

Day 5

Deep Generative Models

Jun Zhu

`dcszj@mail.tsinghua.edu.cn`

Department of Computer Science and Technology

Tsinghua University

Tsinghua Deep Learning Summer School, Beijing, 2018

A bit about the Instructor

- ◆ Jun Zhu, Professor, Depart. of Computer Science & Technology. I received my Ph.D. in DCST of Tsinghua University in 2009. My research interests include statistical machine learning, Bayesian nonparametrics, and data mining
- ◆ I did post-doc at the Machine Learning Department in CMU with Prof. Eric P. Xing. Before that I was invited to visit CMU for twice. I was also invited to visit Stanford for joint research (with Prof. Li Fei-Fei)
- ◆ 2015: Adjunct Professor at CMU
- ◆ Published 100+ research papers on the top-tier ML conferences and journals, including JMLR, TPAMI, ICML, NIPS, etc.
- ◆ Served as Area Chairs for ICML, NIPS, UAI, AAAI, IJCAI; Associate Editor for PAMI, AI Journal
- ◆ Research is supported by National 973, NSFC, “Tsinghua 221 Basic Research Plan for Young Talents”.
- ◆ Homepage: <http://bigml.cs.tsinghua.edu.cn/~jun>



Outline

- ◆ Review of Probability and Statistics
 - MLE
- ◆ Generative Models
- ◆ Deep Generative Models
 - VAE
 - GAN
- ◆ ZhuSuan
 - Programming library
- ◆ Applications

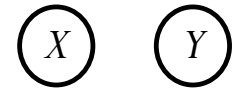
Basics of Probabilities and MLE

Independence

◆ Independent random variables:

$$P(X, Y) = P(X)P(Y)$$

$$P(X|Y) = P(X)$$



- Y and X don't contain information about each other

Observing Y doesn't help predicting X

Observing X doesn't help predicting Y

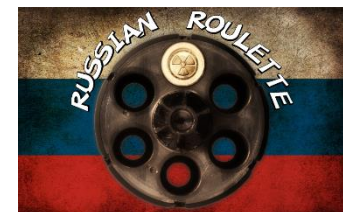
◆ Examples:

- Independent:

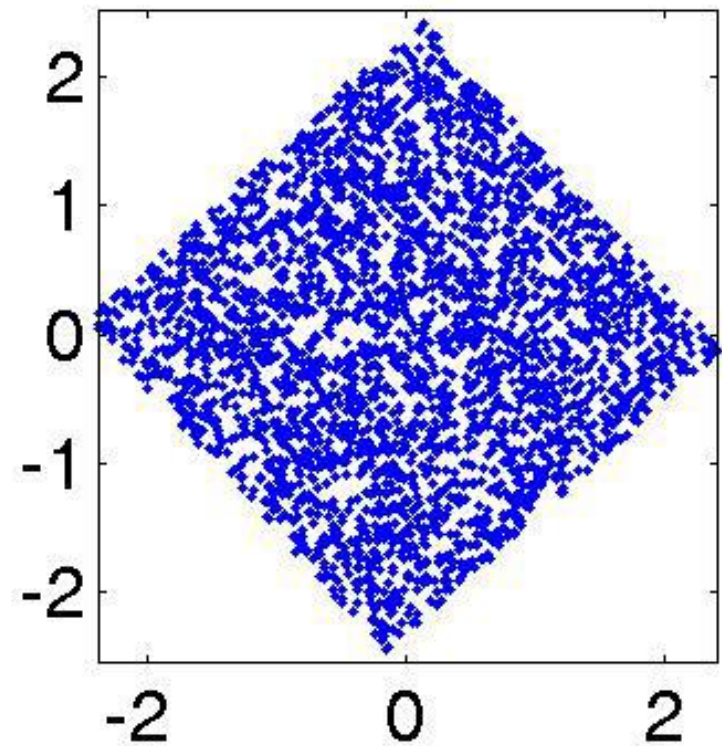
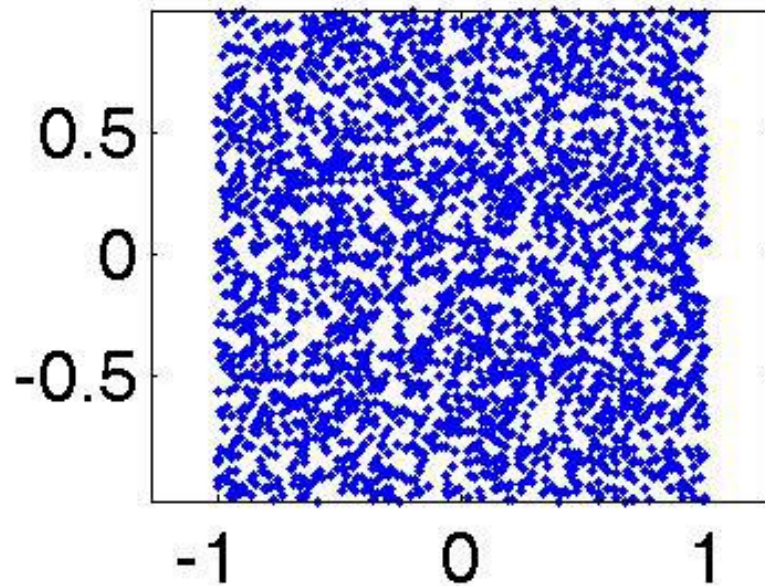
- winning on roulette this week and next week

- Dependent:

- Russian roulette



Dependent / Independent?

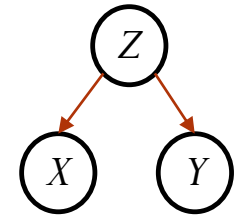


Conditional Independence

◆ Conditionally independent:

$$P(X, Y|Z) = P(X|Z)P(Y|Z)$$

- knowing Z makes X and Y independent



◆ Examples:

London taxi drivers: A survey has pointed out a positive and significant correlation between the number of accidents and wearing coats. They concluded that coats could hinder movements of drivers and be the cause of accidents. A new law was prepared to prohibit drivers from wearing coats when driving.



Finally another study pointed out that people wear coats when it rains...



Maximum Likelihood Estimation (MLE)

Flipping a Coin

- ◆ What's the probability that a coin will fall with a head up (if flipped)?
- ◆ Let us flip it a few times to estimate the probability



The estimated probability is: $3/5$ “frequency of heads”

Questions:



The estimated probability is: $3/5$ “frequency of heads”

- ◆ Why frequency of heads?
- ◆ How good is this estimation?

Question (1)

◆ Why frequency of heads?

- Frequency of heads is exactly the Maximum Likelihood Estimator for this problem
- MLE has nice properties
(interpretation, statistical guarantees, simple)

MLE for Bernoulli Distribution

Data, $D =$



$$D = \{X_i\}_{i=1}^n, X_i \in \{H, T\}$$

$$P(\text{Head}) = \theta \quad P(\text{Tail}) = 1 - \theta$$

- ◆ Flips are i.i.d:
 - ▣ **Independent** events that are **identically distributed** according to Bernoulli distribution
- ◆ **MLE**: choose θ that maximizes the probability of observed data

Maximum Likelihood Estimation (MLE)

- ◆ MLE: choose θ that maximizes the probability of observed data

$$\hat{\theta}_{MLE} = \arg \max_{\theta} P(D|\theta)$$

$$= \arg \max_{\theta} \prod_{i=1}^n P(X_i|\theta) \quad \text{Independent draws}$$

$$= \arg \max_{\theta} \prod_{i:X_i=H} \theta \prod_{i:X_i=T} (1 - \theta) \quad \text{Identically distributed}$$

$$= \arg \max_{\theta} \theta^{N_H} (1 - \theta)^{N_T}$$

Maximum Likelihood Estimation (MLE)

- ◆ MLE: choose θ that maximizes the probability of observed data

$$\begin{aligned}\hat{\theta}_{MLE} &= \arg \max_{\theta} P(D|\theta) \\ &= \arg \max_{\theta} \theta^{N_H} (1 - \theta)^{N_T}\end{aligned}$$

- ◆ Solution?

$$\hat{\theta}_{MLE} = \frac{N_H}{N_H + N_T}$$

- Exactly the “**Frequency of heads**”

Question (2)

◆ How good is the MLE estimation?

$$\hat{\theta}_{MLE} = \frac{N_H}{N_H + N_T}$$

□ Is it biased?

How many flips do I need?

- ◆ I flipped the coins 5 times: 3 heads, 2 tails

$$\hat{\theta}_{MLE} = \frac{3}{5}$$

- ◆ What if I flipped 30 heads and 20 tails?

$$\hat{\theta}_{MLE} = \frac{30}{50}$$

- ◆ Which estimator should we trust more?

A Simple Bound

◆ Let θ^* be the true parameter. For n data points, and

$$\hat{\theta}_{MLE} = \frac{N_H}{N_H + N_T}$$

◆ Then, for any $\epsilon > 0$, we have the Hoeffding's Inequality:

$$P(|\hat{\theta} - \theta^*| \geq \epsilon) \leq 2e^{-2n\epsilon^2}$$

Probably Approximately Correct (PAC) Learning

- ◆ I want to know the coin parameter θ , within $\epsilon=0.1$ error with probability at least $1-\delta$ (e.g., 0.95)
- ◆ How many flips do I need?

$$P(|\hat{\theta} - \theta^*| \geq \epsilon) \leq 2e^{-2n\epsilon^2} \leq \delta$$

- ◆ Sample complexity:

$$n \geq \frac{\ln(2/\delta)}{2\epsilon^2}$$

Deep Generative Models (DGMs)

Why generative models?

“What I cannot create, I do not understand.”

—Richard Feynman

Why generative models?

- ◆ Leverage unlabeled datasets, which are often much larger than labeled ones
 - Unsupervised learning
 - Semi-supervised learning
- ◆ Conditional generative models
 - Speech synthesis: Text \Rightarrow Speech
 - Machine Translation: French \Rightarrow English
 - Image captioning: Image \Rightarrow Text

Generative models are everywhere ...

◆ A simple unigram language model

- Observations (e.g., bag-of-words)

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

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
...		

Generative models are everywhere ...

◆ A simple unigram language model

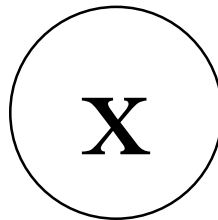
- ▣ Observations (e.g., bag-of-words)

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

- ▣ Joint distribution (likelihood)

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

- ▣ Graphical representation (parameters ignored)



Generative models are everywhere ...

- ◆ Learn a simple generative model

- Given a set of observations

$$X = \{x_1, \dots, x_N\}$$

- Maximize the log-likelihood

$$\max_{\theta} \log p(X; \theta) = \sum \log p(x_i; \theta)$$

- Simple closed-form solutions:

- count frequency for discrete or empirical mean/variance for Gaussian distribution

Generative models are everywhere ...

◆ A fully-observed model is not sufficient

□ Data has hidden structures

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, danieleks, salles}@dcc.ufmg.br

ABSTRACT

The task of retrieving information that really matters to the users is considered hard when taking into consideration the relevant and non-relevant amount of available information. To improve the effectiveness of the information retrieval task, systems have tried to use the combination of many procedures to answer the user's information needs. A task often known as learning to rank (LTR). The most effective learning methods for this task are based on ensemble of trees (e.g., Random Forests) and/or boosting techniques (e.g., Rank-Sieve, MARS, LambdaMART). In this paper, we propose a general framework that combines ensemble of additive trees, specifically Random Forests, with Boosting in a regular way to the task of LTR. In particular, we exploit out-of-bag sampling as well as a selective weight updating strategy (inspired by the out-of-bag sampling) to effectively reduce the training performance. We illustrate that a general framework by considering different loss functions, different ways of splitting the data into training and testing sets, and different ways of combining the results of the ensemble of trees can be used to improve the performance of the LTR task.

Document Retrieval Using Entity-Based Language Models

Keywords
Learning to Rank

Hadas Raviv
Technion, Israel
hadasr@tx.technion.ac.il

Oren Kurland
Technion, Israel
kurland@ie.technion.ac.il

David Carmel
Yahoo Research, Israel
david.carmel@gmail.com

ABSTRACT

We address the ad hoc document retrieval task by devising novel types of entity-based language models. The models utilize information about single terms in the query and documents as well as terms appearing nearby to those (marked) in the text and which often use auxiliary information about entities from the entity repository (e.g., textual description of entities, entities' categories and inter-entity relationships). We show that the models significantly outperform the state-of-the-art in the entity-based retrieval task. We also show that the language models can be effectively used for cluster-based document retrieval and query expansion.

Categories and Subject Descriptors: H.3.3 Information Search and Retrieval; Retrieval models

Keywords: document retrieval; entity-based language models

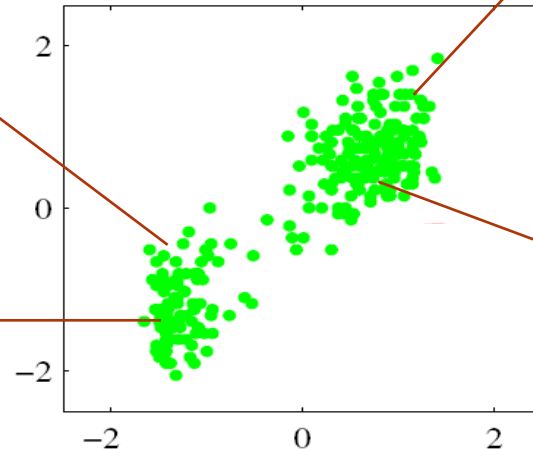
1. INTRODUCTION

Most ad hoc document retrieval methods compare query and document representations. To address the potential vocabulary mismatch between a short query and documents relevant to the query, various semantic document-query similarity measures have been proposed [26]. Specifically, there is a growing body of work on retrieval

in the text and wordings of entities in it, along with raw corpus-based occurrence statistics. This is in contrast to expansion-based and projection-based representations that utilize also terms and entities related to those (marked) in the text and which often use auxiliary information about entities from the entity repository (e.g., textual description of entities, entities' categories and inter-entity relationships). We show that the models significantly outperform the state-of-the-art in the entity-based retrieval task. We also show that the language models can be effectively used for cluster-based document retrieval and query expansion.

The reason for addressing the question just posed is twofold. First, it will shed light on the effectiveness of using entities in their most basic capacity, that is, spatial labels marked by queries and documents. Indeed, findings in past work on ad hoc retrieval regarding the merits of using surface level entity-based representations are inconclusive [16, 42, 47, 3, 14]. Second, such representations can be naturally used in creating retrieval expansion and leads to improve performance, e.g., query expansion and cluster-based document retrieval as we show in this paper.

There are various potential merits in using surface level entity-based representations. For example, these can help to cope with the vocabulary mismatch problem, e.g., the entity (United States of America) can have different expressions in the text, including, "U.S.", "USA", "United States" and



Stochastically Transitive Models for Pairwise Comparisons: Statistical and Computational Issues

Nihar B. Shah¹
Sriram Hariharan²
Adityanand Ganeshaiah²
Martin J. Heule²
¹Dept. of EECS, Univ. of California, Berkeley
²Dept. of Statistics, Carnegie Mellon University

NIBH@EECS.BERKELEY.EDU
SHIRAM@STAT.CMU.EDU
ADITYA@STAT.BERKELEY.EDU
WAINWIG@BERKELEY.EDU

Abstract

There are various parametric models for analyzing pairwise comparison data, including the Bradley-Terry-Luce (BTL) and Thurstone models, but their reliance on strong parametric assumptions is limiting. In this work, we study a flexible model for pairwise comparisons, under which the probabilities of outcomes are required only to satisfy a natural form of stochastic transitivity. This class includes parametric models including the BTL and Thurstone models in special cases, but is considerably more general. We provide various examples of models in this broader stochastically transitive class for which classical parametric models provide poor fits. Despite this greater flexibility, we show that the matrix of probabilities can be estimated at the same rate as in standard parametric model-based, within the BTL and T

competing the minimum optimum. In rough terms, given a set of n objects, and a collection of possibly inconsistent comparisons between pairs of these objects, the goal is to aggregate these comparisons in order to perform effective statistical inference on various underlying properties of the population. A particular property of interest is the underlying pairwise comparison probabilities—that is, the probability that object i is preferred to object j in a pairwise comparison. The Bradley-Terry-Luce (Bradley & Terry, 1952; Luce, 1959) and Thurstone (Thurstone, 1927) models are natural ways of analyzing this type of pairwise comparison data. These models are parametric in nature: more specifically, they assume the existence of an n -dimensional weight vector that measures the quality or strength of each item. The pairwise comparison probabilities are then determined via some fixed (parametric) function of the qualities of the pair of objects.

