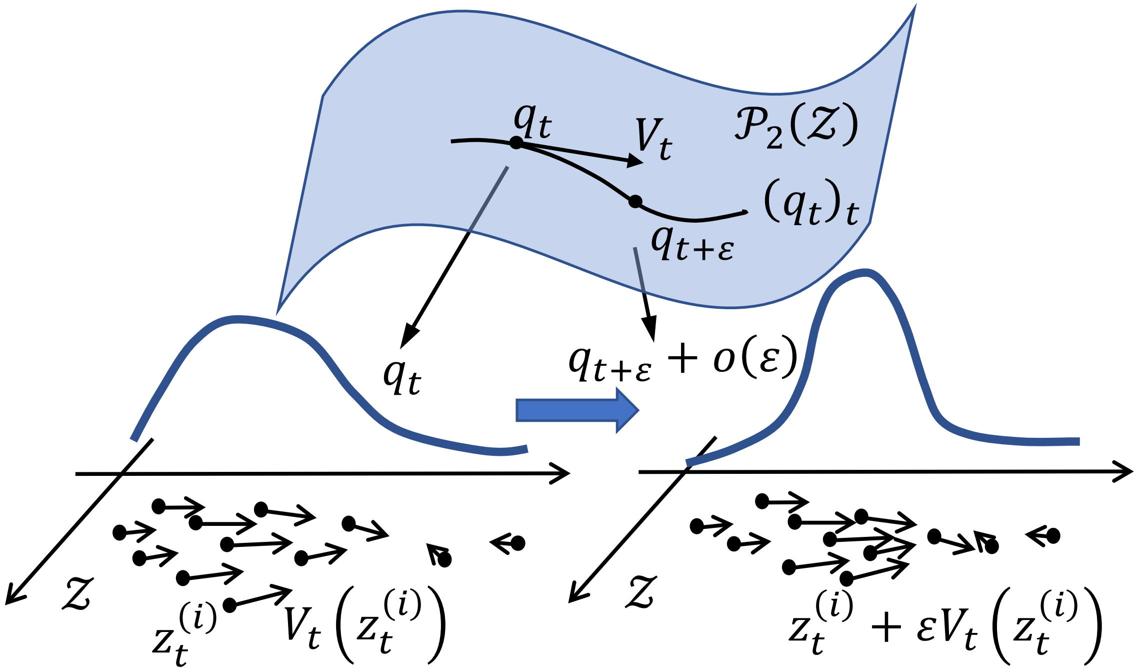
- Particle-based variational inference: use particles $\{z^{(i)}\}_{i=1}^{N}$ to represent $q(z)$.

  To minimize $\mathrm{KL}\big(q(z), p(z|x)\big)$, simulate its gradient flow on the Wasserstein space.

  - Wasserstein space:

    an abstract space of distributions.
  - Wasserstein tangent vector

    $\Leftrightarrow$ vector field.



Bayesian Inference: Variational Inference

$$\min_{q\in\mathcal{Q}} \mathrm{KL}\big(q(z), p(z|x)\big).$$

- Particle-based variational inference: use particles $\{z^{(i)}\}_{i=1}^{N}$ to represent $q(z)$.

$$V := \mathrm{grad}_q \mathrm{KL}(q,p) = \nabla \log(q/p).$$

$$z^{(i)} \leftarrow z^{(i)} + \varepsilon V\big(z^{(i)}\big).$$

$V\big(z^{(i)}\big) \approx$

- SVGD [LW16]: $\sum_j K_{ij} \nabla_{z^{(j)}} \log p\big(z^{(j)}\big|x\big) + \sum_j \nabla_{z^{(j)}} K_{ij}$.
- Blob [CZW+18]: $\nabla_{z^{(i)}} \log p\big(z^{(i)}\big|x\big) - \frac{\sum_j \nabla_{z^{(i)}} K_{ij}}{\sum_k K_{ik}} - \sum_j \frac{\nabla_{z^{(i)}} K_{ij}}{\sum_k K_{jk}}$.
- GFSD [LZC+19]: $\nabla_{z^{(i)}} \log p\big(z^{(i)}\big|x\big) - \frac{\sum_j \nabla_{z^{(i)}} K_{ij}}{\sum_k K_{ik}}$.
- GFSF [LZC+19]: $\nabla_{z^{(i)}} \log p\big(z^{(i)}\big|x\big) + \sum_{j,k} (K^{-1})_{ik} \nabla_{z^{(j)}} K_{kj}$.

$= \sum_j \big(z^{(i)} - z^{(j)}\big) K_{ij}$

for Gaussian Kernel:

Repulsive force!

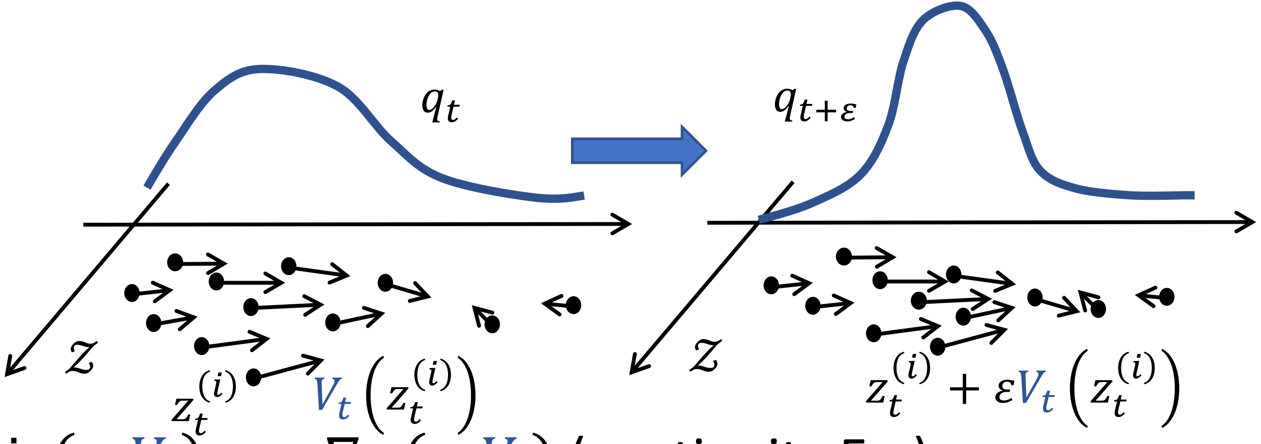
* Bayesian Inference: Variational Inference

$$\min_{q \in \mathcal{Q}} \mathrm{KL}\big(q(z), p(z|x)\big).$$

- Particle-based variational inference: use particles $\left\{z^{(i)}\right\}_{i=1}^{N}$ to represent $q(z)$.

  Non-parametric q: more particles, more flexible.
  - Stein Variational Gradient Descent (SVGD) [LW16]:

    Update the particles by a dynamics $\frac{\mathrm{d}z_t}{\mathrm{d}t} = V_t(z_t)$ so that KL decreases.
    - Distribution evolution: consequence of the dynamics.



$\partial_t q_t = -\mathrm{div}(q_t V_t) = -\nabla \cdot (q_t V_t)$ (continuity Eq.)

* Bayesian Inference: Variational Inference

$$\min_{q\in\mathcal{Q}} \mathrm{KL}\big(q(z), p(z|x)\big).$$

- Particle-based variational inference: use particles $\{z^{(i)}\}_{i=1}^{N}$ to represent $q(z)$.
  - Stein Variational Gradient Descent (SVGD) [LW16]:

Update the particles by a dynamics $\frac{\mathrm{d}z_t}{\mathrm{d}t} = V_t(z_t)$ so that KL decreases.

- Decrease KL:

$$V_t^* := \arg\max_{V_t}\left\{-\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{KL}(q_t,p) = \mathbb{E}_{q_t}\underbrace{[V_t\cdot\nabla\log p + \nabla\cdot V_t]}_{\text{Stein Operator } \mathcal{A}_p[V_t]}\right\}.$$

For tractability,

$$V_t^{\mathrm{SVGD}} := \max\cdot\arg\max_{V_t\in\mathcal{H}^D,\|V_t\|=1}\mathbb{E}_{q_t}[V_t\cdot\nabla\log p + \nabla\cdot V_t]$$

$$= \mathbb{E}_{q(z')}[K(z',\cdot)\nabla_{z'}\log p(z') + \nabla_{z'}K(z',\cdot)].$$

Update rule: $z^{(i)} \mathrel{+}= \varepsilon\left[\sum_j K_{ij}\,\nabla_{z^{(j)}}\log p\big(z^{(j)}\big) + \sum_j \nabla_{z^{(j)}}K_{ij}\right]$.

$= \sum_j \big(z^{(i)} - z^{(j)}\big)K_{ij}$

for Gaussian Kernel:

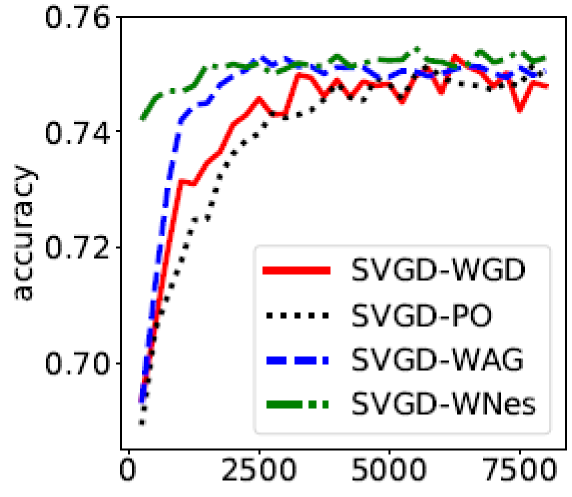
Repulsive force!

* Bayesian Inference: Variational Inference

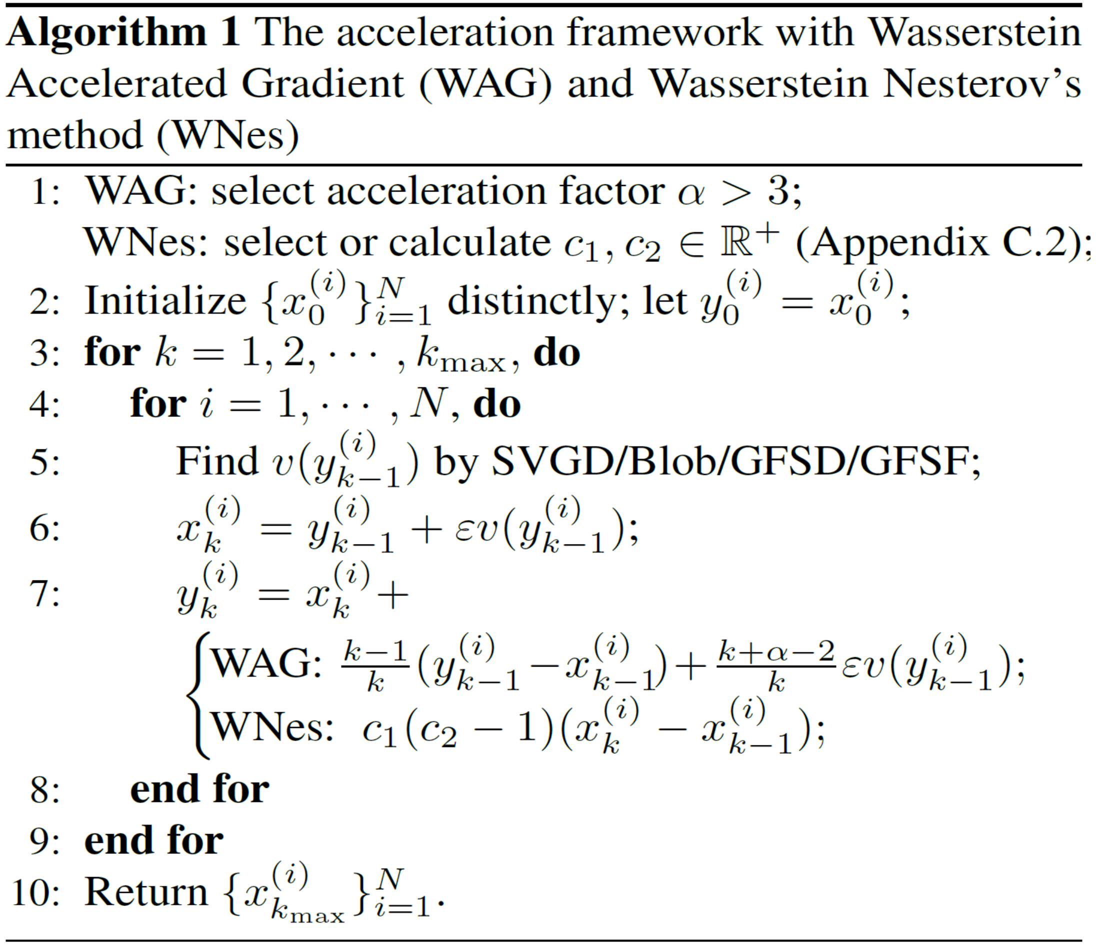
- Particle-based variational inference: use particles $\left\{z^{(i)}\right\}_{i=1}^{N}$ to represent $q(z)$.
  - Unified view as Wasserstein gradient flow (WGF) [LZC+19]: particle-based VIs approximate WGF with a *compulsory* smoothing assumption, in either of the two *equivalent* forms of *smoothing the density* (Blob, GFSD) or *smoothing functions* (SVGD, GFSF).

* Bayesian Inference: Variational Inference

- Particle-based variational inference: use particles $\{z^{(i)}\}_{i=1}^{N}$ to represent $q(z)$.
    - Acceleration on the Wasserstein space [LZC+19]:
        - Apply Riemannian Nesterov's methods to $\mathcal{P}_2(\mathcal{Z})$.



Inference for Bayesian logistic regression

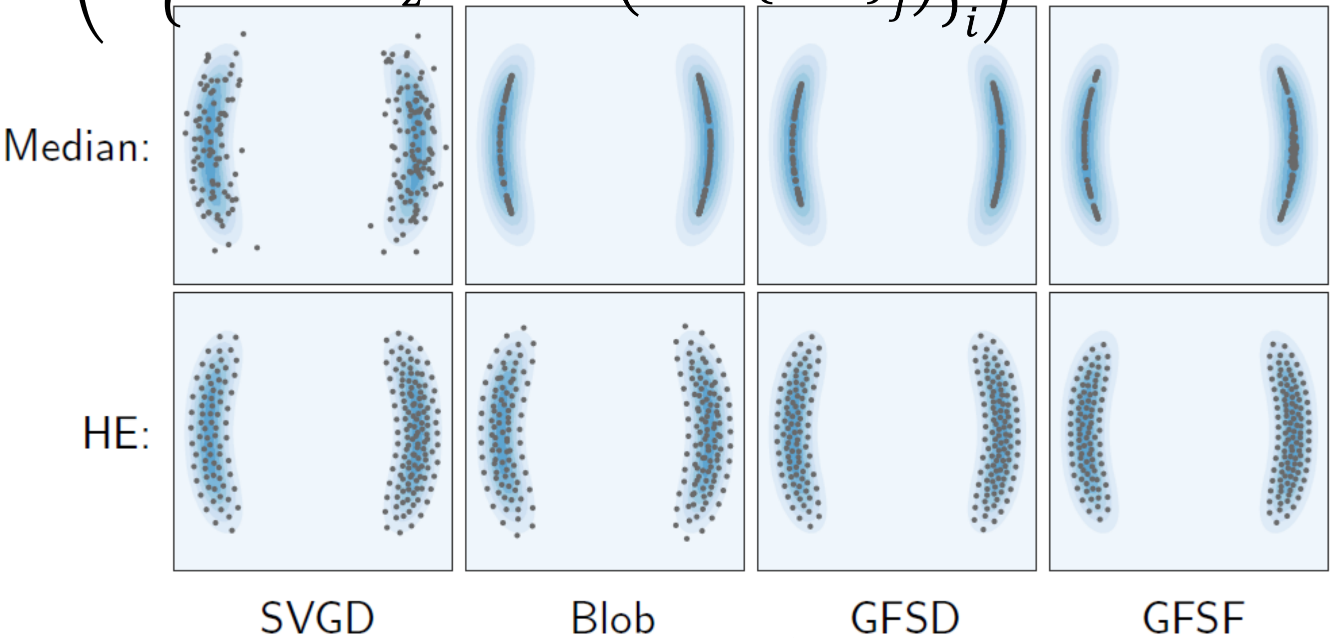


Algorithm 1 The acceleration framework with Wasserstein Accelerated Gradient (WAG) and Wasserstein Nesterov's method (WNes)

```
WAG: select acceleration factor $\alpha > 3$;
WNes: select or calculate $c_1, c_2 \in \mathbb{R}^+$ (Appendix C.2);
Initialize $\{x_0^{(i)}\}_{i=1}^N$ distinctly; let $y_0^{(i)} = x_0^{(i)}$;
for $k = 1, 2, \cdots, k_{\max}$, do
    for $i = 1, \cdots, N$, do
        Find $v(y_{k-1}^{(i)})$ by SVGD/Blob/GFSD/GFSF;
        $x_k^{(i)} = y_{k-1}^{(i)} + \varepsilon v(y_{k-1}^{(i)})$;
        $y_k^{(i)} = x_k^{(i)} + \begin{cases} \text{WAG: } \frac{k-1}{k}(y_{k-1}^{(i)} - x_{k-1}^{(i)}) + \frac{k+\alpha-2}{k}\varepsilon v(y_{k-1}^{(i)}); \\ \text{WNes: } c_1(c_2 - 1)(x_k^{(i)} - x_{k-1}^{(i)}); \end{cases}$
    end for
end for
Return $\{x_{k_{\max}}^{(i)}\}_{i=1}^N$.
```

* Bayesian Inference: Variational Inference

- Particle-based variational inference: use particles $\left\{z^{(i)}\right\}_{i=1}^{N}$ to represent $q(z)$.
  - Kernel bandwidth selection:
    - Median [LW16]: median of pairwise distances of the particles.
    - HE [LZC+19]: the two approx. to $q_{t+\varepsilon}(z)$, i.e., $\tilde{q}\left(z;\left\{z^{(j)}\right\}_j\right) + \varepsilon\Delta_z\tilde{q}\left(z;\left\{z^{(j)}\right\}_j\right)$ (Heat Eq.) and $\tilde{q}\left(z;\left\{z^{(i)} - \varepsilon\nabla_{z^{(i)}}\log\tilde{q}\left(z^{(i)};\left\{z^{(j)}\right\}_j\right)\right\}_i\right)$ (particle evolution), should match.



