

# Deep Latent-Variable Models of Natural Language

Yoon Kim, Sam Wiseman, Alexander Rush



Tutorial 2018

<https://github.com/harvardnlp/DeepLatentNLP>

Introduction

Goals

Background

Models

Variational  
Objective

Inference  
Strategies

Advanced Topics

Case Studies

Conclusion

References

# ① Introduction

Goals

Background

## ② Models

## ③ Variational Objective

## ④ Inference Strategies

## ⑤ Advanced Topics

## ⑥ Case Studies

## ① Introduction

### Goals

### Background

## ② Models

## ③ Variational Objective

## ④ Inference Strategies

## ⑤ Advanced Topics

## ⑥ Case Studies

## Goal of Latent-Variable Modeling

Probabilistic models provide a declarative language for specifying prior knowledge and structural relationships in the context of unknown variables.

Makes it easy to specify:

- Known interactions in the data
- Uncertainty about unknown factors
- Constraints on model properties

## Goal of Latent-Variable Modeling

Probabilistic models provide a declarative language for specifying prior knowledge and structural relationships in the context of unknown variables.

Makes it easy to specify:

- Known interactions in the data
- Uncertainty about unknown factors
- Constraints on model properties

## Latent-Variable Modeling in NLP

Long and rich history of latent-variable models of natural language.

Major successes include, among many others:

- Statistical alignment for translation
- Document clustering and topic modeling
- Unsupervised part-of-speech tagging and parsing

## Goals of Deep Learning

Toolbox of methods for learning rich, non-linear data representations through numerical optimization.

Makes it easy to fit:

- Highly-flexible predictive models
- Transferable feature representations
- Structurally-aligned network architectures

## Goals of Deep Learning

Toolbox of methods for learning rich, non-linear data representations through numerical optimization.

Makes it easy to fit:

- Highly-flexible predictive models
- Transferable feature representations
- Structurally-aligned network architectures



## Deep Learning in NLP

Current dominant paradigm for NLP.

Major successes include, among many others:

- Text classification
- Neural machine translation
- NLU Tasks (QA, NLI, etc)

## Tutorial: Deep Latent-Variable Models for NLP

### Introduction

#### Goals

#### Background

### Models

#### Variational Objective

#### Inference Strategies

### Advanced Topics

#### Case Studies

#### Conclusion

#### References

- How should a contemporary ML/NLP researcher reason about latent-variables?
- What unique challenges come from modeling text with latent variables?
- What techniques have been explored and shown to be effective in recent papers?

We explore these through the lens of *variational inference*.

## Tutorial: Deep Latent-Variable Models for NLP

### Introduction

#### Goals

#### Background

### Models

#### Variational Objective

#### Inference Strategies

### Advanced Topics

#### Case Studies

#### Conclusion

#### References

- How should a contemporary ML/NLP researcher reason about latent-variables?
- What unique challenges come from modeling text with latent variables?
- What techniques have been explored and shown to be effective in recent papers?

We explore these through the lens of *variational inference*.

## Tutorial Take-Aways

- 1 A collection of deep latent-variable **models** for NLP
- 2 An understanding of a **variational** objective
- 3 A toolkit of algorithms for **optimization**
- 4 A formal guide to **advanced** techniques
- 5 A survey of example **applications**
- 6 Code samples and techniques for **practical** use

## Tutorial Non-Objectives

Not covered (for time, not relevance):

- Many classical latent-variable approaches.
- Undirected graphical models such as MRFs
- Non-likelihood based models such as GANs
- Sampling-based inference such as MCMC.
- Details of deep learning architectures.

## ① Introduction

Goals

Background

## ② Models

## ③ Variational Objective

## ④ Inference Strategies

## ⑤ Advanced Topics

## ⑥ Case Studies

## What are deep networks?

Deep networks are parameterized non-linear functions; They transform input  $\mathbf{z}$  into features  $\mathbf{h}$  using parameters  $\pi$ .

Important examples: The multilayer perceptron,

$$\mathbf{h} = \text{MLP}(\mathbf{z}; \pi) = \mathbf{V}\sigma(\mathbf{W}\mathbf{z} + \mathbf{b}) + \mathbf{a} \quad \pi = \{\mathbf{V}, \mathbf{W}, \mathbf{a}, \mathbf{b}\},$$

The recurrent neural network, which maps a sequence of inputs  $\mathbf{z}_{1:T}$  into a sequence of features  $\mathbf{h}_{1:T}$ ,

$$\mathbf{h}_t = \text{RNN}(\mathbf{h}_{t-1}, \mathbf{z}_t; \pi) = \sigma(\mathbf{U}\mathbf{z}_t + \mathbf{V}\mathbf{h}_{t-1} + \mathbf{b}) \quad \pi = \{\mathbf{V}, \mathbf{U}, \mathbf{b}\}$$

## What are deep networks?

## Introduction

## Goals

## Background

## Models

Variational  
ObjectiveInference  
Strategies

## Advanced Topics

## Case Studies

## Conclusion

## References

Deep networks are parameterized non-linear functions; They transform input  $\mathbf{z}$  into features  $\mathbf{h}$  using parameters  $\pi$ .

Important examples: The multilayer perceptron,

$$\mathbf{h} = \text{MLP}(\mathbf{z}; \pi) = \mathbf{V}\sigma(\mathbf{W}\mathbf{z} + \mathbf{b}) + \mathbf{a} \quad \pi = \{\mathbf{V}, \mathbf{W}, \mathbf{a}, \mathbf{b}\},$$

The recurrent neural network, which maps a sequence of inputs  $\mathbf{z}_{1:T}$  into a sequence of features  $\mathbf{h}_{1:T}$ ,

$$\mathbf{h}_t = \text{RNN}(\mathbf{h}_{t-1}, \mathbf{z}_t; \pi) = \sigma(\mathbf{U}\mathbf{z}_t + \mathbf{V}\mathbf{h}_{t-1} + \mathbf{b}) \quad \pi = \{\mathbf{V}, \mathbf{U}, \mathbf{b}\}$$



## What are latent variable models?

Latent variable models give us a joint distribution

$$p(x, z; \theta).$$

- $x$  is our observed data
- $z$  is a collection of latent variables
- $\theta$  are the deterministic parameters of the model, such as the neural network parameters
- Data consists of  $N$  i.i.d samples,

$$p(x^{(1:N)}, z^{(1:N)}; \theta) = \prod_{n=1}^N p(x^{(n)} | z^{(n)}; \theta) p(z^{(n)}; \theta).$$

## What are latent variable models?

Latent variable models give us a joint distribution

$$p(x, z; \theta).$$

- $x$  is our observed data
- $z$  is a collection of latent variables
- $\theta$  are the deterministic parameters of the model, such as the neural network parameters
- Data consists of  $N$  i.i.d samples,

$$p(x^{(1:N)}, z^{(1:N)}; \theta) = \prod_{n=1}^N p(x^{(n)} | z^{(n)}; \theta) p(z^{(n)}; \theta).$$

## What are latent variable models?

Latent variable models give us a joint distribution

$$p(x, z; \theta).$$

- $x$  is our observed data
- $z$  is a collection of latent variables
- $\theta$  are the deterministic parameters of the model, such as the neural network parameters
- Data consists of  $N$  i.i.d samples,

$$p(x^{(1:N)}, z^{(1:N)}; \theta) = \prod_{n=1}^N p(x^{(n)} | z^{(n)}; \theta) p(z^{(n)}; \theta).$$

## Probabilistic Graphical Models

## Introduction

## Goals

## Background

## Models

Variational  
ObjectiveInference  
Strategies

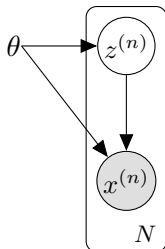
## Advanced Topics

## Case Studies

## Conclusion

## References

- A directed PGM shows the conditional independence structure.
- By chain rule, latent variable model over observations can be represented as,



$$p(x^{(1:N)}, z^{(1:N)}; \theta) = \prod_{n=1}^N p(x^{(n)} | z^{(n)}; \theta) p(z^{(n)}; \theta)$$

- Specific models may factor further.