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

Nihar B. Shah
Dept. of EECS, University of California, Berkeley
Dengyong Zhou
Microsoft Research, Redmond

NIBH@EECS.BERKELEY.EDU
DENGYONG.ZHOU@MICROSOFT.COM

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 systematic or involuntary 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 majority reference answer. We mathematically formalize 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

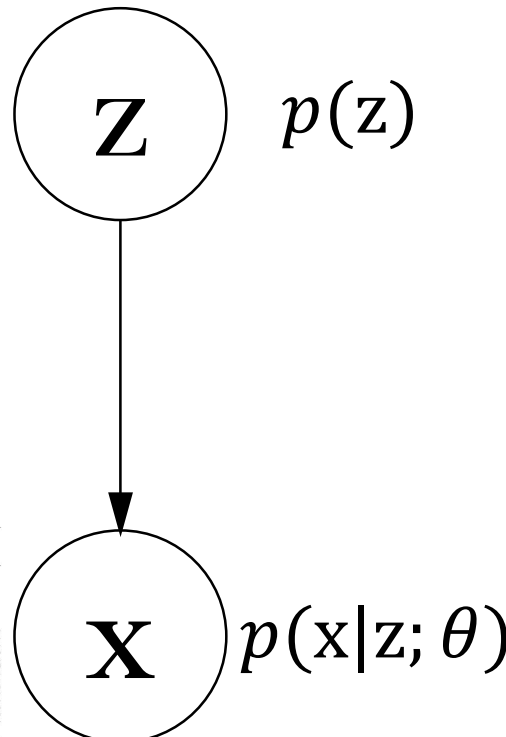
the Internet typically in exchange from some monetary payments. Crowdsourcing is widely used in many real-world applications, and is particularly popular for collecting training labels for machine learning powered systems like web search engines (Berges et al., 2005; Alonso & Mirazo, 2009; Kato, 2011) or to supplement automated algorithms (Khuri et al., 2011; Ling & Rao, 2011; Yao et al., 2008). The labels obtained from crowdsourcing, however, have significant amounts of error (Kato et al., 2011; Vassiri et al., 2011; Wain 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-

□ A simple distribution is not sufficient

Generative models are everywhere ...

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



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

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

Cleison C. A. de Sá, Marcos A. Gonçalves, Daniel X. Sousa, Thiago Sales
Federal University of Bahia Campus
Computer Science Department
Belo Horizonte, Brazil
{cleisonc, mgoncalv, dsousa, tsales}@dcc.ufba.br

ABSTRACT
This task of retrieving information that really matters to the users is considered hard when taking into consideration the relevant and irrelevant information. In response to this task, various have called attention to various of such

learning to rank methods for this task, and have proposed such models were able to compare considered separately, only original training set and

Keywords
Learning to Rank, Broof-L2R

1. INTRODUCTION
There are various parametric models being proposed, such as Boosting, Tree-Learner (BTL) and Boosting. For their ability to learn from complex data, they have been widely used in many applications. In this work, we propose a new parametric model for learning to rank, called Broof-L2R, which is able to learn from complex data, and is able to learn from complex data.

2. RELATED WORK
There are various parametric models being proposed, such as Boosting, Tree-Learner (BTL) and Boosting. For their ability to learn from complex data, they have been widely used in many applications. In this work, we propose a new parametric model for learning to rank, called Broof-L2R, which is able to learn from complex data, and is able to learn from complex data.

3. BROOF-L2R
Broof-L2R is a new parametric model for learning to rank, which is able to learn from complex data, and is able to learn from complex data. It is a new parametric model for learning to rank, which is able to learn from complex data, and is able to learn from complex data.

4. EXPERIMENTAL EVALUATION
We evaluate the performance of Broof-L2R on a dataset of learning to rank. The results show that Broof-L2R outperforms other methods in terms of accuracy and F1 score. This indicates that Broof-L2R is a good model for learning to rank.

5. CONCLUSION
In this paper, we have presented a new parametric model for learning to rank, called Broof-L2R. The results show that Broof-L2R outperforms other methods in terms of accuracy and F1 score. This indicates that Broof-L2R is a good model for learning to rank.

6. REFERENCES
[1] Cleison C. A. de Sá, Marcos A. Gonçalves, Daniel X. Sousa, Thiago Sales. Generalized BROOF-L2R: A General Framework for Learning to Rank Based on Boosting and Random Forests. In Proceedings of the 2019 ACM Conference on Artificial Intelligence, 2019.

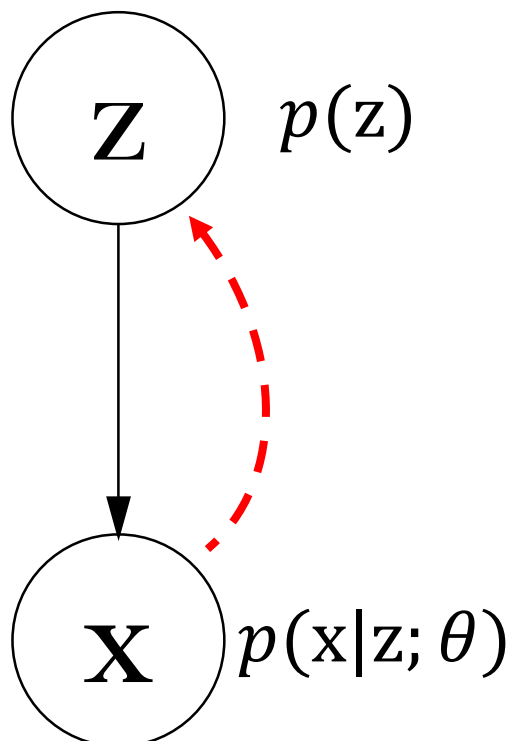
7. ACKNOWLEDGMENTS
This work was supported by the Brazilian National Council of Research (CNPq) under grant number 301301/2018-0.

8. BIOGRAPHICAL NOTES
Cleison C. A. de Sá is a Ph.D. student in the Department of Computer Science at the Federal University of Bahia, Brazil. He is currently working on his thesis, which is titled "Generalized BROOF-L2R: A General Framework for Learning to Rank Based on Boosting and Random Forests".

9. CONTACT
Cleison C. A. de Sá: cleisonc@dcc.ufba.br

Generative models are everywhere ...

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



Bayes' Rule:

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

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

Nihar B. Shah
Dept. of EECS, University of California, Berkeley
Dengyong Zhou
Microsoft Research, Redmond

NIHAR@EECS.BERKELEY.EDU

DENGYONG.ZHOU@MICROSOFT.COM

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 formalize 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 for 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 (Burgess et al., 2005; Alonso & Mizzaro, 2009; Kazai, 2011) or to supplement automated algorithms (Kash et al., 2011; Lang & Rio-Rousse, 2011; Van Ahn et al., 2008). The labels obtained from crowdsourcing, however, have significant amounts of error (Kazai et al., 2011; Vassiri et al., 2011; Wals 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-

Generative models are everywhere ...

- ◆ 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

Clebson C. A. de Sá, Marcos A. Gonçalves, Daniel X. Sousa, Thiago Sales
Federal University of Minas Gerais
Computer Science Department
Belo Horizonte, Brazil
(clebson, mgoncalv, danielxs, sales)@dcc.ufmg.br

ABSTRACT

The task of retrieving information that really matters to the user is considered hard when taking into consideration the complex and heterogeneous amount of available information. In response to this problem, in this paper, we propose a novel framework for learning to rank (LTR). The main goal of this task is to learn a ranking function that orders the documents according to the relevance of the information they contain. In this paper, we propose a novel framework for learning to rank (LTR). The main goal of this task is to learn a ranking function that orders the documents according to the relevance of the information they contain. In this paper, we propose a novel framework for learning to rank (LTR). The main goal of this task is to learn a ranking function that orders the documents according to the relevance of the information they contain.

Keywords: Learning to Rank, Boosting, Random Forests

ABSTRACT

In this paper, we propose a novel framework for learning to rank (LTR). The main goal of this task is to learn a ranking function that orders the documents according to the relevance of the information they contain. In this paper, we propose a novel framework for learning to rank (LTR). The main goal of this task is to learn a ranking function that orders the documents according to the relevance of the information they contain. In this paper, we propose a novel framework for learning to rank (LTR). The main goal of this task is to learn a ranking function that orders the documents according to the relevance of the information they contain.

1. INTRODUCTION

Most of the document retrieval methods consider query and document representations. In addition, the generalization ability of the models is a key factor in the success of the retrieval system. In this paper, we propose a novel framework for learning to rank (LTR). The main goal of this task is to learn a ranking function that orders the documents according to the relevance of the information they contain.

In the text and metadata of entities is a, along with the corresponding information. This is the task of the retrieval system. In this paper, we propose a novel framework for learning to rank (LTR). The main goal of this task is to learn a ranking function that orders the documents according to the relevance of the information they contain. In this paper, we propose a novel framework for learning to rank (LTR). The main goal of this task is to learn a ranking function that orders the documents according to the relevance of the information they contain.

Stochastically Tractable Models for Pairwise Comparisons: Statistical and Computational Issues

Nihar R. Shah,
Shantanu Debbarh
University of California, Berkeley
nshah@cs.berkeley.edu

Abstract

There are various pairwise models for learning to rank, including the Bradley-Terry (BT) and Thurstone models. In this paper, we study the statistical and computational issues of these models. We show that the BT model is not tractable in general, while the Thurstone model is tractable. We also study the issue of learning to rank from pairwise comparisons. We show that the BT model is not tractable in general, while the Thurstone model is tractable. We also study the issue of learning to rank from pairwise comparisons.

Keywords: Learning to Rank, Pairwise Comparisons, Statistical Issues, Computational Issues

Nihar R. Shah,
Shantanu Debbarh
University of California, Berkeley
nshah@cs.berkeley.edu

Abstract

There are various pairwise models for learning to rank, including the Bradley-Terry (BT) and Thurstone models. In this paper, we study the statistical and computational issues of these models. We show that the BT model is not tractable in general, while the Thurstone model is tractable. We also study the issue of learning to rank from pairwise comparisons. We show that the BT model is not tractable in general, while the Thurstone model is tractable. We also study the issue of learning to rank from pairwise comparisons.

Keywords: Learning to Rank, Pairwise Comparisons, Statistical Issues, Computational Issues

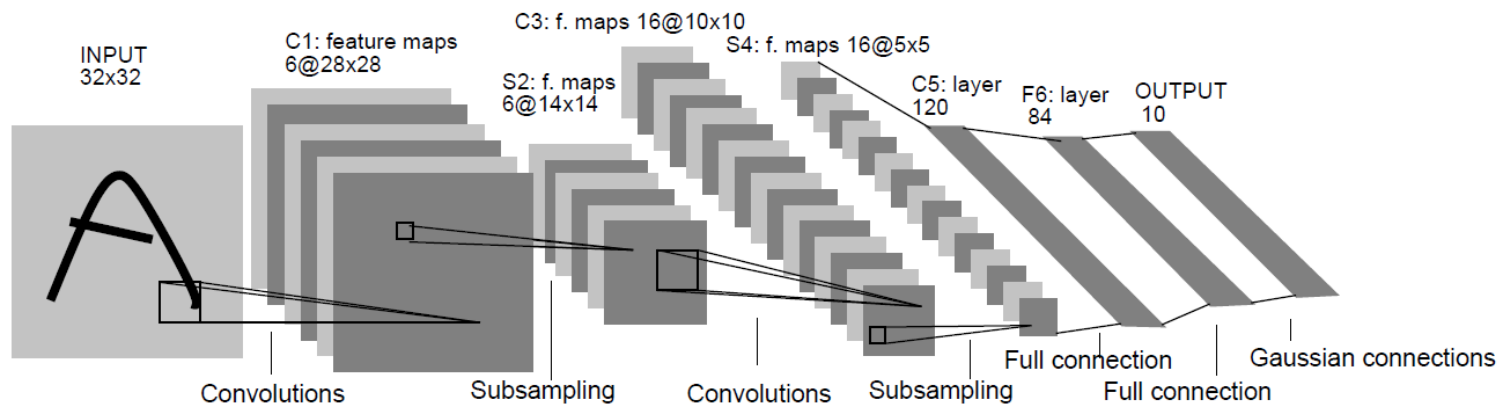
In the text and metadata of entities is a, along with the corresponding information. This is the task of the retrieval system. In this paper, we propose a novel framework for learning to rank (LTR). The main goal of this task is to learn a ranking function that orders the documents according to the relevance of the information they contain. In this paper, we propose a novel framework for learning to rank (LTR). The main goal of this task is to learn a ranking function that orders the documents according to the relevance of the information they contain.

retrieval 0.05
text 0.02
learning 0.01
sports 0.01
...

learning 0.03
theory 0.02
algorithm 0.01
deep 0.01
...

Discriminative Deep Learning

◆ Learn a deep NN to map an input to output



- ❑ Gradient back-propagation
- ❑ Dropout
- ❑ Activation functions:
 - rectified linear

Generative Modeling

- ◆ Have training examples

$$\mathbf{x} \sim p_{\text{data}}(\mathbf{x})$$

- ◆ Want a model that can draw samples:

