Course number: 80240743

Deep Learning

Xiaolin Hu (胡晓林) & Jun Zhu (朱军)

Dept. of Computer Science and Technology

Tsinghua University

Deep Generative Models

Part 2: Variational Auto-Encoders

Jun Zhu

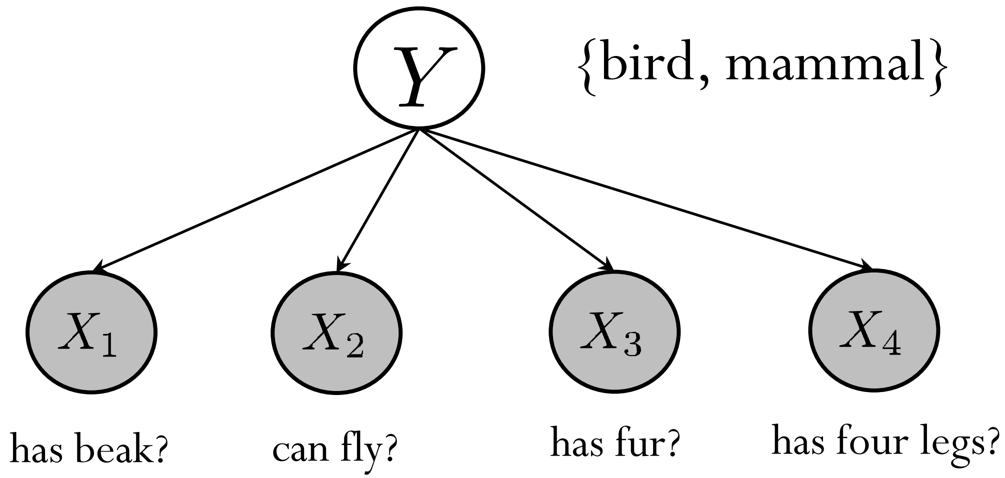
dcszj@mail.tsinghua.edu.cn

Department of Computer Science and Technology

Tsinghua University

Recap of Basics of Generative Models

- Naïve Bayes classifier



A joint distribution:

$$p(\mathrm{x}, y) = \underbrace{p(y)}_{\text{prior}}\underbrace{p(\mathrm{x}|y)}_{\text{likelihood}}$$

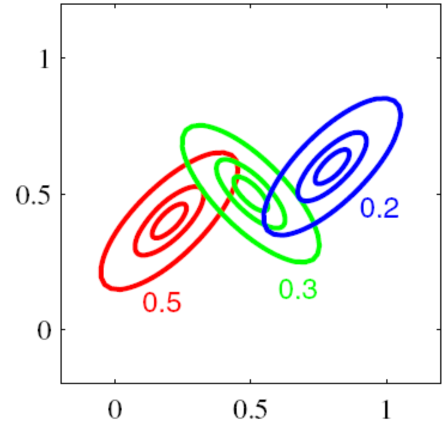
Bayes' decision rule:

$$y^* = \arg\max_{y \in \mathcal{Y}} p(y|\mathrm{x})$$

Recap of Basics of Generative Models

- Mixture of Gaussians

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k)$$

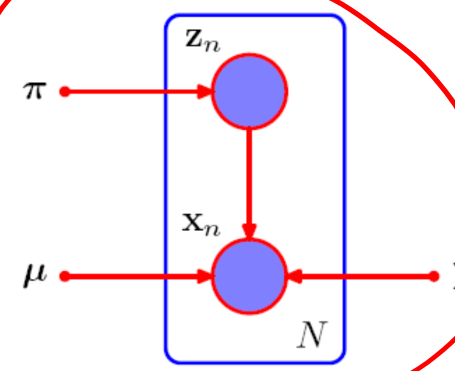


- Equivalent representation:

$$\mathbf{z} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

$$p(\mathbf{x}, \mathbf{z}) = \prod_{k=1}^{K} \pi_k^{z_k} \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \Sigma_k)^{z_k}$$

$$p(\mathbf{x}) = \sum_{\mathbf{z}} p(\mathbf{x}, \mathbf{z})$$

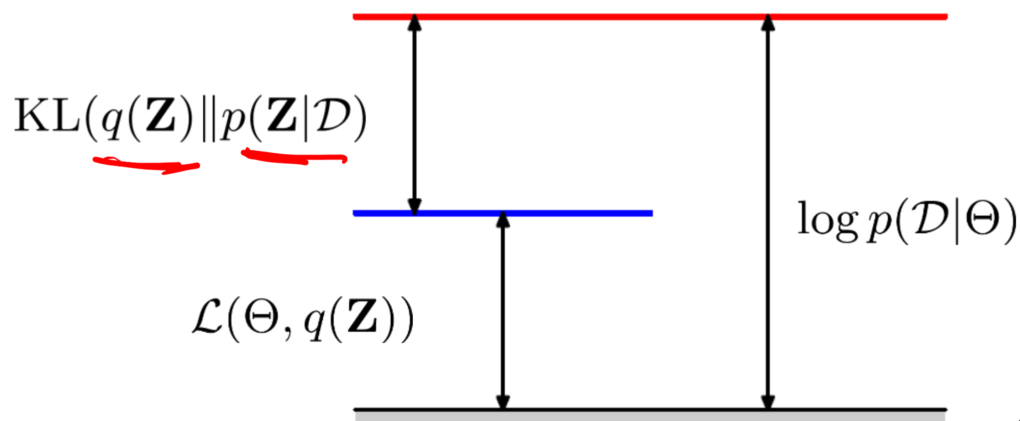


Recap of Basics of Generative Models

- Maximize the lower bound or minimize the gap:

$$\log p(\mathcal{D}|\Theta) \geq \sum_{n=1}^{N} \sum_{\mathbf{z}_n} q(\mathbf{z}_n) \log \left(\frac{p(\mathbf{x}_n, \mathbf{z}_n)}{q(\mathbf{z}_n)} \right) \triangleq \mathcal{L}(\Theta, q(\mathbf{Z}))$$

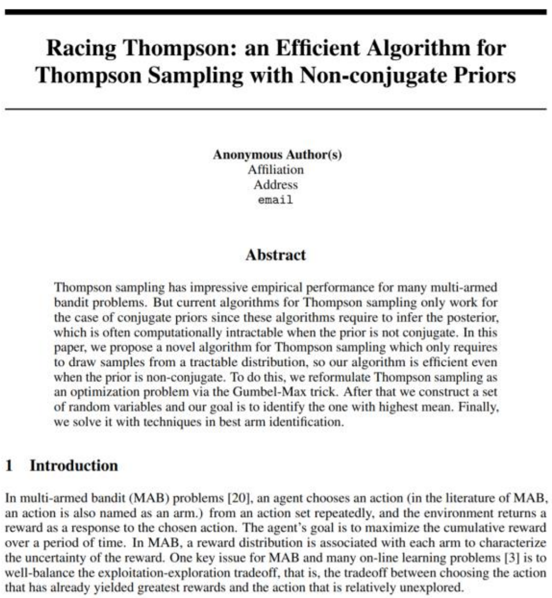
  - Maximize over q(Z) => E-step
  - Maximize over Θ => M-step



Language model revisited

- A simple unigram language model
  - Observations (e.g., bag-of-words)

$$\mathrm{x} = \{x_1, \ldots, x_d\}$$



Racing Thompson: an Efficient Algorithm for Thompson Sampling with Non-conjugate Priors

Anonymous Author(s)
Affiliation
Address
email

Abstract

Thompson sampling has impressive empirical performance for many multi-armed bandit problems. But current algorithms for Thompson sampling only work for the case of conjugate priors since these algorithms require to infer the posterior, which is often computationally intractable when the prior is not conjugate. In this paper, we propose a novel algorithm for Thompson sampling which only requires to draw samples from a tractable distribution, so our algorithm is efficient even when the prior is non-conjugate. To do this, we reformulate Thompson sampling as an optimization problem via the Gumbel-Max trick. After that we construct a set of random variables and our goal is to identify the one with highest mean. Finally, we solve it with techniques in best arm identification.

1 Introduction

In multi-armed bandit (MAB) problems [20], an agent chooses an action (in the literature of MAB, an action is also named as an arm.) from an action set repeatedly, and the environment returns a reward as a response to the chosen action. The agent's goal is to maximize the cumulative reward over a period of time. In MAB, a reward distribution is associated with each arm to characterize the uncertainty of the reward. One key issue for MAB and many on-line learning problems [3] is to well-balance the exploitation-exploration tradeoff, that is, the tradeoff between choosing the action that has already yielded greatest rewards and the action that is relatively unexplored.

Term	D1	D2
game	1	0
decision	0	0
theory	2	0
probability	0	3
analysis	0	2
…		

Language model revisited

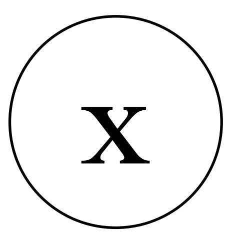
- A simple unigram language model
  - Observations (e.g., bag-of-words)

$$\mathrm{x} = \{x_1, \dots, x_d\}$$

  - Joint distribution (likelihood)

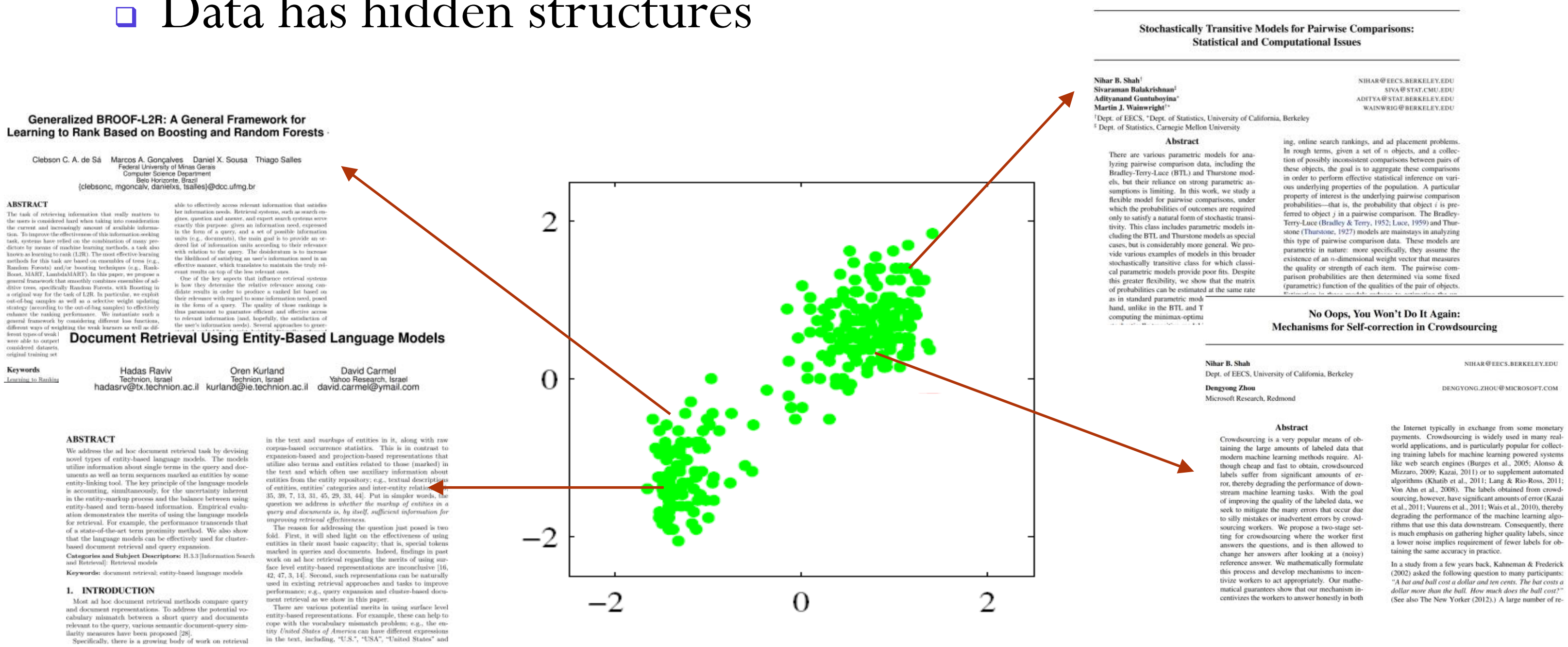
$$p(\mathrm{x}; \theta) = \prod p(x_i \,; \theta)$$

  - Graphical representation (parameters ignored)



Language model revisited

- A fully-observed model is not sufficient
  - Data has hidden structures



  - A simple distribution is not sufficient

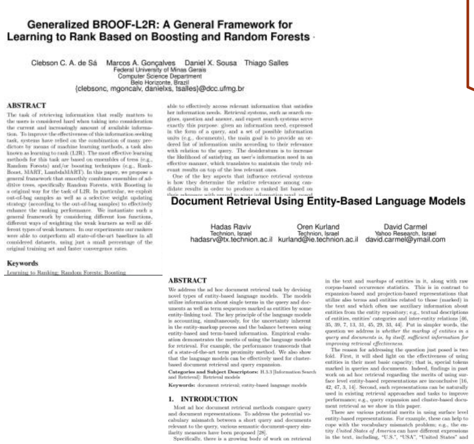
Language model revisited

- Mixture model --- a simple generative model with hidden factors
  - Separate the data into different groups

T1="SIGIR"

T2="ICML"

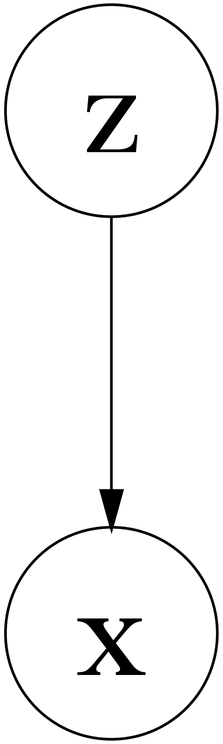
retrieval 0.05
text 0.02
learning 0.01
sports 0.01
…



learning 0.03
theory 0.02
algorithm 0.01
deep 0.01
…

Language model revisited

- Mixture model --- a simple generative model with hidden factors
  - Graphical model representation



$p(\mathrm{z})$

$p(\mathrm{x}|\mathrm{z};\theta)$

$$p(\mathrm{x},\mathrm{z}) = p(\mathrm{z})p(\mathrm{x}|\mathrm{z};\theta)$$


Generalized BROOF-L2R: A General Framework for Learning to Rank Based on Boosting and Random Forests

Clebson C. A. de Sá, Marcos A. Gonçalves, Daniel X. Sousa, Thiago Salles
Federal University of Minas Gerais
Computer Science Department
Belo Horizonte, Brazil
{clebsonc, mgoncalv, danielxs, tsalles}@dcc.ufmg.br

Document Retrieval Using Entity-Based Language Models

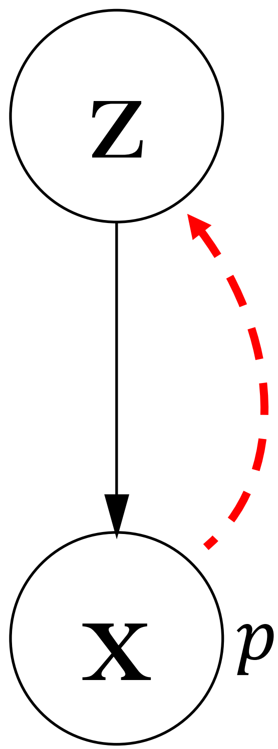
Hadas Raviv, Technion, Israel, hadasrv@tx.technion.ac.il
Oren Kurland, Technion, Israel, kurland@ie.technion.ac.il
David Carmel, Yahoo Research, Israel, david.carmel@ymail.com

Stochastically Transitive Models for Pairwise Comparisons: Statistical and Computational Issues

No Oops, You Won't Do It Again: Mechanisms for Self-correction in Crowdsourcing

Language model revisited

- Mixture model --- a simple generative model with hidden factors
  - Infer the latent Z:



$p(\mathrm{z})$

$p(\mathrm{x}|\mathrm{z};\theta)$

Bayes' Rule:

$$p(\mathrm{z}|\mathrm{x}) = \frac{p(\mathrm{x},\mathrm{z})}{p(\mathrm{x})}$$

$$\propto p(\mathrm{z})p(\mathrm{x}|\mathrm{z};\theta)$$

No Oops, You Won't Do It Again:
Mechanisms for Self-correction in Crowdsourcing

Nihar B. Shah NIHAR@EECS.BERKELEY.EDU
Dept. of EECS, University of California, Berkeley

Dengyong Zhou DENGYONG.ZHOU@MICROSOFT.COM
Microsoft Research, Redmond

Abstract

Crowdsourcing is a very popular means of obtaining the large amounts of labeled data that modern machine learning methods require. Although cheap and fast to obtain, crowdsourced labels suffer from significant amounts of error, thereby degrading the performance of downstream machine learning tasks. With the goal of improving the quality of the labeled data, we seek to mitigate the many errors that occur due to silly mistakes or inadvertent errors by crowdsourcing workers. We propose a two-stage setting for crowdsourcing where the worker first answers the questions, and is then allowed to change her answers after looking at a (noisy) reference answer. We mathematically formulate this process and develop mechanisms to incentivize workers to act appropriately. Our mathematical guarantees show that our mechanism incentivizes the workers to answer honestly in both

the Internet typically in exchange from some monetary payments. Crowdsourcing is widely used in many real-world applications, and is particularly popular for collecting training labels for machine learning powered systems like web search engines (Burges et al., 2005; Alonso & Mizzaro, 2009; Kazai, 2011) or to supplement automated algorithms (Khatib et al., 2011; Lang & Rio-Ross, 2011; Von Ahn et al., 2008). The labels obtained from crowdsourcing, however, have significant amounts of error (Kazai et al., 2011; Vuurens et al., 2011; Wais et al., 2010), thereby degrading the performance of the machine learning algorithms that use this data downstream. Consequently, there is much emphasis on gathering higher quality labels, since a lower noise implies requirement of fewer labels for obtaining the same accuracy in practice.