Bayesian Inference: Variational Inference

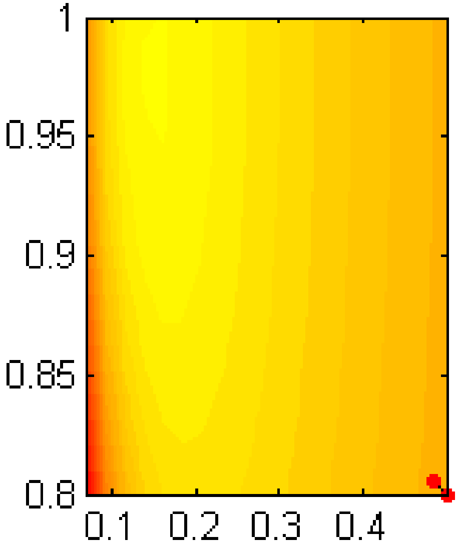
- Particle-based variational inference: use particles $\left\{z^{(i)}\right\}_{i=1}^{N}$ to represent $q(z)$.
  - Unified view as Wasserstein gradient flow: [LZC+19].
  - Asymptotic analysis: SVGD [Liu17] ($N \to \infty, \varepsilon \to 0$).
  - Non-asymptotic analysis
    - w.r.t ε: e.g., [RT96] (as WGF).
    - w.r.t N: [CMG+18, FCSS18, ZZC18].
  - Accelerating ParVIs: [LZC+19, LZZ19].
  - Add particles dynamically: [CMG+18, FCSS18].
  - Solve the Wasserstein gradient by optimal transport: [CZ17, CZW+18].
  - Manifold support space: [LZ18].

Bayesian Inference: MCMC

- Monte Carlo
  - Directly draw (i.i.d.) samples from $p(z|x)$.
  - Almost always impossible to directly do so.
- Markov Chain Monte Carlo (MCMC):

  Simulate a Markov chain whose stationary distribution is $p(z|x)$.
  - Easier to implement: only requires unnormalized $p(z|x)$ ($e.g.$, $p(z,x)$).
  - Asymptotically accurate.
  - Drawback/Challenge: sample auto-correlation.

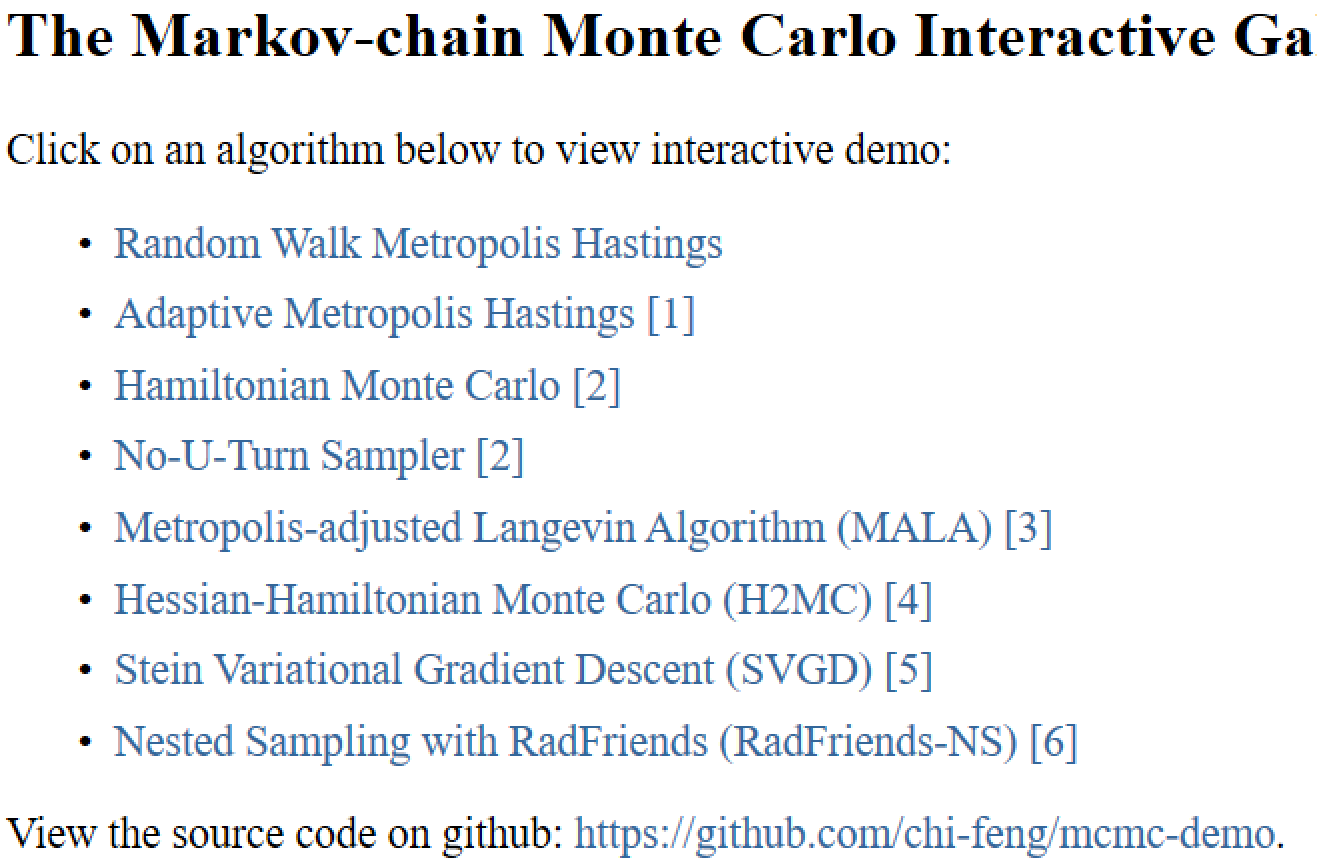
    Less effective than i.i.d. samples.



[GC11]

Bayesian Inference: MCMC

A fantastic MCMC animation site: https://chi-feng.github.io/mcmc-demo/



The Markov-chain Monte Carlo Interactive Gallery

Click on an algorithm below to view interactive demo:

- Random Walk Metropolis Hastings
- Adaptive Metropolis Hastings [1]
- Hamiltonian Monte Carlo [2]
- No-U-Turn Sampler [2]
- Metropolis-adjusted Langevin Algorithm (MALA) [3]
- Hessian-Hamiltonian Monte Carlo (H2MC) [4]
- Stein Variational Gradient Descent (SVGD) [5]
- Nested Sampling with RadFriends (RadFriends-NS) [6]

View the source code on github: https://github.com/chi-feng/mcmc-demo.

Bayesian Inference: MCMC

Classical MCMC

- Metropolis-Hastings framework [MRR+53, Has70]:

  Draw $z^* \sim q\left(z^* | z^{(k)}\right)$ and take $z^{(k+1)}$ as z^* with probability

$$\min\left\{1, \frac{q\left(z^{(k)}\middle|z^*\right)p(z^*|x)}{q\left(z^*\middle|z^{(k)}\right)p\left(z^{(k)}\middle|x\right)}\right\},$$

  else take $z^{(k+1)}$ as $z^{(k)}$.

  Proposal distribution $q(z^*|z)$: e.g., taken as $\mathcal{N}\left(z^*\middle|z, \sigma^2\right)$.

Bayesian Inference: MCMC

Classical MCMC

- Gibbs sampling [GG87]:

Iteratively sample from conditional distributions, which are easier to draw:

$$\begin{aligned}
&z_1^{(1)} \sim p\left(z_1 \middle| \quad z_2^{(0)}, z_3^{(0)}, \dots, z_d^{(0)}, x\right), \\
&z_2^{(1)} \sim p\left(z_2 \middle| z_1^{(1)}, \quad z_3^{(0)}, \dots, z_d^{(0)}, x\right), \\
&z_3^{(1)} \sim p\left(z_3 \middle| z_1^{(1)}, z_2^{(1)}, \quad \dots, z_d^{(0)}, x\right), \\
&\dots, \\
&z_i^{(k+1)} \sim p\left(z_i \middle| z_1^{(k+1)}, \dots, z_{i-1}^{(k+1)}, \quad z_{i+1}^{(k)}, \dots, z_d^{(k)}, x\right).
\end{aligned}$$

Bayesian Inference: MCMC

Dynamics-based MCMC

- Simulates a jump-free continuous-time Markov process (dynamics):

$$\mathrm{d}z = \underbrace{b(z)\,\mathrm{d}t}_{\text{drift}} + \underbrace{\sqrt{2D(z)}\,\mathrm{d}B_t(z)}_{\text{diffusion}},$$

Pos. semi-def. matrix

Brownian motion

$$\Delta z = b(z)\varepsilon + \mathcal{N}(0,2D(z)\varepsilon) + o(\varepsilon),$$

  with appropriate $b(z)$ and $D(z)$ so that $p(z|x)$ is kept stationary/invariant.

- Informative transition using gradient $\nabla_z \log p(z|x)$.
- Some are compatible with *stochastic gradient* (SG): more efficient.

$$\nabla_z \log p(z|x) = \nabla_z \log p(z) + \sum_{n\in\mathcal{D}} \nabla_z \log p\big(x^{(n)}\big|z\big),$$

$$\tilde{\nabla}_z \log p(z|x) = \nabla_z \log p(z) + \frac{|\mathcal{D}|}{|\mathcal{S}|}\sum_{n\in\mathcal{S}} \nabla_z \log p\big(x^{(n)}\big|z\big), \mathcal{S} \subset \mathcal{D}.$$

Bayesian Inference: MCMC

Dynamics-based MCMC

- Langevin Dynamics [RS02] (compatible with SG [WT11, CDC15, TTV16]):

$$z^{(k+1)} = z^{(k)} + \varepsilon \nabla \log p\left(z^{(k)} \middle| x\right) + \mathcal{N}(0,2\varepsilon).$$

- Hamiltonian Monte Carlo [DKPR87, Nea11, Bet17]

  (*incompatible* with SG [CFG14, Bet15]; leap-frog integrator [CDC15]):

$$r^{(0)} \sim \mathcal{N}(0,\Sigma), \qquad \begin{cases} r^{(k+1/2)} = r^{(k)} + (\varepsilon/2) \nabla \log p\left(z^{(k)} \middle| x\right), \\ z^{(k+1)} = z^{(k)} + \varepsilon \Sigma^{-1} r^{(k+1/2)}, \\ r^{(k+1)} = r^{(k+1/2)} + (\varepsilon/2) \nabla \log p\left(z^{(k+1)} \middle| x\right). \end{cases}$$

- Stochastic Gradient Hamiltonian Monte Carlo [CFG14] (compatible with SG):

$$\begin{cases} z^{(k+1)} = z^{(k)} + \varepsilon \Sigma^{-1} r^{(k)}, \\ r^{(k+1)} = r^{(k)} + \varepsilon \nabla \log p\left(z^{(k)} \middle| x\right) - \varepsilon C \Sigma^{-1} r^{(k)} + \mathcal{N}(0,2C\varepsilon). \end{cases}$$

- …

* Bayesian Inference: MCMC

Dynamics-based MCMC

- Langevin dynamics [Lan08]:

$$\mathrm{d}z = \nabla \log p \,\mathrm{d}t + \sqrt{2}\,\mathrm{d}B_t(z).$$

Algorithm (also called Metropolis Adapted Langevin Algorithm) [RS02]:

$$z^{(k+1)} = z^{(k)} + \varepsilon \nabla \log p\left(z^{(k)} \middle| x\right) + \mathcal{N}(0,2\varepsilon),$$

followed by an MH step.

* Bayesian Inference: MCMC

Dynamics-based MCMC

- Hamiltonian Dynamics: $\begin{cases} \mathrm{d}z = \Sigma^{-1} r \, \mathrm{d}t, \\ \mathrm{d}r = \nabla \log p \ \mathrm{d}t. \end{cases}$
- Algorithm: Hamiltonian Monte Carlo [DKPR87, Nea11, Bet17]

  Draw $r^{(0)} \sim \mathcal{N}(0, \Sigma)$ and simulate K steps:

$$\begin{cases} r^{(k+1/2)} = r^{(k)} + (\varepsilon/2) \nabla_z \log p\left(z^{(k)} \middle| x\right), \\ z^{(k+1)} = z^{(k)} + \varepsilon \Sigma^{-1} r^{(k+1/2)}, \\ r^{(k+1)} = r^{(k+1/2)} + (\varepsilon/2) \nabla_z \log p\left(z^{(k+1)} \middle| x\right), \end{cases}$$

  and do an MH step, for one sample of z.

  - Störmer-Verlet (leap-frog) integrator:
    - Makes MH ratio close to 1.
    - Higher-order simulation error [CDC15].
  - More distant exploration than LD (less auto-correlation).

* Bayesian Inference: MCMC

Dynamics-based MCMC: using stochastic gradient (SG).

- Langevin dynamics is compatible with SG [WT11, CDC15, TTV16].
- Hamiltonian Monte Carlo is incompatible with SG [CFG14, Bet15]: the stationary distribution is changed.
- Stochastic Gradient Hamiltonian Monte Carlo [CFG14]:

$$\begin{cases} \mathrm{d}z = \Sigma^{-1} r \, \mathrm{d}t, \\ \mathrm{d}r = \nabla \log p \, \mathrm{d}t - C \Sigma^{-1} r \, \mathrm{d}t + \sqrt{2C} \, \mathrm{d}B_t(r). \end{cases}$$

  - Asymptotically, stationary distribution is p.
  - Non-asymptotically (with Euler integrator), gradient noise is of higher-order of Brownian-motion noise [CDC15].

* Bayesian Inference: MCMC

Dynamics-based MCMC: using stochastic gradient.

- Stochastic Gradient Nose-Hoover Thermostats [DFB+14] (scalar $C > 0$):

$$\begin{cases} \mathrm{d}z = \Sigma^{-1} r \, \mathrm{d}t, \\ \mathrm{d}r = \nabla \log p \, \mathrm{d}t - \xi r \, \mathrm{d}t + \sqrt{2C\Sigma} \, \mathrm{d}B_t(r), \\ \mathrm{d}\xi = \left(\frac{1}{D} r^\top \Sigma^{-1} r - 1 \right) \mathrm{d}t. \end{cases}$$

  - Thermostats $\xi \in \mathbb{R}$: adaptively balance the gradient noise and the Brownian-motion noise.

* Bayesian Inference: MCMC

Dynamics-based MCMC

- Complete recipe for the dynamics [MCF15]:

  For any skew-symmetric matrix Q and pos. semi-def. matrix D, the dynamics
  $$\mathrm{d}z = b(z)\,\mathrm{d}t + \sqrt{2D(z)}\,\mathrm{d}B_t(z),$$
  $$b_i = \frac{1}{p}\sum_j \partial_j \left(p\big(D_{ij} + Q_{ij}\big)\right),$$
  keeps p stationary/invariant.
  - The inverse also holds:
    Any dynamics that keeps p stationary can be cast into this form.
  - If D is pos. def., then p is the unique stationary distribution.
- Integrators and their non-asymptotic analysis (with SG): [CDC15].
- MCMC dynamics as flows on the Wasserstein space: [LZZ19].