$$\mathbf{x}' \sim p_{\text{model}}(\mathbf{x})$$

- where $p_{\text{model}}(\mathbf{x}) \approx p_{\text{data}}(\mathbf{x})$



$$\mathbf{x} \sim p_{\text{data}}(\mathbf{x})$$



$$\mathbf{x}' \sim p_{\text{model}}(\mathbf{x})$$

Two ways to build deep generative models

◆ Traditional one

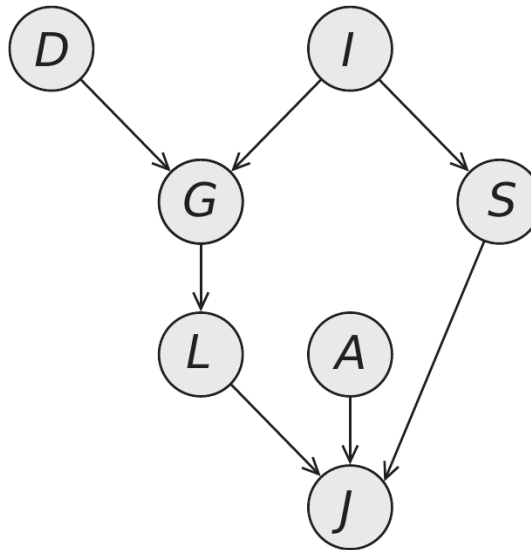
- Hierarchical Bayesian methods

◆ More modern one

- Deep generative models

Hierarchical Bayesian Modeling

- ◆ Build a hierarchy through distributions in analytical forms

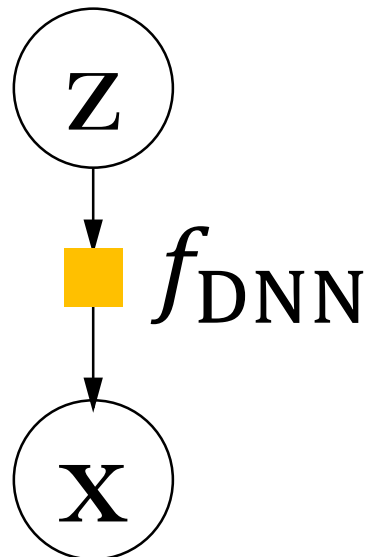


$$P(D, I, G, S, L, A, J) = P(D)P(I)P(G|D, I)P(S|I) \cdot \\ P(A) P(L|G)P(J|L, A, S)$$

Simple, Local Factors: a conditional probability distribution

Deep Generative Models

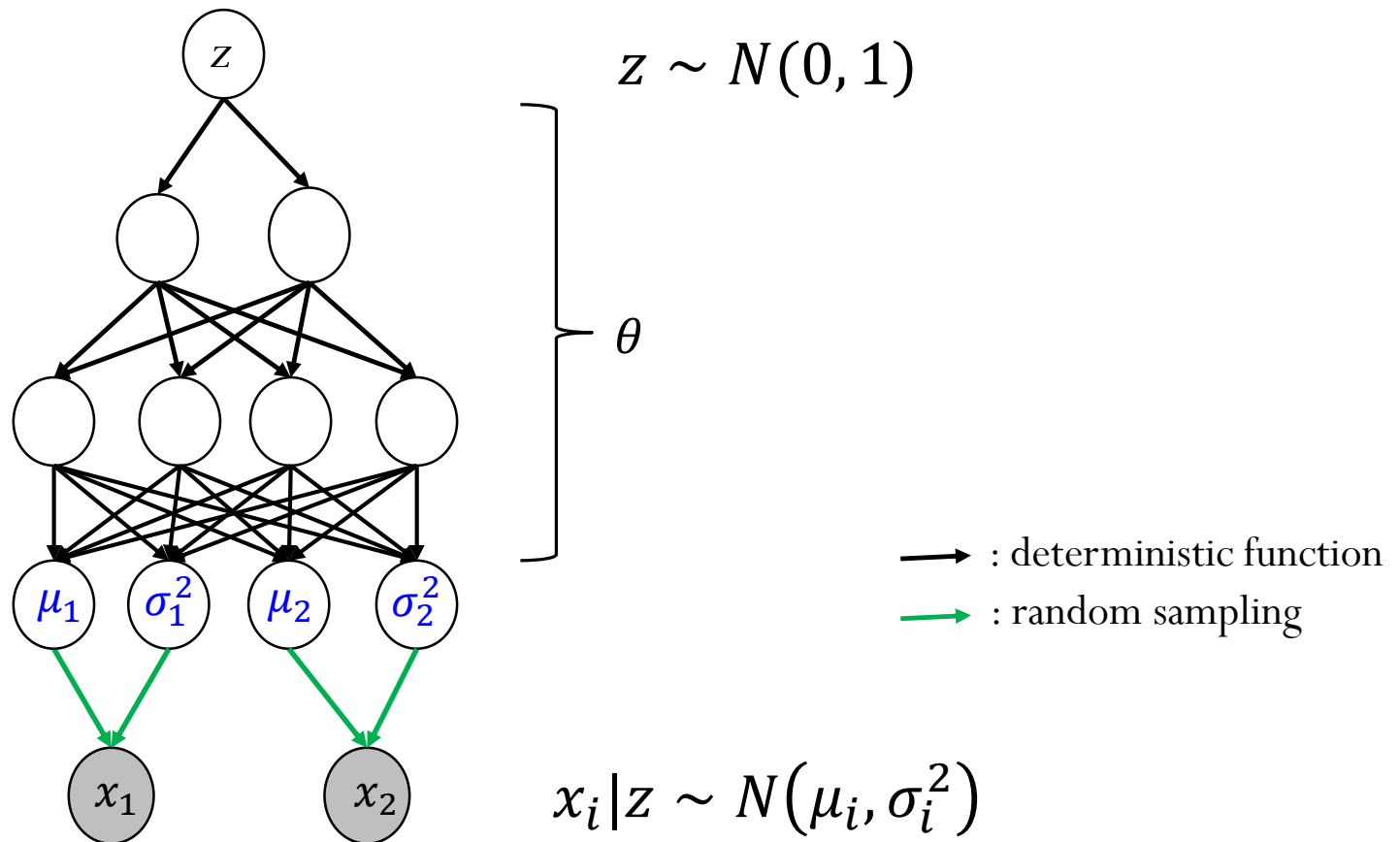
- ◆ More flexible by using differential function mapping between random variables
- ◆ DGMs learn a function transform with deep neural networks



Cause \Rightarrow Disease
Topics \Rightarrow Docs
Objects \Rightarrow Images
Words \Rightarrow Phonemes

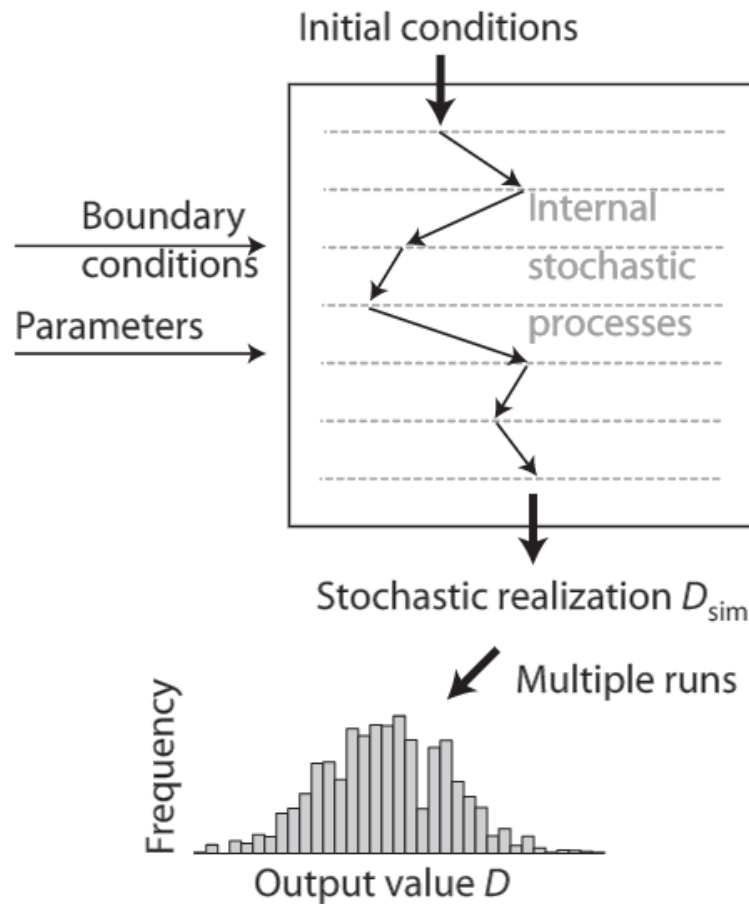
An example with MLP

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



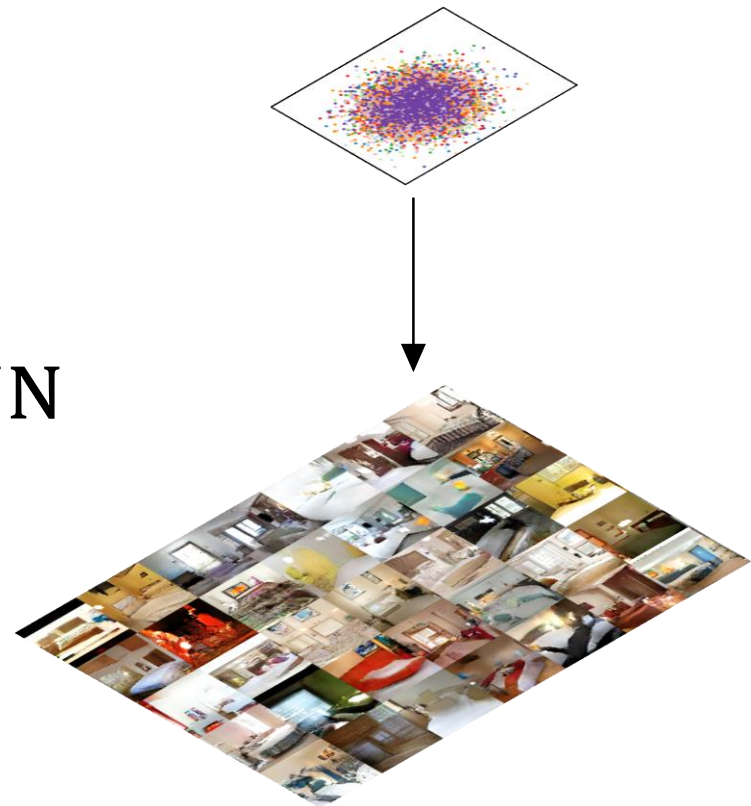
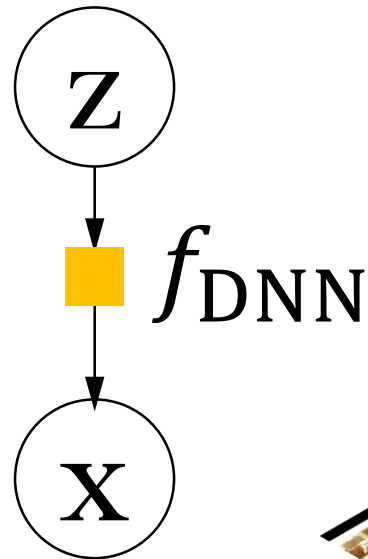
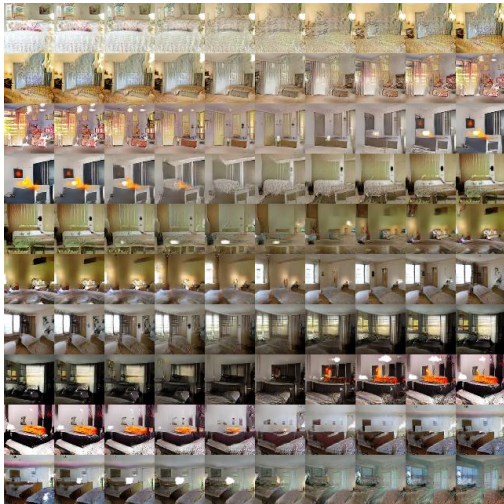
Implicit Deep Generative Models

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

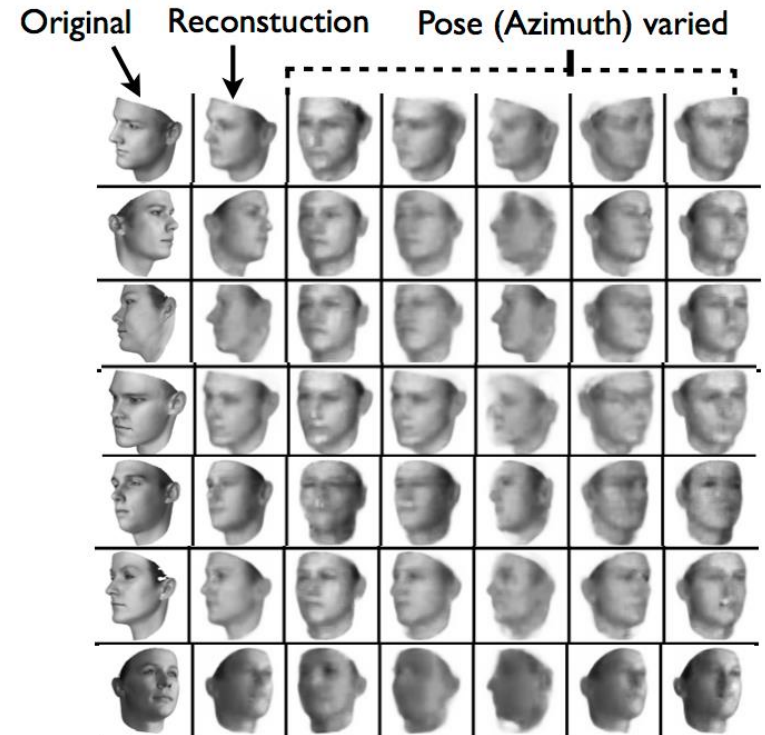
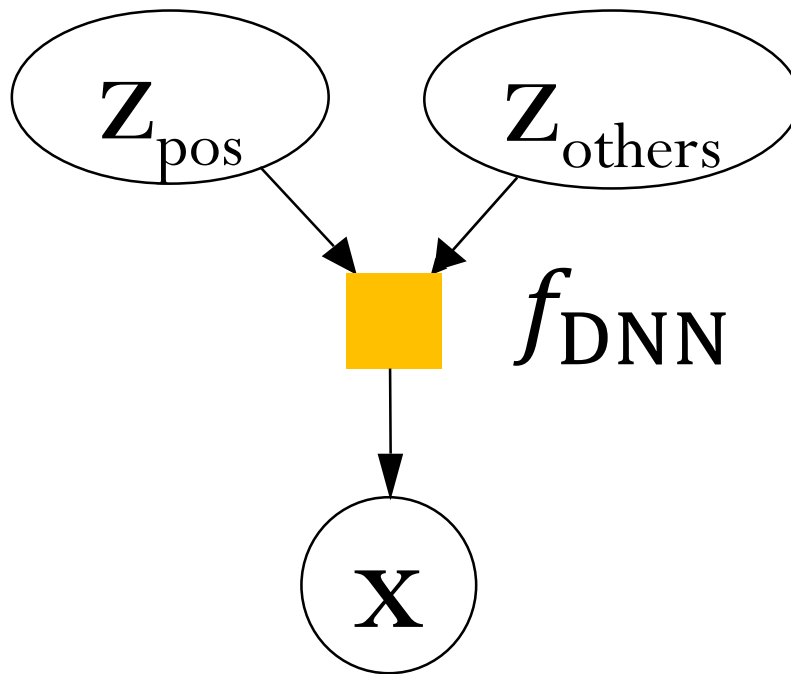


Deep Generative Models

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



Deep Generative Models

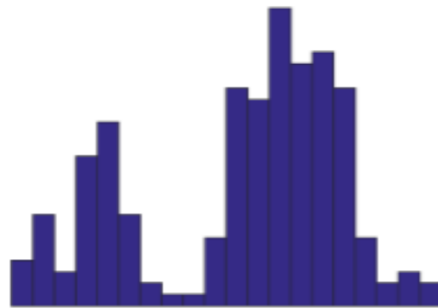


[Image Understanding: Variational Autoencoders,
Kingma13 & Tejas15 & Eslami16]

Learning Deep Generative Models

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

data $D = \{x_i\}$



models $p(x|\theta)$



- ◆ Find a model that minimizes

$$\mathbb{D}\left(\begin{array}{c} \text{data } \{x_i\}_{i=1}^n \\ \text{model } p \end{array}\right)$$

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 : $\hat{D} = \{y_i\}_{i=1}^M$, 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*

$$\log p(\mathbf{x}; \theta) = \log \int p(\mathbf{z}, \mathbf{x}; \theta) d\mathbf{z}$$

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

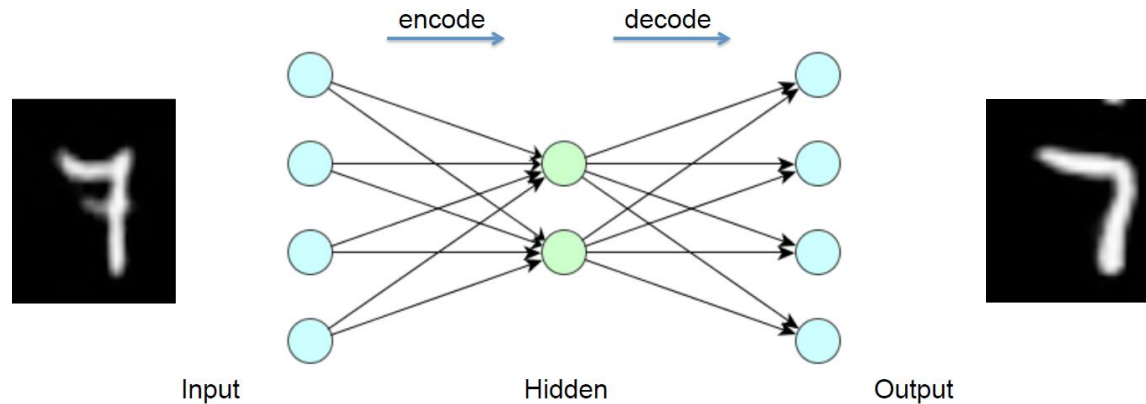
$$\log p(\mathbf{x}; \theta) = L(\theta, \phi, \mathbf{x}) + \text{KL}(q(\mathbf{z}|\mathbf{x}; \phi) \| p(\mathbf{z}|\mathbf{x}; \theta))$$