In a study from a few years back, Kahneman & Frederick (2002) asked the following question to many participants: *"A bat and ball cost a dollar and ten cents. The bat costs a dollar more than the ball. How much does the ball cost?"* (See also The New Yorker (2012).) A large number of re-

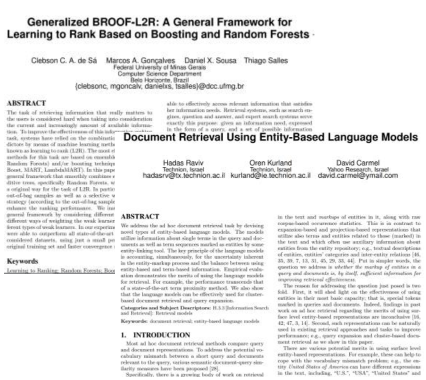
Language model revisited

- Mixture model --- a simple generative model with hidden factors
  - EM algorithm to learn the unknown language models

E-step: Infer the hidden Z

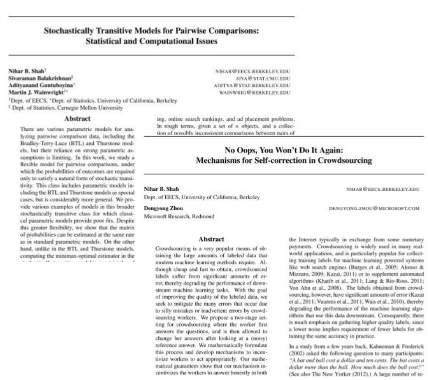
M-step: Update the parameters

T1="SIGIR" T2="ICML"


Generalized BROOF-L2R: A General Framework for Learning to Rank Based on Boosting and Random Forests

Document Retrieval Using Entity-Based Language Models

Hadas Raviv, Oren Kurland, David Carmel


Stochastically Transitive Models for Pairwise Comparisons: Statistical and Computational Issues

No Oops, You Won't Do It Again: Mechanisms for Self-correction in Crowdsourcing

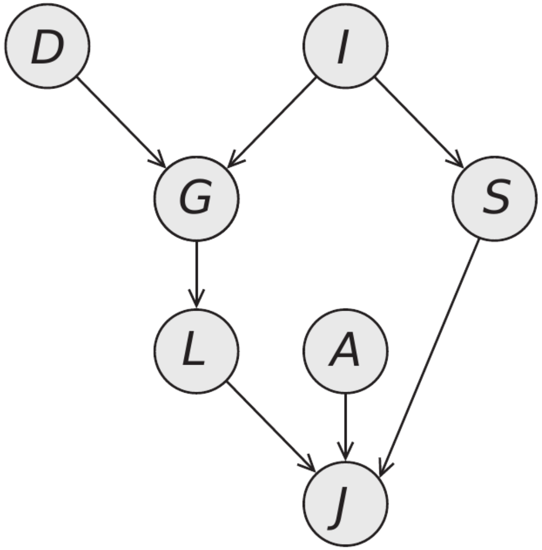
retrieval	0.05
text	0.02
learning	0.01
sports	0.01
…	

learning	0.03
theory	0.02
algorithm	0.01
deep	0.01
…	

Going Deep …

Hierarchical Bayesian Modeling

- Build a hierarchy throng distributions in analytical forms



$$P(D, I, G, S, L, A, J) = P(D)P(I)P(G|D, I)P(S|I) \cdot P(A)\, P(L|G)P(J|L, A, S)$$

Simple, Local Factors: a conditional probability distribution

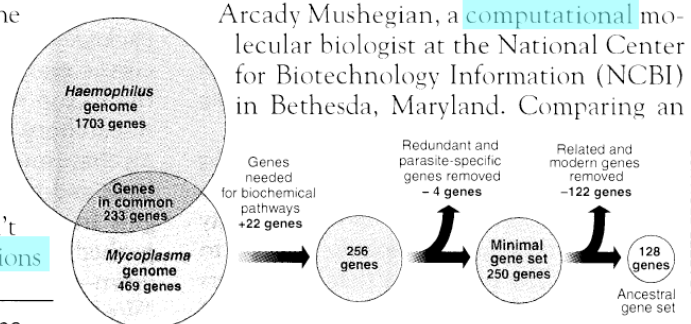
Multiple Topics exist in a Document

- Go deeper into a single document

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.

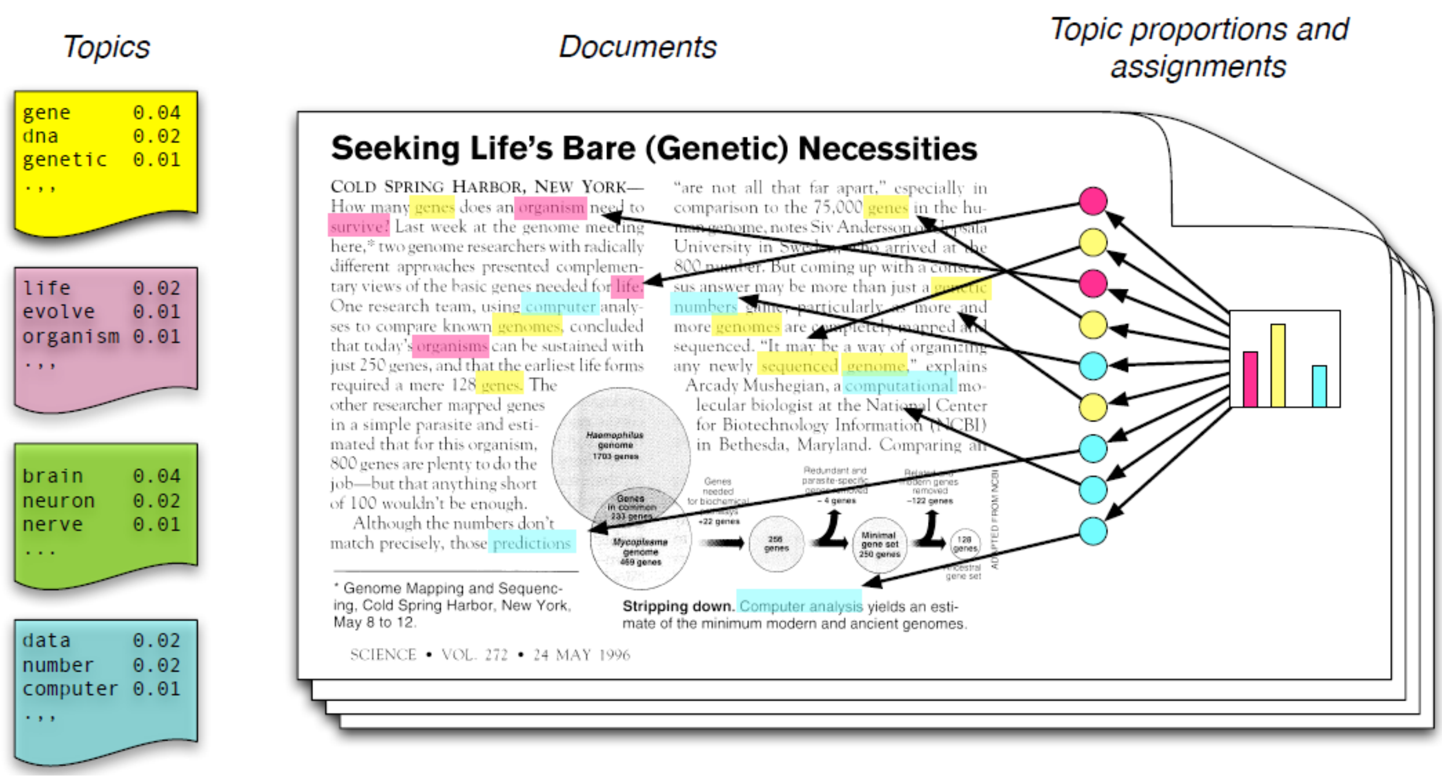
* Genome Mapping and Sequencing, Cold Spring Harbor, New York, May 8 to 12.

SCIENCE • VOL. 272 • 24 MAY 1996

Simple intuition: Documents exhibit multiple topics.

[Slides courtesy: D. Blei]

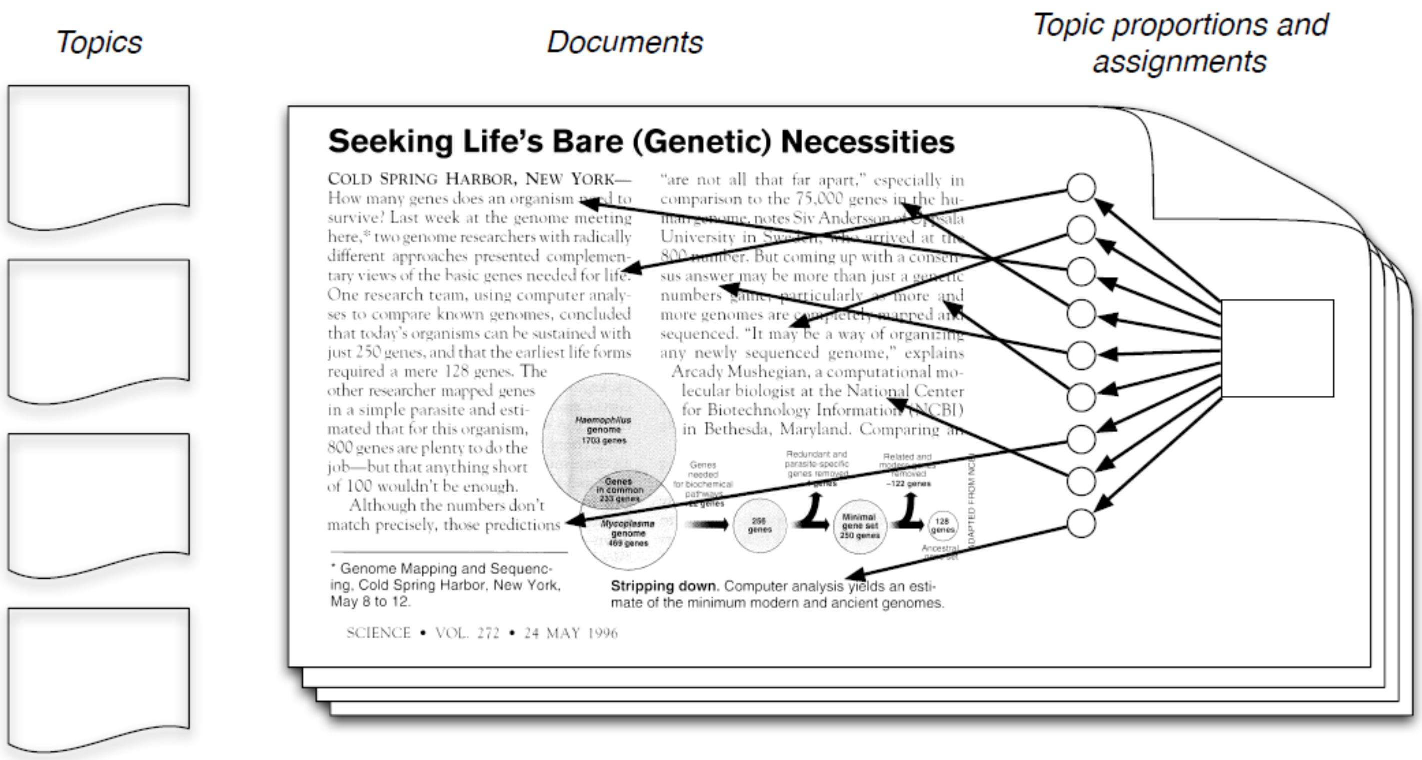
Latent Dirichlet Allocation (LDA)



- Each **topic** is a distribution over words
- Each **document** is a mixture of corpus-wide topics
- Each **word** is drawn from one of those topics

[Slides courtesy: D. Blei]

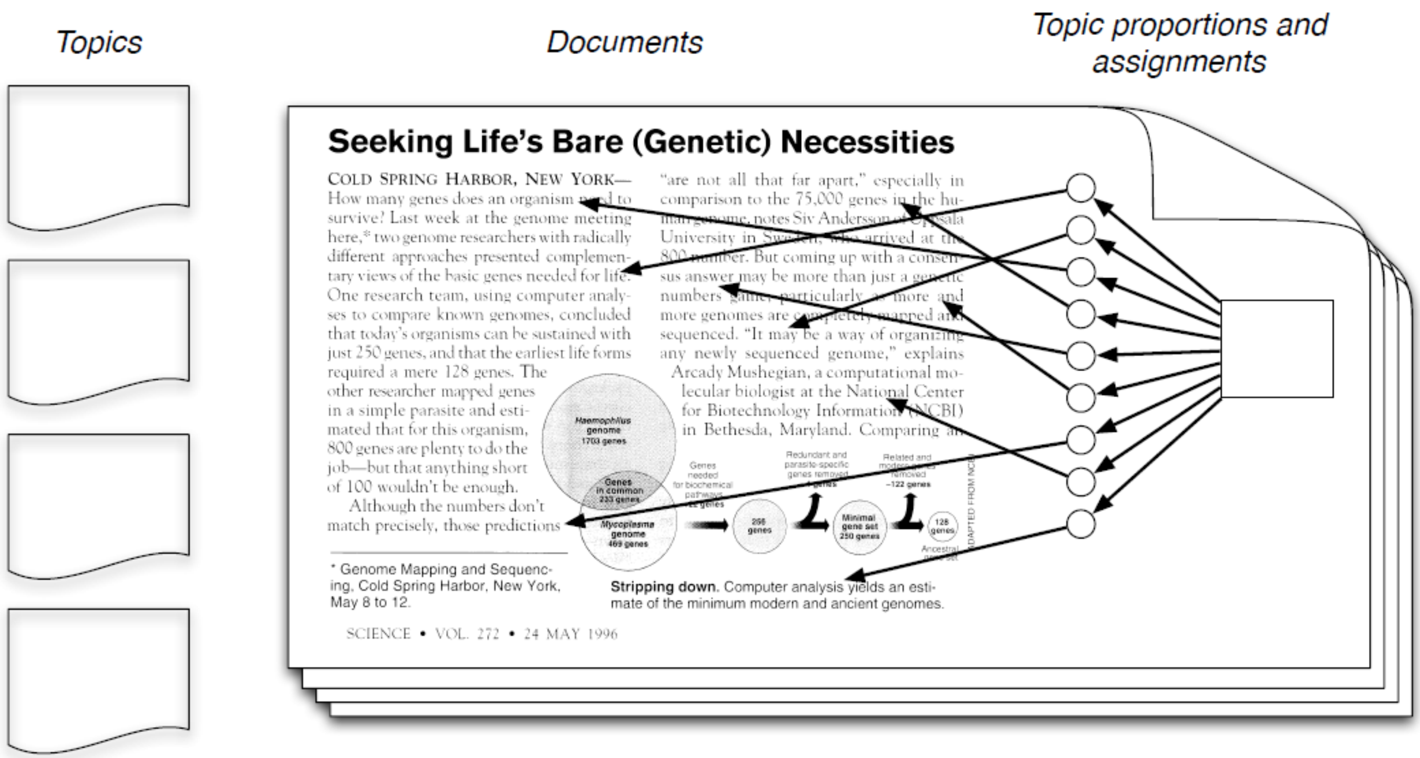
Latent Dirichlet Allocation



- In reality, we only observe the documents
- The other structure are **hidden variables**

[Slides courtesy: D. Blei]

Latent Dirichlet Allocation



- Our goal is to **infer** the hidden variables
- I.e., compute their distribution conditioned on the documents

$$p(\text{topics, proportions, assignments} \mid \text{documents})$$

[Slides courtesy: D. Blei]

- Applications
  - Data visualization
  - Natural language processing
  - Recommendation system
  - Computer vision
  - Information retrieval
  - and many more
- Alternative views
  - Clustering
  - Matrix factorization
  - Discrete PCA

Learning with Big Data

- Data can be very large
- The ClueWeb12 dataset (webpages)
  - #docs (D): billion
  - #vocabulary (V): million
  - #topics (K): million
  - #tokens (T): 100 billion
  - Raw size (html): 27TB
  - Text: 700GB

- Our recent work:
  - WarpLDA on CPU clusters (VLDB 2016)
  - SaberLDA on GPUs (ASPLOS 2017)
  - Scalable dynamic topic models (WWW 2016)
  - Online Bayesian passive-aggressive learning (JMLR 2017)
  - Scalable CTM (NIPS 2013)