Bayesian Inference: MCMC

Dynamics-based MCMC

- Complete framework for MCMC dynamics: [MCF15].
- Interpretation on the Wasserstein space: [JKO98, LZZ19].
- Integrators and their non-asymptotic analysis (with SG): [CDC15].
- For manifold support space:
  - LD: [GC11]; HMC: [GC11, BSU12, BG13, LSSG15]; SGLD: [PT13]; SGHMC: [MCF15, LZS16]; SGNHT: [LZS16]
- Different kinetic energy (other than Gaussian):
  - Monomial Gamma [ZWC+16, ZCG+17].
- Fancy Dynamics:
  - Relativistic: [LPH+16]
  - Magnetic: [TRGT17]

Bayesian Inference: Comparison

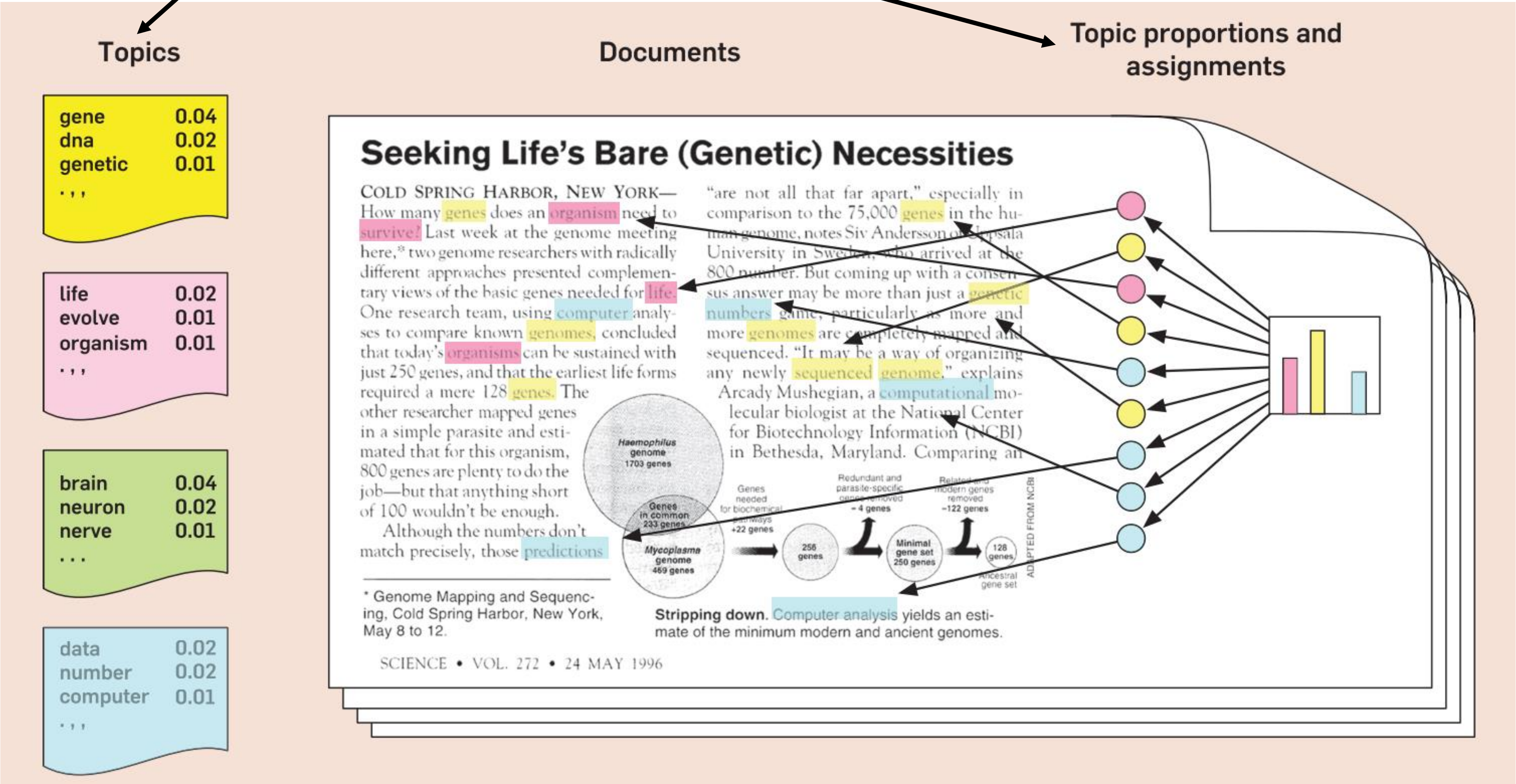
	Parametric VI	**Particle-Based VI**	**MCMC**
Asymptotic Accuracy	No	Yes	Yes
Approximation Flexibility	Limited	Unlimited	Unlimited
Empirical Convergence Speed	High	High	Low
Particle Efficiency	(Do not apply)	High	Low
High-Dimensional Efficiency	High	Low	High

Outline

- Generative Models: Overview
- Plain Generative Models
  - Autoregressive Models
- **Latent Variable Models**
  - Deterministic Generative Models
    - Generative Adversarial Nets
    - Flow-Based Generative Models
  - **Bayesian Generative Models**
    - Bayesian Inference (variational inference, MCMC)
    - **Bayesian Networks**
      - **Topic Models** (LDA, LightLDA, sLDA)
      - Deep Bayesian Models (VAE)
    - Markov Random Fields (Boltzmann machines, deep energy-based models)

Topic Models

Separate *global* (dataset abstraction) and *local* (datum representation) latent variables.



Topics

gene	0.04
dna	0.02
genetic	0.01
...	

life	0.02
evolve	0.01
organism	0.01
...	

brain	0.04
neuron	0.02
nerve	0.01
...	

data	0.02
number	0.02
computer	0.01
...	

Documents

Topic proportions and assignments

Seeking Life's Bare (Genetic) Necessities

COLD SPRING HARBOR, NEW YORK—How many genes does an organism need to survive? Last week at the genome meeting here,* two genome researchers with radically different approaches presented complementary views of the basic genes needed for life. One research team, using computer analyses to compare known genomes, concluded that today's organisms can be sustained with just 250 genes, and that the earliest life forms required a mere 128 genes. The other researcher mapped genes in a simple parasite and estimated that for this organism, 800 genes are plenty to do the job—but that anything short of 100 wouldn't be enough.

Although the numbers don't match precisely, those predictions "are not all that far apart," especially in comparison to the 75,000 genes in the human genome, notes Siv Andersson of Uppsala University in Sweden, who arrived at the 800 number. But coming up with a consensus answer may be more than just a genetic numbers game, particularly as more and more genomes are completely mapped and sequenced. "It may be a way of organizing any newly sequenced genome," explains Arcady Mushegian, a computational molecular biologist at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland. Comparing an

Stripping down. Computer analysis yields an estimate of the minimum modern and ancient genomes.

---

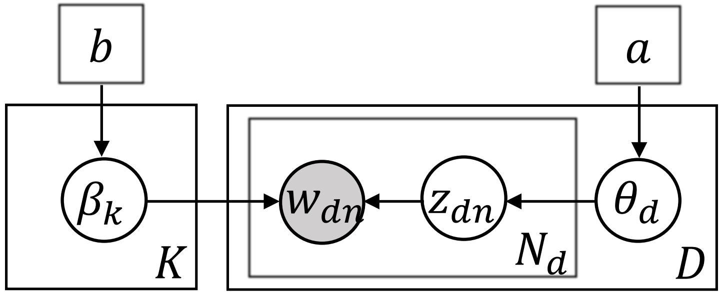
* Genome Mapping and Sequencing, Cold Spring Harbor, New York, May 8 to 12.

SCIENCE • VOL. 272 • 24 MAY 1996

[SG07]

Latent Dirichlet Allocation

Model Structure [BNJ03]:



- Data variable: Words/Documents $w = \{w_{dn}\}_{n=1:N_d, d=1:D}, w_{dn} \in \{1 \dots W\}$.
- Latent variables:
  - *Global*: topics $\beta = \{\beta_k\}_{k=1:K}, \beta_k \in \Delta^W$.
  - *Local*: topic proportions $\theta = \{\theta_d\}, \theta_d \in \Delta^K$,
    topic assignments $z = \{z_{dn}\}, z_{dn} \in \{1 \dots K\}$.
- Prior: $p(\beta_k|b) = \mathrm{Dir}(b), p(\theta_d|a) = \mathrm{Dir}(a), p(z_{dn}|\theta_d) = \mathrm{Mult}(\theta_d)$.
- Likelihood: $p(w_{dn}|z_{dn}, \beta) = \mathrm{Mult}\left(\beta_{z_{dn}}\right)$.

Latent Dirichlet Allocation

Variational inference [BNJ03]:

- Take variational distribution (mean-field approximation):

$$q_{\lambda,\gamma,\phi}(\beta,\theta,z) \coloneqq \prod_{k=1}^{K} \mathrm{Dir}(\beta_k|\lambda_k) \prod_{d=1}^{D} \mathrm{Dir}(\theta_d|\gamma_d) \prod_{n=1}^{N_d} \mathrm{Mult}(z_{dn}|\phi_{dn}) \,.$$

- ELBO$(\lambda,\gamma,\phi;a,b)$ is available in closed form.
- E-step: update λ,γ,ϕ by maximizing ELBO;
- M-step: update a,b by maximizing ELBO.

Latent Dirichlet Allocation

MCMC: Gibbs sampling [GS04]

Model structure $\Longrightarrow p(\beta, \theta, z, w) = AB \left(\prod_{k,w} \beta_{kw}^{N_{kw}+b_w-1} \right) \left(\prod_{d,k} \theta_{dk}^{N_{kd}+a_k-1} \right)$

$\Longrightarrow p(z, w) = AB \left(\prod_k \frac{\prod_w \Gamma(N_{kw}+b_w)}{\Gamma(N_k+W\bar{b})} \right) \left(\prod_d \frac{\prod_k \Gamma(N_{kd}+a_k)}{\Gamma(N_d+K\bar{a})} \right).$

(N_{kw}: #times word w is assigned to topic k; N_{kd}: #times topic k appears in document d.)

- Unacceptable cost to directly compute $p(z|w) = p(z,w)/p(w)$.
- Use Gibbs sampling to draw from $p(z|w)$!

$$p(z_{dn} = k | z^{-dn}, w) \propto \frac{N_{kw}^{-dn} + b_w}{N_k^{-dn} + W\bar{b}} (N_{kd}^{-dn} + a_k).$$

- For β and θ, use MAP estimate:

$$\hat{\beta} \coloneqq \arg\max_{\beta} \log p(\beta|w) \approx \frac{N_{kw} + b_w}{N_k + W\bar{b}},$$

$$\hat{\theta}_{dk} \coloneqq \arg\max_{\theta} \log p(\theta|w) \approx \frac{N_{kd} + a_k}{N_d + K\bar{a}}.$$

Estimated by samples of z

* Latent Dirichlet Allocation

MCMC: Gibbs sampling [GS04]

$$
\begin{aligned}
&p(\beta, \theta, z, w) \\
&= \left(\prod_{k=1}^{K} \mathrm{Dir}(\beta_k | b) \right) \left(\prod_{d=1}^{D} \mathrm{Dir}(\theta_d | a) \left(\prod_{n=1}^{N_d} \mathrm{Mult}(z_{dn} | \theta_d) \mathrm{Mult}\left(w_{dn} \middle| \beta_{z_{dn}}\right) \right) \right) \\
&= AB \left(\prod_{k,w} \beta_{kw}^{b_w - 1} \right) \left(\prod_{d,k} \theta_{dk}^{a_k - 1} \right) \left(\prod_{d,n} \theta_{d z_{dn}} \beta_{z_{dn} w_{dn}} \right) \\
&= AB \left(\prod_{k,w} \beta_{kw}^{N_{kw} + b_w - 1} \right) \left(\prod_{d,k} \theta_{dk}^{N_{kd} + a_k - 1} \right),
\end{aligned}
$$

- $A = \left(\frac{\Gamma(\sum_k a_k)}{\prod_k \Gamma(a_k)} \right)^D, B = \left(\frac{\Gamma(\sum_w b_w)}{\prod_w \Gamma(b_w)} \right)^K$, where $\Gamma(\cdot)$ is the Gamma function.
- $N_{kw} = \sum_{d=1}^{D} \sum_{n=1}^{N_d} \mathbb{I}(w_{dn} = w, z_{dn} = k)$: number of times that word w is assigned to topic k.
- $N_{kd} = \sum_{n=1}^{N_d} \mathbb{I}(z_{dn} = k)$: number of times that topic k appears in document d.

* Latent Dirichlet Allocation

MCMC: Gibbs sampling [GS04]

$$p(\beta, \theta, z, w) = AB \left(\prod_{k,w} \beta_{kw}^{N_{kw}+b_w-1} \right) \left(\prod_{d,k} \theta_{dk}^{N_{kd}+a_k-1} \right),$$

$$N_{kw} = \sum_{d=1}^{D} \sum_{n=1}^{N_d} \mathbb{I}(w_{dn} = w, z_{dn} = k)\, , N_{kd} = \sum_{n=1}^{N_d} \mathbb{I}(z_{dn} = k)\, .$$

- β and θ can be collapsed:

$$p(z, w) = \iint p(\beta, \theta, z, w)\, \mathrm{d}\beta\, \mathrm{d}\theta$$

$$= AB \left(\prod_{k} \frac{\prod_w \Gamma(N_{kw} + b_w)}{\Gamma\left(N_k + W\bar{b}\right)} \right) \left(\prod_{d} \frac{\prod_k \Gamma(N_{kd} + a_k)}{\Gamma(N_d + K\bar{a})} \right).$$

- Unacceptable cost to directly compute $p(z|w) = p(z, w)/p(w)$!

* Latent Dirichlet Allocation

MCMC: Gibbs sampling [GS04]

$$p(z,w) = AB\left(\prod_k \frac{\prod_w \Gamma(N_{kw}+b_w)}{\Gamma\left(N_k + W\bar{b}\right)}\right)\left(\prod_d \frac{\prod_k \Gamma(N_{kd}+a_k)}{\Gamma(N_d + K\bar{a})}\right).$$

- Use Gibbs sampling: iteratively sample from

$$\begin{aligned}
&p\left(z_{11}^{(1)} \middle| \quad z_{12}^{(0)}, z_{13}^{(0)}, z_{14}^{(0)}, \dots, z_{DN_D}^{(0)}, w\right),\\
&p\left(z_{12}^{(1)} \middle| z_{11}^{(1)}, \quad z_{13}^{(0)}, z_{14}^{(0)}, \dots, z_{DN_D}^{(0)}, w\right),\\
&p\left(z_{13}^{(1)} \middle| z_{11}^{(1)}, z_{12}^{(1)}, \quad z_{14}^{(0)}, \dots, z_{DN_D}^{(0)}, w\right),\\
&\dots,\\
&p\left(z_{dn}^{(l+1)} \middle| z_{11}^{(l+1)}, \dots, z_{d(n-1)}^{(l+1)}, \quad z_{d(n+1)}^{(l)}, \dots, w\right) =: p\left(z_{dn} \middle| z^{-dn}, w\right).
\end{aligned}$$

$$p\left(z_{dn}=k \middle| z^{-dn}, w\right) \propto \frac{N_{kw}^{-dn} + b_w}{N_k^{-dn} + W\bar{b}}\left(N_{kd}^{-dn} + a_k\right).$$

* Latent Dirichlet Allocation

$$p(z, w) = AB \left(\prod_{k'} \frac{\prod_{w'} \Gamma\left(N_{k'w'} + b_{w'}\right)}{\Gamma\left(N_{k'} + W\bar{b}\right)} \right) \left(\prod_{d'} \frac{\prod_{k'} \Gamma\left(N_{k'd'} + a_{k'}\right)}{\Gamma(N_{d'} + K\bar{a})} \right)$$