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

reconstruction term

prior regularization

Recap: Auto-Encoder

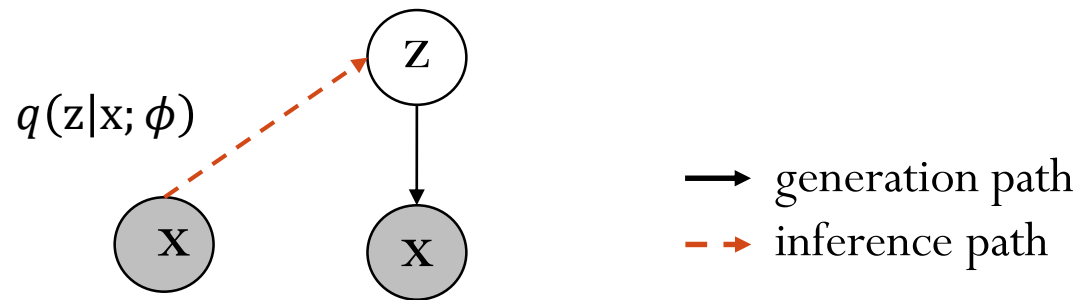


- ◆ Encoder: $h = s(Wx + b)$
- ◆ Decoder: $x' = s(W'h + b')$
- ◆ Training: minimize the reconstruction error (e.g., square loss, cross-entropy loss)
- ◆ Denoising AE: randomly corrupted inputs are restored to learn more robust features

Auto-Encoding Variational Bayes (AEVB)

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

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

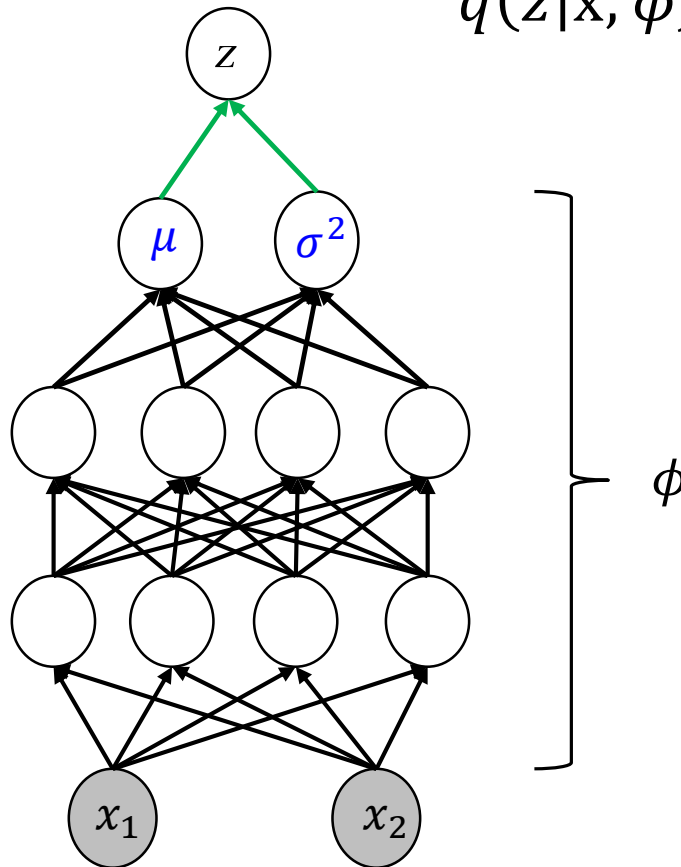


- We call it an **inference (recognition, encoder) network** or a **Q-network**
- All the parameters are learned jointly via SGD with variance reduction

The Encoder Network

◆ A feedforward NN + Gaussian

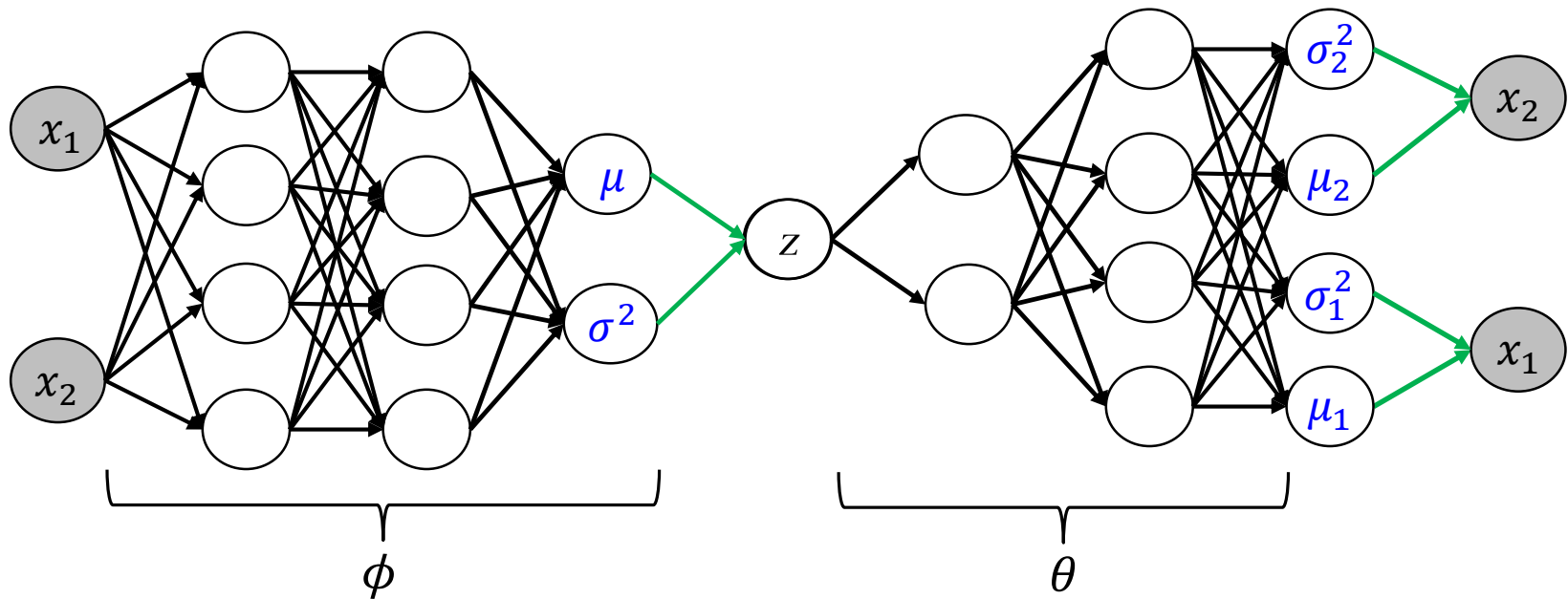
$$q(z|x; \phi) = N(\mu(x; \phi), \sigma^2(x; \phi))$$



→ : deterministic function
→ : random sampling

The Complete Auto-encoder

◆ The Q-P network architecture:



→ : deterministic function

→ : random sampling

Stochastic Variational Inference

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

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

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

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

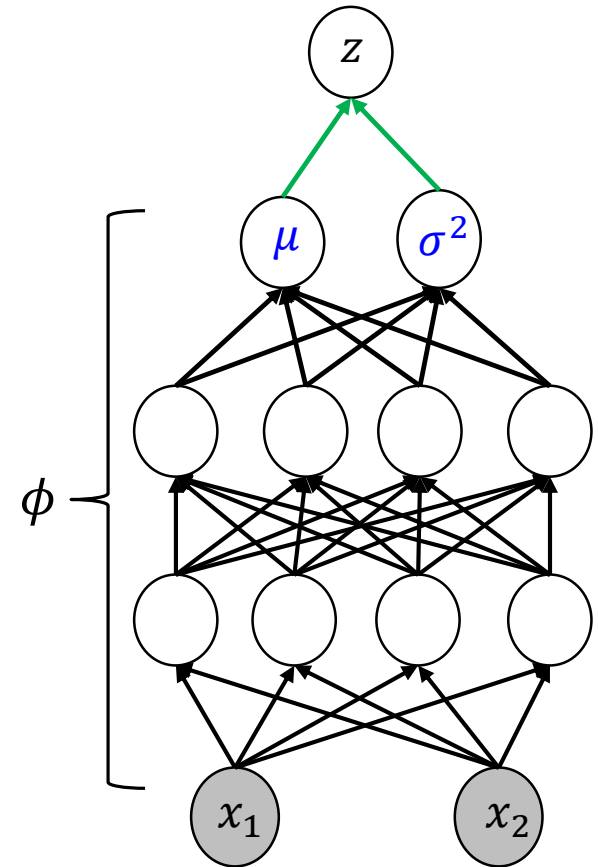
- Use stochastic gradient methods to handle large datasets
- Random mini-batch
 - for each i , infer the posterior $q(\mathbf{z}_i|\mathbf{x}_i; \phi)$; As we parameterize as a neural network, this in fact optimizes ϕ
- ◆ *However, calculating the expectation and its gradients is non-trivial, often intractable*

Example with Gaussian Distributions

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

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

- **Homework:** finish the derivation



Example with Gaussian Distributions

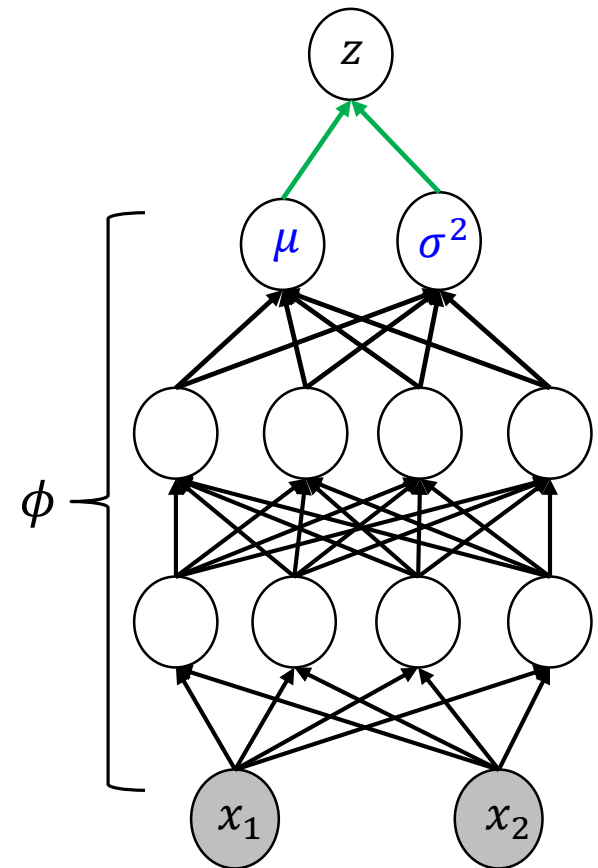
- ◆ Use $N(0, 1)$ as prior for z ; $q(z|x; \phi)$ is Gaussian with parameters $(\mu(x; \phi), \sigma^2(x; \phi))$ determined by NN
 - The expected log-likelihood

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

- If the likelihood is Gaussian

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

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



Example with Gaussian Distributions

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

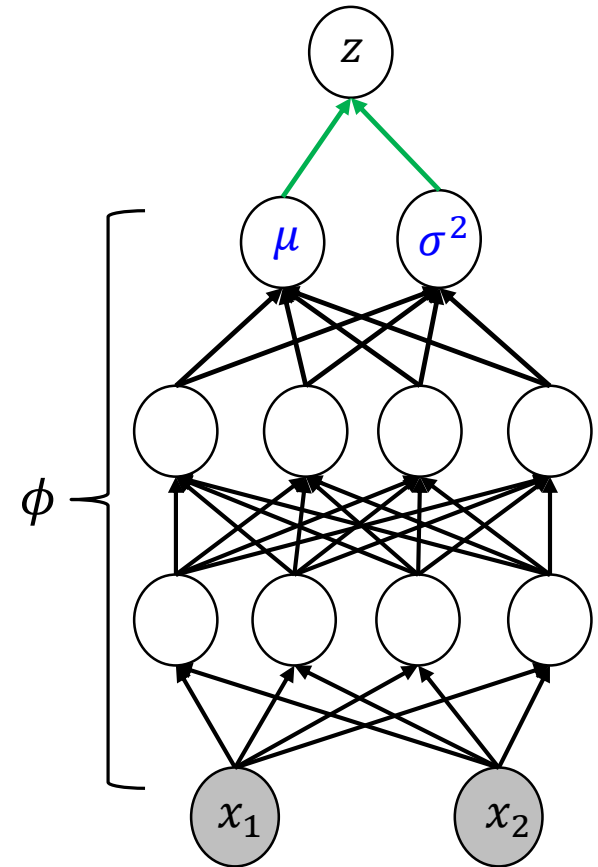
$$\mathbf{E}_{q(z|x; \phi)} [\log p(x|z; \theta)]$$

- Approximate via Monte Carlo methods

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

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

- An unbiased estimator



Example with Gaussian Distributions

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

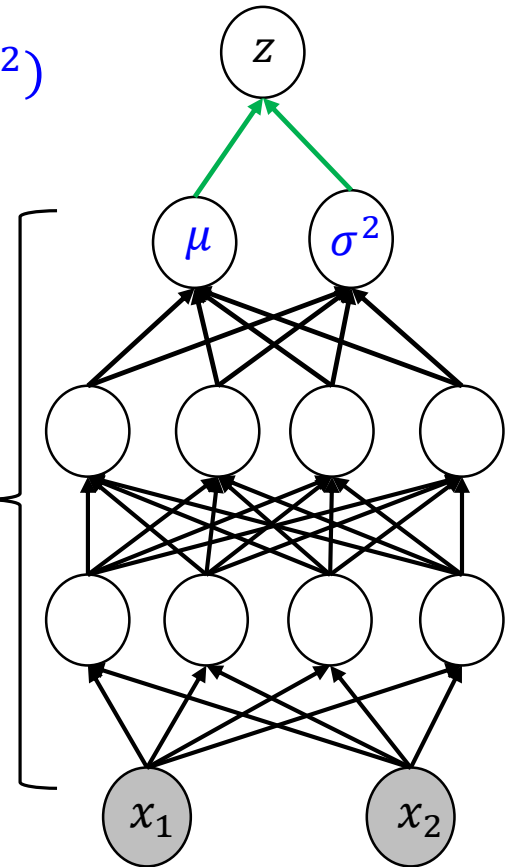
$$-\text{KL}(q(z|x; \phi) \| p(z; \theta)) = \frac{1}{2} (1 + \log \sigma^2 - \mu^2 - \sigma^2)$$

- Easy to calculate gradient
- ◆ The expected log-likelihood term (MC estimate)

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

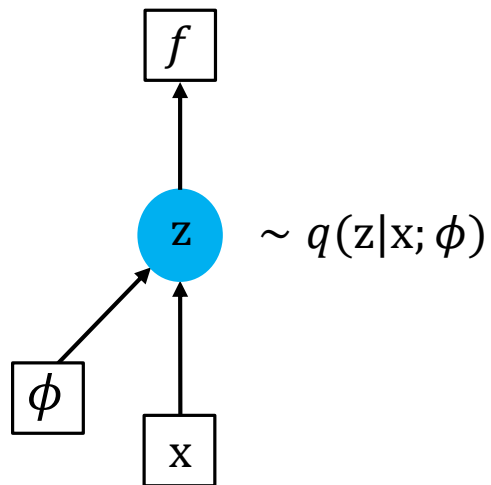
$$z^{(k)} \sim q(z|x; \phi)$$

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



Reparameterization Trick

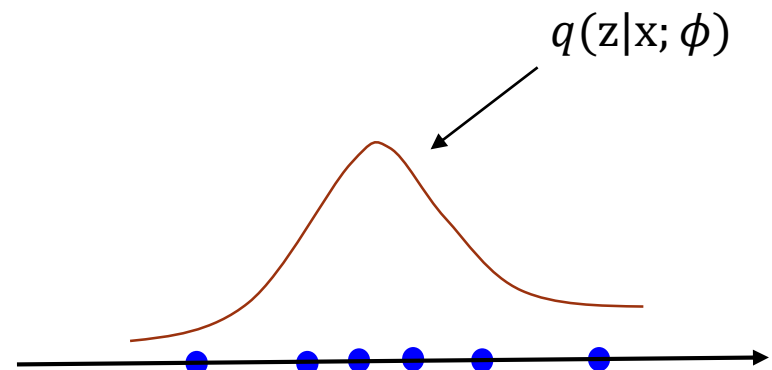
- ◆ Backpropagation not possible through random sampling