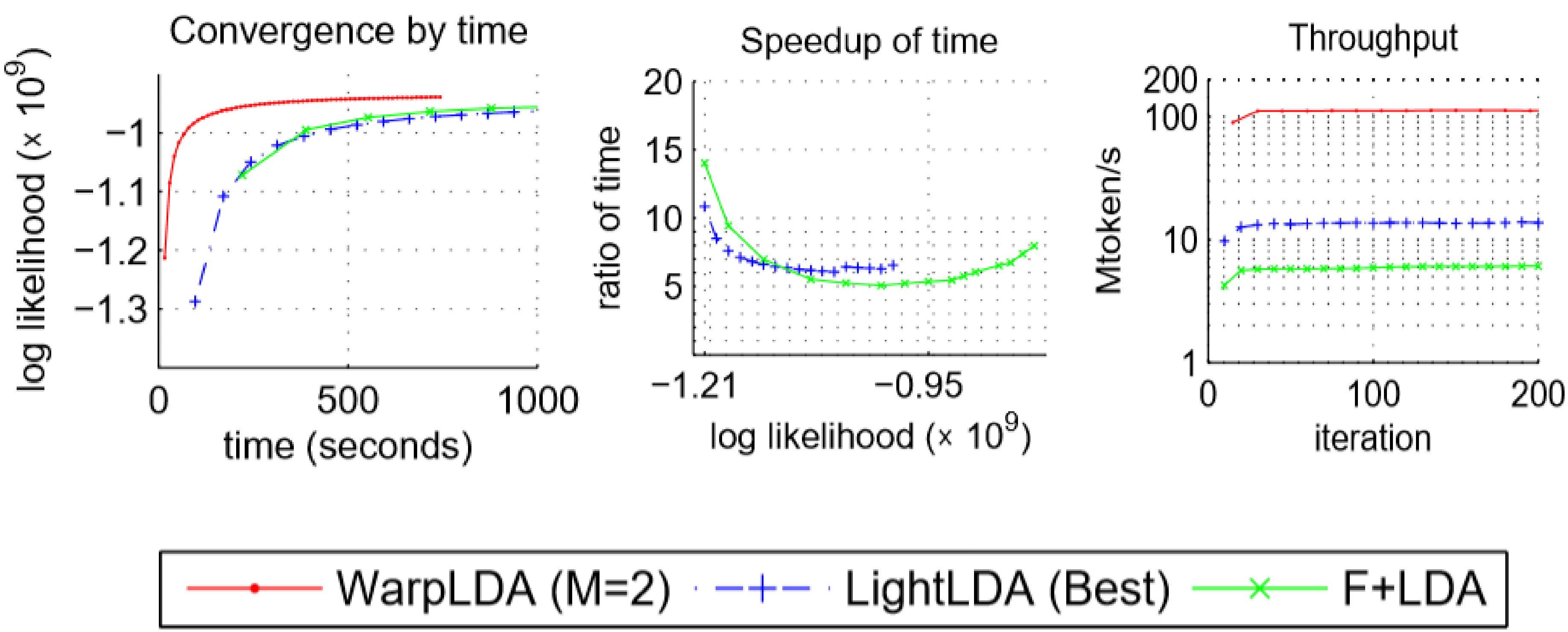
Learning with Big Data

- Data can be very large
- The ClueWeb12 dataset (webpages)
  - #docs (D): billion
  - #vocabulary (V): million
  - #topics (K): million
  - #tokens (T): 100 billion
  - Raw size (html): 27TB
  - Text: 700GB

- Our recent work:
  - WarpLDA on CPU clusters (VLDB 2016)
  - SaberLDA on GPUs (ASPLOS 2017)
  - Scalable dynamic topic models (WWW 2016)
  - Online Bayesian passive-aggressive learning (JMLR 2017)
  - Scalable CTM (NIPS 2013)

WarpLDA

- Much faster convergence than previous methods



WarpLDA

- Large data and model
  - The ClueWeb12 dataset
    - 600M documents
    - 200G tokens = 1600GB space to store
    - 1M vocabulary
    - 1M topics
    - Cdk: 600M * 1M
    - Cwk: 1M * 1M sparse = 100GB-1TB space to store
    - Ck: 1M = 8MB to store

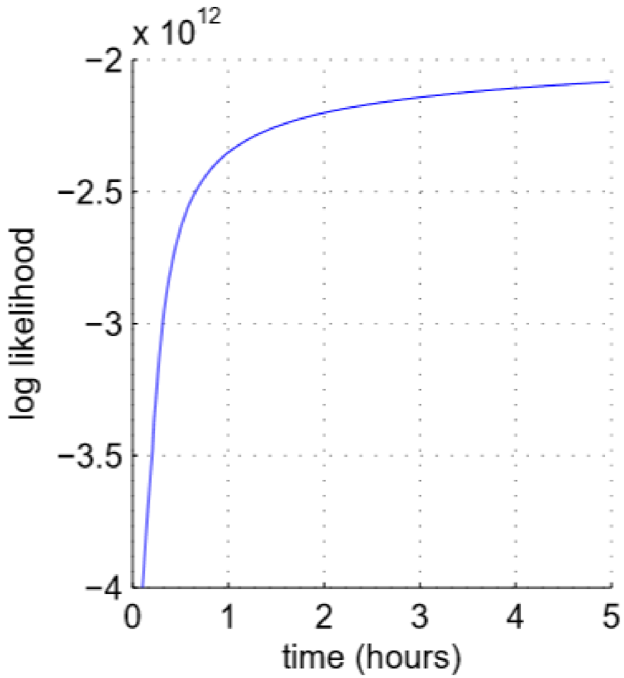
- Accessed sequentially
- Can be distributed

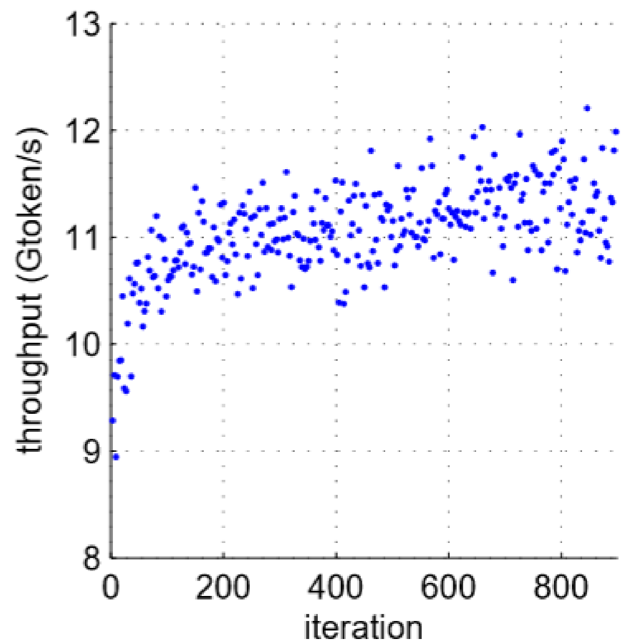
- Accessed randomly, but not in cache!
- Not even fit in single node

- Fits in L3 cache

WarpLDA

- Very scalable
  - 256 machines on Tianhe-2 achieves 12Gtoken/s
  - Previous methods: ~100Mtoken/s





WarpLDA

- Mines meaningful topics from 600M web pages

机器学习：26 topics

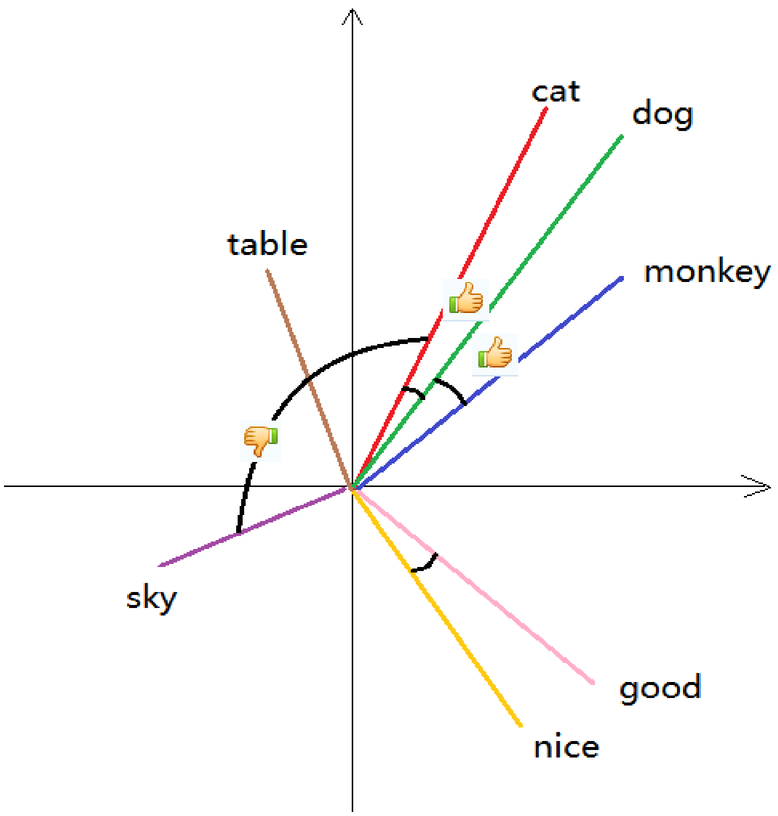
kernel machines learning workshop vector sch nips tutorials call smola
workshop learning conference machine international systems intelligence artificial computational canada
learning machine analysis data algorithms statistics computational theory complexity statistical

清华大学：4 topics

iii iiis tsinghua titled yao itcs titles achievements introduction hrd
university china tsinghua chinese beijing english peking wang shanghai zhang
journal journals international applied tsinghua practice bulletin american issues annals
ihep physics lnt beijing institute peking hep tsinghua energy cas

Generative Topic Embedding

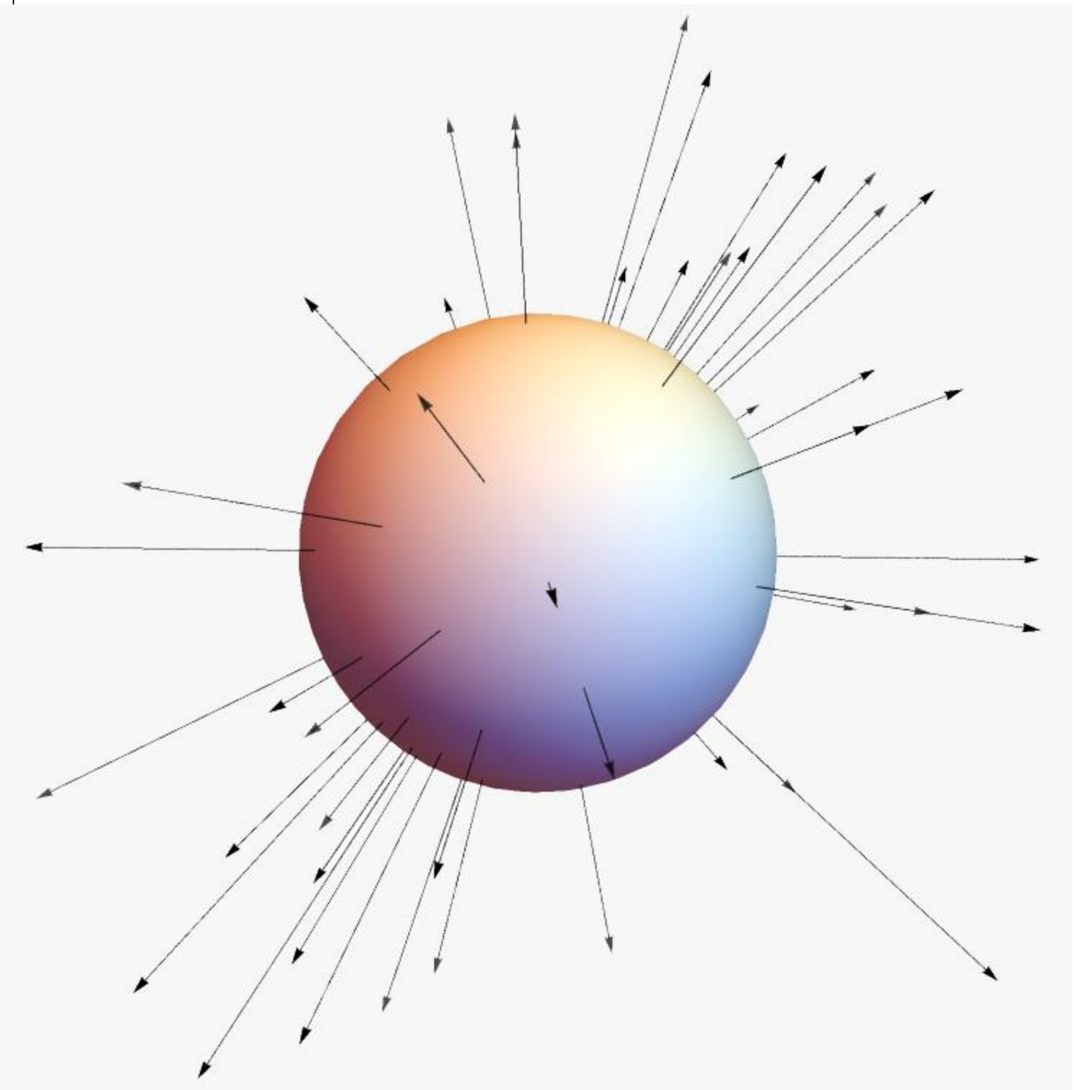
- Word Embedding
  - Widely used in NLP applications
  - Maps words into continuous, low dimensional vectors
  - Vector direction usually encodes semantics
- Embedding methods
  - Paragraph Vector, CNN, …
- LDA + Embedding?
  - Gaussian LDA



Projection of the embedding vectors to 2-D

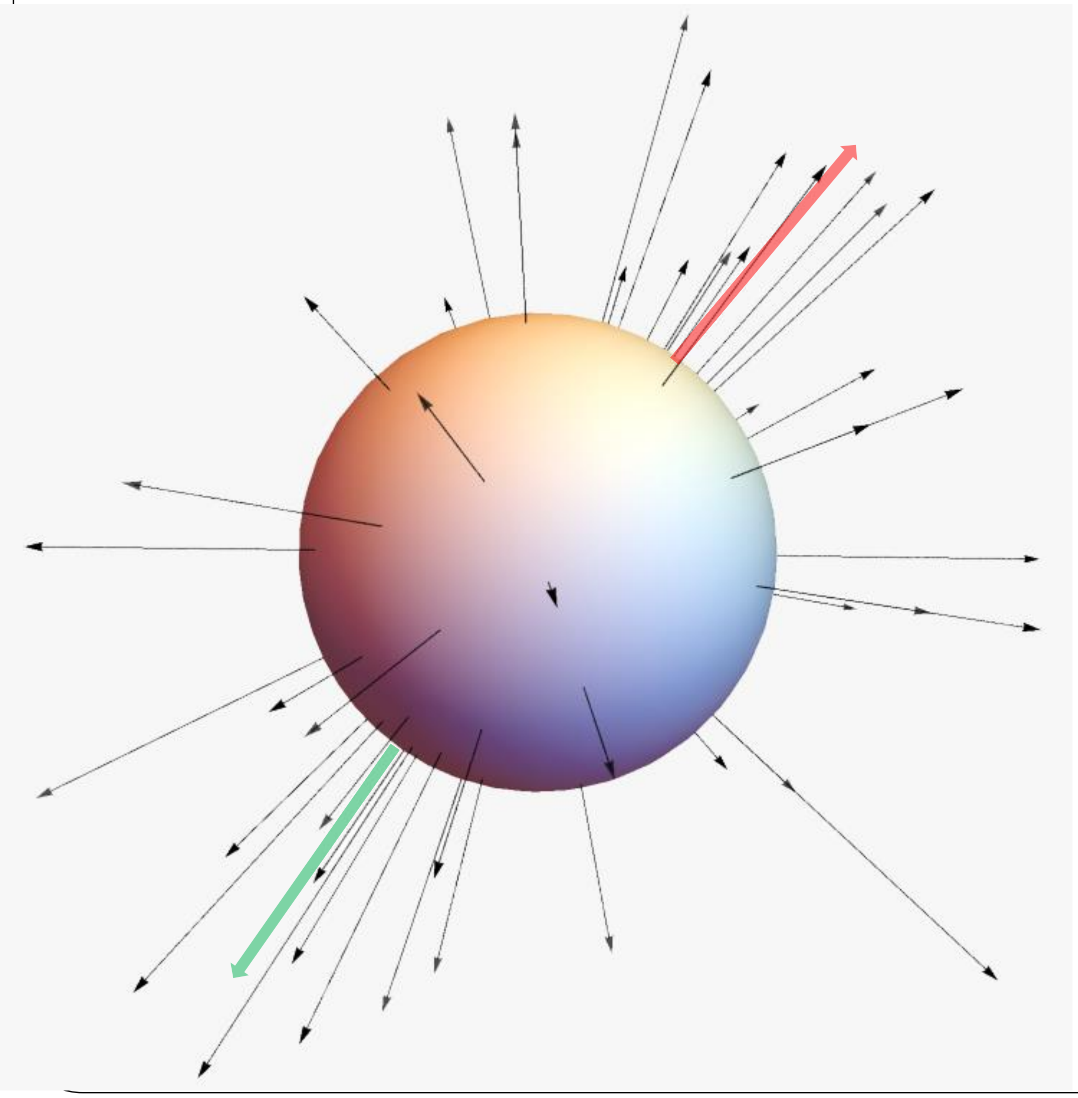
[Li et al., ACL 2016]