## Posterior Inference

For models  $p(x, z; \theta)$ , we'll be interested in the *posterior* over latent variables  $z$ :

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

Why?

- $z$  will often represent interesting information about our data (e.g., the cluster  $x^{(n)}$  lives in, how similar  $x^{(n)}$  and  $x^{(n+1)}$  are).
- Learning the parameters  $\theta$  of the model often requires calculating posteriors as a subroutine. Intuition: if I know likely  $z^{(n)}$  for  $x^{(n)}$ , I can learn by maximizing  $p(x^{(n)} | z^{(n)}; \theta)$ .

Introduction

Goals

Background

Models

Variational  
ObjectiveInference  
Strategies

Advanced Topics

Case Studies

Conclusion

References

## Posterior Inference

For models  $p(x, z; \theta)$ , we'll be interested in the *posterior* over latent variables  $z$ :

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

Why?

- $z$  will often represent interesting information about our data (e.g., the cluster  $x^{(n)}$  lives in, how similar  $x^{(n)}$  and  $x^{(n+1)}$  are).
- Learning the parameters  $\theta$  of the model often requires calculating posteriors as a subroutine. Intuition: if I know likely  $z^{(n)}$  for  $x^{(n)}$ , I can learn by maximizing  $p(x^{(n)} | z^{(n)}; \theta)$ .

Introduction

Goals

Background

Models

Variational  
ObjectiveInference  
Strategies

Advanced Topics

Case Studies

Conclusion

References

## Problem Statement: Two Views

Deep Models & LV Models are naturally **complementary**:

- Rich function approximators with modular parts.
- Declarative methods for specifying model constraints.

Deep Models & LV Models are frustratingly **incompatible**:

- Deep networks make posterior inference intractable.
- Latent variable objectives complicate backpropagation.

## Problem Statement: Two Views

Deep Models & LV Models are naturally **complementary**:

- Rich function approximators with modular parts.
- Declarative methods for specifying model constraints.

Deep Models & LV Models are frustratingly **incompatible**:

- Deep networks make posterior inference intractable.
- Latent variable objectives complicate backpropagation.



## ① Introduction

## ② Models

Discrete Models

Continuous Models

Structured Models

## ③ Variational Objective

## ④ Inference Strategies

## ⑤ Advanced Topics

## ⑥ Case Studies

## A Language Model

Our goal is to model a sentence,  $x_1 \dots x_T$ .

Context: RNN language models are remarkable at this task,

$$x_{1:T} \sim \text{RNNLM}(x_{1:T}; \theta).$$

Defined as,

$$p(x_{1:T}) = \prod_{t=1}^T p(x_t | x_{<t}) = \prod_{t=1}^T \text{softmax}(\mathbf{W} \mathbf{h}_t)_{x_t}$$

where  $\mathbf{h}_t = \text{RNN}(\mathbf{h}_{t-1}, \mathbf{x}_{t-1}; \theta)$



## A Language Model

Our goal is to model a sentence,  $x_1 \dots x_T$ .

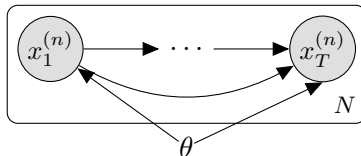
Context: RNN language models are remarkable at this task,

$$x_{1:T} \sim \text{RNNLM}(x_{1:T}; \theta).$$

Defined as,

$$p(x_{1:T}) = \prod_{t=1}^T p(x_t | x_{<t}) = \prod_{t=1}^T \text{softmax}(\mathbf{W} \mathbf{h}_t)_{x_t}$$

where  $\mathbf{h}_t = \text{RNN}(\mathbf{h}_{t-1}, \mathbf{x}_{t-1}; \theta)$



## A Collection of Model Archetypes

Focus: semi-supervised or unsupervised learning, i.e. don't just learn the probabilities, but the process. Range of choices in selecting  $z$

- 1 Discrete LVs  $z$  (*Clustering*)
- 2 Continuous LVs  $z$  (*Dimensionality reduction*)
- 3 Structured LVs  $z$  (*Structured learning*)

## A Collection of Model Archetypes

Focus: semi-supervised or unsupervised learning, i.e. don't just learn the probabilities, but the process. Range of choices in selecting  $z$

- 1 Discrete LVs  $z$  (*Clustering*)
- 2 Continuous LVs  $z$  (*Dimensionality reduction*)
- 3 Structured LVs  $z$  (*Structured learning*)

## ① Introduction

## ② Models

Discrete Models

Continuous Models

Structured Models

## ③ Variational Objective

## ④ Inference Strategies

## ⑤ Advanced Topics

## ⑥ Case Studies

## Model 1: Discrete Clustering

## Introduction

## Models

## Discrete Models

## Continuous Models

## Structured Models

## Variational

## Objective

## Inference

## Strategies

## Advanced Topics

## Case Studies

## Conclusion

## References

## Inference Process:

In an old house in Paris that was  
covered with vines lived twelve little  
girls in two straight lines.

Cluster 23

Discrete latent variable models induce a clustering over sentences  $x^{(n)}$ .

## Example uses:

- Document/sentence clustering [Willett 1988; Aggarwal and Zhai 2012].
- Mixture of expert text generation models [Jacobs et al. 1991; Garmash and Monz 2016; Lee et al. 2016]

## Model 1: Discrete Clustering

## Introduction

## Models

## Discrete Models

## Continuous Models

## Structured Models

## Variational

## Objective

## Inference

## Strategies

## Advanced Topics

## Case Studies

## Conclusion

## References

Inference Process:

In an old house in Paris that was  
covered with vines lived twelve little  
girls in two straight lines.

Cluster 23

Discrete latent variable models induce a clustering over sentences  $x^{(n)}$ .

Example uses:

- Document/sentence clustering [Willett 1988; Aggarwal and Zhai 2012].
- Mixture of expert text generation models [Jacobs et al. 1991; Garmash and Monz 2016; Lee et al. 2016]



## Model 1: Discrete - Mixture of Categoricals

## Introduction

## Models

## Discrete Models

## Continuous Models

## Structured Models

## Variational

## Objective

## Inference

## Strategies

## Advanced Topics

## Case Studies

## Conclusion

## References

Generative process:

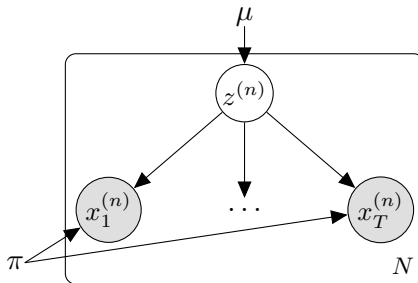
- 1 Draw cluster  $z \in \{1, \dots, K\}$  from a categorical with param  $\mu$ .
- 2 Draw word  $T$  words  $x_t$  from a categorical with word distribution  $\pi_z$ .