□ : deterministic node ● : random node

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

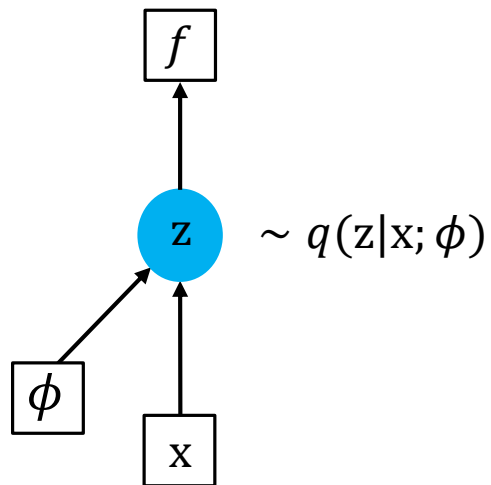
Cannot back-propagate through a
randomly drawn number



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

Reparameterization Trick

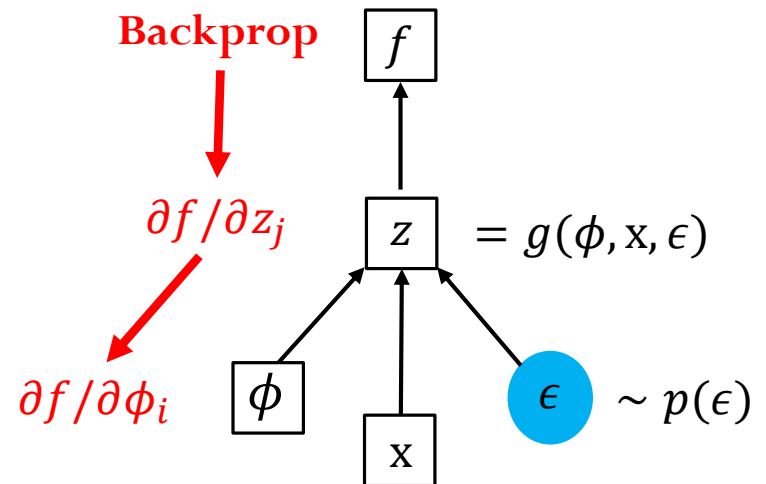
◆ Backpropagation not possible through random sampling



□: deterministic node ●: random node

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

Cannot back-propagate through a randomly drawn number



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

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

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

The General Form

◆ The VAE bound

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

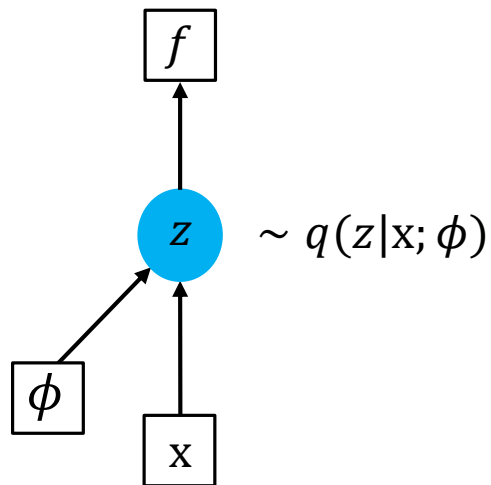
◆ Monte Carlo estimate:

$$\begin{aligned} L(\theta, \phi, \mathbf{x}) &\approx \frac{1}{L} \sum_k \log \frac{p(\mathbf{z}^{(k)}, \mathbf{x}; \theta)}{q(\mathbf{z}^{(k)}|\mathbf{x}; \phi)} \\ \mathbf{z}^{(k)} &\sim q(\mathbf{z}|\mathbf{x}; \phi) \end{aligned}$$

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

Reparameterization Trick

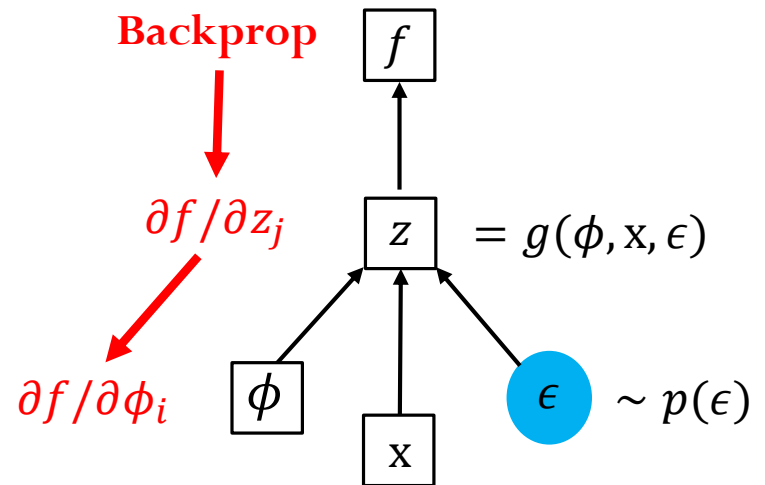
❖ Backpropagation not possible through random sampling



\square : deterministic node \bullet : random node

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

Cannot back-propagate through a randomly drawn number



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

$$z^{(k)} = g(\phi, x, \epsilon^{(k)})$$

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

Reparam-Trick Summary

◆ The VAE bound

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

□ Reparameterized as

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

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

◆ The gradients are

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

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

Importance Weighted Auto-Encoder (IWAE)

- ◆ The VAE lower bound of log-likelihood

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

- ◆ A better variational lower bound (IWAE)

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

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

- 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:

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

Reparametrization Trick

◆ The IWAE bound:

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

$$\text{where } \mathbf{z}^{(k)} \sim q(\mathbf{z}|\mathbf{x}; \phi) \quad w(\mathbf{z}^{(k)}, \mathbf{x}; \theta, \phi) = \frac{p(\mathbf{z}^{(k)}, \mathbf{x}; \theta)}{q(\mathbf{z}^{(k)}|\mathbf{x}; \phi)}$$

◆ Reparameterization form:

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

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

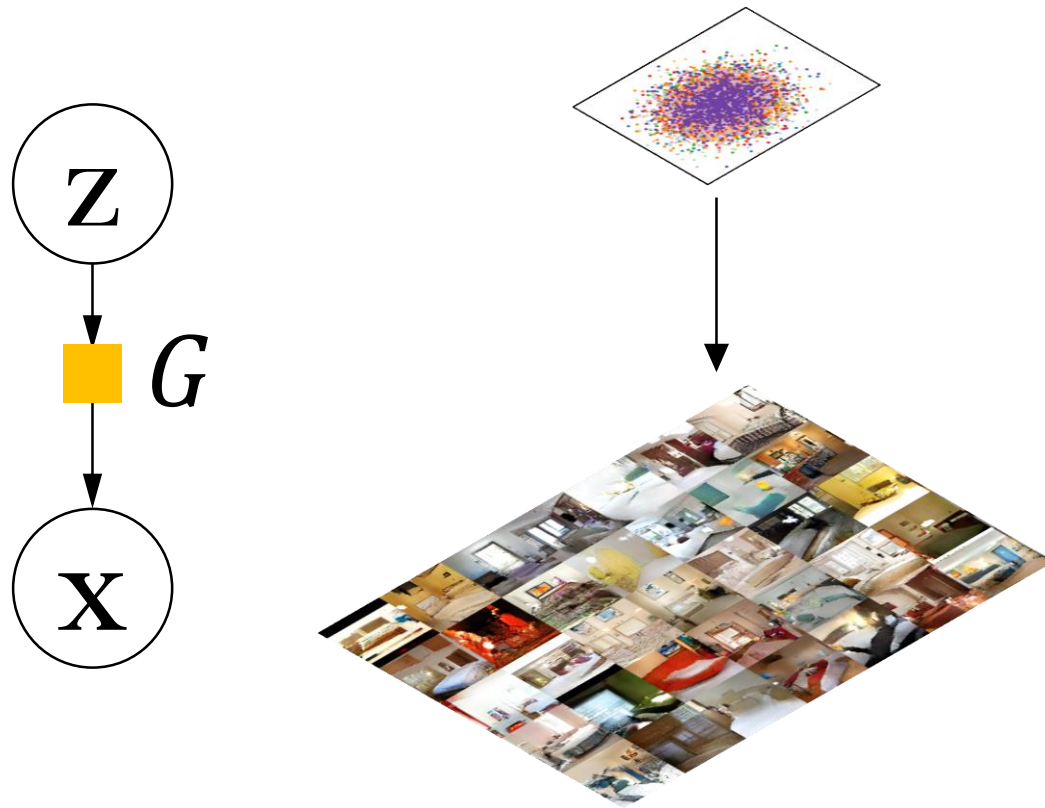
- The gradient can be calculated as in VAE

Generative Adversarial Networks (GAN)

- ◆ A game between two players:
 - A discriminator D
 - A generator G
- ◆ D tries to discriminate between:
 - A sample from the data distribution.
 - And a sample from the generator G .
- ◆ G tries to “trick” D by generating samples that are hard for D to distinguish from data.

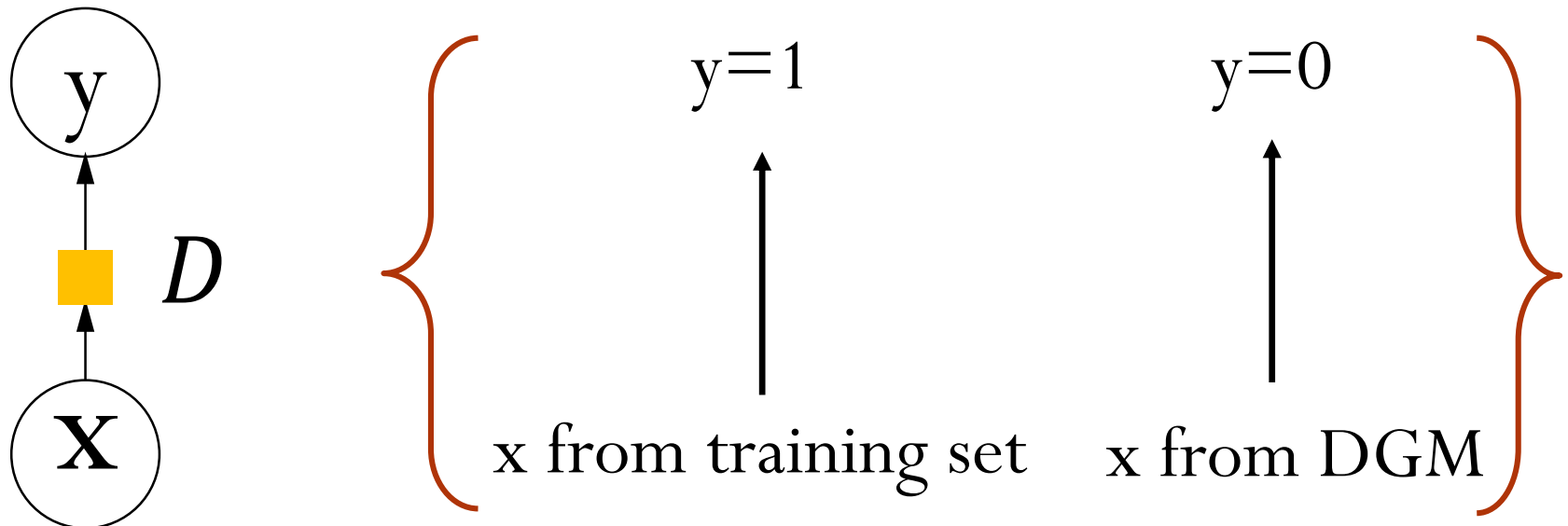
GAN – architecture

- ◆ The generator-network G is a DGM
 - ▣ It generates samples from random noise



GAN – architecture

- ◆ The discriminator-network D is a binary classifier
 - ▣ It aims to assign the correct label to both training samples and the samples from G



- ▣ This is a supervised learning task (binary classification)!

GAN - objective

- ◆ The discriminator-network D is a binary classifier
 - It aims to assign the correct label to both training samples and the samples from G
- ◆ Maximum likelihood estimation (MLE) is the natural choice!

$$\max_D \mathbf{E}_{p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbf{E}_{p(\mathbf{z})} [\log (1 - D(G(\mathbf{z})))]$$

↑
 \mathbf{x} from training set

↖
 $G(\mathbf{z})$ from generator

- $D(\mathbf{x}) = p(y=1 \mid \mathbf{x})$
- aka. cross-entropy loss minimization

GAN - objective

- ◆ The generator aims to fool the discriminator D
 - Generated samples should be identified as “real” by D
 - Maximize the likelihood of being real:

$$\max_G \mathbf{E}_{p(z)} \left[\log \left(D(G(z)) \right) \right]$$

↑

$G(z)$ is a sample from generator

- Or minimize the likelihood of being fake:

$$\min_G \mathbf{E}_{p(z)} \left[\log \left(1 - D(G(z)) \right) \right]$$

GAN – objective

◆ Minimax objective function

$$\min_G \max_D \mathbf{E}_{p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbf{E}_{p(\mathbf{z})} [\log (1 - D(G(\mathbf{z})))]$$

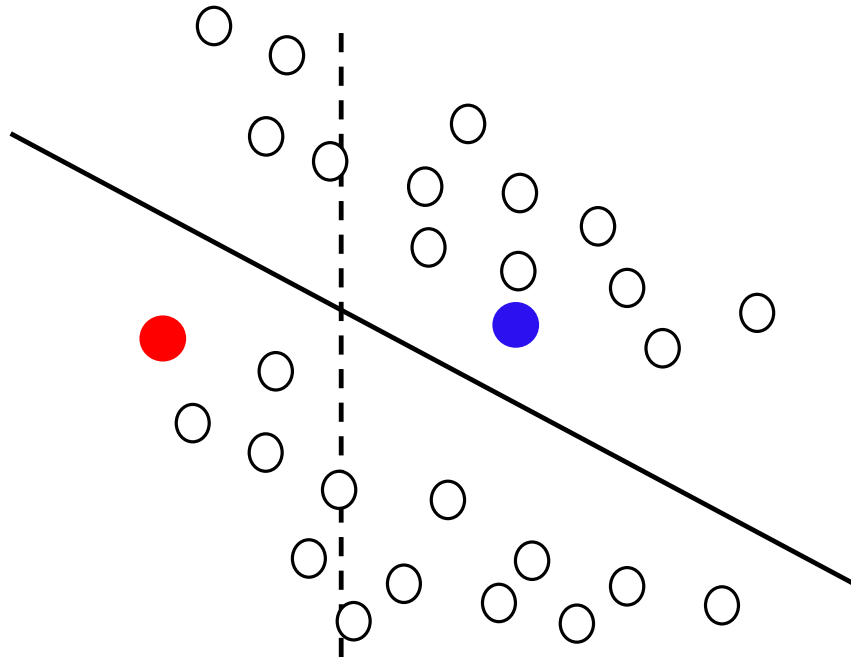
□ Optimal strategy of the discriminator for any $p_{\text{model}}(\mathbf{x})$ is

$$D(\mathbf{x}) = \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_{\text{model}}(\mathbf{x})}$$

- Assume infinite data, infinite model capacity, direct updating generator's distribution
 - Unique global optimum
 - Optimum corresponds to data distribution