Bag-of-Vectors View of a Document



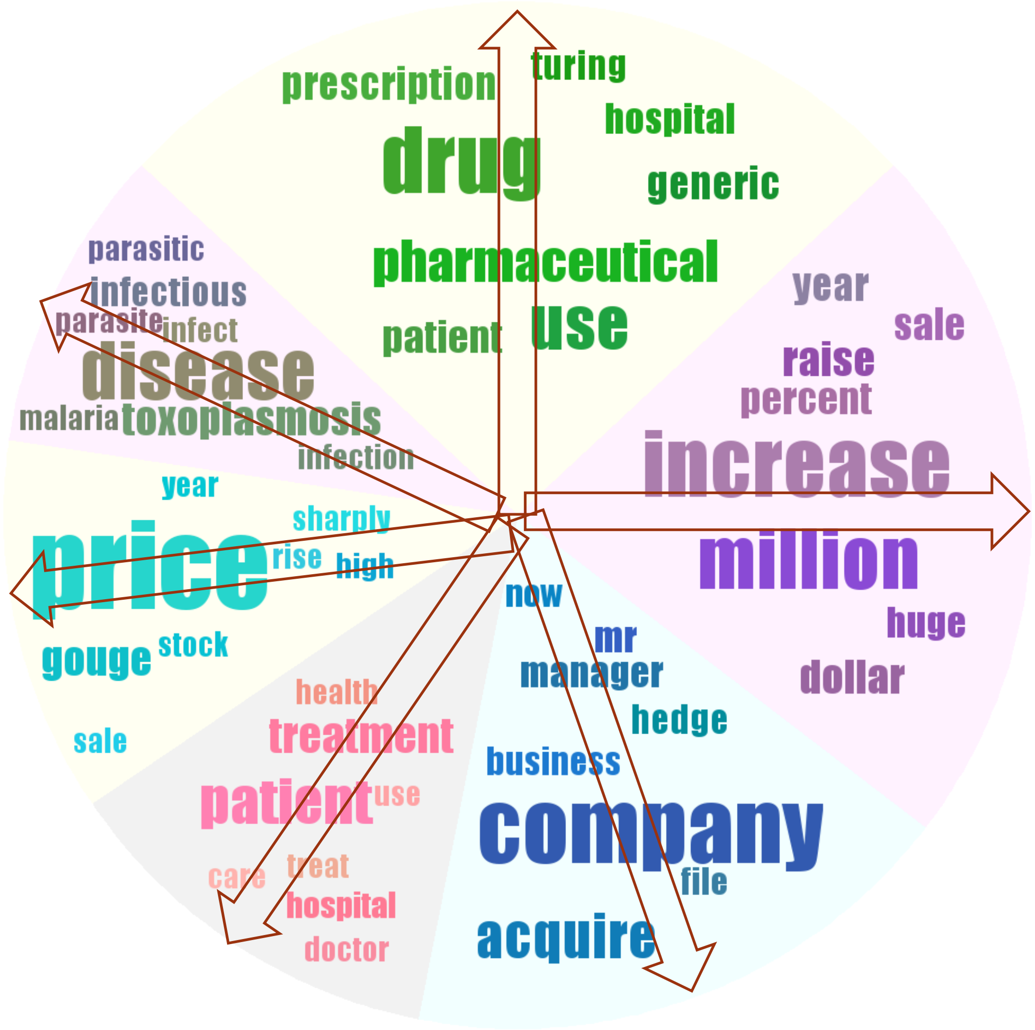
- Represent each word by its embedding vector
- A doc = a set of vectors shooting in different directions
- Vectors form clusters
- Directions containing more vectors ≈ typical semantics

Topic Embeddings in a Single Document



- Find a typical vector for a cluster of vectors
- Left graph: two typical vectors ➡, ➡
- Typical vectors ≈ major semantics = Topics
- Topic embeddings represent the document concisely
- Minor semantics are ignored

Topic Embeddings Illustrated



- On a news article about a pharmaceutical company acquisition

Topic Embeddings Illustrated (2)

- 16944 tweets after Baidu Ads scandal (Wei Zexi incident)

The Topic in a Topic Model

- Topic (a.k.a. **Topic-word distribution**)
  - Each word in a document has a latent topic
  - Each topic is a categorical distribution of words

$$p(w_1|\boldsymbol{t}_k) = p_{k1}, \cdots, p(w_W|\boldsymbol{t}_k) = p_{kW}$$

  - Most topics may have some vague semantics
    - Sports, electronics, computer science…
  - Semantically more related words have higher prob
- Disadvantage
  - Each word needs a probability as a parameter
  - K*(W-1) parameters – too many!

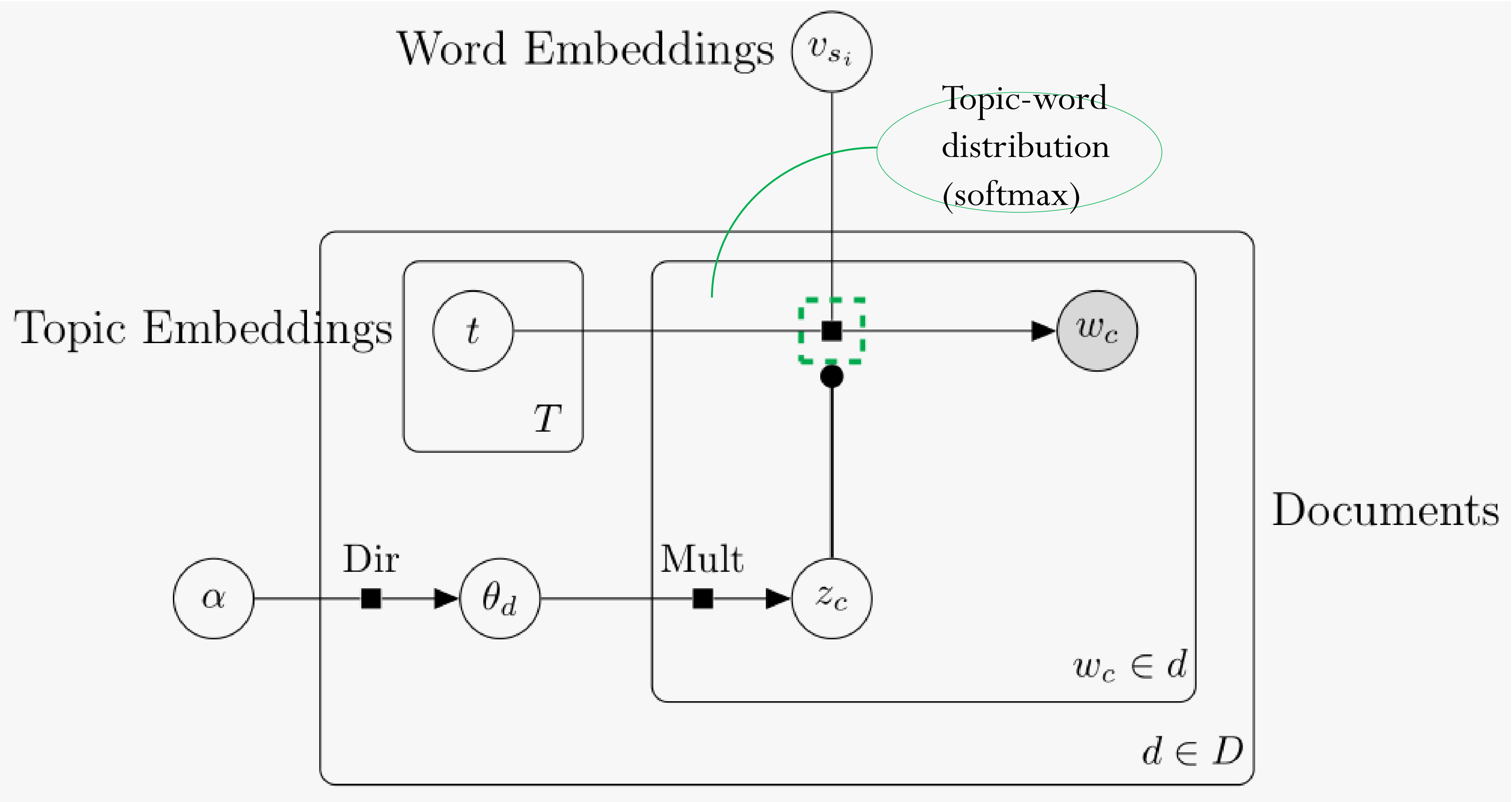
Continuous Representation of Topics

- Probability of each word w_i under topic k:

$$p(w_i|\boldsymbol{t}_k) = \frac{P(w_i)\exp\{\boldsymbol{v}_{w_i}^T \boldsymbol{t}_k\}}{\sum_j P(w_j)\exp\{\boldsymbol{v}_{w_j}^T \boldsymbol{t}_k\}}$$

  - $P(w_i)$: prior prob of w_i
  - $\boldsymbol{v}_{w_i}$: pretrained word embedding
- Predict a word probability by the dot-product of the topic embedding and the word embedding
  - Semantically more related words have higher prob
  - Ensures topic coherence
- Topic embedding: $\boldsymbol{t}_k$ (100~500 dimensional)
  - Parameter number: 500*K

Graphical Model



- Major difference with LDA: formulation of topic-word distribution

Evaluation on Document Classification

	20News			Reuters		
	Prec	Rec	F1	Prec	Rec	F1
BOW	69.1	68.5	68.6	92.5	90.3	91.1
LDA	61.9	61.4	60.3	76.1	74.3	74.8
sLDA	61.4	60.9	60.9	88.3	83.3	85.1
LFTM	63.5	64.8	63.7	84.6	86.3	84.9
MeanWV	70.4	70.3	70.1	92.0	89.6	90.5
Doc2Vec	56.3	56.6	55.4	84.4	50.0	58.5
TWE	69.5	69.3	68.8	91.0	89.1	89.9
GaussianLDA	30.9	26.5	22.7	46.2	31.5	35.3
TopicVec	71.3	71.3	71.2	**92.5**	**92.1**	**92.2**
TV+MeanWV	**71.8**	**71.5**	**71.6**	92.2	91.6	91.6

- It measures how much semantic info is preserved
- TopicVec: Topic proportions as features
- TopicVec & TV+WV are best
- MeanWV (mean word embedding) is not always helpful for TopicVec

Leverage deep neural networks …

Deep Generative Models

- If z is uniformly distributed over (0, 1), then $y = f(z)$ has the distribution

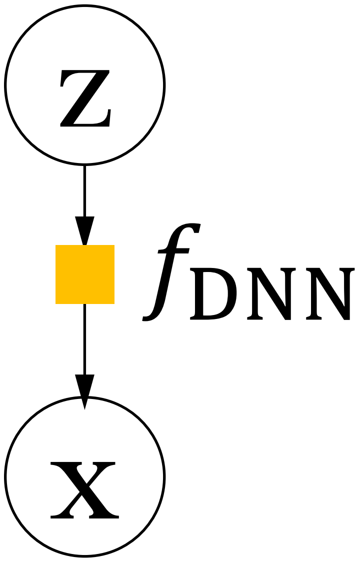
$$p(y) = p(z)\left|\frac{dz}{dy}\right|$$

  - where $p(z) = 1$
- This trick is widely used to draw samples from exponential family distributions (e.g., Gaussian, Exponential)
- *However, the function is typically simple. Can we learn it?*

Deep Generative Models

- **More flexible by using differential function mapping between random variables**

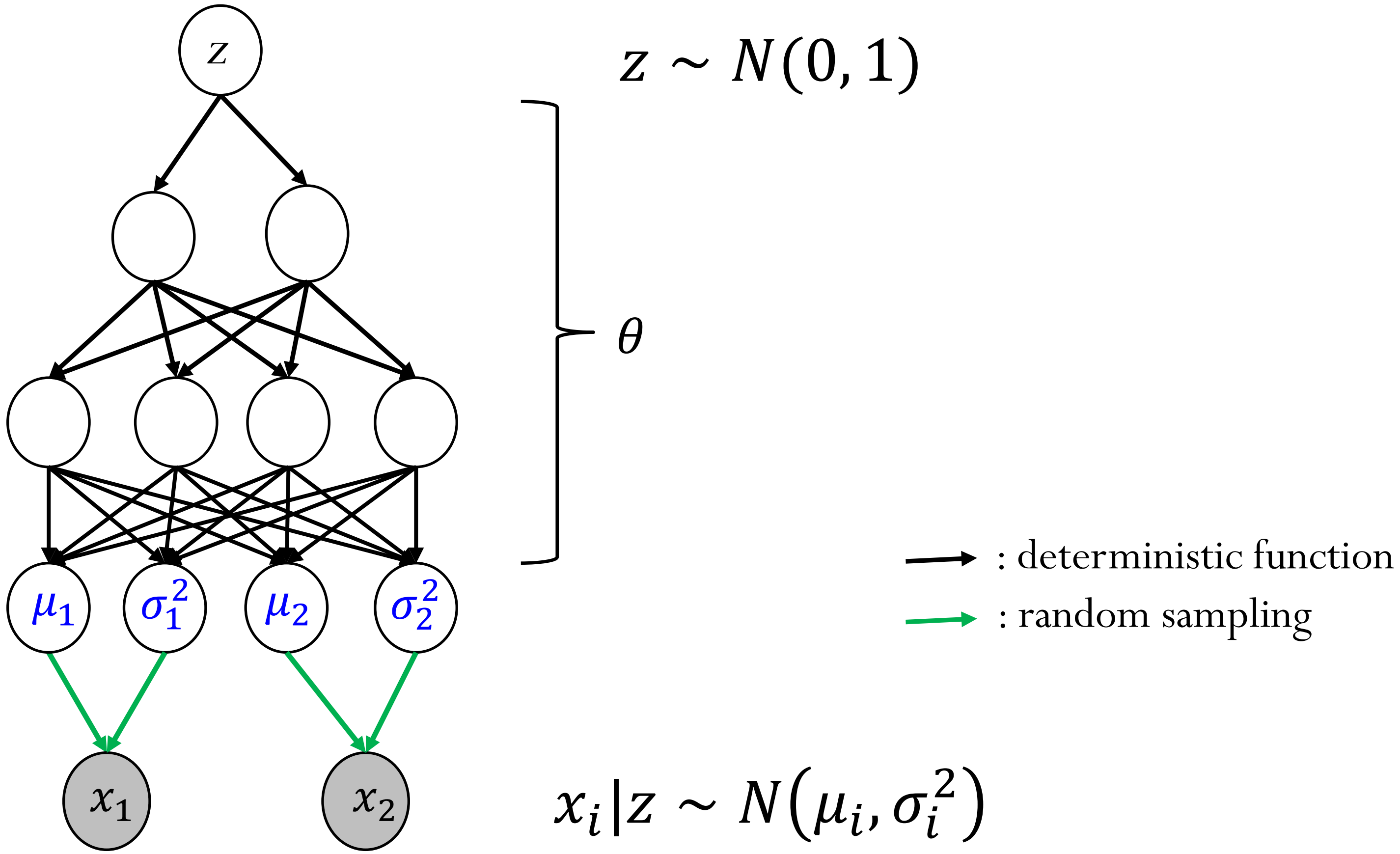
- DGMs learn a function transform with deep neural networks



Cause => Disease
Topics => Docs
Objects => Images
Words => Phonemes

An example with MLP

- 1D latent variable z; 2D observation x
- **Idea**: NN + Gaussian (or Bernoulli) with a diagonal covariance

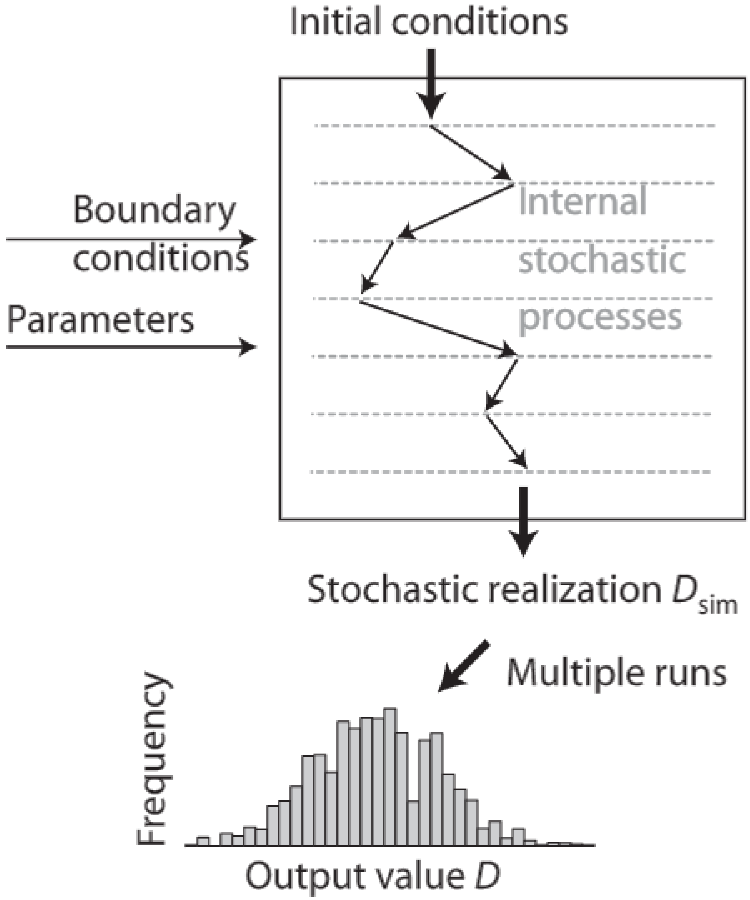


$z \sim N(0,1)$

$x_i | z \sim N(\mu_i, \sigma_i^2)$

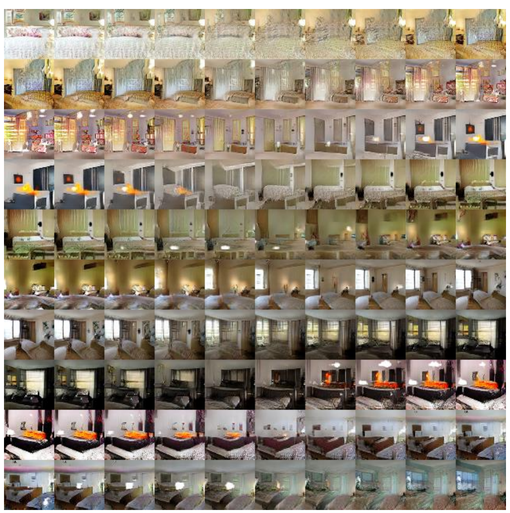
Implicit Deep Generative Models

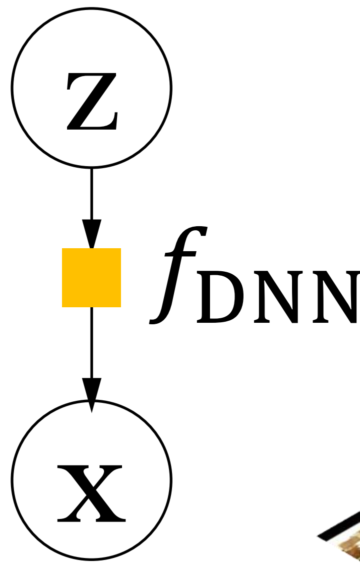
- Generate data with a stochastic process whose likelihood function is not explicitly specified (Hartig et al., 2011)