Parameters:  $\theta = \{\mu \in \Delta^{K-1}, K \times V \text{ stochastic matrix } \pi\}$

Gives rise to the "Naive Bayes" distribution:

$$\begin{aligned} p(x, z; \theta) &= p(z; \mu) \times p(x \mid z; \pi) = \mu_z \times \prod_{t=1}^T \text{Cat}(x_t; \pi) \\ &= \mu_z \times \prod_{t=1}^T \pi_{z, x_t} \end{aligned}$$

## Model 1: Graphical Model View



$$\begin{aligned} \prod_{n=1}^N p(x^{(n)}, z^{(n)}; \mu, \pi) &= \prod_{n=1}^N p(z^{(n)}; \mu) \times p(x^{(n)} | z^{(n)}; \pi) \\ &= \prod_{n=1}^N \mu_{z^{(n)}} \times \prod_{t=1}^T \pi_{z^{(n)}, x_t^{(n)}} \end{aligned}$$

## Deep Model 1: Discrete - Mixture of RNNs

## Introduction

## Models

## Discrete Models

## Continuous Models

## Structured Models

## Variational

## Objective

## Inference

## Strategies

## Advanced Topics

## Case Studies

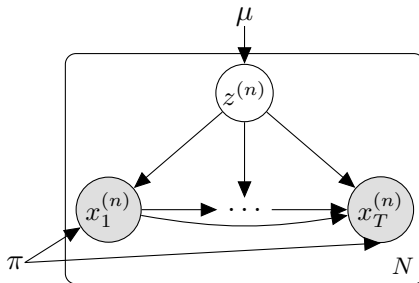
## Conclusion

## References

Generative process:

- 1 Draw cluster  $z \in \{1, \dots, K\}$  from a categorical.
- 2 Draw words  $x_{1:T}$  from RNNLM with parameters  $\pi_z$ .

$$p(x, z; \theta) = \mu_z \times \text{RNNLM}(x_{1:T}; \pi_z)$$



## Difference Between Models

- Dependence structure:
  - Mixture of Categoricals:  $x_t$  independent of other  $x_j$  given  $z$ .
  - Mixture of RNNs:  $x_t$  fully dependent.

Interesting question: how will this affect the learned latent space?

- Number of parameters:
  - Mixture of Categoricals:  $K \times V$ .
  - Mixture of RNNs:  $K \times d^2 + V \times d$  with RNN with  $d$  hidden dims.

## Difference Between Models

- Dependence structure:
  - Mixture of Categoricals:  $x_t$  independent of other  $x_j$  given  $z$ .
  - Mixture of RNNs:  $x_t$  fully dependent.

Interesting question: how will this affect the learned latent space?

- Number of parameters:
  - Mixture of Categoricals:  $K \times V$ .
  - Mixture of RNNs:  $K \times d^2 + V \times d$  with RNN with  $d$  hidden dims.

## Posterior Inference

For both discrete models, can apply Bayes' rule:

$$\begin{aligned} p(z | x; \theta) &= \frac{p(z) \times p(x | z)}{p(x)} \\ &= \frac{p(z) \times p(x | z)}{\sum_{k=1}^K p(z=k) \times p(x | z=k)} \end{aligned}$$

- For mixture of categoricals, posterior uses word counts under each  $\pi_k$ .
- For mixture of RNNs, posterior requires running RNN over  $x$  for each  $k$ .

Introduction

Models

Discrete Models

Continuous Models

Structured Models

Variational

Objective

Inference

Strategies

Advanced Topics

Case Studies

Conclusion

References

## Posterior Inference

For both discrete models, can apply Bayes' rule:

$$\begin{aligned} p(z | x; \theta) &= \frac{p(z) \times p(x | z)}{p(x)} \\ &= \frac{p(z) \times p(x | z)}{\sum_{k=1}^K p(z=k) \times p(x | z=k)} \end{aligned}$$

- For mixture of categoricals, posterior uses word counts under each  $\pi_k$ .
- For mixture of RNNs, posterior requires running RNN over  $x$  for each  $k$ .

## Posterior Inference

For both discrete models, can apply Bayes' rule:

$$\begin{aligned} p(z | x; \theta) &= \frac{p(z) \times p(x | z)}{p(x)} \\ &= \frac{p(z) \times p(x | z)}{\sum_{k=1}^K p(z=k) \times p(x | z=k)} \end{aligned}$$

- For mixture of categoricals, posterior uses word counts under each  $\pi_k$ .
- For mixture of RNNs, posterior requires running RNN over  $x$  for each  $k$ .



## ① Introduction

## ② Models

Discrete Models

**Continuous Models**

Structured Models

## ③ Variational Objective

## ④ Inference Strategies

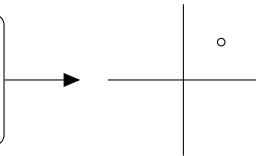
## ⑤ Advanced Topics

## ⑥ Case Studies

## Model 2: Continuous / Dimensionality Reduction

## Inference Process:

In an old house in Paris that was covered with vines lived twelve little girls in two straight lines.



Find a lower-dimensional, well-behaved continuous representation of a sentence.

Latent variables in  $\mathbb{R}^d$  make distance/similarity easy. Examples:

- Recent work in text generation assumes a latent vector per sentence [Bowman et al. 2016; Yang et al. 2017; Hu et al. 2017].
- Certain sentence embeddings (e.g., Skip-Thought vectors [Kiros et al. 2015]) can be interpreted in this way.

## Model 2: Continuous / Dimensionality Reduction

## Introduction

## Models

Discrete Models

**Continuous Models**

Structured Models

## Variational

## Objective

## Inference

## Strategies

## Advanced Topics

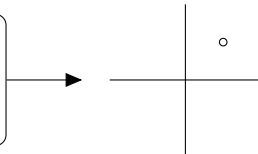
## Case Studies

## Conclusion

## References

Inference Process:

In an old house in Paris that was  
covered with vines lived twelve little  
girls in two straight lines.



Find a lower-dimensional, well-behaved continuous representation of a sentence.

Latent variables in  $\mathbb{R}^d$  make distance/similarity easy. Examples:

- Recent work in text generation assumes a latent vector per sentence [Bowman et al. 2016; Yang et al. 2017; Hu et al. 2017].
- Certain sentence embeddings (e.g., Skip-Thought vectors [Kiros et al. 2015]) can be interpreted in this way.

## Model 2: Continuous "Mixture"

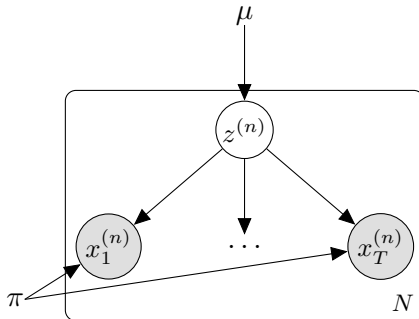
Generative Process:

- 1 Draw continuous latent variable  $\mathbf{z}$  from Normal with param  $\mu$ .
- 2 For each  $t$ , draw word  $x_t$  from categorical with param  $\text{softmax}(\mathbf{W}\mathbf{z})$ .

Parameters:  $\theta = \{\mu \in \mathbb{R}^d, \pi\}, \pi = \{\mathbf{W} \in \mathbb{R}^{V \times d}\}$

Intuition:  $\mu$  is a global distribution,  $\mathbf{z}$  captures local word distribution of the sentence.

## Graphical Model View



Gives rise to the joint distribution:

$$\prod_{n=1}^N p(x^{(n)}, z^{(n)}; \theta) = \prod_{n=1}^N p(z^{(n)}; \mu) \times p(x^{(n)} | z^{(n)}; \pi)$$

## Deep Model 2: Continuous "Mixture" of RNNs

Generative Process:

- 1 Draw latent variable  $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{I})$ .
- 2 Draw each token  $x_t$  from a conditional RNNLM.

RNN is also conditioned on latent  $\mathbf{z}$ ,

$$\begin{aligned} p(x, \mathbf{z}; \pi, \boldsymbol{\mu}, \mathbf{I}) &= p(\mathbf{z}; \boldsymbol{\mu}, \mathbf{I}) \times p(x \mid \mathbf{z}; \pi) \\ &= \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \mathbf{I}) \times \text{CRNNLM}(x_{1:T}; \pi, \mathbf{z}) \end{aligned}$$

where

$$\begin{aligned} \text{CRNNLM}(x_{1:T}; \pi, \mathbf{z}) &= \prod_{t=1}^T \text{softmax}(\mathbf{W} \mathbf{h}_t)_{x_t} \\ \mathbf{h}_t &= \text{RNN}(\mathbf{h}_{t-1}, [\mathbf{x}_{t-1}; \mathbf{z}]; \pi) \end{aligned}$$

Introduction

Models

Discrete Models

**Continuous Models**

Structured Models

Variational

Objective

Inference

Strategies

Advanced Topics

Case Studies

Conclusion

References

## Deep Model 2: Continuous "Mixture" of RNNs

Generative Process:

- 1 Draw latent variable  $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{I})$ .
- 2 Draw each token  $x_t$  from a conditional RNNLM.