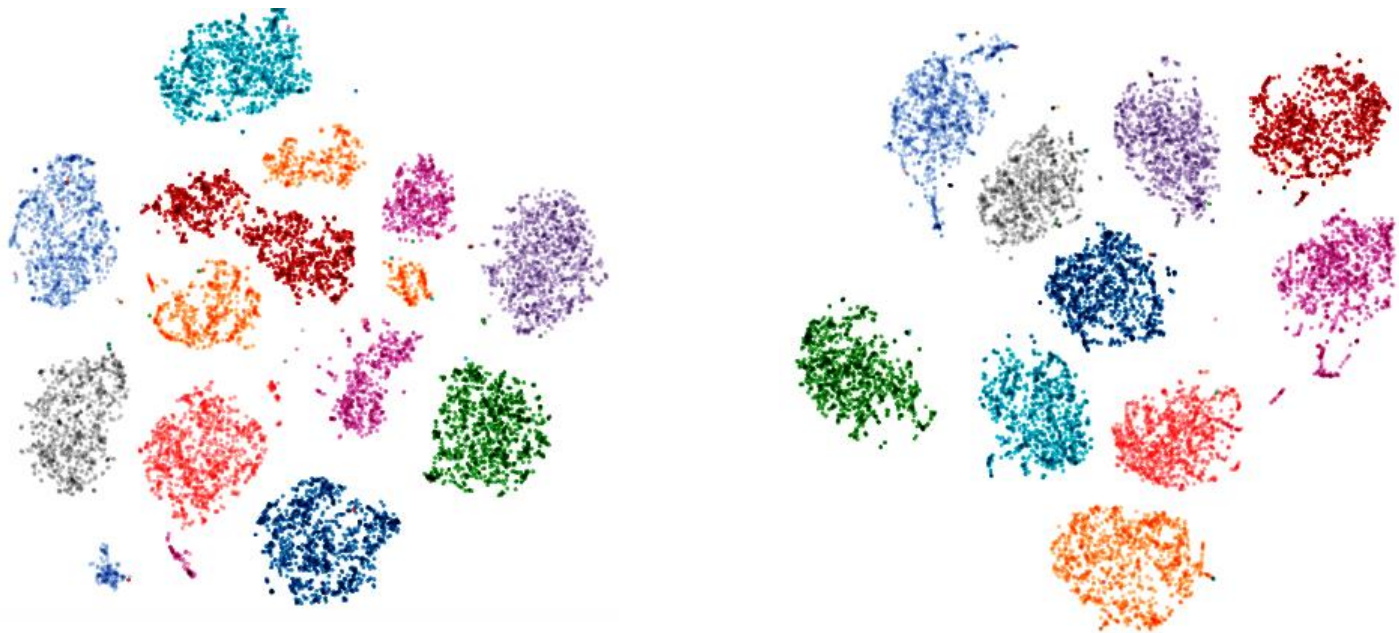
Semi-supervised Learning

◆ A toy example



Representation Matters

- ◆ t-SNE embedding of learned representations by different DGM models on CIFAR10



Triple Generative Adversarial Nets

◆ A minimax game for semi-supervised learning

□ GAN is for unsupervised learning

$$p_{\text{model}}(\mathbf{x}) = p_{\text{data}}(\mathbf{x})$$

□ We aim to learn the joint distribution

$$p_{\text{model}}(\mathbf{x}, \mathbf{y}) = p_{\text{data}}(\mathbf{x}, \mathbf{y})$$

◆ We need three players

□ factorization form with conditionals

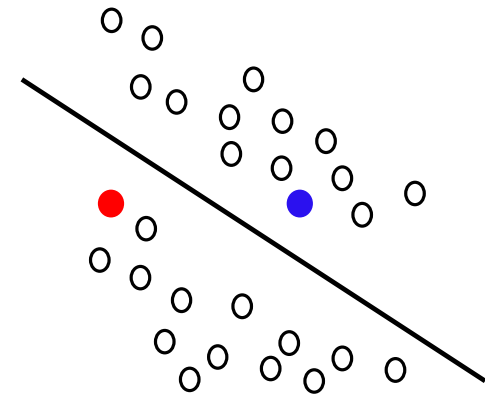
$$\begin{aligned} p(\mathbf{x}, \mathbf{y}) &= p(\mathbf{x})p(\mathbf{y}|\mathbf{x}) \\ &= p(\mathbf{y})p(\mathbf{x}|\mathbf{y}) \end{aligned}$$

← A classifier

← A class-conditional generator

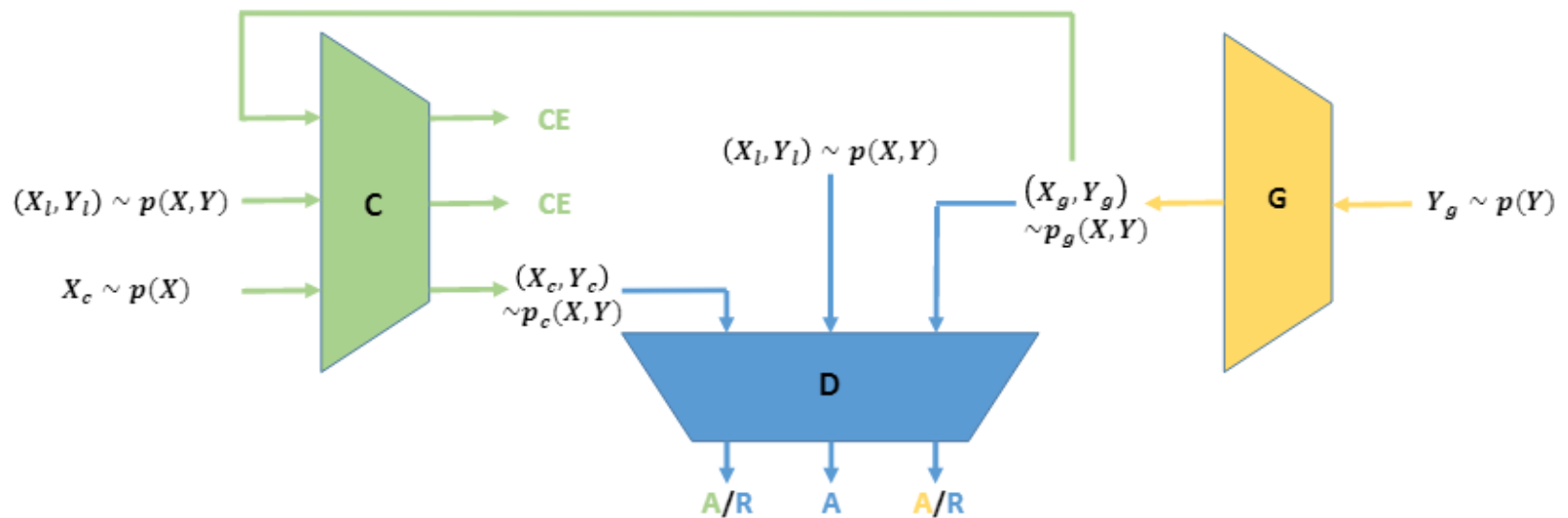
□ Two generators to generate (\mathbf{x}, \mathbf{y})

□ A discriminator to distinguish fake (\mathbf{x}, \mathbf{y})



Triple-GAN

◆ The network architecture



- Both C and G are generators
- D is the discriminator
- CE: cross-entropy loss for learning classifier

A minimax game

◆ The optimization problem

$$\min_{C, G} \max_D U(C, G, D) = E_p[\log D(x, y)] + \alpha E_{p_c}[\log(1 - D(x, y))] + (1 - \alpha) E_{p_g}[\log(1 - D(x, y))]$$

- The hyper-parameter α is often set at $1/2$
- The standard supervised loss can be incorporated

$$\min_{C, G} \max_D \tilde{U}(C, G, D) = U(C, G, D) + E_p[-\log p_c(y|x)]$$

Major theoretical results

Theorem

The equilibrium of $\tilde{U}(C, G, D)$ is achieved if and only if $p(x, y) = p_g(x, y) = p_c(x, y)$ with $D_{C,G}^(x, y) = \frac{1}{2}$ and the optimum value is $-\log 4$.*

Lemma

For any fixed C and G , the optimal discriminator D is:

$$D_{C,G}^*(x, y) = \frac{p(x, y)}{p(x, y) + p_\alpha(x, y)},$$

where $p_\alpha(x, y) := (1 - \alpha)p_g(x, y) + \alpha p_c(x, y)$.

Some Practical Tricks for SSL

- Pseudo discriminative loss: using $(x, y) \sim p_g(x, y)$ as labeled data to train C
 - Explicit loss, equivalent to $KL(p_g(x, y) || p_c(x, y))$
 - Complementary to the implicit regularization by D
- Collapsing to the empirical distribution $p(x, y)$
 - Sample $(x, y) \sim p_c(x, y)$ as true data for D
 - Biased solution: target shifting towards $p_c(x, y)$
- Unlabeled data loss on C
 - Confidence (Springenberg [2015])
 - Consistence (Laine and Aila [2016])

Some Results

◆ Semi-supervised classification

Table 1: Error rates (%) on partially labeled MNIST, SHVN and CIFAR10 datasets. The results with [†] are trained with more than 500,000 extra unlabeled data on SVHN.

Algorithm	MNIST $n = 100$	SVHN $n = 1000$	CIFAR10 $n = 4000$
<i>M1+M2</i> [11]	3.33 (± 0.14)	36.02 (± 0.10)	
<i>VAT</i> [18]	2.33		24.63
<i>Ladder</i> [23]	1.06 (± 0.37)		20.40 (± 0.47)
<i>Conv-Ladder</i> [23]	0.89 (± 0.50)		
<i>ADGM</i> [17]	0.96 (± 0.02)	22.86 [†]	
<i>SDGM</i> [17]	1.32 (± 0.07)	16.61(± 0.24) [†]	
<i>MMCVA</i> [15]	1.24 (± 0.54)	4.95 (± 0.18) [†]	
<i>CatGAN</i> [26]	1.39 (± 0.28)		19.58 (± 0.58)
<i>Improved-GAN</i> [25]	0.93 (± 0.07)	8.11 (± 1.3)	18.63 (± 2.32)
<i>ALI</i> [5]		7.3	18.3
<i>Triple-GAN (ours)</i>	0.91 (± 0.58)	5.77 (± 0.17)	16.99 (± 0.36)

Some Results

◆ Class-conditional generation



(d) Dog



(e) Horse



(f) Ship

Some Results

◆ Disentangle class and style



Figure: Same y for each row. Same z for each column.

Some Results

- ◆ Latent space interpolation on MNIST



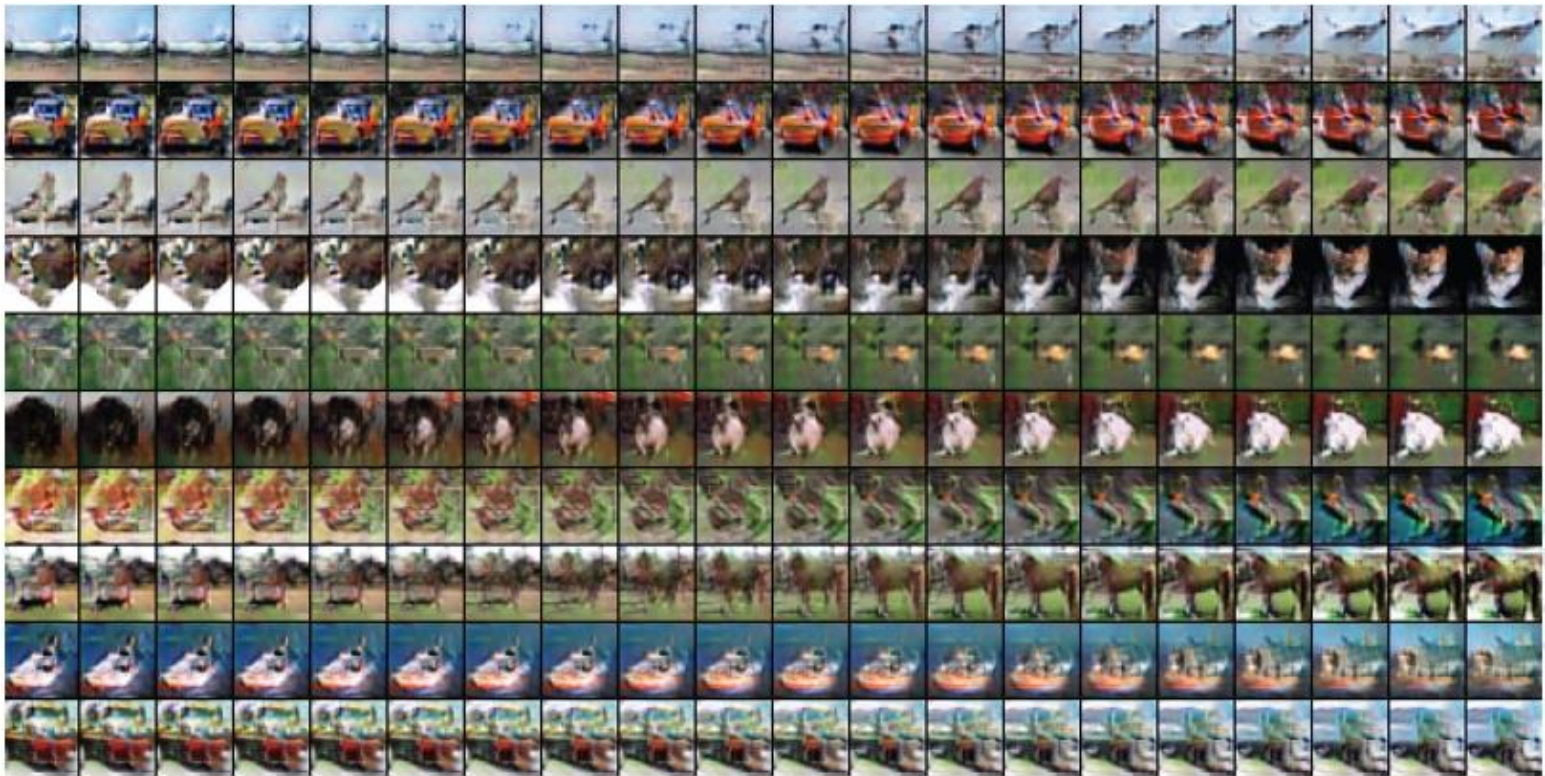
Some Results

- ◆ Latent space interpolation on SVHN



Some Results

◆ Latent space interpolation on CIFAR10

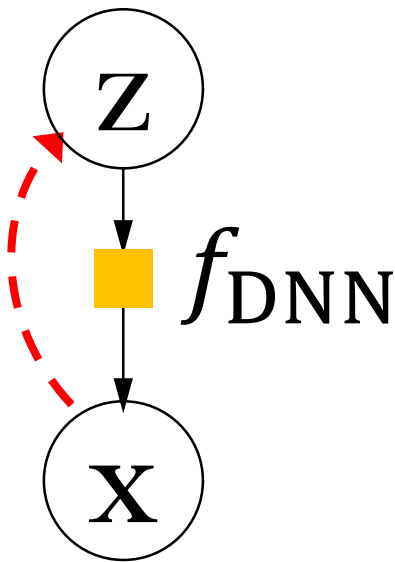


ZhuSuan

<http://zhusuan.readthedocs.io>

Bayesian inference

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



Given **Disease**, what is the **Cause**?

Given **Object**, what are the **Components**?

Given **Docs**, what are the **Topics**?