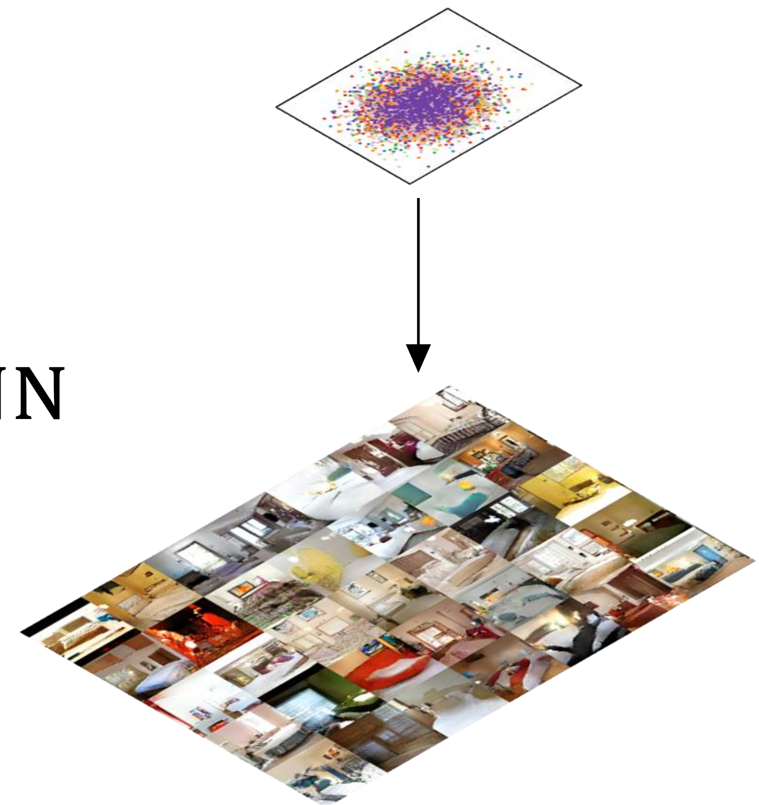


Deep Generative Models

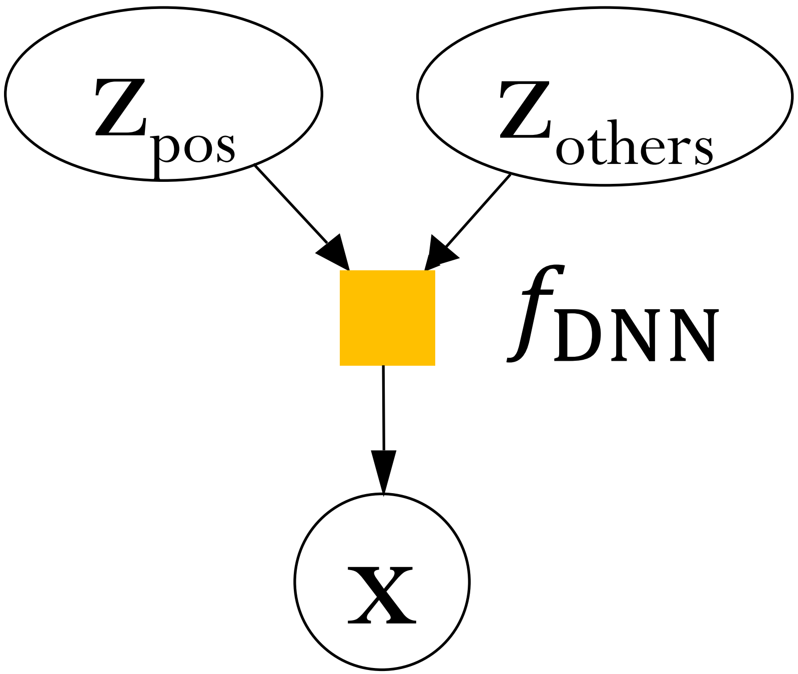
[Image Generation:
Generative Adversarial Nets,
Goodfellow13 & Radford15]

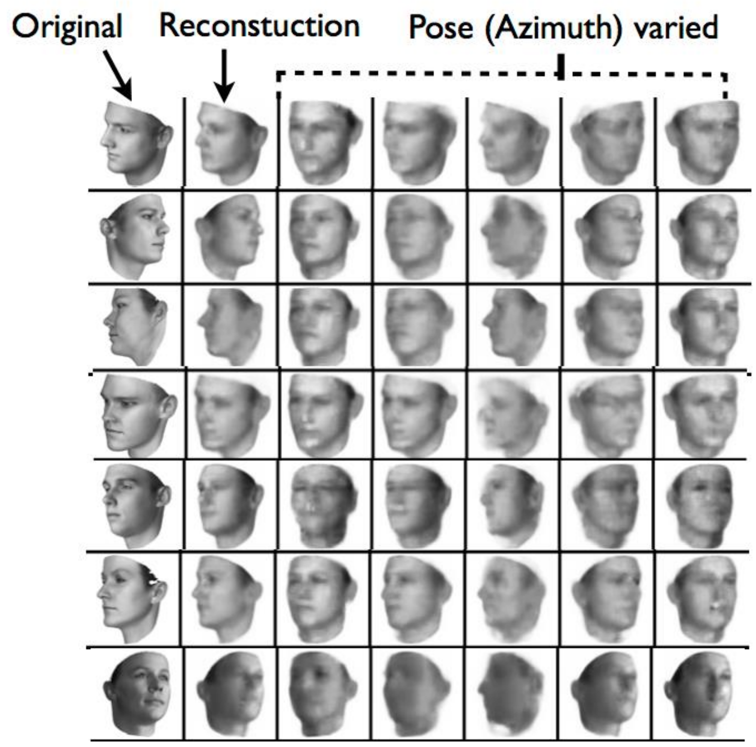






Deep Generative Models



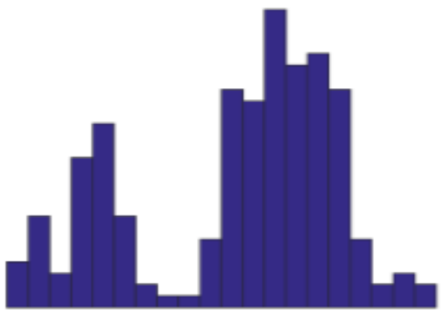


[Image Understanding: Variational Autoencoders, Kingma13 & Tejas15 & Eslami16]

Learning Deep Generative Models

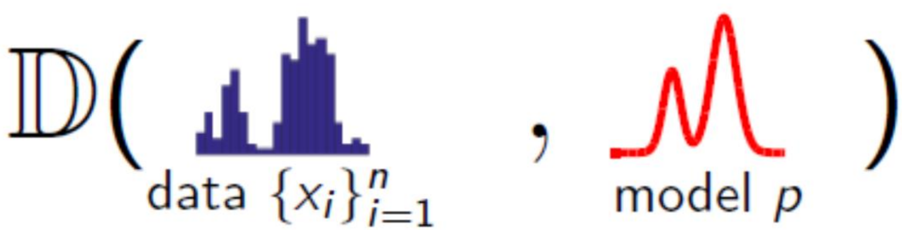
- **Given** a set *D* of unlabeled samples, learn the unknown parameters (or a distribution)

data $D=\{\mathrm{x_i}\}$



models $p(\mathbf{x}|\theta)$

- Find a model that minimizes



Learning Deep Generative Models

- Maximum likelihood estimation (MLE):

$$\hat{\theta} = \operatorname{argmax} p(D|\theta)$$

  - has an explicit likelihood model
- Minimax objective (e.g., GAN)
  - A two-player game to reach equilibrium
  - A three-player game for semi-supervised learning (NIPS'17)
- Moment-matching:
  - draw samples from p: $\widehat{D} = \{y_i\}_{i=1}^{M}$, where $y_i \sim p(x|\theta)$
  - Kernel MMD (NIPS'16):
    - rich enough to distinguish any two distributions in certain RKHS
  - PMD (NIPS'17)

Variational Bayes

- Consider the log-likelihood of ***a single example***

$$\log p(\mathrm{x};\theta) = \log \int p(\mathrm{z},\mathrm{x};\theta)dz$$

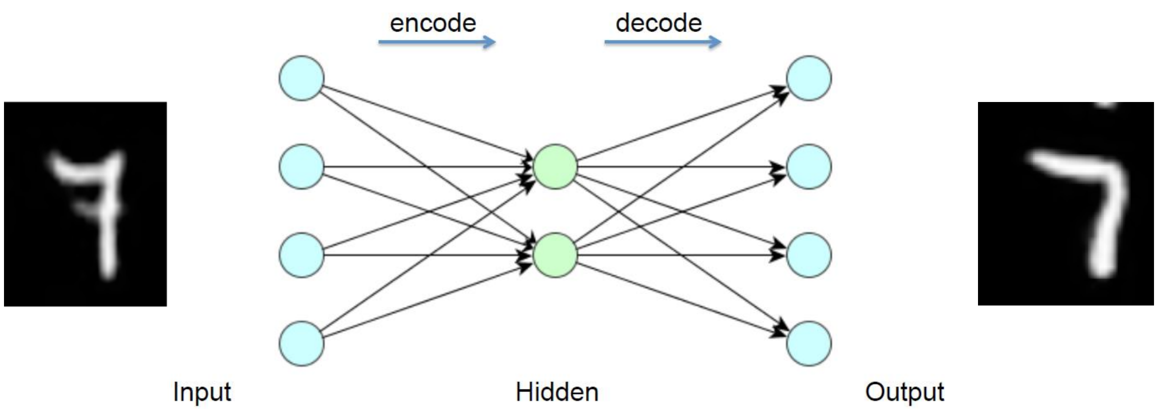
- Log-integral/sum is annoying to handle directly
- Derive a variational lower bound $L(\theta,\phi,\mathrm{x})$

$$\log p(\mathrm{x};\theta) = L(\theta,\phi,\mathrm{x}) + \mathrm{KL}(q(\mathrm{z}|\mathrm{x};\phi)||p(\mathrm{z}|\mathrm{x};\theta))$$

$$\begin{aligned} L(\theta,\phi,\mathrm{x}) &= \mathbf{E}_{q(\mathrm{z}|\mathrm{x};\phi)}[\log p(\mathrm{z},\mathrm{x};\theta) - \log q(\mathrm{z}|\mathrm{x};\phi)] \\ &= \mathbf{E}_{q(\mathrm{z}|\mathrm{x};\phi)}[\log p(\mathrm{x}|\mathrm{z};\theta) + \log p(\mathrm{z};\theta) - \log q(\mathrm{z}|\mathrm{x};\phi)] \\ &= \mathbf{E}_{q(\mathrm{z}|\mathrm{x};\phi)}[\log p(\mathrm{x}|\mathrm{z};\theta)] - \mathrm{KL}(q(\mathrm{z}|\mathrm{x};\phi)||p(\mathrm{z};\theta)) \end{aligned}$$

reconstruction term **prior regularization**

Recap: Auto-Encoder

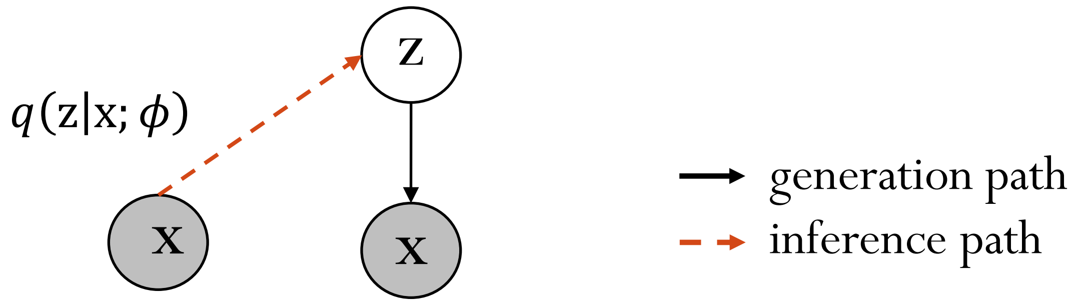


- Encoder: $\mathrm{h} = s(W\mathrm{x} + b)$
- Decoder: $\mathrm{x}' = s(W'\mathrm{h} + b')$
- Training: minimize the reconstruction error (e.g., square loss, cross-entropy loss)
- Denoising AE: randomly corrupted inputs are restored to learn more robust features

Auto-Encoding Variational Bayes (AEVB)

- What's unique in AEVB is that ***the variational distribution is parameterized by a deep neural network***

$$q(\mathrm{z}|\mathrm{x};\phi) \approx p(\mathrm{z}|\mathrm{x};\theta)$$



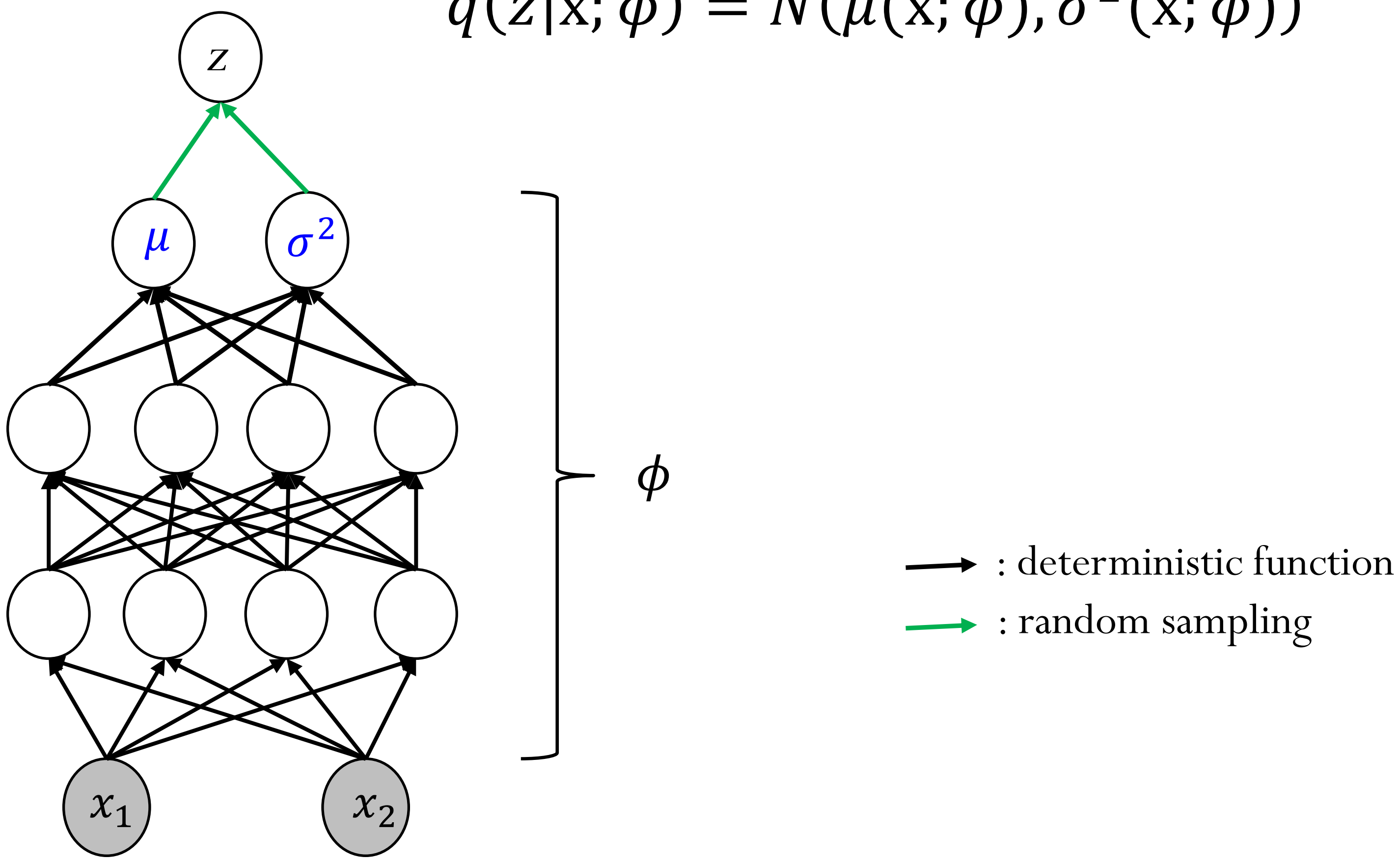
- We call it an inference (recognition, encoder) network or a Q-network
- All the parameters are learned jointly via SGD with variance reduction

[Kingma & Welling, ICLR 2013; Rezende et al., ICML 2014]