RNN is also conditioned on latent  $\mathbf{z}$ ,

$$\begin{aligned} p(x, \mathbf{z}; \pi, \boldsymbol{\mu}, \mathbf{I}) &= p(\mathbf{z}; \boldsymbol{\mu}, \mathbf{I}) \times p(x \mid \mathbf{z}; \pi) \\ &= \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \mathbf{I}) \times \text{CRNNLM}(x_{1:T}; \pi, \mathbf{z}) \end{aligned}$$

where

$$\begin{aligned} \text{CRNNLM}(x_{1:T}; \pi, \mathbf{z}) &= \prod_{t=1}^T \text{softmax}(\mathbf{W} \mathbf{h}_t)_{x_t} \\ \mathbf{h}_t &= \text{RNN}(\mathbf{h}_{t-1}, [\mathbf{x}_{t-1}; \mathbf{z}]; \pi) \end{aligned}$$

## Deep Model 2: Continuous "Mixture" of RNNs

Generative Process:

- 1 Draw latent variable  $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{I})$ .
- 2 Draw each token  $x_t$  from a conditional RNNLM.

RNN is also conditioned on latent  $\mathbf{z}$ ,

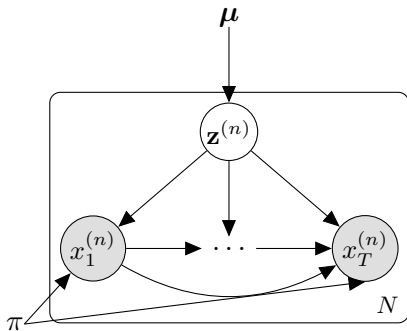
$$\begin{aligned} p(x, \mathbf{z}; \pi, \boldsymbol{\mu}, \mathbf{I}) &= p(\mathbf{z}; \boldsymbol{\mu}, \mathbf{I}) \times p(x \mid \mathbf{z}; \pi) \\ &= \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \mathbf{I}) \times \text{CRNNLM}(x_{1:T}; \pi, \mathbf{z}) \end{aligned}$$

where

$$\begin{aligned} \text{CRNNLM}(x_{1:T}; \pi, \mathbf{z}) &= \prod_{t=1}^T \text{softmax}(\mathbf{W} \mathbf{h}_t)_{x_t} \\ \mathbf{h}_t &= \text{RNN}(\mathbf{h}_{t-1}, [\mathbf{x}_{t-1}; \mathbf{z}]; \pi) \end{aligned}$$



## Graphical Model View



## Posterior Inference

For continuous models, Bayes' rule is harder to compute,

$$p(z | x; \theta) = \frac{p(z; \mu) \times p(x | z; \pi)}{\int_z p(z; \mu) \times p(x | z; \pi) dz}$$

- Shallow and deep Model 2 variants mirror Model 1 variants exactly, but with continuous  $z$ .
- Integral intractable (in general) for both shallow and deep variants.

## Posterior Inference

For continuous models, Bayes' rule is harder to compute,

$$p(z | x; \theta) = \frac{p(z; \mu) \times p(x | z; \pi)}{\int_z p(z; \mu) \times p(x | z; \pi) dz}$$

- Shallow and deep Model 2 variants mirror Model 1 variants exactly, but with continuous  $z$ .
- Integral intractable (in general) for both shallow and deep variants.

## ① Introduction

## ② Models

Discrete Models

Continuous Models

**Structured Models**

## ③ Variational Objective

## ④ Inference Strategies

## ⑤ Advanced Topics

## ⑥ Case Studies

## Model 3: Structure Learning

Inference Process:

In an old house in Paris that was  
covered with vines lived twelve little  
girls in two straight lines.



Structured latent variable models are used to infer unannotated structure:

- Unsupervised POS tagging [Brown et al. 1992; Merialdo 1994; Smith and Eisner 2005]
- Unsupervised dependency parsing [Klein and Manning 2004; Headden III et al. 2009]

Or when structure is useful for *interpreting* our data:

- Segmentation of documents into topical passages [Hearst 1997]
- Alignment [Vogel et al. 1996]

## Model 3: Structured - Hidden Markov Model

Generative Process:

- 1 For each  $t$ , draw  $z_t \in \{1, \dots, K\}$  from a categorical with param  $\mu_{z_{t-1}}$ .
- 2 Draw observed token  $x_t$  from categorical with param  $\pi_{z_t}$ .

Parameters:  $\theta = \{K \times K \text{ stochastic matrix } \mu, K \times V \text{ stochastic matrix } \pi\}$

Gives rise to the joint distribution:

$$\begin{aligned} p(x, z; \theta) &= \prod_{t=1}^T p(z_t | z_{t-1}; \mu_{z_{t-1}}) \times \prod_{t=1}^T p(x_t | z_t; \pi_{z_t}) \\ &= \prod_{t=1}^T \mu_{z_{t-1}, z_t} \times \prod_{t=1}^T \pi_{z_t, x_t} \end{aligned}$$

## Model 3: Structured - Hidden Markov Model

## Introduction

## Models

Discrete Models

Continuous Models

Structured Models

## Variational

## Objective

## Inference

## Strategies

## Advanced Topics

## Case Studies

## Conclusion

## References

Generative Process:

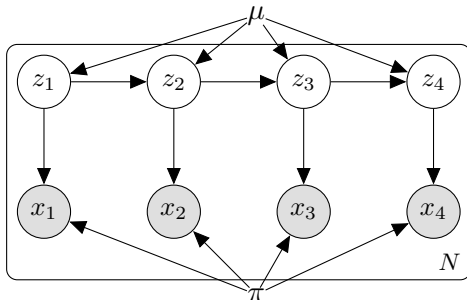
- 1 For each  $t$ , draw  $z_t \in \{1, \dots, K\}$  from a categorical with param  $\mu_{z_{t-1}}$ .
- 2 Draw observed token  $x_t$  from categorical with param  $\pi_{z_t}$ .

Parameters:  $\theta = \{K \times K \text{ stochastic matrix } \mu, K \times V \text{ stochastic matrix } \pi\}$ 

Gives rise to the joint distribution:

$$\begin{aligned} p(x, z; \theta) &= \prod_{t=1}^T p(z_t | z_{t-1}; \mu_{z_{t-1}}) \times \prod_{t=1}^T p(x_t | z_t; \pi_{z_t}) \\ &= \prod_{t=1}^T \mu_{z_{t-1}, z_t} \times \prod_{t=1}^T \pi_{z_t, x_t} \end{aligned}$$

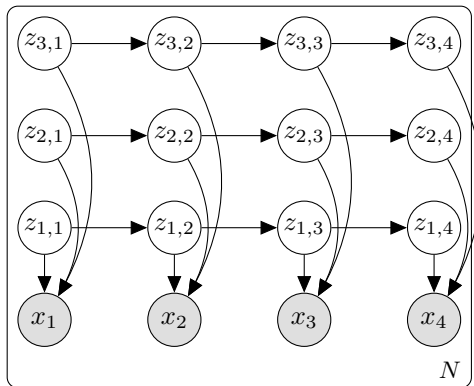
## Graphical Model View



$$\begin{aligned}
 p(x, z; \theta) &= \prod_{t=1}^T p(z_t | z_{t-1}; \mu_{z_{t-1}}) \times \prod_{t=1}^T p(x_t | z_t; \pi_{z_t}) \\
 &= \prod_{t=1}^T \mu_{z_{t-1}, z_t} \times \prod_{t=1}^T \pi_{z_t, x_t}
 \end{aligned}$$



## Further Extension: Factorial HMM



$$p(x, z; \theta) = \prod_{l=1}^L \prod_{t=1}^T p(z_{l,t} | z_{l,t-1}) \times \prod_{t=1}^T p(x_t | z_{1:L,t})$$