(Denote w_{dn} as w:)

$$= AB \prod_{k'} \frac{\left(\prod_{w' \neq w} \Gamma\left(N_{k'w'} + b_{w'}\right)\right) \cdot \Gamma\left(N_{k'w}^{-dn} + \mathbb{I}(z_{dn} = k') + b_w\right)}{\Gamma\left(N_{k'}^{-dn} + \mathbb{I}(z_{dn} = k') + W\bar{b}\right)} \cdot \left(\prod_{d' \neq d} \frac{\prod_{k'} \Gamma\left(N_{k'd'} + a_{k'}\right)}{\Gamma(N_{d'} + K\bar{a})} \right) \cdot \frac{\prod_{k'} \Gamma\left(N_{k'd}^{-dn} + \mathbb{I}(z_{dn} = k') + a_{k'}\right)}{\Gamma(N_d + K\bar{a})}$$

$$= AB \prod_{k'} \frac{\left(\prod_{w' \neq w} \Gamma\left(N_{k'w'} + b_{w'}\right)\right) \cdot \Gamma\left(N_{k'w}^{-dn} + b_w\right) \cdot \left(N_{k'w}^{-dn} + b_w\right)^{\mathbb{I}(z_{dn} = k')}}{\Gamma\left(N_{k'}^{-dn} + W\bar{b}\right) \cdot \left(N_{k'}^{-dn} + W\bar{b}\right)^{\mathbb{I}(z_{dn} = k')}}$$

$$\cdot \left(\prod_{d' \neq d} \frac{\prod_{k'} \Gamma\left(N_{k'd'} + a_{k'}\right)}{\Gamma(N_{d'} + K\bar{a})} \right) \cdot \frac{\prod_{k'} \Gamma\left(N_{k'd}^{-dn} + a_{k'}\right) \cdot \left(N_{k'd}^{-dn} + a_{k'}\right)^{\mathbb{I}(z_{dn} = k')}}{\Gamma(N_d + K\bar{a})}$$

$$= AB \left(\prod_{k'} \frac{\left(\prod_{w' \neq w} \Gamma\left(N_{k'w'} + b_{w'}\right)\right) \cdot \Gamma\left(N_{k'w}^{-dn} + b_w\right)}{\Gamma\left(N_{k'}^{-dn} + W\bar{b}\right)} \right) \cdot \prod_{k'} \left(\frac{N_{k'w}^{-dn} + b_w}{N_{k'}^{-dn} + W\bar{b}} \right)^{\mathbb{I}(z_{dn} = k')}$$

$$\cdot \left(\left(\prod_{d' \neq d} \frac{\prod_{k'} \Gamma\left(N_{k'd'} + a_{k'}\right)}{\Gamma(N_{d'} + K\bar{a})} \right) \cdot \frac{\prod_{k'} \Gamma\left(N_{k'd}^{-dn} + a_{k'}\right)}{\Gamma(N_d + K\bar{a})} \right) \cdot \prod_{k'} \left(N_{k'd}^{-dn} + a_{k'}\right)^{\mathbb{I}(z_{dn} = k')}.$$

$\Rightarrow$

$$p\left(z_{dn} = k \middle| z^{-dn}, w\right) \propto \frac{N_{kw}^{-dn} + b_w}{N_k^{-dn} + W\bar{b}} \left(N_{kd}^{-dn} + a_k\right).$$

* Latent Dirichlet Allocation

MCMC: Gibbs sampling [GS04]

- For β, use the MAP estimate:

$$\hat{\beta} = \arg\max_{\beta} \log p(\beta|w) .$$

Estimate $p(\beta|w) = \mathbb{E}_{p(z|w)}[p(\beta, z, w)]$ with one sample of z from $p(z|w)$:

$$\Longrightarrow \hat{\beta}_k = \frac{N_{kw} + b_w - 1}{N_k + W\bar{b} - W} \approx \frac{N_{kw} + b_w}{N_k + W\bar{b}} .$$

- For θ, use the MAP estimate:

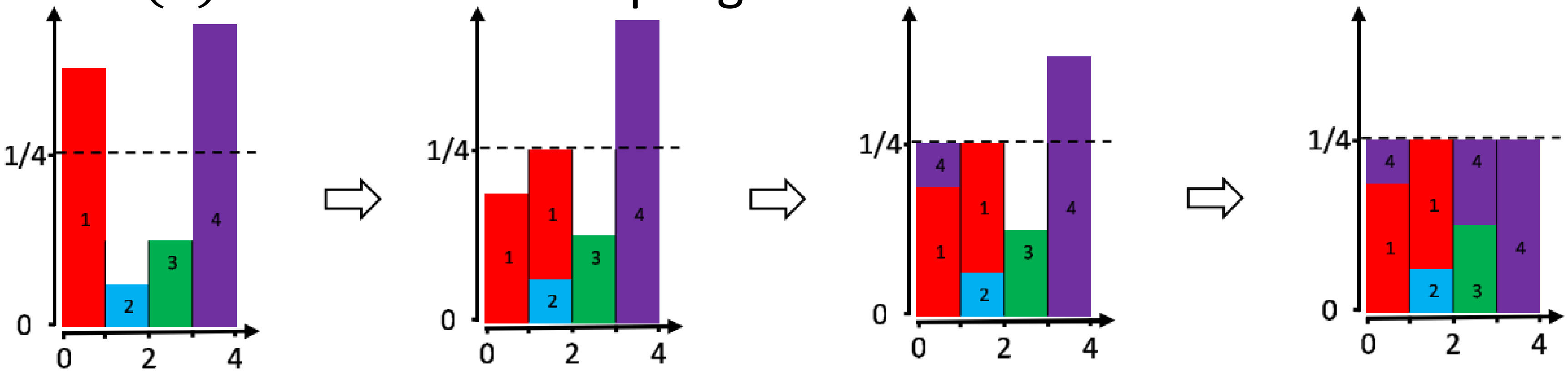
$$\hat{\theta}_{dk} = \frac{N_{kd} + a_k - 1}{N_d + K\bar{a} - K} \approx \frac{N_{kd} + a_k}{N_d + K\bar{a}} .$$

Latent Dirichlet Allocation

MCMC: LightLDA [YGH+15]

$$p\left(z_{dn}=k \middle| z^{-dn}, w\right) \propto \left(N_{kd}^{-dn}+a_k\right) \frac{N_{kw}^{-dn}+b_w}{N_k^{-dn}+W\bar{b}}.$$

- Direct implementation: $O(K)$ time.
- Amortized $O(1)$ multinomial sampling: alias table.



$$\left[\frac{3}{8},\frac{1}{16},\frac{1}{8},\frac{7}{16}\right] \Rightarrow \text{Alias Table: } \left[\left(4,\frac{3}{16}\right),\left(1,\frac{1}{16}\right),\left(4,\frac{1}{8}\right),\left(4,\frac{1}{4}\right)\right] = [(h_i, v_i)]$$

- $O(1)$ sampling: $i \sim \mathrm{Unif}\{1,\ldots,K\}, v \sim \mathrm{Unif}[0,1], z = i$ if $v < v_i$ else h_i.
- $O(K)$ time to build the Alias Table $\Longrightarrow$ Amortized $O(1)$ time for K samples.
- What if the target changes (slightly): use Metropolis Hastings (MH) to correct.

* Latent Dirichlet Allocation

MCMC: LightLDA [YGH+15]

$$p\big(z_{dn} = k \big| z^{-dn}, w\big) \propto \left(N_{kd}^{-dn} + a_k\right) \frac{N_{kw}^{-dn} + b_w}{N_k^{-dn} + W\bar{b}},$$

- Proposal in MH:

$$q(z_{dn} = k) \propto \underbrace{(M_{kd} + a_k)}_{\text{doc}-\text{proposal}} \ \underbrace{\frac{M_{kw} + b_w}{M_k + W\bar{b}}}_{\text{word}-\text{proposal}} \ .$$

  Update $M_{kd} = N_{kd}, M_{kw} = N_{kw}, M_k = N_k$ every K draws.

  - Doc-proposal:
    - MH ratio = $\frac{\left(N_{k'd}^{-dn}+a_{k'}\right)\left(N_{k'w}^{-dn}+\beta_w\right)\left(N_k^{-dn}+W\bar{b}\right)(M_{kd}+a_k)}{\left(N_{kd}^{-dn}+a_k\right)\left(N_{kw}^{-dn}+\beta_w\right)\left(N_{k'}^{-dn}+W\bar{b}\right)\left(M_{k'd}+a_{k'}\right)}$. $O(1)$.
    - Sample from ∝ M_{kd}: take z_{dn} where $n \sim \mathrm{Unif}\{1, \dots, N_d\}$. Directly $O(1)$.
    - Sample from ∝ a_k (dense): use Alias Table. Amortized $O(1)$.

* Latent Dirichlet Allocation

MCMC: LightLDA [YGH+15]

$$p\left(z_{dn} = k \middle| z^{-dn}, w\right) \propto \left(N_{kd}^{-dn} + a_k\right) \frac{N_{kw}^{-dn} + b_w}{N_k^{-dn} + W\bar{b}},$$

- Proposal in MH:

$$q(z_{dn} = k) \propto \underbrace{(M_{kd} + a_k)}_{\text{doc−proposal}} \underbrace{\frac{M_{kw} + b_w}{M_k + W\bar{b}}}_{\text{word−proposal}} .$$

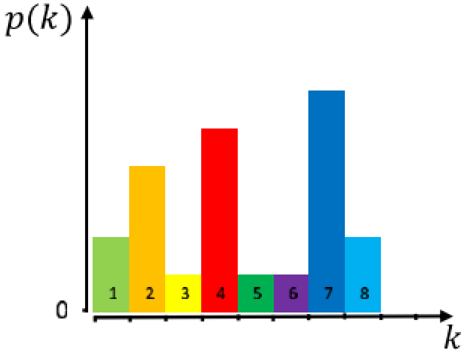
  Update $M_{kd} = N_{kd}, M_{kw} = N_{kw}, M_k = N_k$ every K draws.

  - Word-proposal:
    - MH ratio $=\frac{\left(N_{k'd}^{-dn}+a_{k'}\right)\left(N_{k'w}^{-dn}+\beta_w\right)\left(N_k^{-dn}+W\bar{b}\right)(M_{kw}+b_w)(M_{k'}+W\bar{b})}{\left(N_{kd}^{-dn}+a_k\right)\left(N_{kw}^{-dn}+\beta_w\right)\left(N_{k'}^{-dn}+W\bar{b}\right)(M_{k'w}+b_w)(M_k+W\bar{b})}$. $O(1)$.
    - $\frac{M_{kw}+b_w}{M_k+W\bar{b}} = \frac{M_{kw}}{M_k+W\bar{b}} + \frac{b_w}{M_k+W\bar{b}}$. Sample from either term: use Alias Table. Amortized $O(1)$.

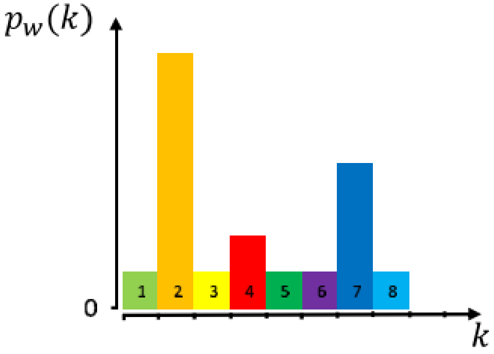
* Latent Dirichlet Allocation

MCMC: LightLDA [YGH+15]

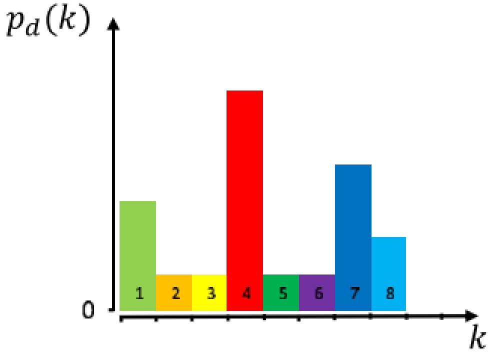
- Overall procedure for Gibbs sampling (cycle proposal):



true conditional distribution



word-proposal distribution



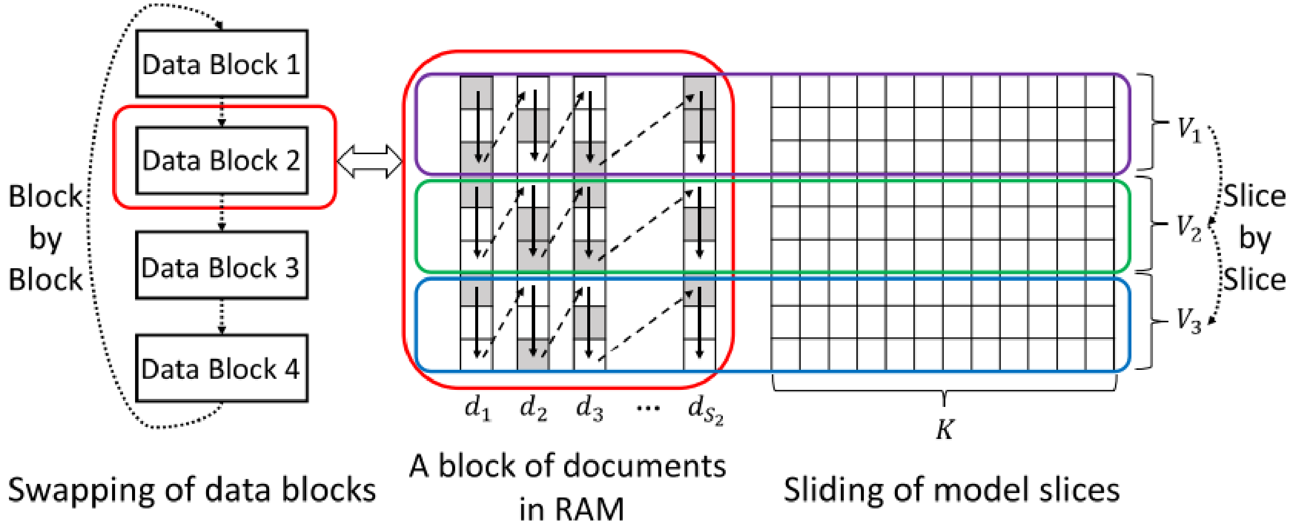
doc-proposal distribution

  - Alternatively use word-proposal and doc-proposal: better coverage on the modes.
  - For each z_{dn}, run the MH chain $L \leq K$ times and take the last sample.

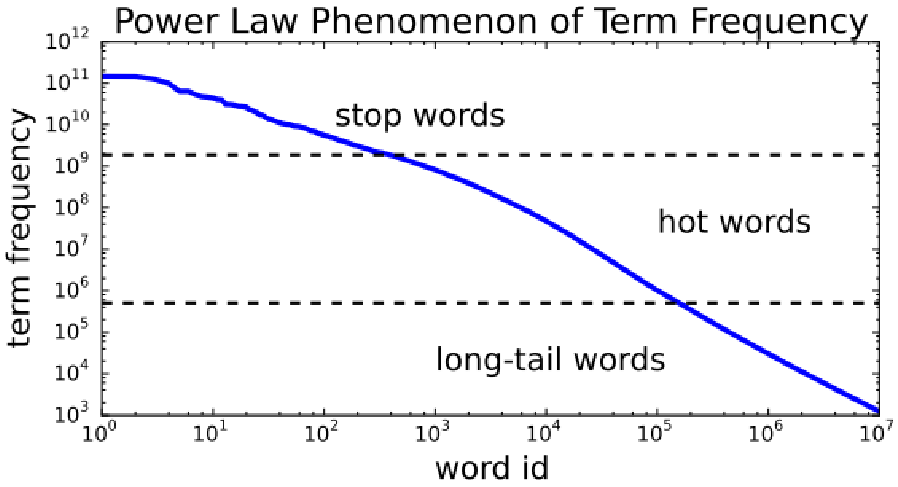
* Latent Dirichlet Allocation

MCMC: LightLDA [YGH+15]

- System implementation
  - Send the model to data:



  - Hybrid data structure:



Latent Dirichlet Allocation

- Dynamics-Based MCMC and Particle-Based VI: target $p(\beta|w)$.

$$\nabla_\beta \log p(\beta|w) = \mathbb{E}_{\underset{\substack{\uparrow \\ \boxed{\text{Gibbs Sampling}}}}{p(z|\beta,w)}}\left[\nabla_\beta \log \underset{\substack{\uparrow \\ \boxed{\text{Closed-form known}}}}{p(\beta,z,w)}\right].$$

  - Stochastic Gradient Riemannian Langevin Dynamics [PT13],
    Stochastic Gradient Nose-Hoover Thermostats [DFB+14],
    Stochastic Gradient Riemannian Hamiltonian Monte Carlo [MCF15].
  - Accelerated particle-based VI [LZC+19, LZZ19].