The Encoder Network

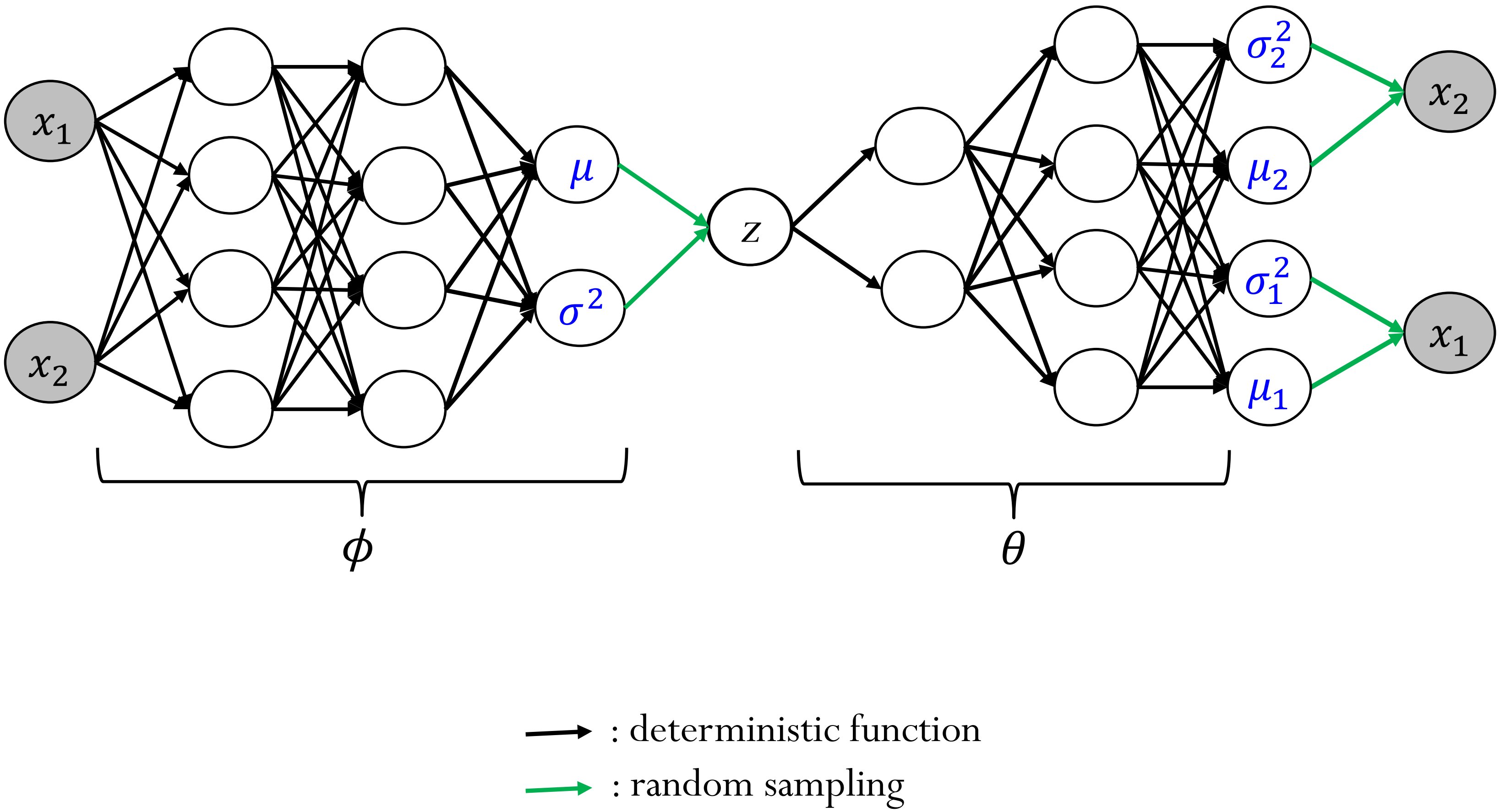
- A feedforward NN + Gaussian

$$q(z|\mathrm{x};\phi) = N(\mu(\mathrm{x};\phi), \sigma^2(\mathrm{x};\phi))$$



The Complete Auto-encoder

- The Q-P network architecture:



Stochastic Variational Inference

- Variational lower-bound for ***a single example***

$$L(\theta, \phi, \mathrm{x}) = \mathbf{E}_{q(\mathrm{z}|\mathrm{x};\phi)}[\log p(\mathrm{x}|\mathrm{z};\theta)] - \mathrm{KL}(q(\mathrm{z}|\mathrm{x};\phi)\|p(\mathrm{z};\theta))$$

- Variational lower-bound for ***a set of examples***

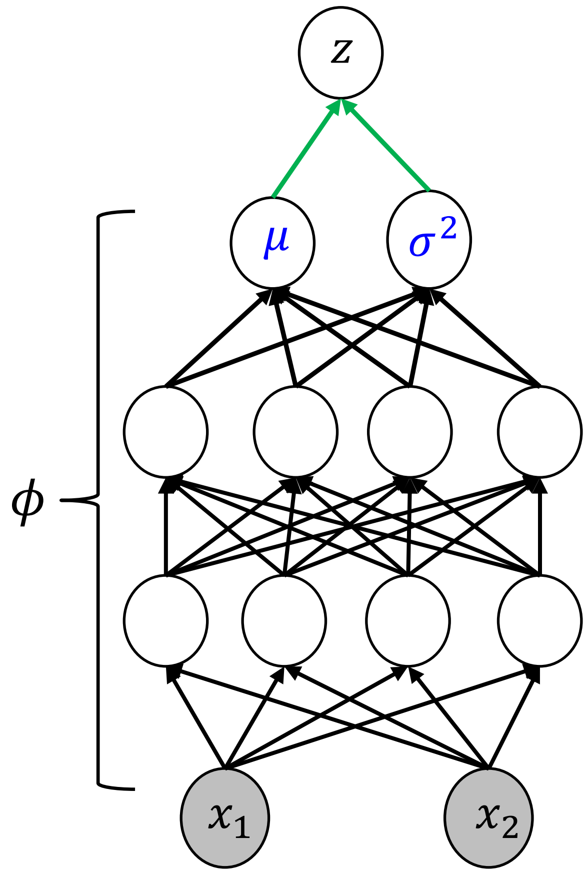
$$L(\theta, \phi, D) = \sum_i \mathbf{E}_{q(\mathrm{z}_i|\mathrm{x}_i;\phi)}[\log p(\mathrm{x}_i|\mathrm{z}_i;\theta)] - \mathrm{KL}(q(\mathrm{z}_i|\mathrm{x}_i;\phi)\|p(\mathrm{z}_i;\theta))$$

  - Use stochastic gradient methods to handle large datasets
  - Random mini-batch
    - for each i, infer the posterior $q(\mathrm{z}_i|\mathrm{x}_i;\phi)$; As we parameterize as a neural network, this in fact optimizes ϕ
- *However, calculating the expectation and its gradients is non-trivial, often intractable*

Example with Gaussian Distributions

- Use $N(0,1)$ as prior for z; $q(\mathrm{z}|\mathrm{x};\phi)$ is Gaussian with parameters $(\mu(\mathrm{x};\phi),\sigma^2(\mathrm{x};\phi))$ determined by NN
  - The KL-divergence

$$-\mathrm{KL}\big(q(z|\mathrm{x};\phi)\big\|p(z;\theta)\big)=\frac{1}{2}(1+\log\sigma^2-\mu^2-\sigma^2)$$



  - **Homework**: finish the derivation

Example with Gaussian Distributions

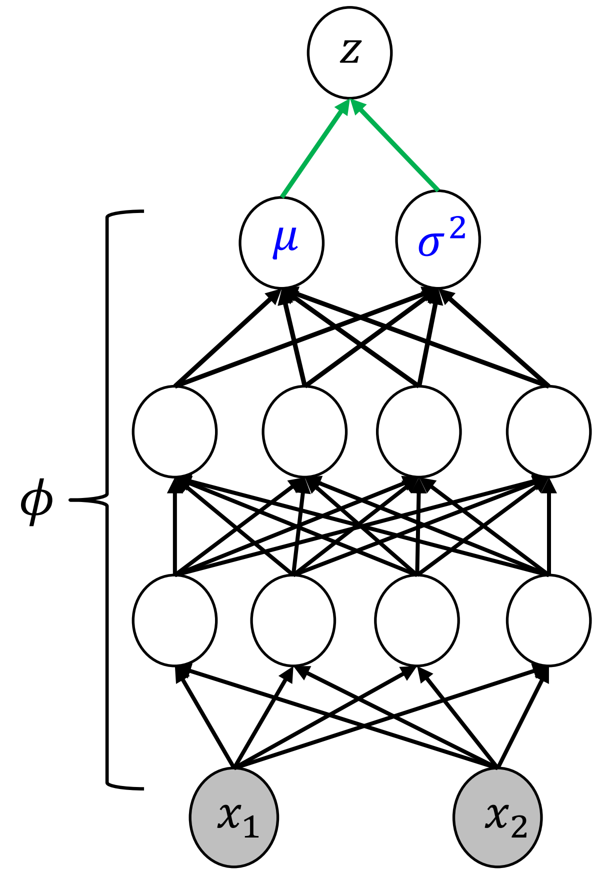
- Use $N(0,1)$ as prior for z; $q(\mathrm{z}|\mathrm{x};\phi)$ is Gaussian with parameters $(\mu(\mathrm{x};\phi), \sigma^2(\mathrm{x};\phi))$ determined by NN
  - The expected log-likelihood

$$\mathbf{E}_{q(\mathrm{z}|\mathrm{x};\phi)}[\log p(\mathrm{x}|z;\theta)]$$

  - If the likelihood is Gaussian

$$-\log p(\mathrm{x}_i|z_i) = \sum_j \frac{1}{2}\log \sigma_j^2 + \frac{(x_{ij} - \mu_{xi})^2}{2\sigma_j^2}$$

  - *The expectation is still hard to compute because of nonlinearity functions*



Example with Gaussian Distributions

- Use $N(0,1)$ as prior for z; $q(\mathrm{z}|\mathrm{x};\phi)$ is Gaussian with parameters $(\mu(\mathrm{x};\phi), \sigma^2(\mathrm{x};\phi))$ determined by NN
  - The expected log-likelihood

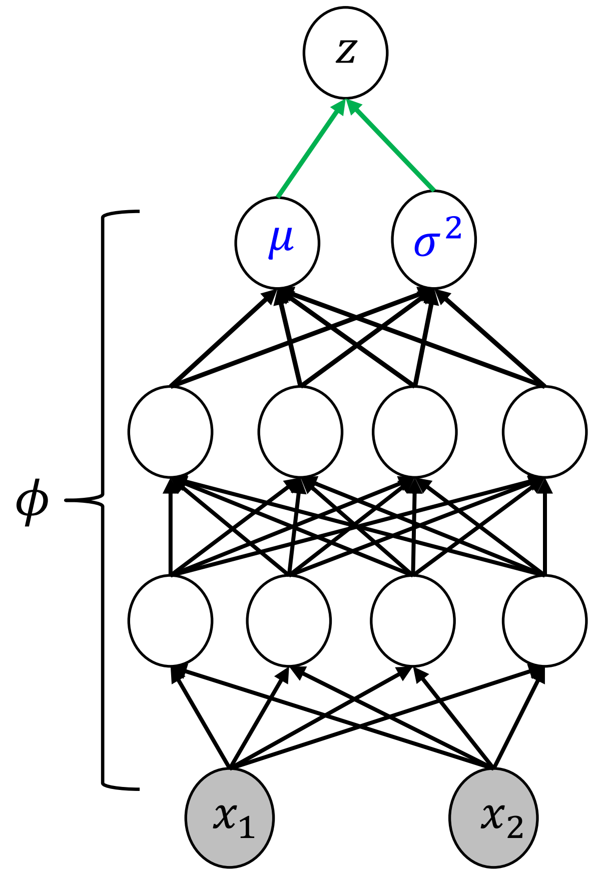
$$\mathbf{E}_{q(\mathrm{z}|\mathrm{x};\phi)}[\log p(\mathrm{x}|z;\theta)]$$

  - Approximate via Monte Carlo methods

$$\mathbf{E}_{q(\mathrm{z}|\mathrm{x};\phi)}[\log p(\mathrm{x}|\mathrm{z};\theta)] \approx \frac{1}{L}\sum_{k} \log p\left(\mathrm{x}|\mathrm{z}^{(k)}\right)$$

$$z^{(k)} \sim q(z|\mathrm{x};\phi)$$

  - An unbiased estimator



Example with Gaussian Distributions

- The KL-regularization term (closed-form):

$$-\mathrm{KL}(q(z|\mathrm{x};\phi)\|p(z;\theta)) = \frac{1}{2}(1+\log\sigma^2-\mu^2-\sigma^2)$$

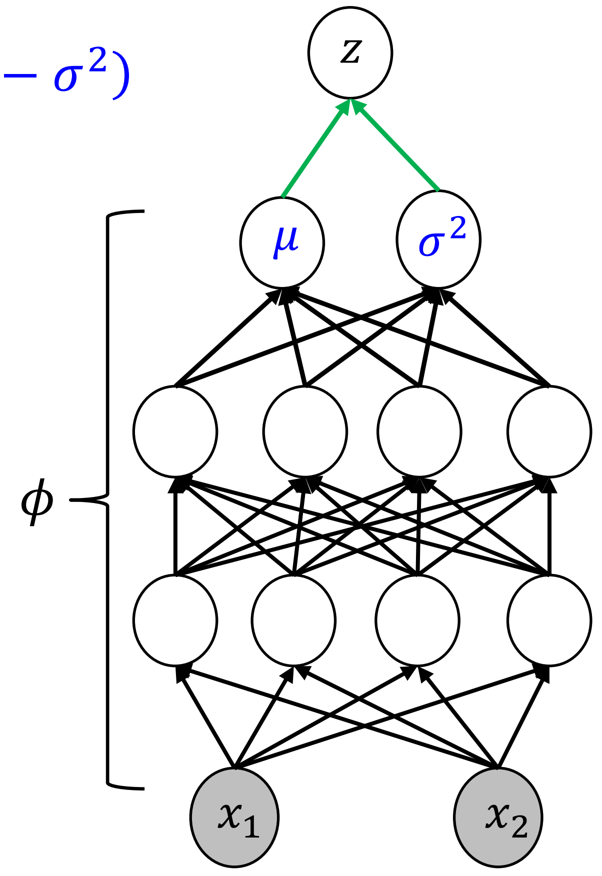
  - Easy to calculate gradient

- The expected log-likelihood term (MC estimate)

$$\mathbf{E}_{q(\mathrm{z}|\mathrm{x};\phi)}[\log p(\mathrm{x}|\mathrm{z};\theta)] \approx \frac{1}{L}\sum_k \log p\left(x\middle|\mathrm{z}^{(k)}\right)$$

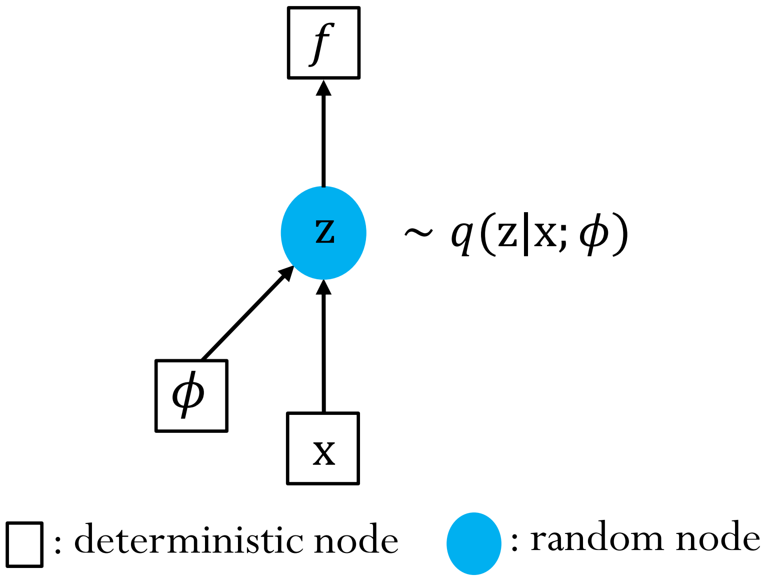
$$z^{(k)} \sim q(z|\mathrm{x};\phi)$$

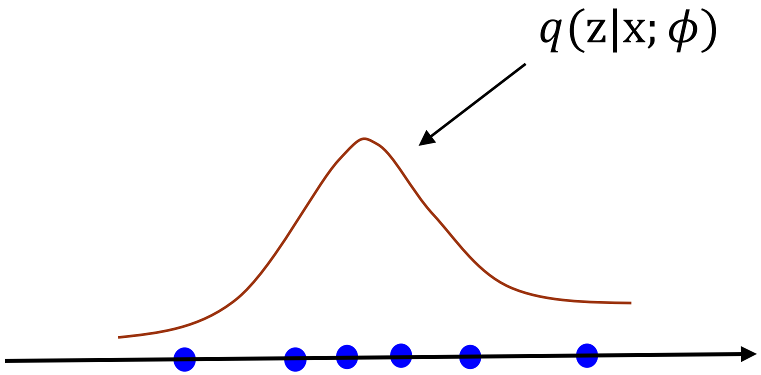
  - Gradient needs back-propagation!
  - *However, $z^{(k)}$ is a random variable, we can't take gradient over a randomly drawn number*