## Deep Model 3: Deep HMM

## Introduction

## Models

Discrete Models

Continuous Models

Structured Models

## Variational

## Objective

## Inference

## Strategies

## Advanced Topics

## Case Studies

## Conclusion

## References

Parameterize transition and emission distributions with neural networks (c.f., Tran et al. [2016])

- Model transition distribution as

$$p(z_t | z_{t-1}) = \text{softmax}(\text{MLP}(z_{t-1}; \mu))$$

- Model emission distribution as

$$p(x_t | z_t) = \text{softmax}(\text{MLP}(z_t; \pi))$$

**Note:**  $K \times K$  transition parameters for standard HMM vs.  $O(K \times d + d^2)$  for deep version.

## Deep Model 3: Deep HMM

## Introduction

## Models

Discrete Models

Continuous Models

Structured Models

## Variational

## Objective

## Inference

## Strategies

## Advanced Topics

## Case Studies

## Conclusion

## References

Parameterize transition and emission distributions with neural networks (c.f., Tran et al. [2016])

- Model transition distribution as

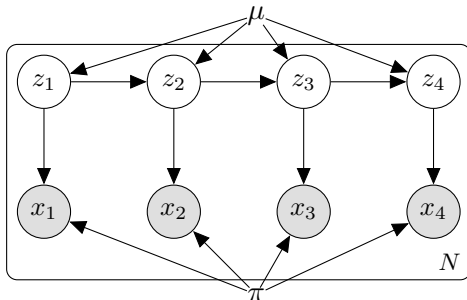
$$p(z_t | z_{t-1}) = \text{softmax}(\text{MLP}(z_{t-1}; \mu))$$

- Model emission distribution as

$$p(x_t | z_t) = \text{softmax}(\text{MLP}(z_t; \pi))$$

**Note:**  $K \times K$  transition parameters for standard HMM vs.  $O(K \times d + d^2)$  for deep version.

## Graphical Model View



$$\begin{aligned}
 p(x, z; \theta) &= \prod_{t=1}^T p(z_t | z_{t-1}; \mu_{z_{t-1}}) \times \prod_{t=1}^T p(x_t | z_t; \pi_{z_t}) \\
 &= \prod_{t=1}^T \mu_{z_{t-1}, z_t} \times \prod_{t=1}^T \pi_{z_t, x_t}
 \end{aligned}$$

## Posterior Inference

For structured models, Bayes' rule may tractable,

$$p(z | x; \theta) = \frac{p(z; \mu) \times p(x | z; \pi)}{\sum_{z'} p(z'; \mu) \times p(x | z'; \pi)}$$

- Unlike previous models,  $z$  contains interdependent “parts.”
- For *both* shallow and deep Model 3 variants, it's possible to calculate  $p(x; \theta)$  exactly, with a dynamic program.
- For some structured models, like Factorial HMM, the dynamic program may still be intractable.

## Posterior Inference

## Introduction

## Models

Discrete Models

Continuous Models

**Structured Models**

## Variational

## Objective

## Inference

## Strategies

## Advanced Topics

## Case Studies

## Conclusion

## References

For structured models, Bayes' rule may be tractable,

$$p(z | x; \theta) = \frac{p(z; \mu) \times p(x | z; \pi)}{\sum_{z'} p(z'; \mu) \times p(x | z'; \pi)}$$

- Unlike previous models,  $z$  contains interdependent “parts.”
- For *both* shallow and deep Model 3 variants, it's possible to calculate  $p(x; \theta)$  exactly, with a dynamic program.
- For some structured models, like Factorial HMM, the dynamic program may still be intractable.

## ① Introduction

## ② Models

## ③ Variational Objective

Maximum Likelihood

ELBO

## ④ Inference Strategies

## ⑤ Advanced Topics

## ⑥ Case Studies

## ① Introduction

## ② Models

## ③ Variational Objective

Maximum Likelihood

ELBO

## ④ Inference Strategies

## ⑤ Advanced Topics

## ⑥ Case Studies



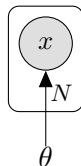
## Learning with Maximum Likelihood

Objective: Find model parameters  $\theta$  that maximize the likelihood of the data,

$$\theta^* = \arg \max_{\theta} \sum_{n=1}^N \log p(x^{(n)}; \theta)$$

## Learning Deep Models

$$L(\theta) = \sum_{n=1}^N \log p(x^{(n)}; \theta)$$



- Dominant framework is gradient-based optimization:

$$\theta^{(i)} = \theta^{(i-1)} + \eta \nabla_{\theta} L(\theta)$$

- $\nabla_{\theta} L(\theta)$  calculated with backpropagation.
- Tactics: mini-batch based training, adaptive learning rates [Duchi et al. 2011; Kingma and Ba 2015].

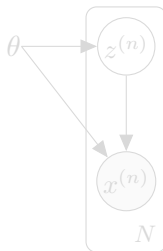
## Learning Deep Latent-Variable Models: Marginalization

Likelihood requires summing out the latent variables,

$$p(x; \theta) = \sum_{z \in \mathcal{Z}} p(x, z; \theta) \quad (= \int p(x, z; \theta) dz \text{ if continuous } z)$$

In general, **hard to optimize** log-likelihood for the training set,

$$L(\theta) = \sum_{n=1}^N \log \sum_{z \in \mathcal{Z}} p(x^{(n)}, z; \theta)$$



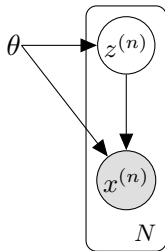
## Learning Deep Latent-Variable Models: Marginalization

Likelihood requires summing out the latent variables,

$$p(x; \theta) = \sum_{z \in \mathcal{Z}} p(x, z; \theta) \quad (= \int p(x, z; \theta) dz \text{ if continuous } z)$$

In general, **hard to optimize** log-likelihood for the training set,

$$L(\theta) = \sum_{n=1}^N \log \sum_{z \in \mathcal{Z}} p(x^{(n)}, z; \theta)$$



## ① Introduction

## ② Models

## ③ Variational Objective

Maximum Likelihood

**ELBO**

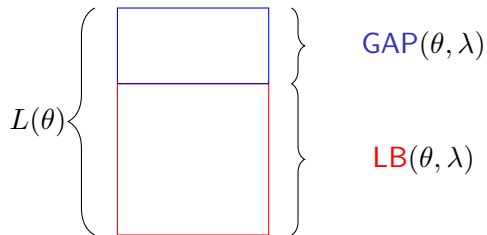
## ④ Inference Strategies

## ⑤ Advanced Topics

## ⑥ Case Studies

## Variational Inference

High-level: decompose objective into **lower-bound** and **gap**.


$$L(\theta) \left\{ \begin{array}{l} \text{GAP}(\theta, \lambda) \\ \text{LB}(\theta, \lambda) \end{array} \right.$$

$$L(\theta) = \text{LB}(\theta, \lambda) + \text{GAP}(\theta, \lambda) \text{ for some } \lambda$$

Provides framework for deriving a rich set of optimization algorithms.

## Marginal Likelihood: Variational Decomposition

## Introduction

## Models

Variational  
Objective

## Maximum Likelihood

## ELBO

Inference  
Strategies

## Advanced Topics

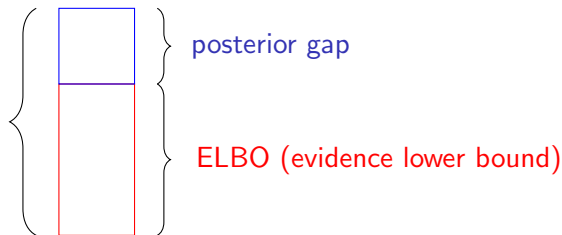
## Case Studies

## Conclusion

## References

For any<sup>1</sup> distribution  $q(z | x; \lambda)$  over  $z$ ,

$$L(\theta) = \mathbb{E}_q \left[ \log \frac{p(x, z; \theta)}{q(z | x; \lambda)} \right] + \text{KL}[q(z | x; \lambda) \| p(z | x; \theta)]$$



Since KL is always non-negative,  $L(\theta) \geq \text{ELBO}(\theta, \lambda)$ .