Find **Cause** of **Disease**

Extract **Topics** from **Docs**

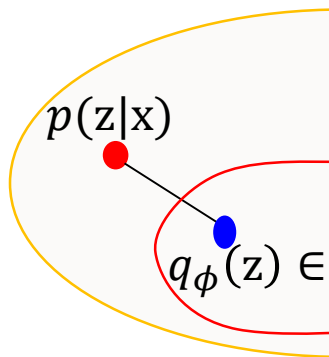
Identify **Objects** from **Images**

Recognize **Words** in **Speeches**

Inference in Old Days

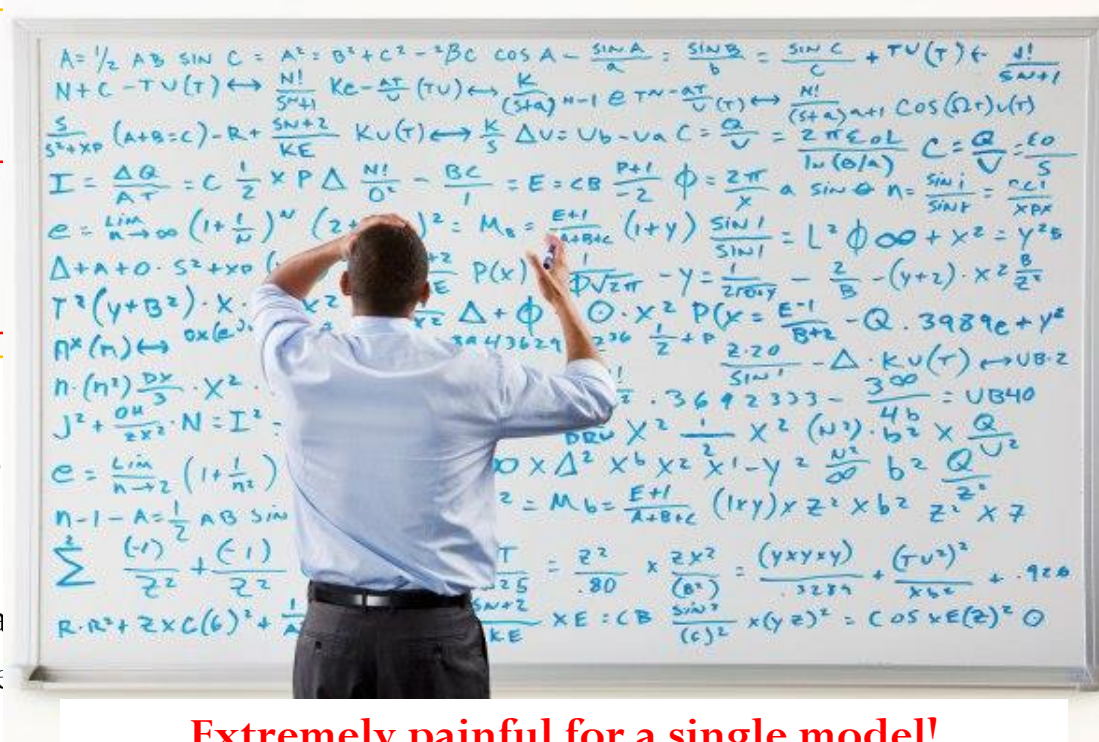
Variational Inference

(Too much math!!!)



$$\min_{\phi} \text{KL}(q_{\phi} \| p(z|x))$$

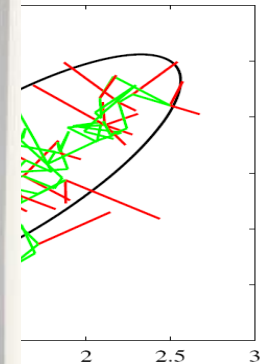
Varia
postc



Extremely painful for a single model!

MCMC

(Many dynamics!!!)



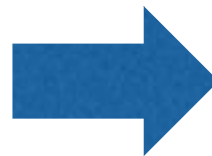
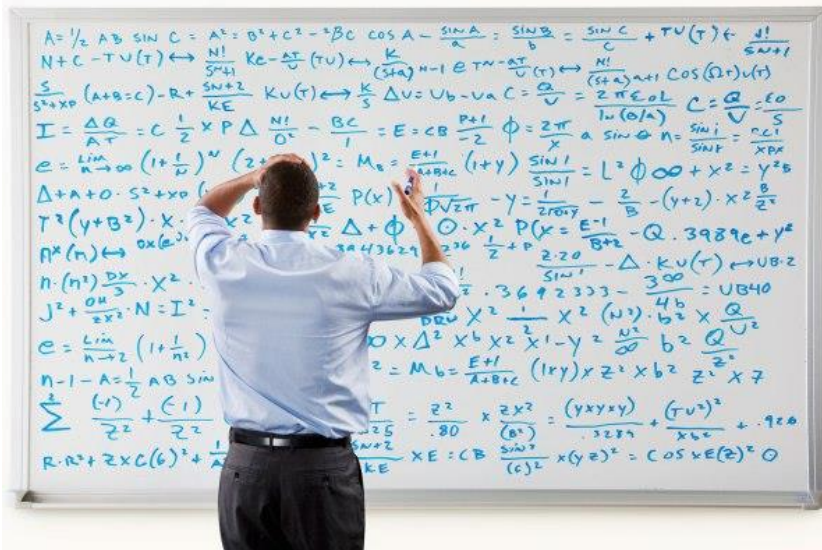
$$\nabla_q [p^T G^{-1} p] dt + \mathcal{N}(0, 2CG dt)$$



ZHUSUAN

Inference in ZhuSuan

Turn painful math deviations into Easy and Intuitive (Probabilistic) Programming



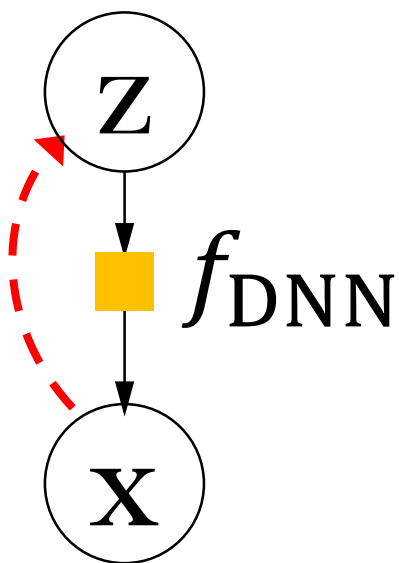
```
0110011001111000101101100011000110110
011000110110011000110110011001101101
Computers are good
at following instructions,
but not
at reading your mind.
01100110110011001100110011001101101
0110011001111000101101100011000110110
~ Donald Knuth
0110001101100110001101100110001101101
10011000110110011001100110011001100
0110011001111000101101100011000110110
01100011011001100011001100110001101101
```




ZHUSUAN

ZhuSuan

ZhuSuan is a python library for generative models, built upon TensorFlow.



Unlike existing DL libraries, which are mainly for supervised tasks, ZhuSuan is featured for:

- its **deep root into Bayesian Inference**
- supporting various kinds of generative models: traditional **hierarchical Bayesian models** & recent **deep generative models**.

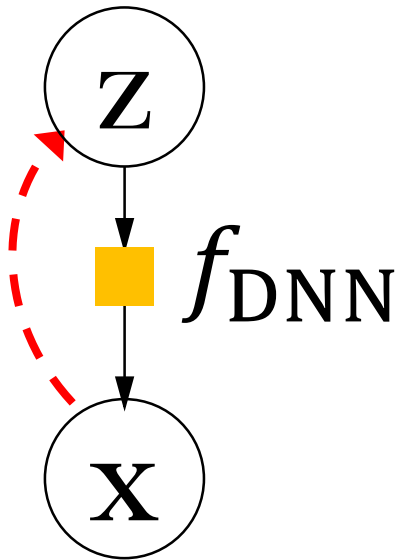


ZHUSUAN

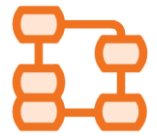
ZhuSuan

With **ZhuSuan**, users can enjoy

- **powerful fitting** and **multi-GPU training** of deep learning



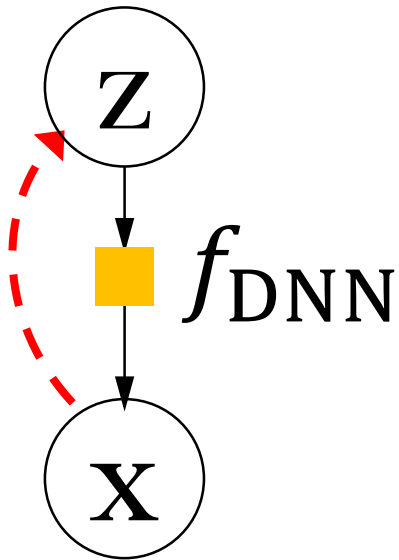
- while at the same time they can use **generative models** to
 - ✓ model the **complex world**
 - ✓ exploit **unlabeled data**
 - ✓ deal with **uncertainty** by performing principled Bayesian inference
 - ✓ **generate** new samples



ZHUSUAN

Model Primitives: BayesianNet

- A DAG representing a **Bayesian Network**
- Two types of nodes:
 - **Deterministic nodes**: Can be composed of **any Tensorflow operations**.
 - **Stochastic nodes**: Use **StochasticTensor**'s from ZhuSuan's library.
- Start a BayesianNet environment

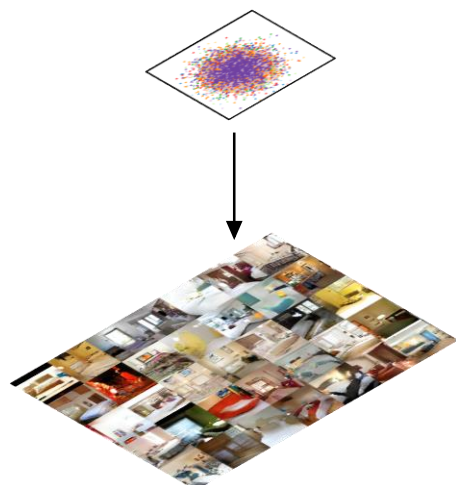


```
import zhusuan as zs
with zs.BayesianNet() as model:
    # build the model
```



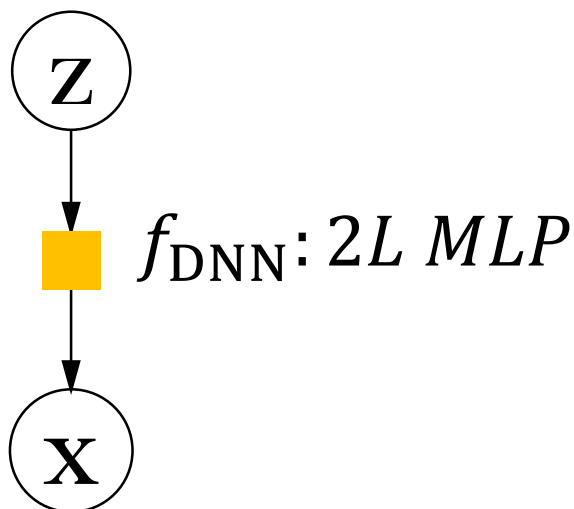
ZHUSUAN

Example: Variational Autoencoders



$$z \sim \mathcal{N}(z|0, I)$$
$$x_{logits} = f_{NN}(z)$$
$$x \sim \text{Bernoulli}(x|\text{sigmoid}(x_{logits}))$$

```
import tensorflow as tf
from tensorflow.contrib import layers
import zhusuan as zs
```



```
with zs.BayesianNet() as model:
    z_mean = tf.zeros([n, n_z])
    z_logstd = tf.zeros([n, n_z])
    z = zs.Normal('z', z_mean, z_logstd)
    h = layers.fully_connected(z, 500)
    x_logits = layers.fully_connected(
        h, n_x, activation_fn=None)
    x = zs.Bernoulli('x', x_logits)
```



ZHUSUAN

Variational Inference in ZhuSuan

```
with zs.BayesianNet() as variational:  
    # build variational ...  
qz_samples, log_qz = variational.query(  
    'z', outputs=True, local_log_prob=True)
```

⊙ Build variational posterior
as BayesianNet

```
lower_bound = zs.sgvb(log_joint, observed={'x': x},  
    latent={'z': [qz_samples, log_qz]})
```

⊙ Call variational objectives

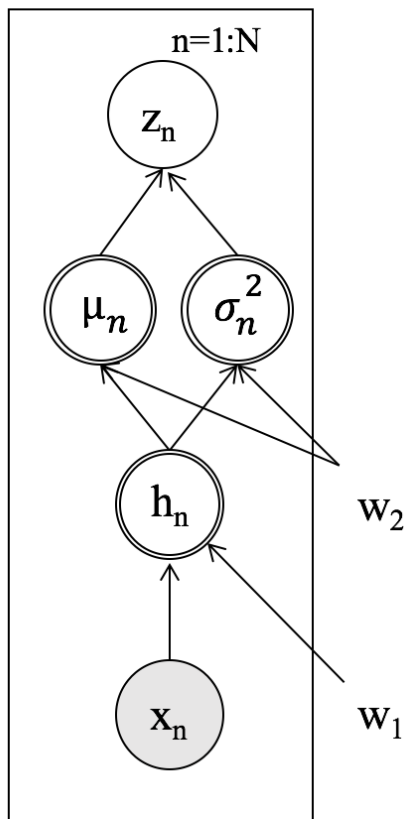
```
optimizer = tf.train.AdamOptimizer(learning_rate=0.001)  
run_op = optimizer.minimize(-lower_bound)  
With tf.Session() as sess:  
    for iter in range(iters):  
        sess.run(run_op)
```

⊙ Run **gradient descent!**



ZHUSUAN

Example: Variational Autoencoders



Structured variational posterior as a BayesianNet ALSO.

```
import tensorflow as tf
from tensorflow.contrib import layers
import zhusuan as zs

with zs.BayesianNet() as variational:
    x = tf.placeholder([None, n_x], tf.float32)
    h = layers.fully_connected(x, 500)
    z_mean = layers.fully_connected(
        h, n_z, activation_fn=None)
    z_logstd = layers.fully_connected(
        h, n_z, activation_fn=None)
    z = zs.Normal('z', z_mean, z_logstd)
```



ZHUSUAN

Variational Inference Algorithms

ZhuSuan supports a broad class of variational objectives, ranging from widely used evidence lower bounds to recent state-of-arts.

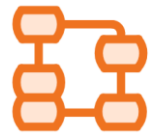
Works for continuous latent variables

- ▣ **zs.sgvb**: Stochastic gradient variational Bayes.
- ▣ **zs.iwae**: Importance weighted lower bounds.

Works for both continuous and discrete latent variables

- ▣ **zs.nvil**: Variance reduced score function estimator/REINFORCE.
- ▣ **zs.vimco**: Variance reduced multi-sample score function estimator.

* This is like optimization algorithms (SGD, momentum, Adam, etc.) in deep learning software. Users need not dive into the technical details of these algorithms because ZhuSuan provides easy-to-use APIs for users to directly try on their generative models.



ZHUSUAN

HMC like a TensorFlow optimizer

```
# like creating the variable to optimize over.
```

```
z = tf.Variable(0.)
```

- ⦿ Create the variable to store samples

```
# like optimizer = tf.train.AdamOptimizer(...)
```

```
hmc = zs.HMC(step_size=1e-3, n_leapfrogs=10)
```

- ⦿ Initialize **HMC**

```
# like optimize_op = optimizer.minimize(...)
```

```
sample_op, hmc_info = hmc.sample(  
    log_joint, observed={'x': x}, latent={'z': z})
```

- ⦿ Call **sample()** method to return a sample operation

```
with tf.Session() as sess:
```

```
    for iter in range(iters):
```

```
        # like sess.run(optimize_op)
```

```
        _ = sess.run(sample_op)
```

- ⦿ Run the sample operation **like an optimizer!**

Applications: ZhuSuan as a Research Platform



ZHUSUAN

ZhuSuan is featured for both Bayesian Statistics and Deep Learning. **State-of-the-Art** models can be found in ZhuSuan's examples.

- ◆ Bayesian Logistic Regression
- ◆ Bayesian Neural Nets for Multivariate Regression
- ◆ (Convolutional) Variational Autoencoders (VAE)
- ◆ Semi-supervised learning for images with VAEs
- ◆ Deep Sigmoid Belief Networks
- ◆ Generative Adversarial Networks (GAN)
- ◆ Gaussian processes (GPs)
- ◆ Topic Models
- ◆ More to come ...



ZHUSUAN

ZhuSuan: GitHub Page

thu-ml / zhusuan

Watch

107

★ Star

1,103

🍴 Fork

219

<> Code

🔔 Issues 5

🔗 Pull requests 4

📁 Projects 0

📊 Insights

Branch: master ▾

zhusuan / docs / index.rst

📖 README.md



Jiaxin Change README to markdown for displaying the logo.