Reparameterization Trick

- Backpropagation not possible through random sampling





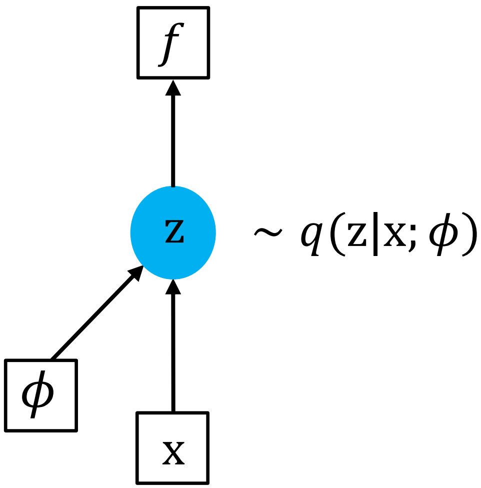
$$\{-1.5, -0.5, 0.3, 0.6, 1.5, \dots\}$$

$$z^{(k)} \sim N\left(\mu(\text{x}, \phi), \sigma^2(\text{x}, \phi)\right)$$

Cannot back-propagate through a randomly drawn number

Reparameterization Trick

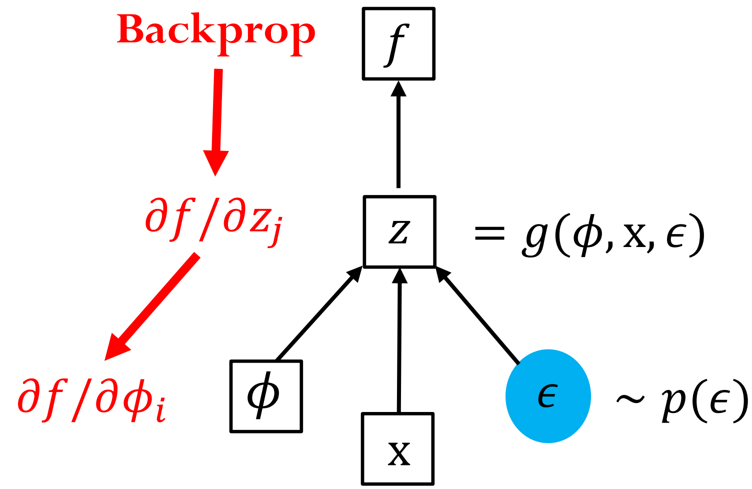
- Backpropagation not possible through random sampling



□ : deterministic node ● : random node

$$z^{(k)} \sim N\left(\mu(\mathrm{x},\phi), \sigma^2(\mathrm{x},\phi)\right)$$

Cannot back-propagate through a randomly drawn number



$$\epsilon^{(k)} \sim N(0,1)$$

$$z^{(k)} = \mu(\mathrm{x},\phi) + \sigma(\mathrm{x},\phi) \cdot \epsilon^{(k)}$$

Z has the same distribution, but now can back-prop
Separate into a deterministic part and noise

The General Form

- The VAE bound

$$L(\theta, \phi, \mathrm{x}) = \mathbf{E}_{q(\mathrm{z}|\mathrm{x};\phi)}[\log p(\mathrm{z}, \mathrm{x}; \theta) - \log q(\mathrm{z}|\mathrm{x}; \phi)]$$

$$= \mathbf{E}_{q(\mathrm{z}|\mathrm{x};\phi)} \left[\log \frac{p(\mathrm{z}, \mathrm{x}; \theta)}{q(\mathrm{z}|\mathrm{x}; \phi)}\right]$$

- Monte Carlo estimate:

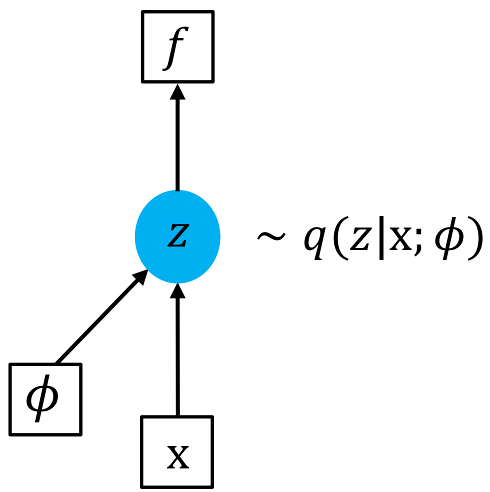
$$L(\theta, \phi, \mathrm{x}) \approx \frac{1}{L} \sum_{k} \log \frac{p\left(\mathrm{z}^{(k)}, \mathrm{x}; \theta\right)}{q\left(\mathrm{z}^{(k)} \middle| \mathrm{x}; \phi\right)}$$

$$z^{(k)} \sim q(z|\mathrm{x}; \phi)$$

  - Again, we cannot back-prop through the randomly drawn numbers

Reparameterization Trick

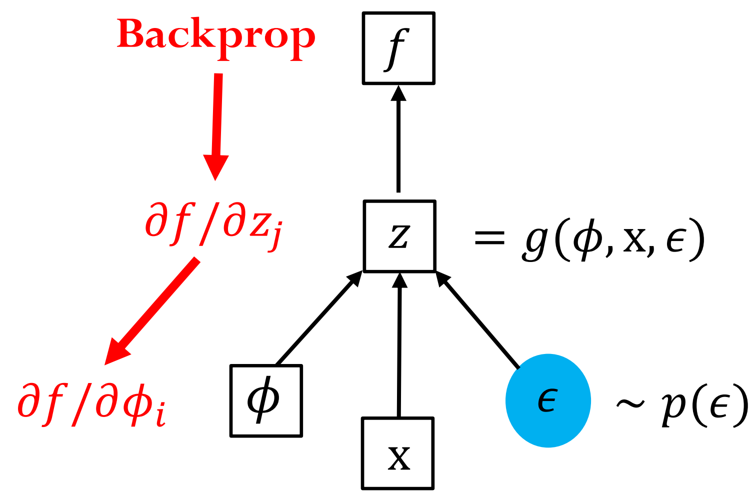
- Backpropagation not possible through random sampling



: deterministic node   : random node

$$z^{(k)} \sim q(z|\mathrm{x};\phi)$$

Cannot back-propagate through a randomly drawn number



$$\epsilon^{(k)} \sim p(\epsilon)$$
$$z^{(k)} = g\left(\phi, x, \epsilon^{(k)}\right)$$

Z has the same distribution, but now can back-prop
Separate into a deterministic part and noise

Reparam-Trick Summary

- The VAE bound

$$L(\theta, \phi, \mathrm{x}) = \mathbf{E}_{q(\mathrm{z}|\mathrm{x};\phi)}\left[\log \frac{p(\mathrm{z}, \mathrm{x}; \theta)}{q(\mathrm{z}|\mathrm{x}; \phi)}\right]$$

  - Reparameterized as

$$L(\theta, \phi, \mathrm{x}) = \mathbf{E}_{p(\epsilon)}\left[\log \frac{p(g(\mathrm{x}, \epsilon, \phi), \mathrm{x}; \theta)}{q(g(\mathrm{x}, \epsilon, \phi)|\mathrm{x}; \phi)}\right]$$

    - where ϵ is a simple distribution (e.g., standard normal) and g is a deep NN

- The gradients are

$$\nabla_\theta L(\theta, \phi, \mathrm{x}) = \mathbf{E}_{p(\epsilon)}\left[\nabla_\theta \log \frac{p(g(\mathrm{x}, \epsilon, \phi), \mathrm{x}; \theta)}{q(g(\mathrm{x}, \epsilon, \phi)|\mathrm{x}; \phi)}\right]$$

  - Back-prop is applied over the deep NN
  - Similar for ϕ

Importance Weighted Auto-Encoder (IWAE)

- The VAE lower bound of log-likelihood

$$L(\theta, \phi, \mathrm{x}) = \mathbf{E}_{q(\mathrm{z}|\mathrm{x};\phi)}\left[\log \frac{p(\mathrm{z}, \mathrm{x}; \theta)}{q(\mathrm{z}|\mathrm{x}; \phi)}\right]$$

- A better variational lower bound (IWAE)

$$L_K(\theta, \phi, \mathrm{x}) = \mathbf{E}_{q(\mathrm{z}|\mathrm{x};\phi)}\left[\log\left(\frac{1}{K}\sum_{k=1:K} \frac{p\left(\mathrm{z}^{(k)}, \mathrm{x}; \theta\right)}{q\left(\mathrm{z}^{(k)}\middle|\mathrm{x}; \phi\right)}\right)\right]$$

where $z^{(k)} \sim q(z|\mathrm{x}; \phi)$

  - This is a lower-bound of the log-likelihood
  - When K=1, recovers the VAE bound
  - When $K = \infty$, recovers the log-likelihood
  - A monotonic sequence:

$$L_K(\theta, \phi, \mathrm{x}) \leq L_{K+1}(\theta, \phi, \mathrm{x}), \qquad \forall \theta, \phi, \mathrm{x}$$

[Burda et al., arXiv, 2015]

Reparametrization Trick

- The IWAE bound:

$$L_K(\theta, \phi, \mathrm{x}) = \mathbf{E}_{q(\mathrm{z}|\mathrm{x};\phi)}\left[\log\left(\frac{1}{K}\sum_{k=1:K} w\left(\mathrm{z}^{(k)}, \mathrm{x}; \theta\right)\right)\right]$$

where $z^{(k)} \sim q(z|\mathrm{x};\phi)$ $\quad w\left(\mathrm{z}^{(k)}, \mathrm{x}; \theta, \phi\right) = \frac{p\left(\mathrm{z}^{(k)}, \mathrm{x}; \theta\right)}{q\left(\mathrm{z}^{(k)}\middle|\mathrm{x}; \phi\right)}$

- Reparameterization form:

$$L_K(\theta, \phi, \mathrm{x}) = \mathbf{E}_{\boldsymbol{p}(\boldsymbol{\epsilon})}\left[\log\left(\frac{1}{K}\sum_{k=1:K} w\left(g(\epsilon^{(k)}, \mathrm{x}, \phi), \mathrm{x}; \theta\right)\right)\right]$$