<sup>1</sup>Technical condition:  $\text{supp}(q(z)) \subset \text{supp}(p(z | x; \theta))$

## Evidence Lower Bound: Proof

Introduction

Models

Variational  
Objective

Maximum Likelihood

**ELBO**Inference  
Strategies

Advanced Topics

Case Studies

Conclusion

References

$$\log p(x; \theta) = \mathbb{E}_q \log p(x) \quad (\text{Expectation over } z)$$

$$= \mathbb{E}_q \log \frac{p(x, z)}{p(z | x)} \quad (\text{Mult/div by } p(z|x), \text{ combine numerator})$$

$$= \mathbb{E}_q \log \left( \frac{p(x, z)}{q(z | x)} \frac{q(z | x)}{p(z | x)} \right) \quad (\text{Mult/div by } q(z|x))$$

$$= \mathbb{E}_q \log \frac{p(x, z)}{q(z | x)} + \mathbb{E}_q \log \frac{q(z | x)}{p(z | x)} \quad (\text{Split Log})$$

$$= \mathbb{E}_q \log \frac{p(x, z; \theta)}{q(z | x; \lambda)} + \text{KL}[q(z | x; \lambda) \parallel p(z | x; \theta)]$$



## Evidence Lower Bound: Proof

Introduction

Models

Variational  
Objective

Maximum Likelihood

**ELBO**Inference  
Strategies

Advanced Topics

Case Studies

Conclusion

References

$$\begin{aligned}\log p(x; \theta) &= \mathbb{E}_q \log p(x) \quad (\text{Expectation over } z) \\&= \mathbb{E}_q \log \frac{p(x, z)}{p(z | x)} \quad (\text{Mult/div by } p(z|x), \text{ combine numerator}) \\&= \mathbb{E}_q \log \left( \frac{p(x, z)}{q(z | x)} \frac{q(z | x)}{p(z | x)} \right) \quad (\text{Mult/div by } q(z|x)) \\&= \mathbb{E}_q \log \frac{p(x, z)}{q(z | x)} + \mathbb{E}_q \log \frac{q(z | x)}{p(z | x)} \quad (\text{Split Log}) \\&= \mathbb{E}_q \log \frac{p(x, z; \theta)}{q(z | x; \lambda)} + \text{KL}[q(z | x; \lambda) \parallel p(z | x; \theta)]\end{aligned}$$

## Evidence Lower Bound: Proof

Introduction

Models

Variational  
Objective

Maximum Likelihood

ELBO

Inference  
Strategies

Advanced Topics

Case Studies

Conclusion

References

$$\begin{aligned}\log p(x; \theta) &= \mathbb{E}_q \log p(x) \quad (\text{Expectation over } z) \\ &= \mathbb{E}_q \log \frac{p(x, z)}{p(z | x)} \quad (\text{Mult/div by } p(z|x), \text{ combine numerator}) \\ &= \mathbb{E}_q \log \left( \frac{p(x, z)}{q(z | x)} \frac{q(z | x)}{p(z | x)} \right) \quad (\text{Mult/div by } q(z|x)) \\ &= \mathbb{E}_q \log \frac{p(x, z)}{q(z | x)} + \mathbb{E}_q \log \frac{q(z | x)}{p(z | x)} \quad (\text{Split Log}) \\ &= \mathbb{E}_q \log \frac{p(x, z; \theta)}{q(z | x; \lambda)} + \text{KL}[q(z | x; \lambda) \parallel p(z | x; \theta)]\end{aligned}$$

## Evidence Lower Bound: Proof

Introduction

Models

Variational  
Objective

Maximum Likelihood

ELBO

Inference  
Strategies

Advanced Topics

Case Studies

Conclusion

References

$$\begin{aligned}\log p(x; \theta) &= \mathbb{E}_q \log p(x) \quad (\text{Expectation over } z) \\ &= \mathbb{E}_q \log \frac{p(x, z)}{p(z | x)} \quad (\text{Mult/div by } p(z|x), \text{ combine numerator}) \\ &= \mathbb{E}_q \log \left( \frac{p(x, z)}{q(z | x)} \frac{q(z | x)}{p(z | x)} \right) \quad (\text{Mult/div by } q(z|x)) \\ &= \mathbb{E}_q \log \frac{p(x, z)}{q(z | x)} + \mathbb{E}_q \log \frac{q(z | x)}{p(z | x)} \quad (\text{Split Log}) \\ &= \mathbb{E}_q \log \frac{p(x, z; \theta)}{q(z | x; \lambda)} + \text{KL}[q(z | x; \lambda) \parallel p(z | x; \theta)]\end{aligned}$$

## Evidence Lower Bound: Proof

Introduction

Models

Variational  
Objective

Maximum Likelihood

ELBO

Inference  
Strategies

Advanced Topics

Case Studies

Conclusion

References

$$\begin{aligned}\log p(x; \theta) &= \mathbb{E}_q \log p(x) \quad (\text{Expectation over } z) \\ &= \mathbb{E}_q \log \frac{p(x, z)}{p(z | x)} \quad (\text{Mult/div by } p(z|x), \text{ combine numerator}) \\ &= \mathbb{E}_q \log \left( \frac{p(x, z)}{q(z | x)} \frac{q(z | x)}{p(z | x)} \right) \quad (\text{Mult/div by } q(z|x)) \\ &= \mathbb{E}_q \log \frac{p(x, z)}{q(z | x)} + \mathbb{E}_q \log \frac{q(z | x)}{p(z | x)} \quad (\text{Split Log}) \\ &= \mathbb{E}_q \log \frac{p(x, z; \theta)}{q(z | x; \lambda)} + \text{KL}[q(z | x; \lambda) \| p(z | x; \theta)]\end{aligned}$$

## Evidence Lower Bound over Observations

$$\text{ELBO}(\theta, \lambda; x) = \mathbb{E}_{q(z)} \left[ \log \frac{p(x, z; \theta)}{q(z | x; \lambda)} \right]$$

- ELBO is a function of the generative model parameters,  $\theta$ , and the variational parameters,  $\lambda$ .

$$\begin{aligned} \sum_{n=1}^N \log p(x^{(n)}; \theta) &\geq \sum_{n=1}^N \text{ELBO}(\theta, \lambda; x^{(n)}) \\ &= \sum_{n=1}^N \mathbb{E}_{q(z | x^{(n)}; \lambda)} \left[ \log \frac{p(x^{(n)}, z; \theta)}{q(z | x^{(n)}; \lambda)} \right] \\ &= \text{ELBO}(\theta, \lambda; x^{(1:N)}) = \text{ELBO}(\theta, \lambda) \end{aligned}$$

## Setup: Selecting Variational Family

Introduction

Models

Variational  
Objective

Maximum Likelihood

**ELBO**Inference  
Strategies

Advanced Topics

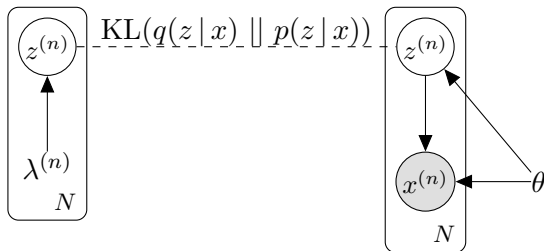
Case Studies

Conclusion

References

- Just as with  $p$  and  $\theta$ , we can select any form of  $q$  and  $\lambda$  that satisfies ELBO conditions.
- Different choices of  $q$  will lead to different algorithms.
- We will explore several forms of  $q$ :
  - Posterior
  - Point Estimate / MAP
  - Amortized
  - Mean Field (later)