* Latent Dirichlet Allocation

MCMC: Stochastic Gradient Riemannian Langevin Dynamics [PT13]

$$\mathrm{d}x = G^{-1}\nabla \log p \ \mathrm{d}t + \nabla \cdot G^{-1} \ \mathrm{d}t + \mathcal{N}(0, 2G^{-1} \ \mathrm{d}t).$$

- To draw from $p(\beta|w)$,

$$\nabla_\beta \log p(\beta|w) = \frac{1}{p(\beta|w)} \nabla_\beta \int p(\beta, z|w) \ \mathrm{d}z = \int \frac{1}{p(\beta|w)} \nabla_\beta p(\beta, z|w) \ \mathrm{d}z$$

$$= \int \frac{p(\beta, z|w)}{p(\beta|w)} \frac{\nabla_\beta p(\beta, z|w)}{p(\beta, z|w)} \ \mathrm{d}z = \mathbb{E}_{p(z|\beta, w)}[\nabla_\beta \log p(\beta, z, w)].$$

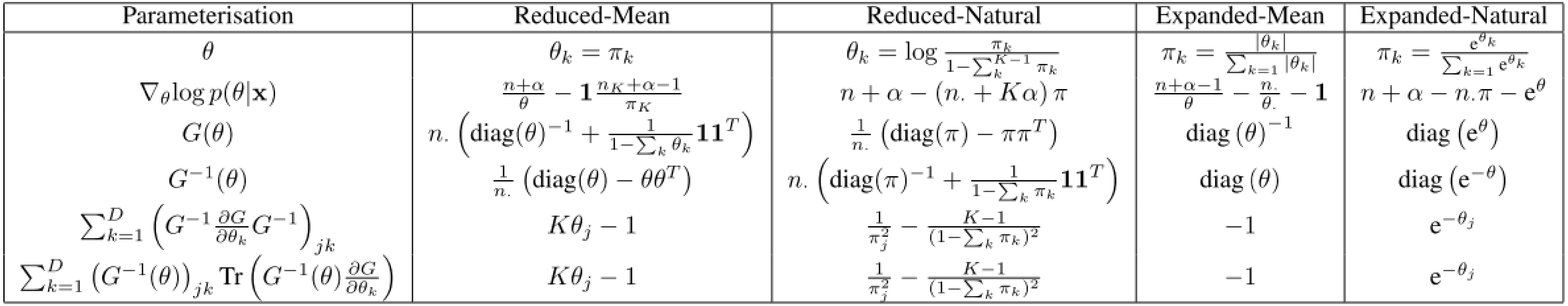
  - $p(\beta, z, w)$ is available in closed form.
  - $p(z|\beta, w)$ can be drawn using Gibbs sampling.
- Each β_k is on a simplex: use reparameterization to convert to the Euclidean space (that's where G comes from), e.g., $\beta_{kw} = \frac{\pi_{kw}}{\sum_w \pi_{kw}}$.

* Latent Dirichlet Allocation

MCMC: Stochastic Gradient Riemannian Langevin Dynamics [PT13]

$$\mathrm{d}x = G^{-1} \nabla \log p \ \mathrm{d}t + \nabla \cdot G^{-1} \ \mathrm{d}t + \mathcal{N}(0, 2G^{-1} \ \mathrm{d}t).$$

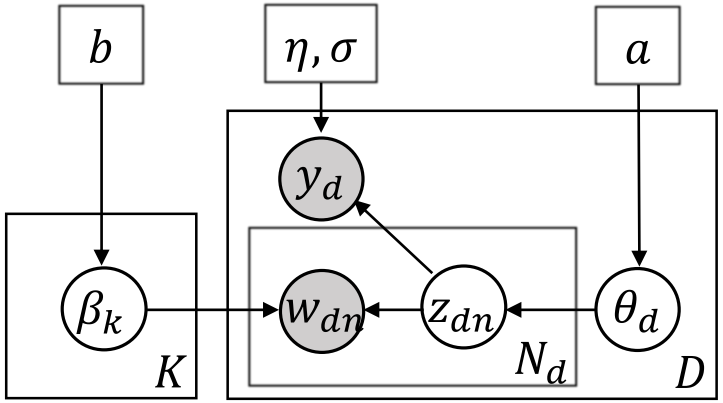
- Various parameterizations:

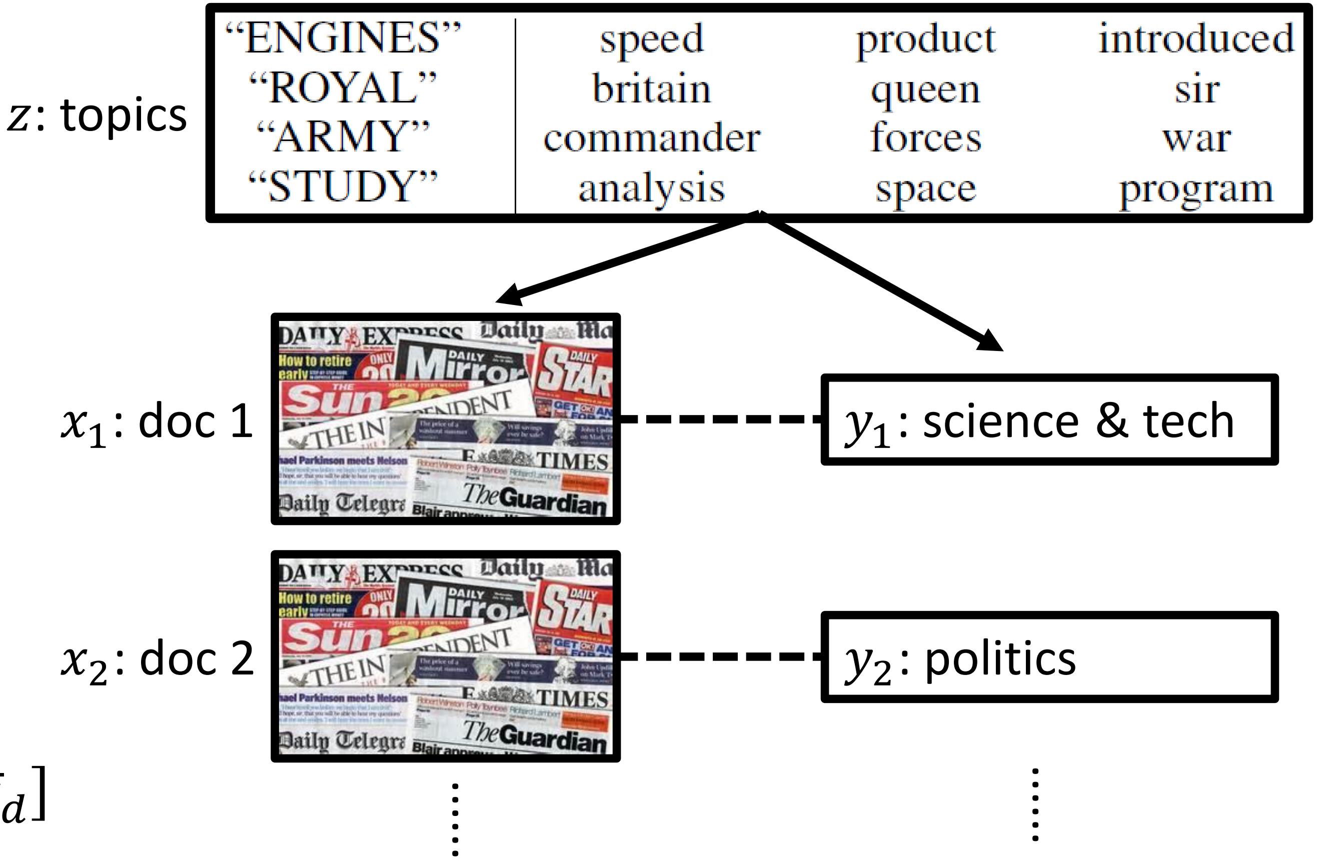


Parameterisation	Reduced-Mean	Reduced-Natural	Expanded-Mean	Expanded-Natural
θ	$\theta_k = \pi_k$	$\theta_k = \log \frac{\pi_k}{1-\sum_k^{K-1} \pi_k}$	$\pi_k = \frac{\lvert\theta_k\rvert}{\sum_{k=1} \lvert\theta_k\rvert}$	$\pi_k = \frac{\mathrm{e}^{\theta_k}}{\sum_{k=1} \mathrm{e}^{\theta_k}}$
$\nabla_\theta \log p(\theta \mid \mathbf{x})$	$\frac{n+\alpha}{\theta} - \mathbf{1} \frac{n_K+\alpha-1}{\pi_K}$	$n + \alpha - (n. + K\alpha)\, \pi$	$\frac{n+\alpha-1}{\theta} - \frac{n.}{\theta.} - \mathbf{1}$	$n + \alpha - n.\pi - \mathrm{e}^\theta$
$G(\theta)$	$n. \left(\mathrm{diag}(\theta)^{-1} + \frac{1}{1-\sum_k \theta_k} \mathbf{1}\mathbf{1}^T\right)$	$\frac{1}{n.} \left(\mathrm{diag}(\pi) - \pi\pi^T\right)$	$\mathrm{diag}\,(\theta)^{-1}$	$\mathrm{diag}\,(\mathrm{e}^\theta)$
$G^{-1}(\theta)$	$\frac{1}{n.} \left(\mathrm{diag}(\theta) - \theta\theta^T\right)$	$n. \left(\mathrm{diag}(\pi)^{-1} + \frac{1}{1-\sum_k \pi_k} \mathbf{1}\mathbf{1}^T\right)$	$\mathrm{diag}\,(\theta)$	$\mathrm{diag}\,(\mathrm{e}^{-\theta})$
$\sum_{k=1}^D \left(G^{-1} \frac{\partial G}{\partial \theta_k} G^{-1}\right)_{jk}$	$K\theta_j - 1$	$\frac{1}{\pi_j^2} - \frac{K-1}{(1-\sum_k \pi_k)^2}$	-1	$\mathrm{e}^{-\theta_j}$
$\sum_{k=1}^D \left(G^{-1}(\theta)\right)_{jk} \mathrm{Tr} \left(G^{-1}(\theta) \frac{\partial G}{\partial \theta_k}\right)$	$K\theta_j - 1$	$\frac{1}{\pi_j^2} - \frac{K-1}{(1-\sum_k \pi_k)^2}$	-1	$\mathrm{e}^{-\theta_j}$

Supervised Latent Dirichlet Allocation

Model structure [MB08]:





z: topics

"ENGINES" "ROYAL" "ARMY" "STUDY"	speed britain commander analysis	product queen forces space	introduced sir war program

x_1: doc 1 — y_1: science & tech

x_2: doc 2 — y_2: politics

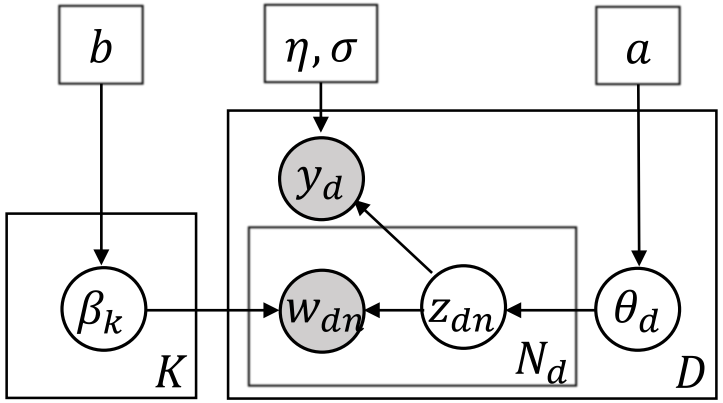
⋮ ⋮

- Variational inference: similar to LDA.
- Prediction: for test document w_d,

$$\hat{y}_d \coloneqq \mathbb{E}_{p(y_d|w_d)}[y_d] = \eta^\top \mathbb{E}_{p(z_d|w_d)}[\bar{z}_d]$$
$$\approx \eta^\top \mathbb{E}_{q(z_d|w_d)}[\bar{z}_d].$$

First do inference (find $q(z_d|w_d)$), then estimate $\hat{y}_d$.

* Supervised Latent Dirichlet Allocation



Model structure [MB08]:

- Generating process:
  - Draw topics: $\beta_k \sim \mathrm{Dir}(b), k = 1, \dots, K$;
  - For each document d,
    - Draw topic proportion $\theta_d \sim \mathrm{Dir}(a)$;
    - For each word n in document d,
      - Draw topic assignment $z_{dn} \sim \mathrm{Mult}(\theta_d)$;
      - Draw word $w_{dn} \sim \mathrm{Mult}(z_{dn})$.
    - Draw the response $y_d \sim \mathcal{N}\left(\eta^\top \bar{z}_d, \sigma^2\right), \bar{z}_d \coloneqq \frac{1}{N_d} \sum_{n=1}^{N_d} z_{dn}$ (one-hot).

$$p(\beta, \theta, z, w, \mathrm{y})$$
$$= \left(\prod_{k=1}^{K} \mathrm{Dir}(\beta_k | b) \right) \left(\prod_{d=1}^{D} \mathrm{Dir} \left(\prod_{n=1}^{N_d} \mathrm{Mult}(z_{dn} | \theta_d) \mathrm{Mult}\left(w_{dn} | \beta_{z_{dn}}\right) \right) \mathcal{N}\left(y_d | \eta^\top \bar{z}_d, \sigma^2\right) \right).$$

* Supervised Latent Dirichlet Allocation

Variational inference [MB08]: similar to LDA.

- Same variational distribution

$$q_{\lambda,\gamma,\phi}(\beta,\theta,z) \coloneqq \prod_{k=1}^{K} \mathrm{Dir}(\beta_k|\lambda_k) \prod_{d=1}^{D} \mathrm{Dir}(\theta_d|\gamma_d) \prod_{n=1}^{n_d} \mathrm{Mult}(z_{dn}|\phi_{dn})\,.$$

  ELBO$\left(\lambda,\gamma,\phi;a,b,\eta,\sigma^2\right)$ is available in closed form.

- E-step: update λ,γ,ϕ by maximizing ELBO.
- M-step: update a,b,η,σ^2 by maximizing ELBO.
- Prediction: given a new document w_d,

$$\hat{y}_d \coloneqq \mathbb{E}_{p(y_d|w_d)}[y_d] = \eta^\top \mathbb{E}_{p(z_d|w_d)}[\bar{z}_d] \approx \eta^\top \mathbb{E}_{q(z_d|w_d)}[\bar{z}_d].$$

  First do inference: find $q(z_d|w_d)$ i.e. ϕ_d, then estimate $\hat{y}_d$.

* Supervised Latent Dirichlet Allocation

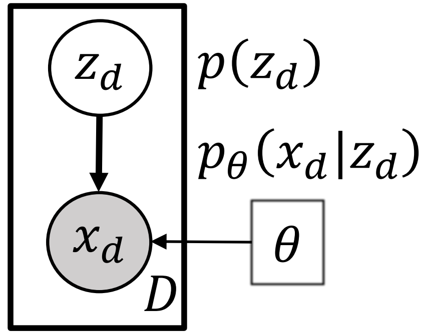
Variational inference with posterior regularization [ZAX12]

- Regularized Bayes (RegBayes) [ZCX14]:
  - Recall: $p\left(z \middle| \left\{x^{(n)}, y^{(n)}\right\}\right)$ $= \underset{q(z)}{\arg\min}\left\{-\mathcal{L}[q] = \mathrm{KL}\big(q(z), p(z)\big) - \sum_n \mathbb{E}_q\left[\log p\left(x^{(n)}, y^{(n)} | z\right)\right]\right\}.$
  - Regularize posterior towards better prediction: $\min_{q(z)} \mathrm{KL}\big(q(z), p(z)\big) - \sum_n \mathbb{E}_q\left[\log p\left(x^{(n)}, y^{(n)} | z\right)\right] + \lambda \ell\left(q(z); \left\{x^{(n)}, y^{(n)}\right\}\right).$
- Maximum entropy discrimination LDA (MedLDA) [ZAX12]:
  - $\ell\left(q; \left\{w^{(n)}, y^{(n)}\right\}\right) = \sum_n \ell_\varepsilon\left(y^{(n)} - \hat{y}^{(n)}\left(q, w^{(n)}\right)\right)$ $= \sum_n \ell_\varepsilon\left(y^{(n)} - \eta^\top \mathbb{E}_{q\left(z^{(n)} | w^{(n)}\right)}\left[\bar{z}^{(n)}\right]\right),$ where $\ell_\varepsilon(r) = \max\{0, |r| - \varepsilon\}$ is the hinge (max-margin) loss.
  - Facilitates both prediction and topic representation.

Outline

- Generative Models: Overview
- Plain Generative Models
  - Autoregressive Models
- **Latent Variable Models**
  - Deterministic Generative Models
    - Generative Adversarial Nets
    - Flow-Based Generative Models
  - **Bayesian Generative Models**
    - Bayesian Inference (variational inference, MCMC)
    - **Bayesian Networks**
      - Topic Models (LDA, LightLDA, sLDA)
      - **Deep Bayesian Models** (VAE)
    - Markov Random Fields (Boltzmann machines, deep energy-based models)

Variational Auto-Encoder



More *flexible* Bayesian model using *deep learning* tools.