1 contributor

86 lines (63 sloc) | 2.49 KB

Welcome to ZhuSuan

ZhuSuan is a python probabilistic programming library for advantages of Bayesian methods and deep learning. ZhuSuan which are mainly designed for deterministic neural network primitives and algorithms for building probabilistic models algorithms include:

- Variational inference with programmable variational pc (SGVB, REINFORCE, VIMCO, etc.).
- Importance sampling for learning and evaluating models, with programmable proposals.
- Hamiltonian Monte Carlo (HMC) with parallel chains, and optional automatic parameter tuning.

ZhuSuan

ZhuSuan is a python library for **Generative Models**, built upon Tensorflow. Unlike existing deep learning libraries, which are mainly designed for supervised tasks, ZhuSuan is featured for its deep root into Bayesian Inference, thus supporting various kinds of generative models: both the traditional **hierarchical Bayesian models** and recent **deep generative models**.

With ZhuSuan, users can enjoy powerful fitting and multi-GPU training of deep learning, while at the same time they can use generative models to model the complex world, exploit unlabeled data and deal with uncertainty by performing principled Bayesian inference.

Supported Inference

(Stochastic) Variational Inference (VI & SVI)

- Kinds of variational posteriors we support:
 - **Mean-field** posterior: Fully-factorized.
 - **Structured** posterior: With user specified dependencies.
- Variational objectives we support:
 - **SGVB**: Stochastic gradient variational Bayes
 - **IWAE**: Importance weighted objectives
 - **NVIL**: Score function estimator with variance reduction
 - **VIMCO**: Multi-sample score function estimator with variance



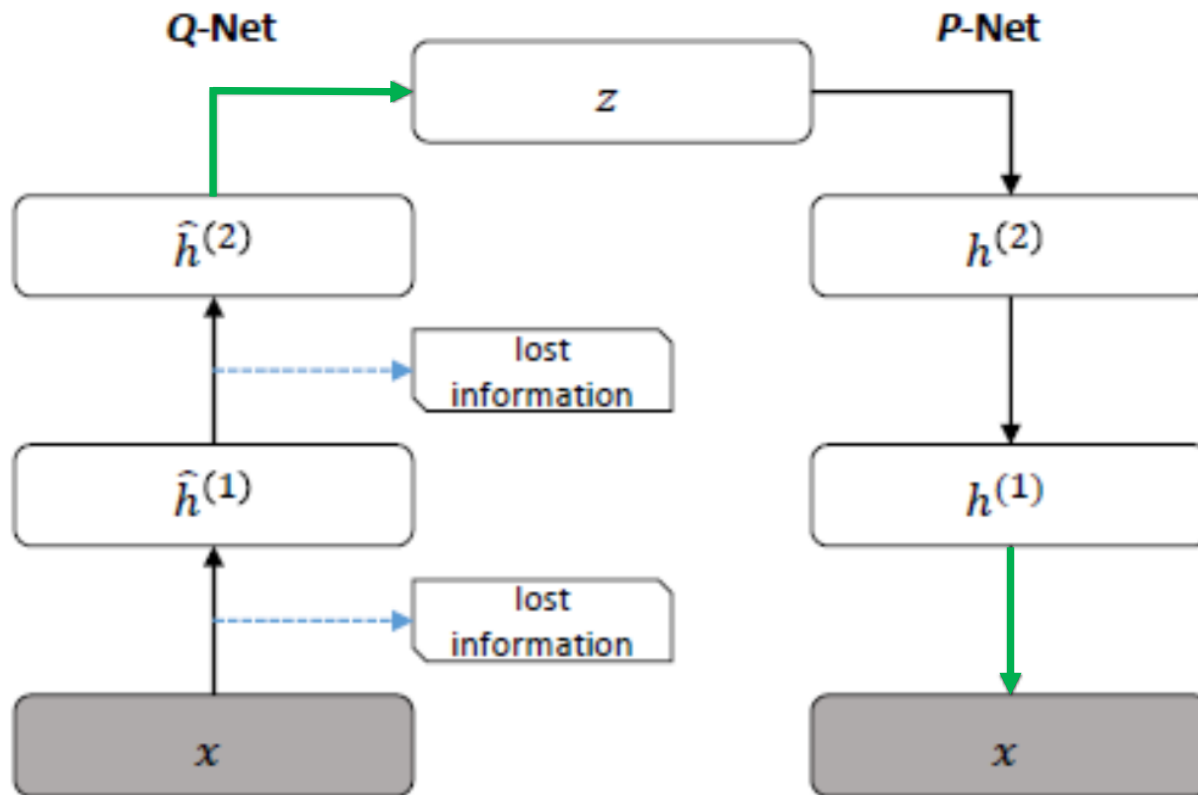
github.com/thu-ml/zhusuan

Contributions & Stars
Welcome!

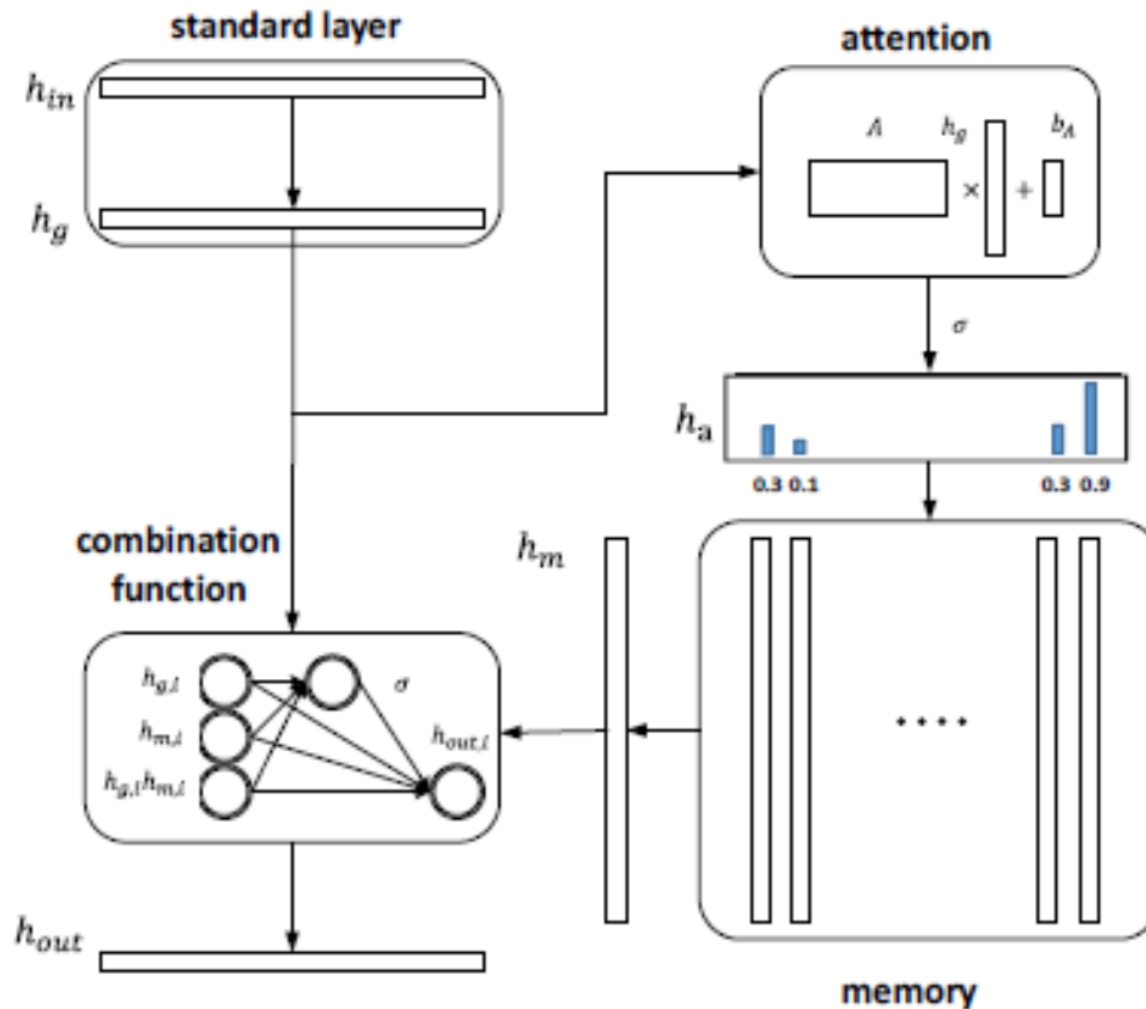
More Applications

Symmetric Q-P Network

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

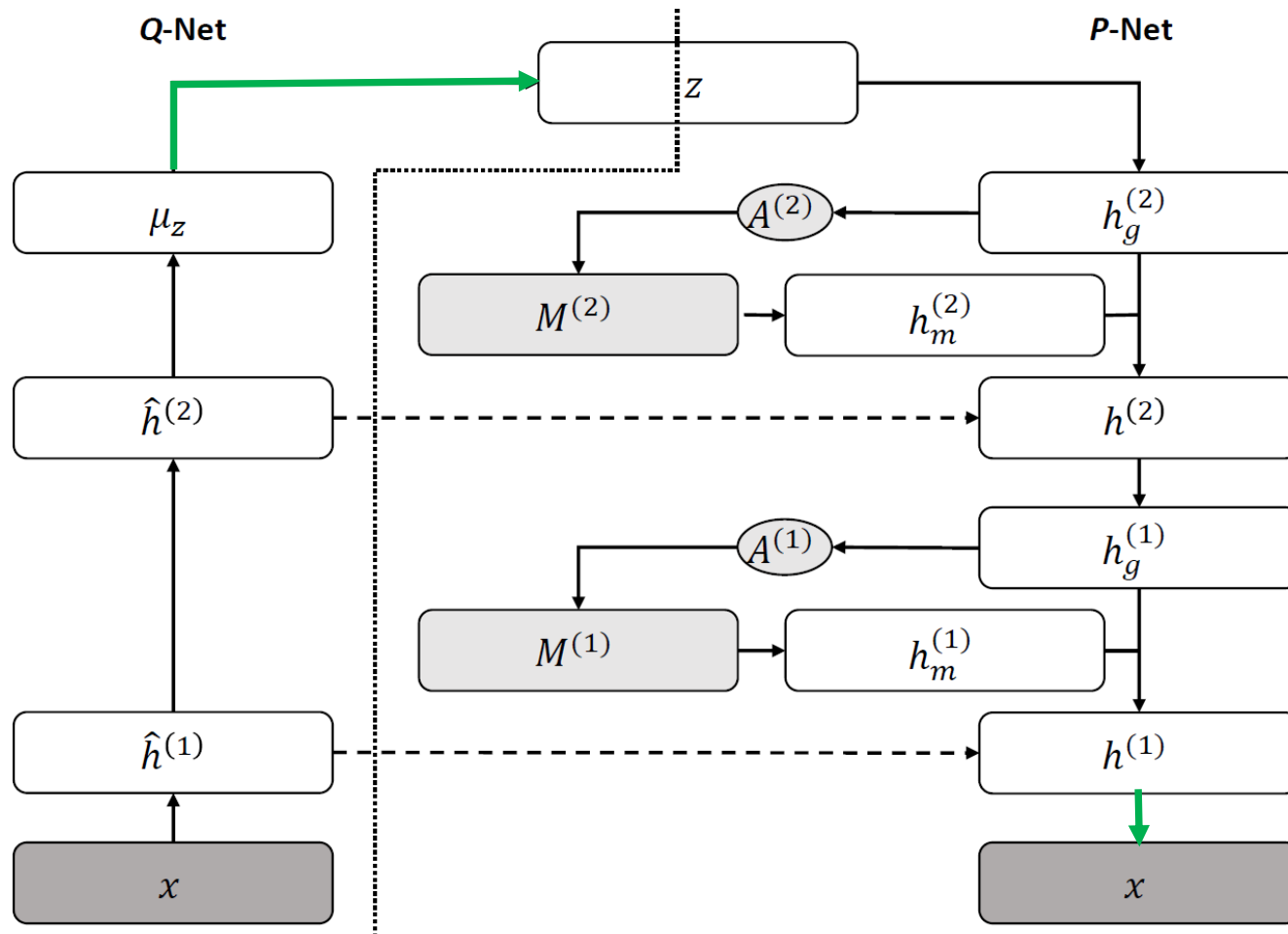


A Layer with Memory and Attention



A Stacked Deep Model with Memory

◆ Asymmetric architecture



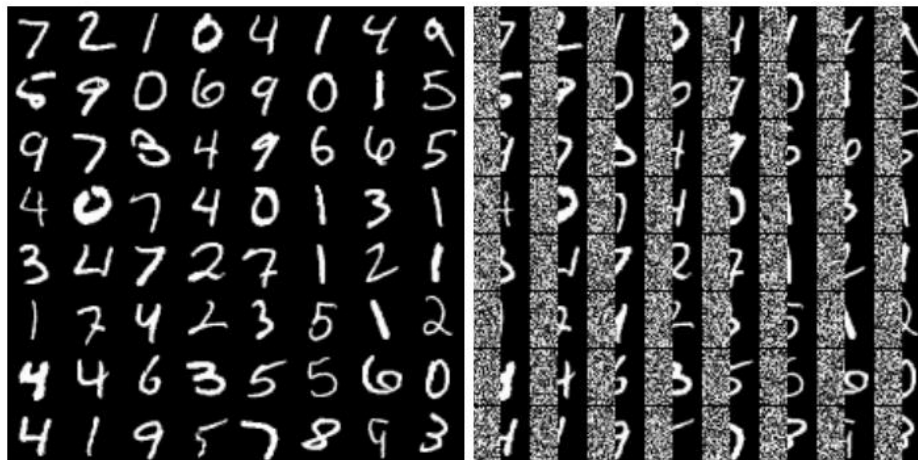
Some Results

◆ Density estimation

MODELS	MNIST	OCR-LETTERS
<i>VAE</i>	-85.69	-30.09
<i>MEM-VAE(ours)</i>	-84.41	-29.09
<i>IWAE-5</i>	-84.43	-28.69
<i>MEM-IWAE-5(ours)</i>	-83.26	-27.65
<i>IWAE-50</i>	-83.58	-27.60
<i>MEM-IWAE-50(ours)</i>	-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

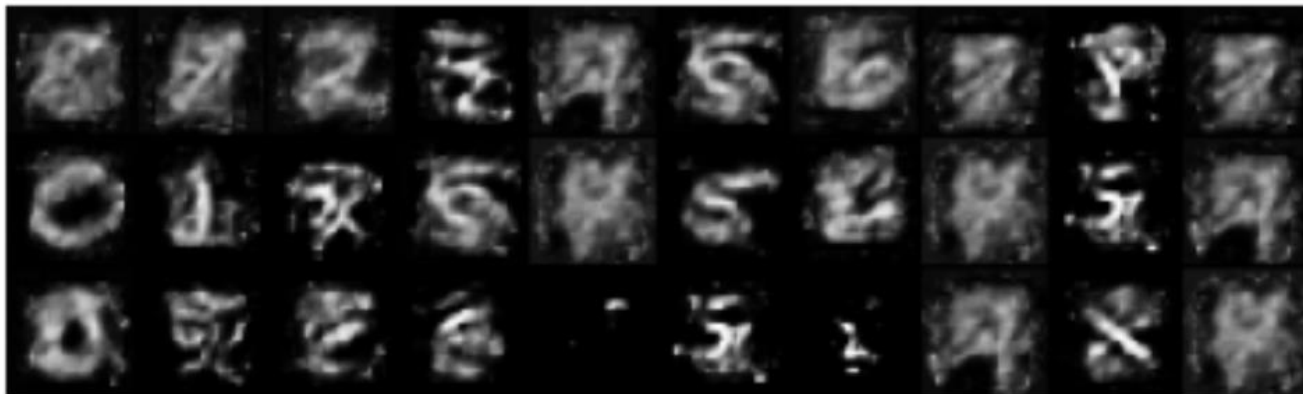
(d) Results of MEM-VAE

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/>