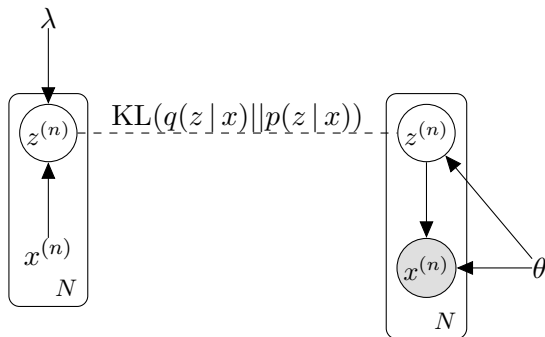
## Example Family : Full Posterior Form



$\lambda = [\lambda^{(1)}, \dots, \lambda^{(N)}]$  is a concatenation of local variational parameters  $\lambda^{(n)}$ , e.g.

$$q(z^{(n)} | x^{(n)}; \lambda) = q(z^{(n)} | x^{(n)}; \lambda^{(n)}) = \mathcal{N}(\lambda^{(n)}, 1)$$

## Example Family: Amortized Parameterization [Kingma and Welling 2014]



$\lambda$  parameterizes a global network (encoder/inference network) that is run over  $x^{(n)}$  to produce the local variational distribution, e.g.

$$q(z^{(n)} | x^{(n)}; \lambda) = \mathcal{N}(\mu^{(n)}, 1), \quad \mu^{(n)} = \text{enc}(x^{(n)}; \lambda)$$



① Introduction

② Models

③ Variational Objective

④ Inference Strategies

Exact Gradient

Sampling

Conjugacy

⑤ Advanced Topics

⑥ Case Studies

## Maximizing the Evidence Lower Bound

Central quantity of interest: almost all methods are maximizing the ELBO

$$\arg \max_{\theta, \lambda} \text{ELBO}(\theta, \lambda)$$

Aggregate ELBO objective,

$$\begin{aligned} \arg \max_{\theta, \lambda} \text{ELBO}(\theta, \lambda) &= \arg \max_{\theta, \lambda} \sum_{n=1}^N \text{ELBO}(\theta, \lambda; x^{(n)}) \\ &= \arg \max_{\theta, \lambda} \sum_{n=1}^N \mathbb{E}_q \left[ \log \frac{p(x^{(n)}, z^{(n)}; \theta)}{q(z^{(n)} | x^{(n)}; \lambda)} \right] \end{aligned}$$

## Maximizing the Evidence Lower Bound

Central quantity of interest: almost all methods are maximizing the ELBO

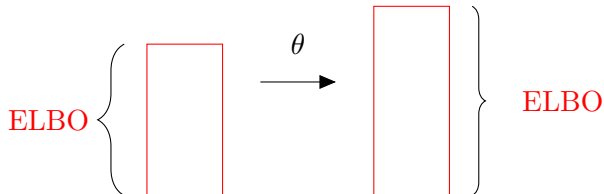
$$\arg \max_{\theta, \lambda} \text{ELBO}(\theta, \lambda)$$

Aggregate ELBO objective,

$$\begin{aligned} \arg \max_{\theta, \lambda} \text{ELBO}(\theta, \lambda) &= \arg \max_{\theta, \lambda} \sum_{n=1}^N \text{ELBO}(\theta, \lambda; x^{(n)}) \\ &= \arg \max_{\theta, \lambda} \sum_{n=1}^N \mathbb{E}_q \left[ \log \frac{p(x^{(n)}, z^{(n)}; \theta)}{q(z^{(n)} | x^{(n)}; \lambda)} \right] \end{aligned}$$

## Maximizing ELBO: Model Parameters

$$\arg \max_{\theta} \mathbb{E}_q \left[ \log \frac{p(x, z; \theta)}{q(z | x; \lambda)} \right] = \arg \max_{\theta} \mathbb{E}_q [\log p(x, z; \theta)]$$



Intuition: Maximum likelihood problem under variables drawn from  $q(z | x; \lambda)$ .

## Model Estimation: Gradient Ascent on Model Parameters

## Introduction

## Models

Variational  
ObjectiveInference  
StrategiesExact Gradient  
Sampling  
Conjugacy

## Advanced Topics

## Case Studies

## Conclusion

## References

Easy: Gradient respect to  $\theta$ 

$$\begin{aligned}\nabla_{\theta} \text{ELBO}(\theta, \lambda; x) &= \nabla_{\theta} \mathbb{E}_q \left[ \log p(x, z; \theta) \right] \\ &= \mathbb{E}_q \left[ \nabla_{\theta} \log p(x, z; \theta) \right]\end{aligned}$$

- Since  $q$  not dependent on  $\theta$ ,  $\nabla$  moves inside expectation.
- Estimate with samples from  $q$ . Term  $\log p(x, z; \theta)$  is easy to evaluate. (In practice single sample is often sufficient).
- In special cases, can exactly evaluate expectation.

## Model Estimation: Gradient Ascent on Model Parameters

## Introduction

## Models

Variational  
ObjectiveInference  
Strategies

Exact Gradient

Sampling

Conjugacy

## Advanced Topics

## Case Studies

## Conclusion

## References

Easy: Gradient respect to  $\theta$ 

$$\begin{aligned}\nabla_{\theta} \text{ELBO}(\theta, \lambda; x) &= \nabla_{\theta} \mathbb{E}_q \left[ \log p(x, z; \theta) \right] \\ &= \mathbb{E}_q \left[ \nabla_{\theta} \log p(x, z; \theta) \right]\end{aligned}$$

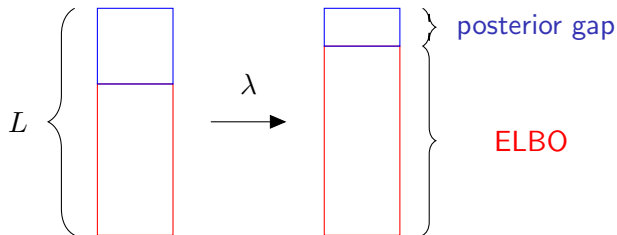
- Since  $q$  not dependent on  $\theta$ ,  $\nabla$  moves inside expectation.
- Estimate with samples from  $q$ . Term  $\log p(x, z; \theta)$  is easy to evaluate. (In practice single sample is often sufficient).
- In special cases, can exactly evaluate expectation.

## Maximizing ELBO: Variational Distribution

$$\arg \max_{\lambda} \text{ELBO}(\theta, \lambda)$$

$$= \arg \max_{\lambda} \log p(x; \theta) - \text{KL}[q(z | x; \lambda) \parallel p(z | x; \theta)]$$

$$= \arg \min_{\lambda} \text{KL}[q(z | x; \lambda) \parallel p(z | x; \theta)]$$



Intuition:  $q$  should approximate the posterior  $p(z|x)$ . However, may be difficult if  $q$  or  $p$  is a deep model.

Model Inference: Gradient Ascent on  $\lambda$ ?

Hard: Gradient respect to  $\lambda$

$$\begin{aligned}\nabla_{\lambda} \text{ELBO}(\theta, \lambda; x) &= \nabla_{\lambda} \mathbb{E}_q \left[ \log p(x, z; \theta) \right] \\ &\neq \mathbb{E}_q \left[ \nabla_{\lambda} \log p(x, z; \theta) \right]\end{aligned}$$

- Cannot naively move  $\nabla$  inside the expectation, since  $q$  depends on  $\lambda$ .
- This section: Inference in practice:
  - ① Exact gradient
  - ② Sampling: score function, reparameterization
  - ③ Conjugacy: closed-form, coordinate ascent



Model Inference: Gradient Ascent on  $\lambda$ ?