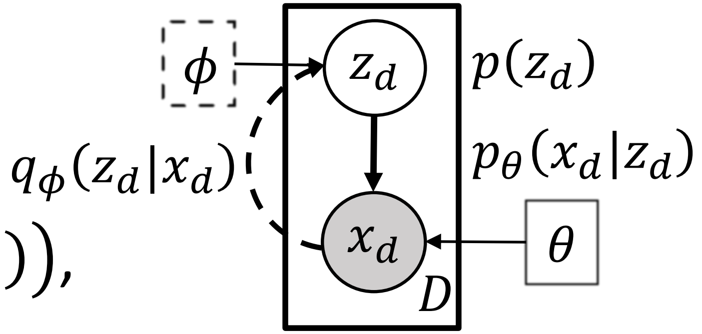
- Model structure (decoder) [KW14]:

$$z_d \sim p(z_d) = \mathcal{N}(z_d|0, I),$$

$$x_d \sim p_\theta(x_d|z_d) = \mathcal{N}\big(x_d|\mu_\theta(z_d), \Sigma_\theta(z_d)\big),$$

where $\mu_\theta(z_d)$ and $\Sigma_\theta(z_d)$ are modeled by neural networks.

Variational Auto-Encoder



- Variational inference (encoder) [KW14]:

  $q_\phi(z|x) \coloneqq \prod_{d=1}^{D} q_\phi(z_d|x_d) = \prod_{d=1}^{D} \mathcal{N}\big(z_d \big| \nu_\phi(x_d), \Gamma_\phi(x_d)\big),$

  where $\nu_\phi(x_d), \Gamma_\phi(x_d)$ are also NNs.
  - Amortized inference: approximate local posteriors $\{p(z_d|x_d)\}_{d=1}^{D}$ globally by ϕ.
  - Objective:

$$\mathbb{E}_{\hat{p}(x)}[\mathrm{ELBO}(x)] \approx \frac{1}{D}\sum_{d=1}^{D} \mathbb{E}_{q_\phi(z_d|x_d)}\big[\log p_\theta(z_d)p_\theta(x_d|z_d) - \log q_\phi(z_d|x_d)\big].$$

  - Gradient estimation with the reparameterization trick:

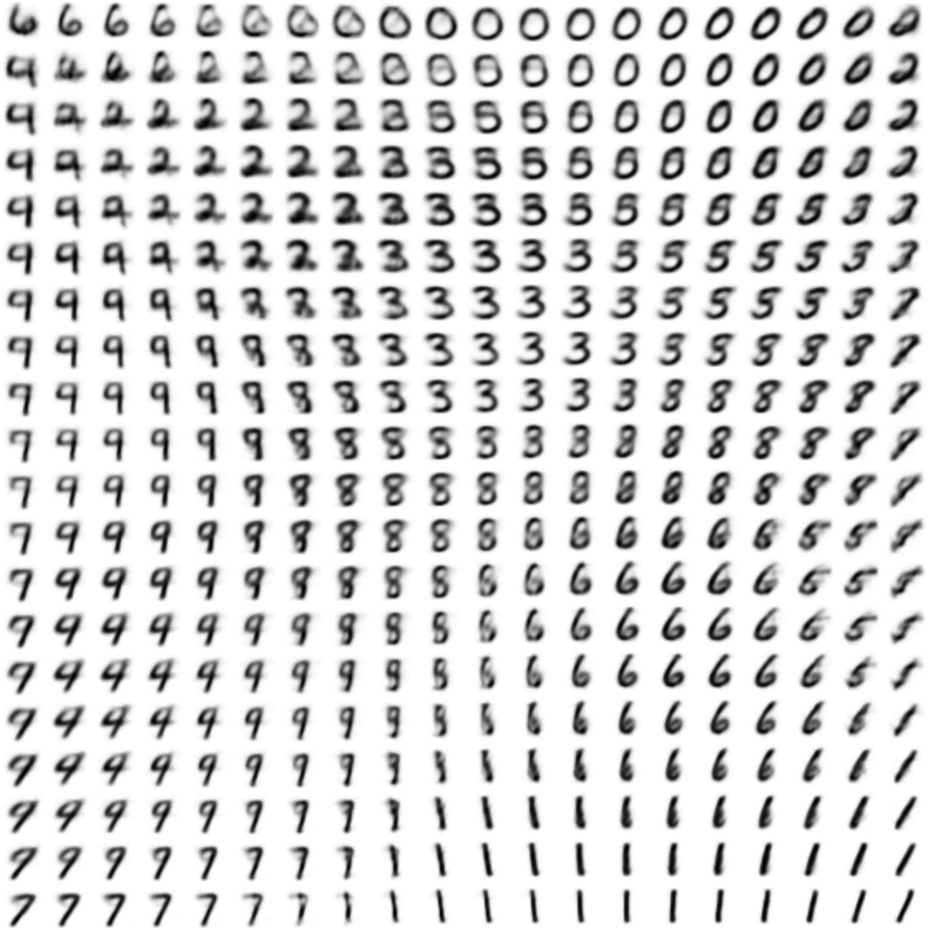
$$z_d \sim q_\phi(z_d|x_d) \Leftrightarrow z_d = g_\phi(x_d,\epsilon) \coloneqq \nu_\phi(x_d) + \epsilon\sqrt{\Gamma_\phi(x_d)}, \epsilon \sim q(\epsilon) = \mathcal{N}(\epsilon|0,I).$$

$$\nabla_{\phi,\theta}\mathbb{E}_{\hat{p}(x)}[\mathrm{ELBO}(x)] \approx \tfrac{1}{D}\textstyle\sum_{d=1}^{D} \mathbb{E}_{q(\epsilon)}\left[\nabla_{\phi,\theta}\left(\log p_\theta(z_d)p_\theta(x_d|z_d) - \log q_\phi(z_d|x_d)\,|_{z_d=g_\phi(x_d,\epsilon)}\right)\right].$$

(Smaller variance than REINFORCE-like estimator [Wil92]: $\nabla_\theta \mathbb{E}_{q_\theta}[f] = \mathbb{E}_{q_\theta}[f \nabla_\theta \log q_\theta]$.)

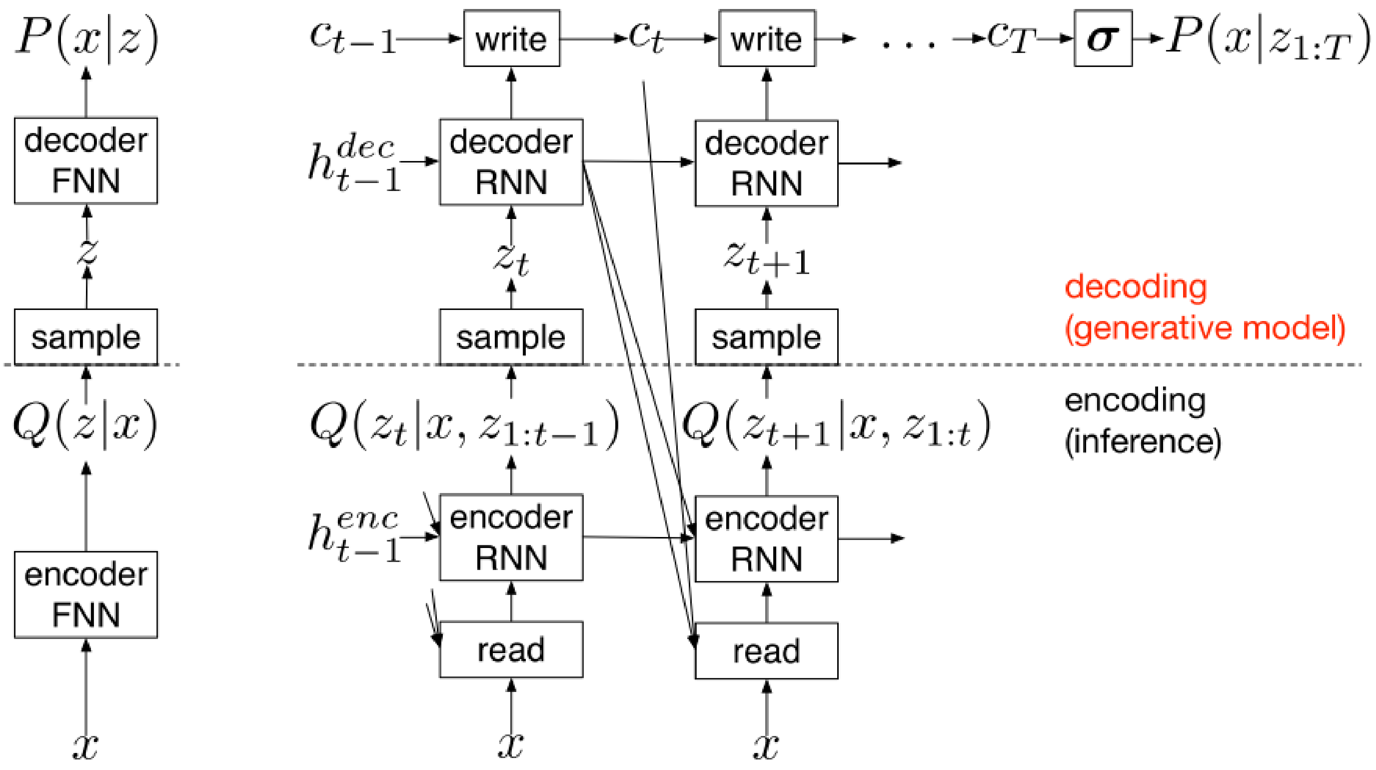
Variational Auto-Encoder

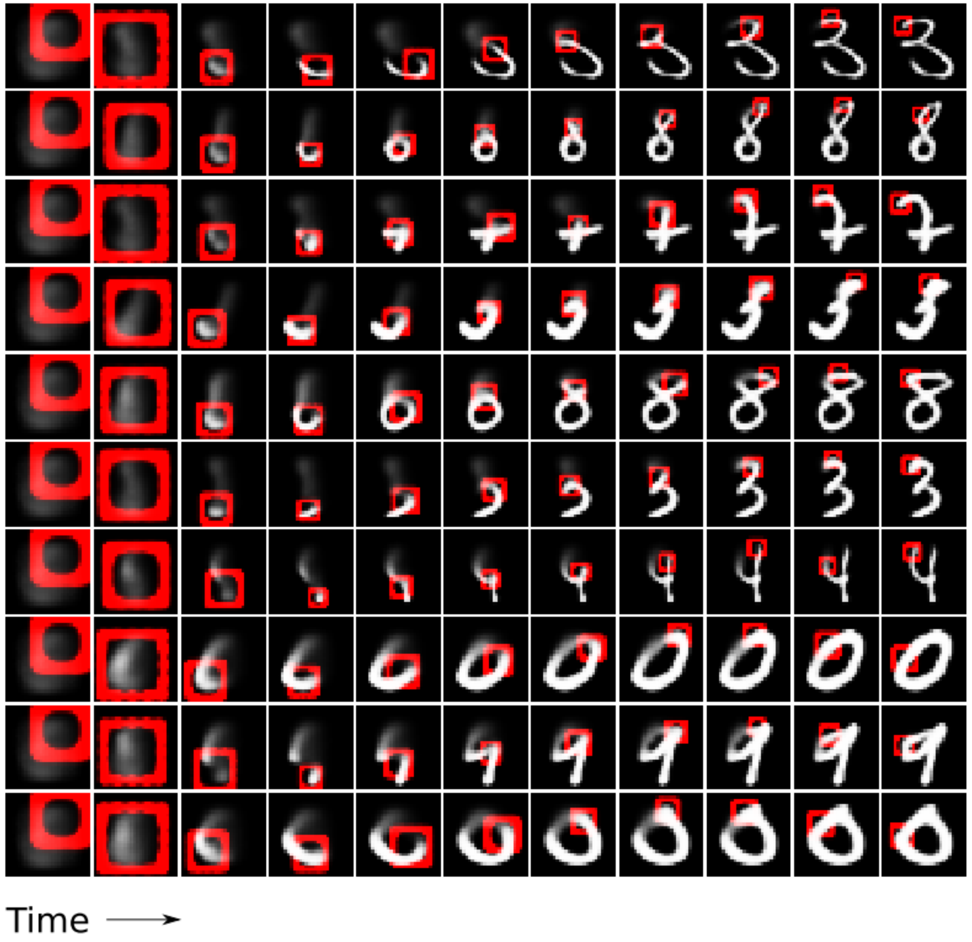
- Generation results [KW14]



* Variational Auto-Encoder

- With spatial attention structure [GDG+15]





* Variational Auto-Encoder

- Inference with importance-weighted ELBO [BGS15]
  - ELBO: $\mathcal{L}_\theta[q_\phi(z)] = \mathbb{E}_{q_\phi(z)}[\log p_\theta(z,x)] - \mathbb{E}_{q_\phi(z)}[\log q_\phi(z)]$.
  - A tighter lower bound:

$$\mathcal{L}_\theta^{(k)}[q_\phi] := \mathbb{E}_{z^{(1)},\ldots,z^{(k)} \sim^{\text{i.i.d.}} q_\phi}\left[\log \frac{1}{k}\sum_{i=1}^{k}\frac{p_\theta\left(z^{(k)},x\right)}{q_\phi\left(z^{(k)}\right)}\right].$$

Ordering relation:

$$\mathcal{L}_\theta[q_\phi] = \mathcal{L}_\theta^{(1)}[q_\phi] \le \mathcal{L}_\theta^{(2)}[q_\phi] \le \cdots \le \mathcal{L}_\theta^{(\infty)}[q_\phi] \underset{\uparrow}{=} \log p_\theta(x)\,.$$

If $\frac{p(z,x)}{q(z|x)}$ is bounded.

Variational Auto-Encoder

- Parametric Variational Inference: towards more flexible approximations.
  - Explicit VI:

    Normalizing flows [RM15, KSJ+16].

    Using a tighter ELBO [BGS15].
  - Implicit VI:

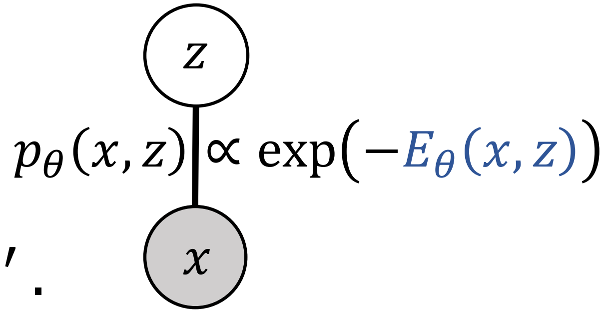
    Adversarial Auto-Encoder [MSJ+15], Adversarial Variational Bayes [MNG17], Wasserstein Auto-Encoder [TBGS17], [SSZ18a], [LT18], [SSZ18b].

- MCMC [LTL17] and Particle-Based VI [FWL17, PGH+17]:
  - Train the encoder as a sample generator.
  - Amortize the update on samples to ϕ.

Outline

- Generative Models: Overview
- Plain Generative Models
  - Autoregressive Models
- **Latent Variable Models**
  - Deterministic Generative Models
    - Generative Adversarial Nets
    - Flow-Based Generative Models
  - **Bayesian Generative Models**
    - Bayesian Inference (variational inference, MCMC)
    - Bayesian Networks
      - Topic Models (LDA, LightLDA, sLDA)
      - Deep Bayesian Models (VAE)
    - **Markov Random Fields** (Boltzmann machines, deep energy-based models)

Markov Random Fields


$p_\theta(x,z) \propto \exp(-E_\theta(x,z))$

Specify $p_\theta(x,z)$ by an energy function $E_\theta(x,z)$:

$$p_\theta(x,z) = \frac{1}{Z_\theta}\exp\big(-E_\theta(x,z)\big), Z_\theta = \int \exp\big(-E_\theta(x',z')\big)\, \mathrm{d}x'\mathrm{d}z'.$$

- Only correlation and no causality: $p(x,z)$ is either $p(z)p(x|z)$ or $p(x)p(z|x)$.

\+ Flexible and simple in modeling dependency.

\- Harder to learn and generate than BayesNets.
  - Learning: even $p_\theta(x,z)$ is unavailable.

$$\nabla_\theta \mathbb{E}_{\hat{p}(x)}[\log p_\theta(x)] = -\mathbb{E}_{\hat{p}(x)p_\theta(z|x)}[\nabla_\theta E_\theta(x,z)] + \mathbb{E}_{p_\theta(x,z)}[\nabla_\theta E_\theta(x,z)].$$

=0 if $E = \log p$.

(augmented) data distribution

(Bayesian inference)

model distribution

(generation)

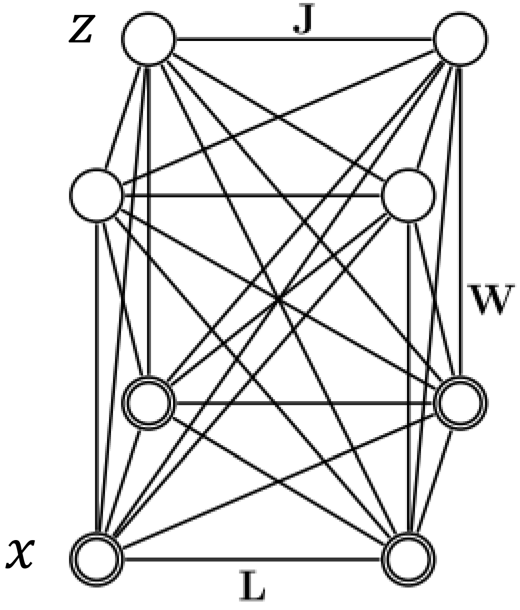
  - Bayesian inference: generally same as BayesNets.
  - Generation: rely on MCMC or train a generator.