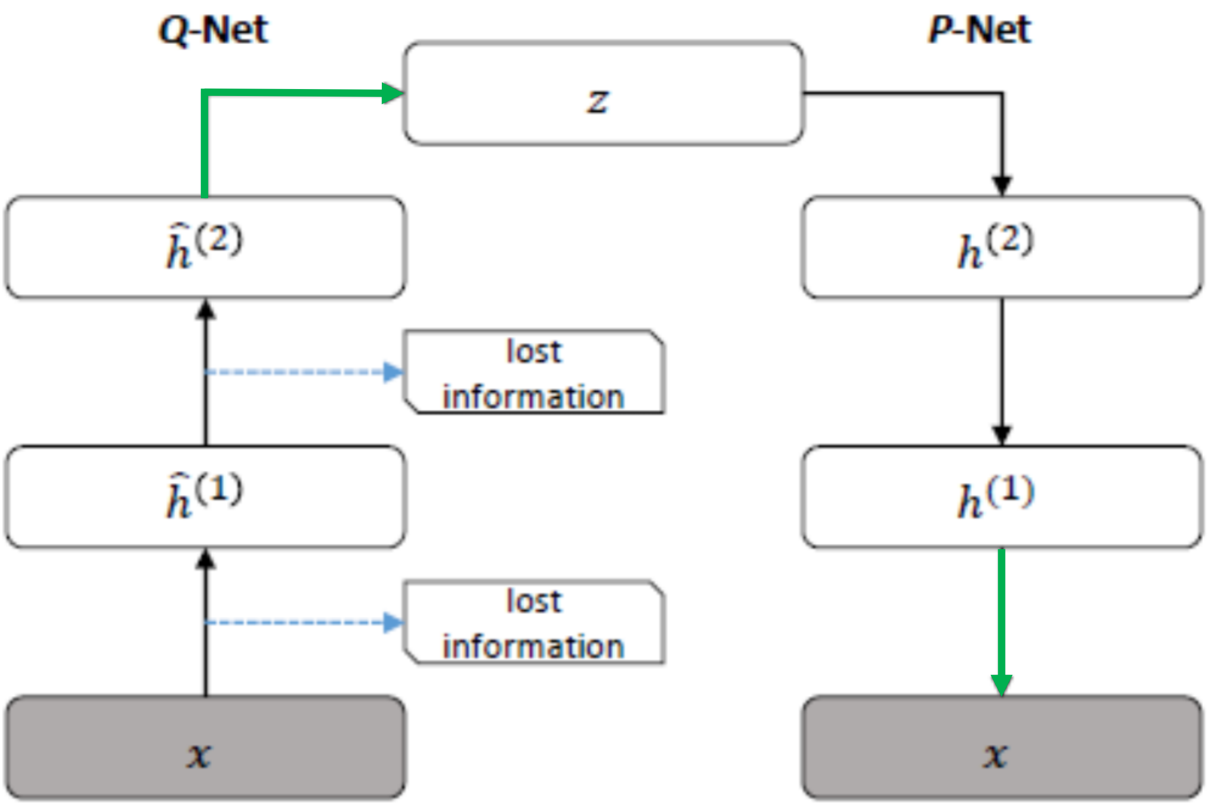
where $\epsilon^{(k)} \sim p(\epsilon)$

  - The gradient can be calculated as in VAE

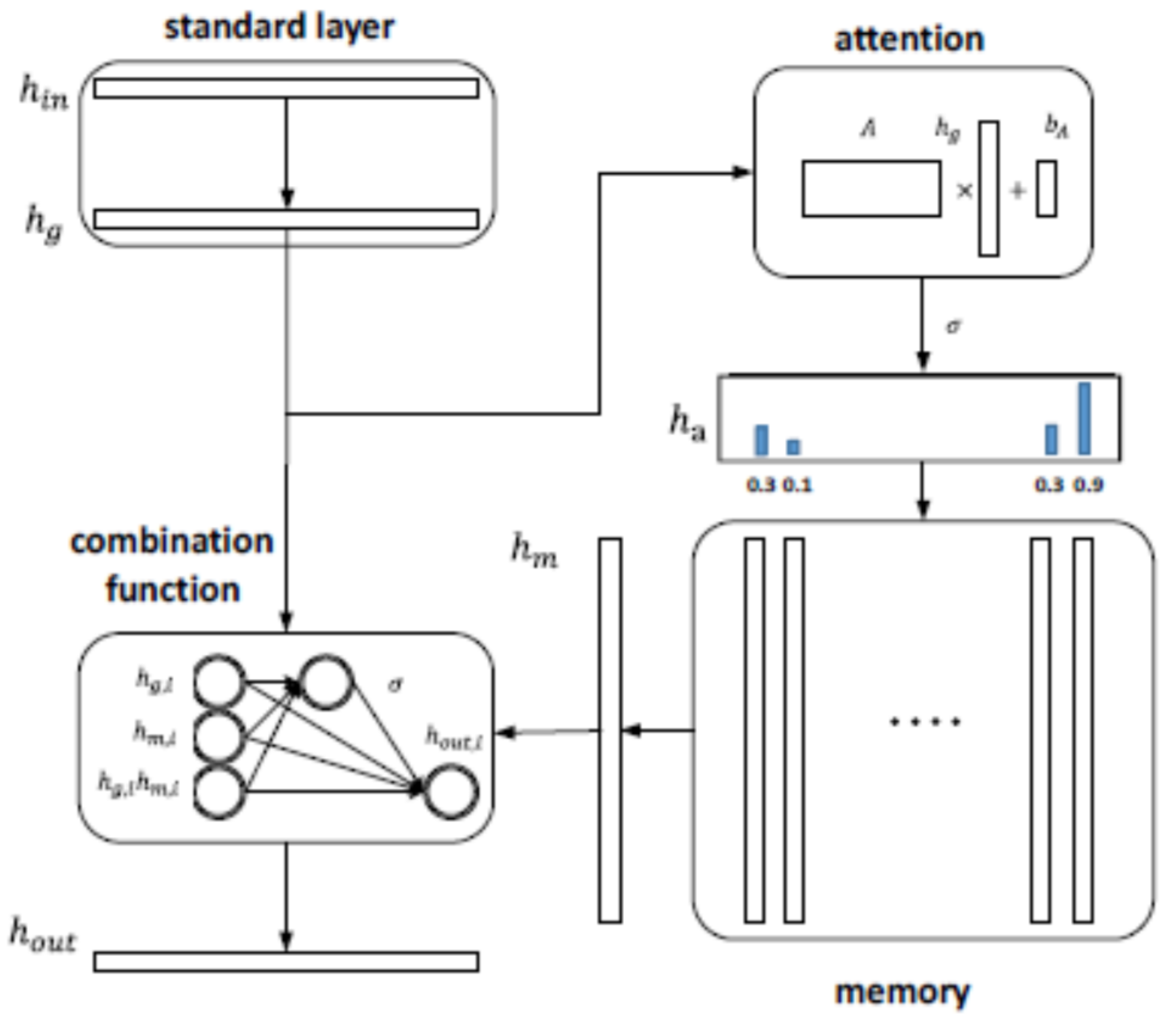
More Applications

Symmetric Q-P Network

- **Problem**: detail information is lost during abstraction



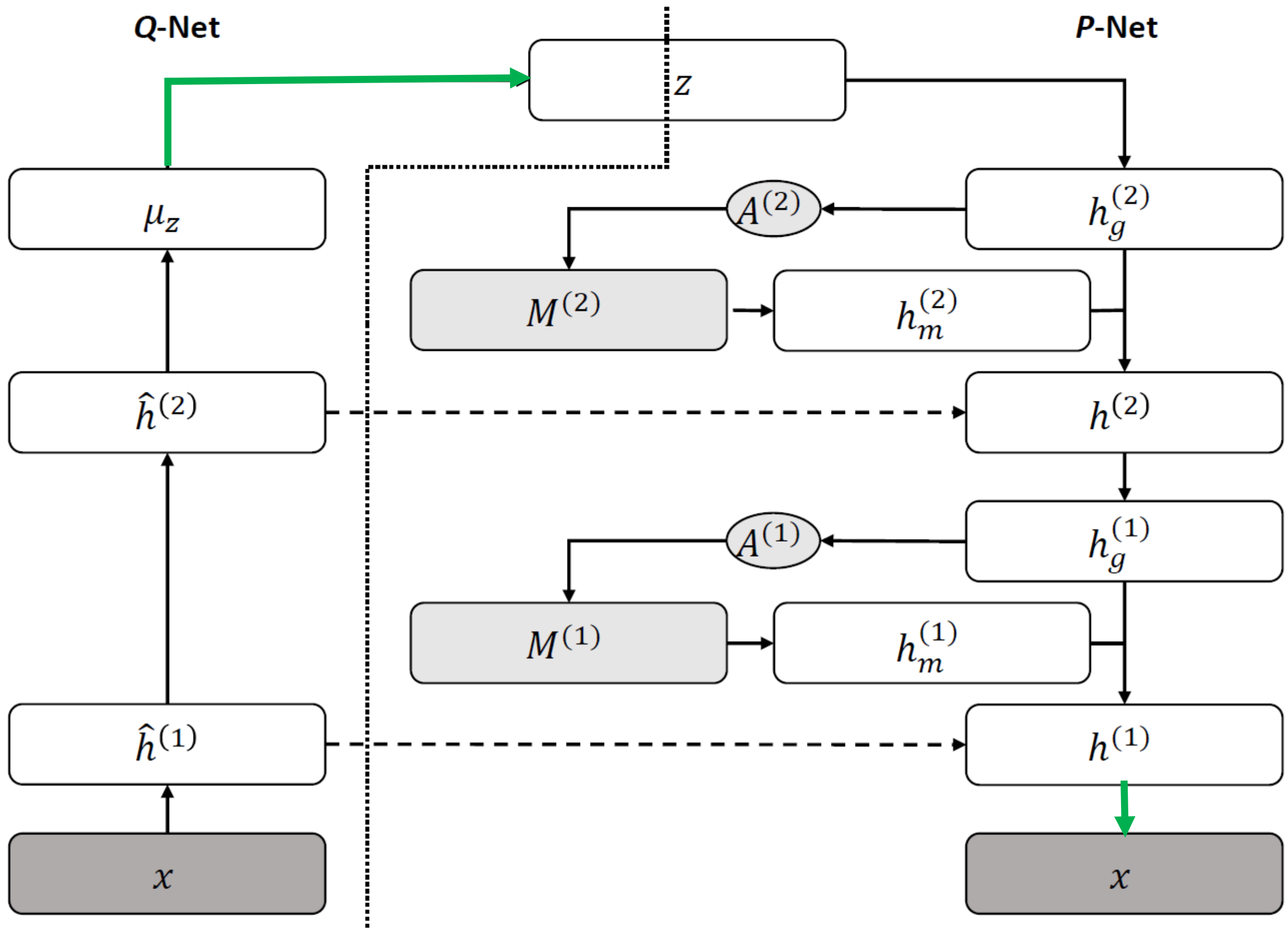
A Layer with Memory and Attention



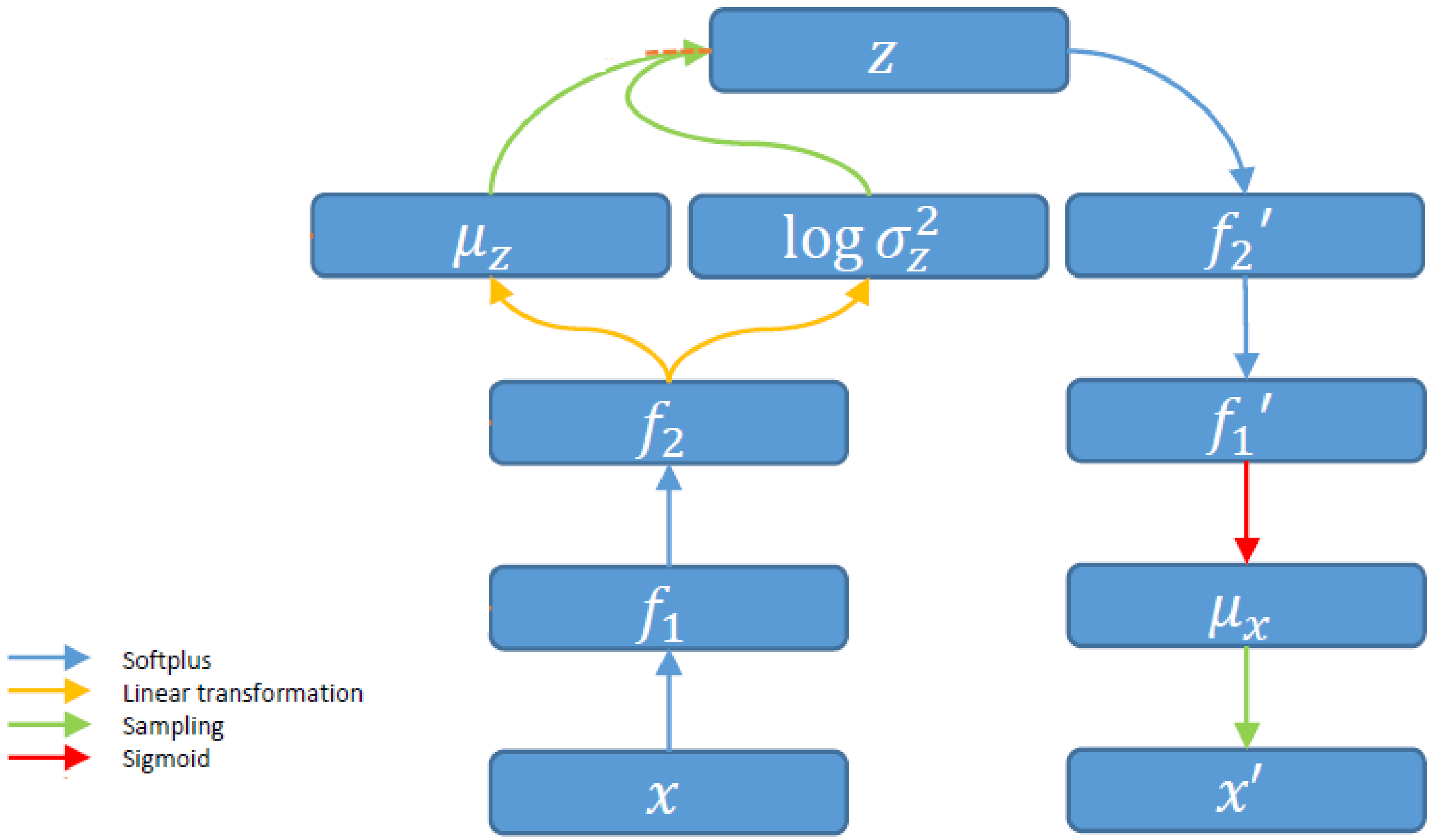
[Li, et al., ICML 2016]

A Stacked Deep Model with Memory

- Asymmetric architecture

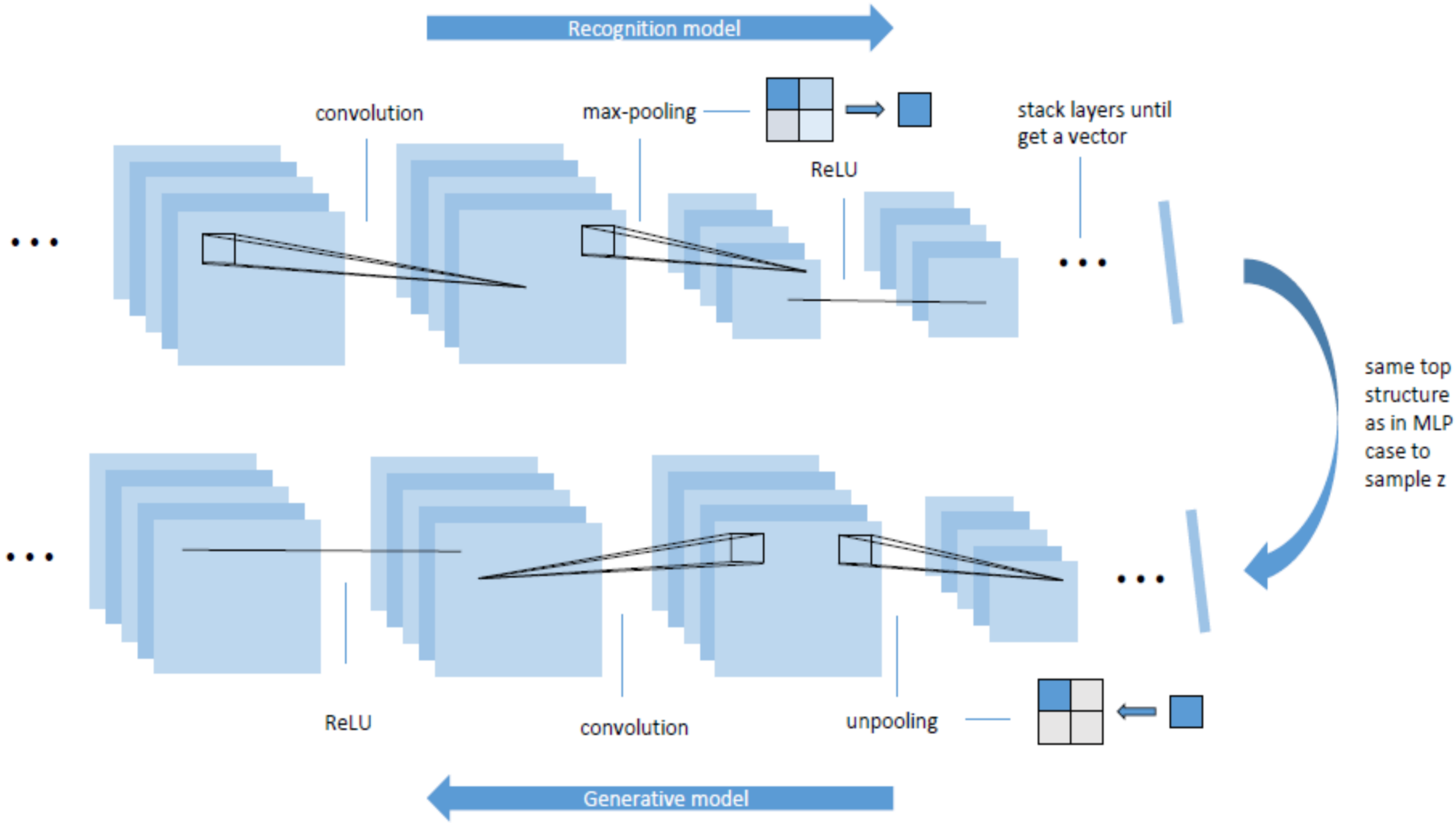


2-Layer MLP: Q-P network architecture



*Same as in Auto-Encoding Variational Bayes (VA) [Kingma & Welling, 2014]

5-Layer CNN: Q-P network architecture



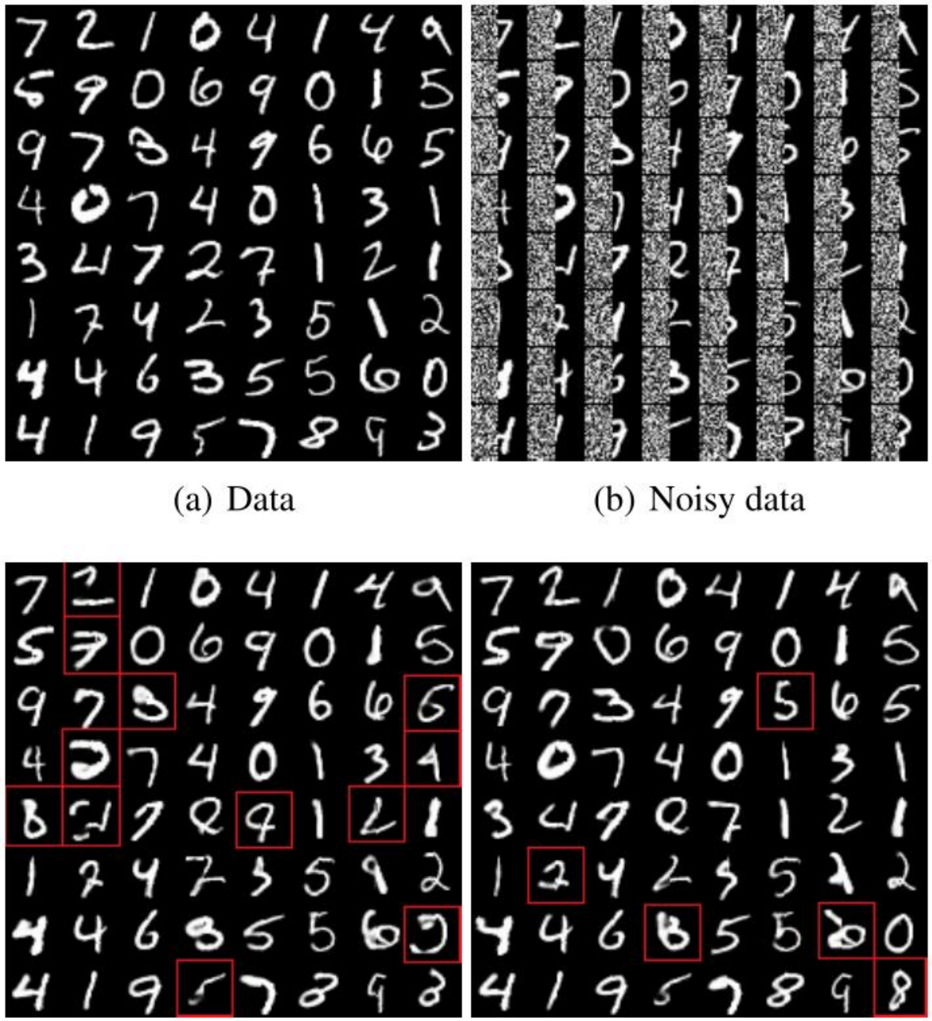
Some Results

- Density estimation

Models	MNIST	OCR-letters
VAE	-85.69	-30.09
MEM-VAE(ours)	**-84.41**	**-29.09**
IWAE-5	-84.43	-28.69
MEM-IWAE-5(ours)	**-83.26**	**-27.65**
IWAE-50	-83.58	-27.60
MEM-IWAE-50(ours)	**-82.84**	**-26.90**

  - Better than symmetric VAE networks
  - Comparable with state-of-the-art with much fewer parameters

Missing Value Imputation



(a) Data (b) Noisy data

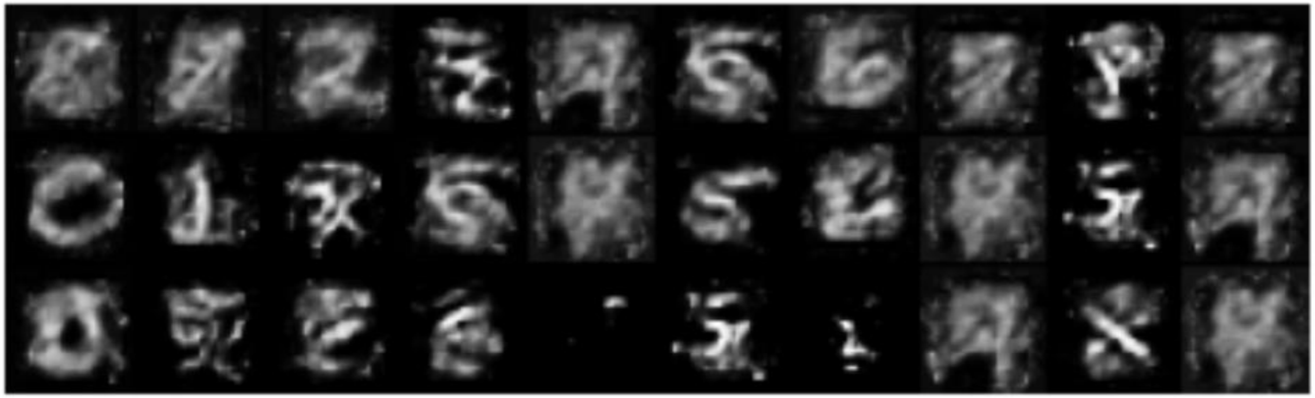
(c) Results of VAE (d) Results of MEM-VAE

Learnt Memory Slots

- Average preference over classes of the first 3 slots:

"0"	"1"	"2"	"3"	"4"	"5"	"6"	"7"	"8"	"9"
0.27	0.82	0.33	0.11	0.34	0.15	0.49	0.27	0.09	0.28
0.24	0.09	0.06	0.11	0.30	0.13	0.12	0.27	0.09	0.21
0.18	0.05	0.06	0.11	0.07	0.07	0.05	0.11	0.09	0.18

- Corresponding images:



References

- Kingma, D. P. and Welling, M. *Auto-encoding variational Bayes*. In ICLR, 2013.
- Rezende, D. J., Mohamed, S., and Wierstra, D. *Stochastic backpropagation and approximate inference in deep generative models*. In ICML, 2014.
- Burda, Y., Grosse, R., and Salakhutdinov, R. *Importance weighted autoencoders*. In arXiv:1509.00519, 2015.
- Goodfellow, I. J., Abadie, J. P., Mirza, M., Xu, B., Farley, D. W., S.ozair, Courville, A., and Bengio, Y. *Generative adversarial nets*. In NIPS, 2014
- Bengio, Y., Thlhodeau-Laufer, E., Alain, G., and Yosinski, J. *Deep generative stochastic networks trainable by backprop*. In ICML, 2014.
- Li, C., Zhu, J., and Zhang, B. *Learning to generate with memory*. In ICML, 2016

Thanks!

ZhuSuan: A Library for Bayesian Deep Learning. J. Shi, J. Chen, J. Zhu, S. Sun, Y. Luo, Y. Gu, Y. Zhou. arXiv preprint, arXiv:1709.05870 , 2017

Online Documents: http://zhusuan.readthedocs.io/