Markov Random Fields

- Learning: $\nabla_\theta \mathbb{E}_{\hat{p}(x)}[\log p_\theta(x)] = -\mathbb{E}_{\hat{p}(x)p_\theta(z|x)}[\nabla_\theta E_\theta(x,z)] + \mathbb{E}_{p_\theta(x,z)}[\nabla_\theta E_\theta(x,z)]$.

  Bayesian Inference Generation

  - Boltzmann Machine: Gibbs sampling for both inference and generation [HS83].



$$E_\theta(x,z) = -x^\top W z - \frac{1}{2}x^\top L x - \frac{1}{2}z^\top J z.$$

$$\Longrightarrow$$

$$p_\theta(z_j|x, z_{-j}) = \mathrm{Bern}\left(\sigma\left(\sum_{i=1}^{D} W_{ij}x_i + \sum_{m\neq j}^{P} J_{jm}z_j\right)\right),$$

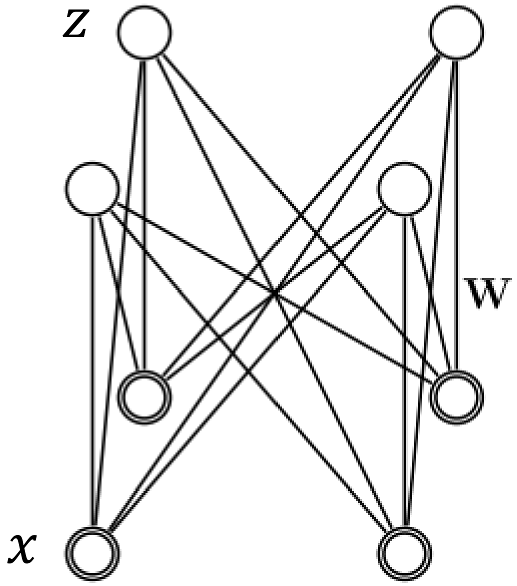
$$p_\theta(x_i|z, x_{-i}) = \mathrm{Bern}\left(\sigma\left(\sum_{j=1}^{P} W_{ij}z_j + \sum_{k\neq i}^{D} L_{ik}x_k\right)\right).$$

Markov Random Fields

- Learning: $\nabla_\theta \mathbb{E}_{\hat{p}(x)}[\log p_\theta(x)] = -\mathbb{E}_{\hat{p}(x)p_\theta(z|x)}[\nabla_\theta E_\theta(x,z)] + \mathbb{E}_{p_\theta(x,z)}[\nabla_\theta E_\theta(x,z)]$.

Bayesian Inference ↑ Generation ↑

  - Restricted Boltzmann Machine [Smo86]:



$$E_\theta(x,z) = -x^\top W z + {b^{(x)}}^\top x + {b^{(z)}}^\top z.$$

- Bayesian Inference is exact:

$$p_\theta(z_k|x) = \mathrm{Bern}\left(\sigma\left(x^\top W_{:k} + b_k^{(z)}\right)\right).$$

- Generation: Gibbs sampling.

Iterate:

$$p_\theta(z_k|x) = \mathrm{Bern}\left(\sigma\left(x^\top W_{:k} + b_k^{(z)}\right)\right),$$

$$p_\theta(x_k|z) = \mathrm{Bern}\left(\sigma\left(W_{k:} z + b_k^{(x)}\right)\right).$$

Markov Random Fields

Deep Energy-Based Models:

No latent variable; $E_\theta(x)$ is modeled by a neural network.

$$\nabla_\theta \mathbb{E}_{\hat{p}(x)}[\log p_\theta(x)] = -\mathbb{E}_{\hat{p}(x)}[\nabla_\theta E_\theta(x)] + \mathbb{E}_{p_\theta(x')}[\nabla_\theta E_\theta(x')].$$

- [KB16]: learn a generator

$$x \sim q_\phi(x) \Leftrightarrow z \sim q(z), x = g_\phi(z),$$

to mimic the generation from $p_\theta(x)$:

$$\arg\min_\phi \mathrm{KL}\left(q_\phi, p_\theta\right) = \arg\min_\phi \mathbb{E}_{q(z)}\left[E_\theta\left(g_\phi(z)\right)\right] - \underbrace{\mathbb{H}[q_\phi]}_{\substack{\text{approx. by batch} \\ \text{normalization Gaussian}}}$$

Markov Random Fields

Deep Energy-Based Models:

No latent variable; $E_\theta(x)$ is modeled by a neural network.

$$\nabla_\theta \mathbb{E}_{\hat{p}(x)}[\log p_\theta(x)] = -\mathbb{E}_{\hat{p}(x)}[\nabla_\theta E_\theta(x)] + \mathbb{E}_{p_\theta(x')}[\nabla_\theta E_\theta(x')].$$

- [DM19]: estimate $\mathbb{E}_{p_\theta(x')}[\cdot]$ by samples drawn by the Langevin Dynamics.

* Markov Random Fields

Deep Energy-Based Models:

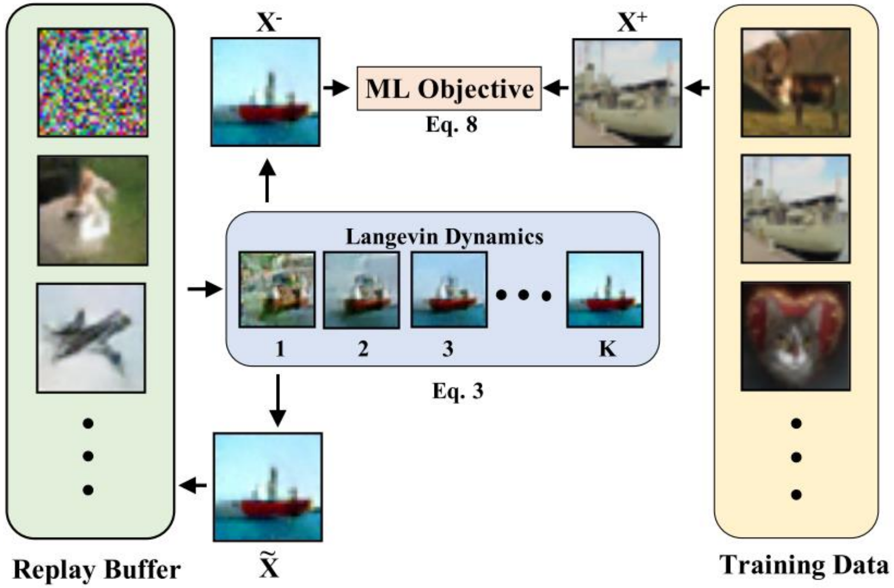
No latent variable; $E_\theta(x)$ is modeled by a neural network.

$$\nabla_\theta \mathbb{E}_{\hat{p}(x)}[\log p_\theta(x)] = -\mathbb{E}_{\hat{p}(x)}[\nabla_\theta E_\theta(x)] + \mathbb{E}_{p_\theta(x')}[\nabla_\theta E_\theta(x')].$$

- [DM19]: estimate $\mathbb{E}_{p_\theta(x')}[\cdot]$ by samples drawn by the Langevin Dynamics

$$x^{(k+1)} = x^{(k)} - \varepsilon \nabla_x E_\theta\left(x^{(k)}\right) + \mathcal{N}(0, 2\varepsilon).$$

  - Replay buffer for initializing the LD chain.
  - L_2-regularization on the energy function.

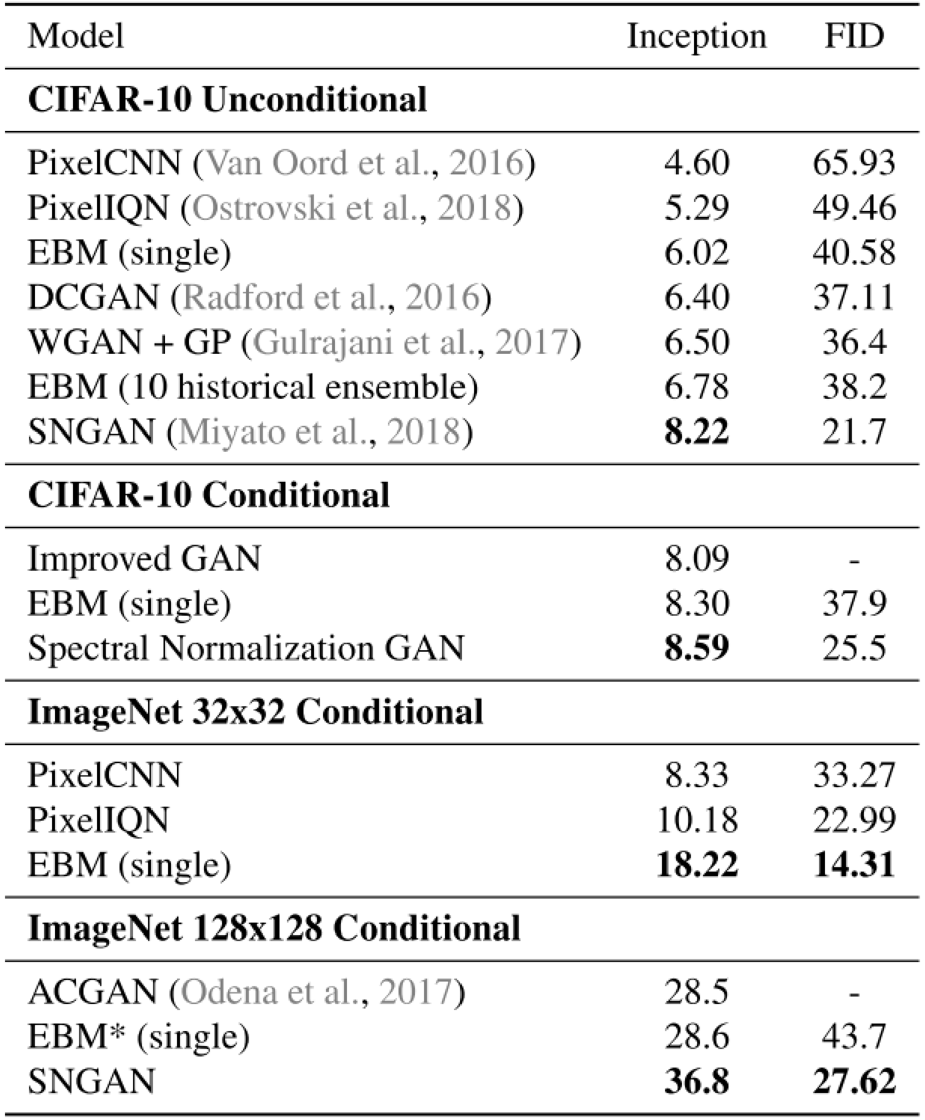


* Markov Random Fields

Deep Energy-Based Models:

- [DM19]

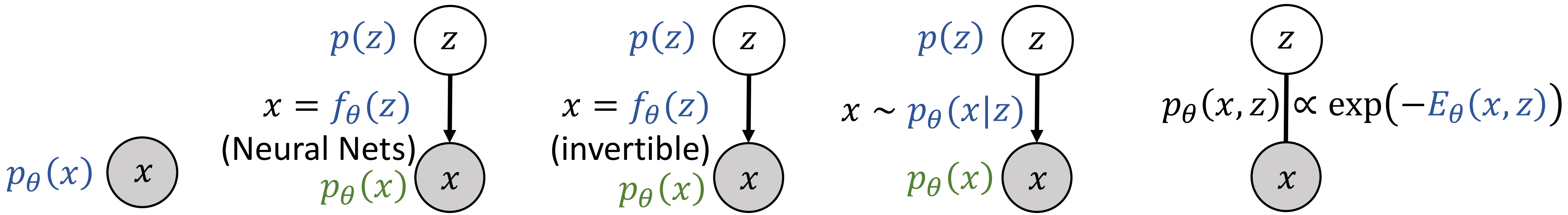
ImageNet32x32 Generation



Model	Inception	FID
CIFAR-10 Unconditional		
PixelCNN (Van Oord et al., 2016)	4.60	65.93
PixelIQN (Ostrovski et al., 2018)	5.29	49.46
EBM (single)	6.02	40.58
DCGAN (Radford et al., 2016)	6.40	37.11
WGAN + GP (Gulrajani et al., 2017)	6.50	36.4
EBM (10 historical ensemble)	6.78	38.2
SNGAN (Miyato et al., 2018)	**8.22**	21.7
CIFAR-10 Conditional		
Improved GAN	8.09	-
EBM (single)	8.30	37.9
Spectral Normalization GAN	**8.59**	25.5
ImageNet 32x32 Conditional		
PixelCNN	8.33	33.27
PixelIQN	10.18	22.99
EBM (single)	**18.22**	**14.31**
ImageNet 128x128 Conditional		
ACGAN (Odena et al., 2017)	28.5	-
EBM* (single)	28.6	43.7
SNGAN	**36.8**	**27.62**

Generative Model: Summary

Plain Generative Models	Latent Variable Models			
Autoregressive Models	Deterministic Generative Models		Bayesian Generative Models	
	GANs	Flow-Based	BayesNets	MRFs
+ Easy learning + Easy generation - No abstract representation - Slow generation	+ Abstract representation and manipulated generation - Harder learning			
	+ Flexible modeling + Easy and good generation		+ Robust to small data and adversarial attack + Principled inference + Prior knowledge	
	- Hard inference - Hard learning	+ Easy inference + Stable learning - Hard model design	+ Causal information + Easier learning + Easy generation	+ Simple dependency modeling - Harder learning - Hard generation



Questions?

References

References

- Plain Generative Models
  - Autoregressive Models

    - [Fre98] Frey, Brendan J. (1998). *Graphical models for machine learning and digital communication*. MIT press.
    - [LM11] Larochelle, H. and Murray, I. The neural autoregressive distribution estimator. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2011.
    - [UML14] Uria, B., Murray, I., & Larochelle, H. (2014). A deep and tractable density estimator. In *International Conference on Machine Learning* (pp. 467-475).
    - [GGML15] Germain, M., Gregor, K., Murray, I., & Larochelle, H. (2015). MADE: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning* (pp. 881-889).
    - [OKK16] Oord, A. V. D., Kalchbrenner, N., & Kavukcuoglu, K. (2016). Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*.
    - [ODZ+16] Oord, A. V. D., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., & Kavukcuoglu, K. (2016). WaveNet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*.

References

- Deterministic Generative Models
  - Generative Adversarial Networks
    - [GPM+14] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems* (pp. 2672-2680).
    - [ACB17] Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein generative adversarial networks. In *International Conference on Machine Learning* (pp. 214-223).
  - Flow-Based Generative Models
    - [DKB15] Dinh, L., Krueger, D., & Bengio, Y. (2015). NICE: Non-linear independent components estimation. *ICLR workshop*.
    - [DSB17] Dinh, L., Sohl-Dickstein, J., & Bengio, S. (2017). Density estimation using real NVP. In *Proceedings of the International Conference on Learning Representations*.
    - [PPM17] Papamakarios, G., Pavlakou, T., & Murray, I. (2017). Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems* (pp. 2338-2347).
    - [KD18] Kingma, D. P., & Dhariwal, P. (2018). Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems* (pp. 10215-10224).

References

- Bayesian Inference: Variational Inference
  - Explicit Parametric VI:
    - [SJJ96] Saul, L. K., Jaakkola, T., & Jordan, M. I. (1996). Mean field theory for sigmoid belief networks. *Journal of artificial intelligence research*, 4, 61-76.
    - [BNJ03] Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3(Jan), pp.993-1022.
    - [GHB12] Gershman, S., Hoffman, M., & Blei, D. (2012). Nonparametric variational inference. arXiv preprint arXiv:1206.4665.
    - [HBWP13] Hoffman, M. D., Blei, D. M., Wang, C., & Paisley, J. (2013). Stochastic variational inference. *The Journal of Machine Learning Research*, *14*(1), 1303-1347.
    - [RGB14] Ranganath, R., Gerrish, S., & Blei, D. (2014). Black box variational inference. In *Artificial Intelligence and Statistics* (pp. 814-822).
    - [RM15] Rezende, D.J., & Mohamed, S. (2015). Variational inference with normalizing flows. In *Proceedings of the International Conference on Machine Learning* (pp. 1530-1538).
    - [KSJ+16] Kingma, D.P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., & Welling, M. (2016). Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems* (pp. 4743-4751).

References

- Bayesian Inference: Variational Inference
  - Implicit Parametric VI: density ratio estimation
    - [MSJ+15] Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., & Frey, B. (2016). Adversarial Autoencoders. In *Proceedings of the International Conference on Learning Representations*.
    - [MNG17] Mescheder, L., Nowozin, S., & Geiger, A. (2017). Adversarial variational Bayes: Unifying variational autoencoders and generative adversarial networks. In *Proceedings of the International Conference on Machine Learning* (pp. 2391-2400).
    - [Hus17] Huszár, F. (2017). Variational inference using implicit distributions. *arXiv preprint* arXiv:1702.08235.
    - [TRB17] Tran, D., Ranganath, R., & Blei, D. (2017). Hierarchical implicit models and likelihood-free variational inference. In *Advances in Neural Information Processing Systems* (pp. 5523-5533).
    - [SSZ18a] Shi, J., Sun, S., & Zhu, J. (2018). Kernel Implicit Variational Inference. In *Proceedings of the International Conference on Learning Representations*.
  - Implicit Parametric VI: gradient estimation
    - [VLBM08] Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P. A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning* (pp. 1096-1103). ACM.
    - [LT18] Li, Y., & Turner, R. E. (2018). Gradient estimators for implicit models. In *Proceedings of the International Conference on Learning Representations*.
    - [SSZ18b] Shi, J., Sun, S., & Zhu, J. (2018). A spectral approach to gradient estimation for implicit distributions. In *Proceedings of the 35th International Conference on Machine Learning* (pp. 4651-4660).

References

- Bayesian Inference: Variational Inference
  - Particle-Based VI

    - [LW16] Liu, Q., & Wang, D. (2016). Stein variational gradient descent: A general purpose Bayesian inference algorithm. In *Advances In Neural Information Processing Systems* (pp. 2378-2386).
    - [Liu17] Liu, Q. (2017). Stein variational gradient descent as gradient flow. In *Advances in neural information processing systems* (pp. 3115-3123).
    - [CZ17] Chen, C., & Zhang, R. (2017). Particle optimization in stochastic gradient MCMC. *arXiv preprint arXiv:1711.10927*.
    - [FWL17] Feng, Y., Wang, D., & Liu, Q. (2017). Learning to Draw Samples with Amortized Stein Variational Gradient Descent. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*.
    - [PGH+17] Pu, Y., Gan, Z., Henao, R., Li, C., Han, S., & Carin, L. (2017). VAE learning via Stein variational gradient descent. In *Advances in Neural Information Processing Systems* (pp. 4236-4245).
    - [LZ18] Liu, C., & Zhu, J. (2018). Riemannian Stein Variational Gradient Descent for Bayesian Inference. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence* (pp. 3627-3634).

References

- Bayesian Inference: Variational Inference
  - Particle-Based VI
    - [CMG+18] Chen, W. Y., Mackey, L., Gorham, J., Briol, F. X., & Oates, C. J. (2018). Stein points. *arXiv preprint arXiv:1803.10161*.
    - [FCSS18] Futami, F., Cui, Z., Sato, I., & Sugiyama, M. (2018). Frank-Wolfe Stein sampling. *arXiv preprint arXiv:1805.07912*.
    - [CZW+18] Chen, C., Zhang, R., Wang, W., Li, B., & Chen, L. (2018). A unified particle-optimization framework for scalable Bayesian sampling. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*.
    - [ZZC18] Zhang, J., Zhang, R., & Chen, C. (2018). Stochastic particle-optimization sampling and the non-asymptotic convergence theory. *arXiv preprint arXiv:1809.01293*.
    - [LZC+19] Liu, C., Zhuo, J., Cheng, P., Zhang, R., Zhu, J., & Carin, L. (2019). Understanding and Accelerating Particle-Based Variational Inference. In *Proceedings of the 36th International Conference on Machine Learning* (pp. 4082-4092).

References

- Bayesian Inference: MCMC
  - Classical MCMC
    - [MRR+53] Metropolis, N., Rosenbluth, A. W., Rosenbluth, M.N., Teller, A.H., & Teller, E. (1953). Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6), pp.1087-1092.
    - [Has70] Hastings, W. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1), pp.97-109.
    - [GG87] Geman, S., & Geman, D. (1987). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. In *Readings in computer vision* (pp. 564-584).
    - [ADDJ03] Andrieu, C., De Freitas, N., Doucet, A., & Jordan, M. I. (2003). An introduction to MCMC for machine learning. *Machine learning*, *50*(1-2), 5-43.

References

- Bayesian Inference: MCMC
  - Dynamics-Based MCMC: full-batch

    - [Lan08] Langevin, P. (1908). Sur la théorie du mouvement Brownien. *Compt. Rendus*, *146*, 530-533.
    - [DKPR87] Duane, S., Kennedy, A.D., Pendleton, B.J., Roweth, D. (1987). Hybrid Monte Carlo. *Physics letters B*, 195(2), pp.216-222.
    - [RT96] Roberts, G. O., & Tweedie, R. L. (1996). Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli*, *2*(4), 341-363.
    - [RS02] Roberts, G.O., & Stramer, O. (2002). Langevin diffusions and Metropolis-Hastings algorithms. *Methodology and computing in applied probability*, 4(4), pp.337-357.
    - [Nea11] Neal, R.M. (2011). MCMC using Hamiltonian dynamics. Handbook of Markov chain Monte Carlo, 2(11), p.2.
    - [ZWC+16] Zhang, Y., Wang, X., Chen, C., Henao, R., Fan, K., & Carin, L. (2016). Towards unifying Hamiltonian Monte Carlo and slice sampling. In *Advances in Neural Information Processing Systems* (pp. 1741-1749).
    - [TRGT17] Tripuraneni, N., Rowland, M., Ghahramani, Z., & Turner, R. (2017, August). Magnetic Hamiltonian Monte Carlo. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70* (pp. 3453-3461).
    - [Bet17] Betancourt, M. (2017). A conceptual introduction to Hamiltonian Monte Carlo. *arXiv preprint arXiv:1701.02434*.

References

- Bayesian Inference: MCMC
  - Dynamics-Based MCMC: full-batch (manifold support)
    - [GC11] Girolami, M., & Calderhead, B. (2011). Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *73*(2), 123-214.
    - [BSU12] Brubaker, M., Salzmann, M., & Urtasun, R. (2012, March). A family of MCMC methods on implicitly defined manifolds. In *Artificial intelligence and statistics* (pp. 161-172).
    - [BG13] Byrne, S., & Girolami, M. (2013). Geodesic Monte Carlo on embedded manifolds. *Scandinavian Journal of Statistics*, *40*(4), 825-845.
    - [LSSG15] Lan, S., Stathopoulos, V., Shahbaba, B., & Girolami, M. (2015). Markov chain Monte Carlo from Lagrangian dynamics. *Journal of Computational and Graphical Statistics*, *24*(2), 357-378.

References

- Bayesian Inference: MCMC
  - Dynamics-Based MCMC: stochastic gradient
    - [WT11] Welling, M., & Teh, Y. W. (2011). Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the International Conference on Machine Learning* (pp. 681-688).
    - [CFG14] Chen, T., Fox, E., & Guestrin, C. (2014). Stochastic gradient Hamiltonian Monte Carlo. In *Proceedings of the International conference on machine learning* (pp. 1683-1691).
    - [DFB+14] Ding, N., Fang, Y., Babbush, R., Chen, C., Skeel, R. D., & Neven, H. (2014). Bayesian sampling using stochastic gradient thermostats. In *Advances in neural information processing systems* (pp. 3203-3211).
    - [Bet15] Betancourt, M. (2015). The fundamental incompatibility of scalable Hamiltonian Monte Carlo and naive data subsampling. In *International Conference on Machine Learning* (pp. 533-540).
    - [TTV16] Teh, Y. W., Thiery, A. H., & Vollmer, S. J. (2016). Consistency and fluctuations for stochastic gradient Langevin dynamics. *The Journal of Machine Learning Research*, *17*(1), 193-225.
    - [LPH+16] Lu, X., Perrone, V., Hasenclever, L., Teh, Y. W., & Vollmer, S. J. (2016). Relativistic Monte Carlo. *arXiv preprint arXiv:1609.04388*.
    - [ZCG+17] Zhang, Y., Chen, C., Gan, Z., Henao, R., & Carin, L. (2017, August). Stochastic gradient monomial Gamma sampler. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70* (pp. 3996-4005).
    - [LTL17] Li, Y., Turner, R.E., & Liu, Q. (2017). Approximate inference with amortised MCMC. *arXiv preprint arXiv:1702.08343*.

References

- Bayesian Inference: MCMC
  - Dynamics-Based MCMC: stochastic gradient (manifold support)
    - [PT13] Patterson, S., & Teh, Y.W. (2013). Stochastic gradient Riemannian Langevin dynamics on the probability simplex. In *Advances in neural information processing system*s (pp. 3102-3110).
    - [MCF15] Ma, Y. A., Chen, T., & Fox, E. (2015). A complete recipe for stochastic gradient MCMC. In *Advances in Neural Information Processing Systems* (pp. 2917-2925).
    - [LZS16] Liu, C., Zhu, J., & Song, Y. (2016). Stochastic Gradient Geodesic MCMC Methods. In *Advances in Neural Information Processing Systems* (pp. 3009-3017).
  - Dynamics-Based MCMC: general theory
    - [JKO98] Jordan, R., Kinderlehrer, D., & Otto, F. (1998). The variational formulation of the Fokker-Planck equation. *SIAM journal on mathematical analysis*, *29*(1), 1-17.
    - [MCF15] Ma, Y. A., Chen, T., & Fox, E. (2015). A complete recipe for stochastic gradient MCMC. In *Advances in Neural Information Processing Systems* (pp. 2917-2925).
    - [CDC15] Chen, C., Ding, N., & Carin, L. (2015). On the convergence of stochastic gradient MCMC algorithms with high-order integrators. In *Advances in Neural Information Processing Systems* (pp. 2278-2286).
    - [LZZ19] Liu, C., Zhuo, J., & Zhu, J. (2019). Understanding MCMC Dynamics as Flows on the Wasserstein Space. In *Proceedings of the 36th International Conference on Machine Learning* (pp. 4093-4103).

References

- Bayesian Models
  - Bayesian Networks: Topic Models

    - [BNJ03] Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3(Jan), pp.993-1022.
    - [GS04] Griffiths, T.L., & Steyvers, M. (2004). Finding scientific topics. *Proceedings of the National academy of Sciences*, 101 (suppl 1), pp.5228-5235.
    - [SG07] Steyvers, M., & Griffiths, T. (2007). Probabilistic topic models. *Handbook of latent semantic analysis*, *427*(7), 424-440.
    - [MB08] Mcauliffe, J.D., & Blei, D.M. (2008). Supervised topic models. In *Advances in neural information processing systems* (pp. 121-128).
    - [ZAX12] Zhu, J., Ahmed, A., & Xing, E. P. (2012). MedLDA: maximum margin supervised topic models. *Journal of Machine Learning Research*, *13*(Aug), 2237-2278.
    - [PT13] Patterson, S., & Teh, Y.W. (2013). Stochastic gradient Riemannian Langevin dynamics on the probability simplex. In *Advances in neural information processing systems* (pp. 3102-3110).
    - [ZCX14] Zhu, J., Chen, N., & Xing, E. P. (2014). Bayesian inference with posterior regularization and applications to infinite latent SVMs. *The Journal of Machine Learning Research*, *15*(1), 1799-1847.

References

- Bayesian Models
  - Bayesian Networks: Topic Models
    - [LARS14] Li, A.Q., Ahmed, A., Ravi, S., & Smola, A.J. (2014). Reducing the sampling complexity of topic models. In *Proceedings of the ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 891-900).
    - [YGH+15] Yuan, J., Gao, F., Ho, Q., Dai, W., Wei, J., Zheng, X., Xing, E.P., Liu, T.Y., & Ma, W.Y. (2015). LightLDA: Big topic models on modest computer clusters. In *Proceedings of the 24th International Conference on World Wide Web* (pp. 1351-1361).
    - [CLZC16] Chen, J., Li, K., Zhu, J., & Chen, W. (2016). WarpLDA: a cache efficient o(1) algorithm for latent Dirichlet allocation. *Proceedings of the VLDB Endowment*, 9(10), pp.744-755.

References

- Bayesian Models
  - Bayesian Networks: Variational Auto-Encoders

    - [KW14] Kingma, D.P., & Welling, M. (2014). Auto-encoding variational Bayes. In *Proceedings of the International Conference on Learning Representations*.
    - [GDG+15] Gregor, K., Danihelka, I., Graves, A., Rezende, D. J., & Wierstra, D. (2015). DRAW: A recurrent neural network for image generation. In *Proceedings of the 32nd International Conference on Machine Learning*.
    - [BGS15] Burda, Y., Grosse, R., & Salakhutdinov, R. (2015). Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*.
    - [DFD+18] Davidson, T.R., Falorsi, L., De Cao, N., Kipf, T., & Tomczak, J.M. (2018). Hyperspherical variational auto-encoders. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*.
    - [MSJ+15] Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., & Frey, B. (2016). Adversarial Autoencoders. In *Proceedings of the International Conference on Learning Representations*.
    - [CDH+16] Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., & Abbeel, P. (2016). InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems* (pp. 2172-2180).
    - [MNG17] Mescheder, L., Nowozin, S., & Geiger, A. (2017). Adversarial variational Bayes: Unifying variational autoencoders and generative adversarial networks. In *Proceedings of the International Conference on Machine Learning* (pp. 2391-2400).

References

- Bayesian Models
  - Bayesian Networks: Variational Auto-Encoders

    - [TBGS17] Tolstikhin, I., Bousquet, O., Gelly, S., & Schoelkopf, B. (2017). Wasserstein Auto-Encoders. *arXiv preprint arXiv:1711.01558*.
    - [FWL17] Feng, Y., Wang, D., & Liu, Q. (2017). Learning to Draw Samples with Amortized Stein Variational Gradient Descent. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*.
    - [PGH+17] Pu, Y., Gan, Z., Henao, R., Li, C., Han, S., & Carin, L. (2017). VAE learning via Stein variational gradient descent. In *Advances in Neural Information Processing Systems* (pp. 4236-4245).
    - [KSDV18] Kocaoglu, M., Snyder, C., Dimakis, A. G., & Vishwanath, S. (2018). CausalGAN: Learning causal implicit generative models with adversarial training. In *Proceedings of the International Conference on Learning Representations*.
    - [LWZZ18] Li, C., Welling, M., Zhu, J., & Zhang, B. (2018). Graphical generative adversarial networks. In *Advances in Neural Information Processing Systems* (pp. 6069-6080).

References

- Bayesian Models
  - Markov Random Fields

    - [HS83] Hinton, G., & Sejnowski, T. (1983). Optimal perceptual inference. In *IEEE Conference on Computer Vision and Pattern Recognition*.
    - [Smo86] Smolensky, P. (1986). Information processing in dynamical systems: Foundations of harmony theory. In *Parallel Distributed Processing*, volume 1, chapter 6, pages 194-281. MIT Press.
    - [Hin02] Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8), 1771-1800.
    - [LCH+06] LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M., & Huang, F. (2006). A tutorial on energy-based learning. *Predicting structured data*, *1*(0).
    - [HOT06] Hinton, G. E., Osindero, S., & Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, 18(7), 1527-1554.
    - [SH09] Salakhutdinov, R., & Hinton, G. (2009, April). Deep Boltzmann machines. *In AISTATS* (pp. 448-455).
    - [Sal15] Salakhutdinov, R. (2015). Learning deep generative models. *Annual Review of Statistics and Its Application*, 2, 361-385.
    - [KB16] Kim, T., & Bengio, Y. (2016). Deep directed generative models with energy-based probability estimation. *arXiv preprint arXiv:1606.03439*.
    - [DM19] Du, Y., & Mordatch, I. (2019). Implicit generation and generalization in energy-based models. *arXiv preprint arXiv:1903.08689*.

References

- Others
  - Bayesian Models
    - [KYD+18] Kim, T., Yoon, J., Dia, O., Kim, S., Bengio, Y., & Ahn, S. (2018). Bayesian model-agnostic meta-learning. In *Advances in Neural Information Processing Systems* (pp. 7332-7342).
    - [LST15] Lake, B.M., Salakhutdinov, R., & Tenenbaum, J.B. (2015). Human-level concept learning through probabilistic program induction. *Science*, 350(6266), pp. 1332-1338.
  - Bayesian Neural Network
    - [LG17] Li, Y., & Gal, Y. (2017). Dropout inference in Bayesian neural networks with alpha-divergences. In *Proceedings of the International Conference on Machine Learning* (pp. 2052-2061).
  - Related References
    - [Wil92] Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, *8*(3-4), 229-256.
    - [HV93] Hinton, G., & Van Camp, D. (1993). Keeping neural networks simple by minimizing the description length of the weights. In *in Proc. of the 6th Ann. ACM Conf. on Computational Learning Theory*.
    - [NJ01] Ng, A. Y., & Jordan, M. I. (2002). On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in neural information processing systems* (pp. 841-848).

The End