Introduction

Models

Variational  
ObjectiveInference  
Strategies

Exact Gradient

Sampling

Conjugacy

Advanced Topics

Case Studies

Conclusion

References

Hard: Gradient respect to  $\lambda$ 

$$\begin{aligned}\nabla_{\lambda} \text{ELBO}(\theta, \lambda; x) &= \nabla_{\lambda} \mathbb{E}_q \left[ \log p(x, z; \theta) \right] \\ &\neq \mathbb{E}_q \left[ \nabla_{\lambda} \log p(x, z; \theta) \right]\end{aligned}$$

- Cannot naively move  $\nabla$  inside the expectation, since  $q$  depends on  $\lambda$ .
- This section: Inference in practice:
  - ① Exact gradient
  - ② Sampling: score function, reparameterization
  - ③ Conjugacy: closed-form, coordinate ascent

## ① Introduction

## ② Models

## ③ Variational Objective

## ④ Inference Strategies

Exact Gradient

Sampling

Conjugacy

## ⑤ Advanced Topics

## ⑥ Case Studies

## Strategy 1: Exact Gradient

$$\begin{aligned}\nabla_{\lambda} \text{ELBO}(\theta, \lambda; x) &= \nabla_{\lambda} \mathbb{E}_{q(z|x; \lambda)} \left[ \log \frac{p(x, z; \theta)}{q(z|x; \lambda)} \right] \\ &= \nabla_{\lambda} \left( \sum_{z \in \mathcal{Z}} q(z|x; \lambda) \log \frac{p(x, z; \theta)}{q(z|x; \lambda)} \right)\end{aligned}$$

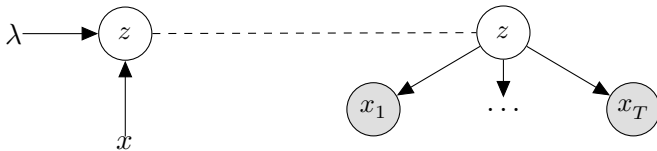
- Naive enumeration: Linear in  $|\mathcal{Z}|$ .
- Depending on structure of  $q$  and  $p$ , potentially faster with dynamic programming.
- Applicable mainly to Model 1 and 3 (Discrete and Structured), or Model 2 with point estimate.

## Strategy 1: Exact Gradient

$$\begin{aligned}\nabla_{\lambda} \text{ELBO}(\theta, \lambda; x) &= \nabla_{\lambda} \mathbb{E}_{q(z|x; \lambda)} \left[ \log \frac{p(x, z; \theta)}{q(z|x; \lambda)} \right] \\ &= \nabla_{\lambda} \left( \sum_{z \in \mathcal{Z}} q(z|x; \lambda) \log \frac{p(x, z; \theta)}{q(z|x; \lambda)} \right)\end{aligned}$$

- Naive enumeration: Linear in  $|\mathcal{Z}|$ .
- Depending on structure of  $q$  and  $p$ , potentially faster with dynamic programming.
- Applicable mainly to Model 1 and 3 (Discrete and Structured), or Model 2 with point estimate.

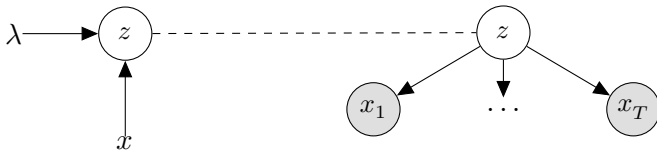
## Example: Model 1 - Naive Bayes



Let  $q(z | x; \lambda) = \text{Cat}(\nu)$  where  $\nu = \text{enc}(x; \lambda)$

$$\begin{aligned}\nabla_{\lambda} \text{ELBO}(\theta, \lambda; x) &= \nabla_{\lambda} \mathbb{E}_{q(z | x; \lambda)} \left[ \log \frac{p(x, z; \theta)}{q(z | x; \lambda)} \right] \\ &= \nabla_{\lambda} \left( \sum_{z \in \mathcal{Z}} q(z | x; \lambda) \log \frac{p(x, z; \theta)}{q(z | x; \lambda)} \right) \\ &= \nabla_{\lambda} \left( \sum_{z \in \mathcal{Z}} \nu_z \log \frac{p(x, z; \theta)}{\nu_z} \right)\end{aligned}$$

## Example: Model 1 - Naive Bayes



Let  $q(z | x; \lambda) = \text{Cat}(\nu)$  where  $\nu = \text{enc}(x; \lambda)$

$$\begin{aligned}\nabla_{\lambda} \text{ELBO}(\theta, \lambda; x) &= \nabla_{\lambda} \mathbb{E}_{q(z | x; \lambda)} \left[ \log \frac{p(x, z; \theta)}{q(z | x; \lambda)} \right] \\ &= \nabla_{\lambda} \left( \sum_{z \in \mathcal{Z}} q(z | x; \lambda) \log \frac{p(x, z; \theta)}{q(z | x; \lambda)} \right) \\ &= \nabla_{\lambda} \left( \sum_{z \in \mathcal{Z}} \nu_z \log \frac{p(x, z; \theta)}{\nu_z} \right)\end{aligned}$$

## ① Introduction

## ② Models

## ③ Variational Objective

## ④ Inference Strategies

Exact Gradient

**Sampling**

Conjugacy

## ⑤ Advanced Topics

## ⑥ Case Studies

## Strategy 2: Sampling

$$\begin{aligned}\nabla_{\lambda} \text{ELBO}(\theta, \lambda; x) &= \nabla_{\lambda} \mathbb{E}_q \left[ \log \frac{\log p(x, z; \theta)}{\log q(z | x; \lambda)} \right] \\ &= \nabla_{\lambda} \mathbb{E}_q \left[ \log p(x, z; \theta) \right] - \nabla_{\lambda} \mathbb{E}_q \left[ \log q(z | x; \theta) \right]\end{aligned}$$

- How can we approximate this gradient with sampling? Naive algorithm fails to provide non-zero gradient.

$$z^{(1)}, \dots, z^{(J)} \sim q(z | x; \lambda)$$

$$\nabla_{\lambda} \frac{1}{J} \sum_{j=1}^J \left[ \log p(x, z^{(j)}; \theta) \right] = 0$$

- Manipulate expression so we can move  $\nabla_{\lambda}$  inside  $\mathbb{E}_q$  before sampling.



## Strategy 2: Sampling

$$\begin{aligned}\nabla_{\lambda} \text{ELBO}(\theta, \lambda; x) &= \nabla_{\lambda} \mathbb{E}_q \left[ \log \frac{\log p(x, z; \theta)}{\log q(z | x; \lambda)} \right] \\ &= \nabla_{\lambda} \mathbb{E}_q \left[ \log p(x, z; \theta) \right] - \nabla_{\lambda} \mathbb{E}_q \left[ \log q(z | x; \theta) \right]\end{aligned}$$

- How can we approximate this gradient with sampling? Naive algorithm fails to provide non-zero gradient.

$$z^{(1)}, \dots, z^{(J)} \sim q(z | x; \lambda)$$

$$\nabla_{\lambda} \frac{1}{J} \sum_{j=1}^J \left[ \log p(x, z^{(j)}; \theta) \right] = 0$$

- Manipulate expression so we can move  $\nabla_{\lambda}$  inside  $\mathbb{E}_q$  before sampling.

## Strategy 2a: Sampling — Score Function Gradient Estimator

First term. Use basic identity:

$$\nabla \log q = \frac{\nabla q}{q} \Rightarrow \nabla q = q \nabla \log q$$

Policy-gradient style training [Williams 1992]

$$\nabla_{\lambda} \mathbb{E}_q \left[ \log p(x, z; \theta) \right] = \sum_z \nabla_{\lambda} q(z | x; \lambda) \log p(x, z; \theta)$$

## Strategy 2a: Sampling — Score Function Gradient Estimator

First term. Use basic identity:

$$\nabla \log q = \frac{\nabla q}{q} \Rightarrow \nabla q = q \nabla \log q$$

Policy-gradient style training [Williams 1992]

$$\nabla_{\lambda} \mathbb{E}_q \left[ \log p(x, z; \theta) \right] = \sum_z \underbrace{\nabla_{\lambda} q(z | x; \lambda)}_{q \nabla \log q} \log p(x, z; \theta)$$

## Strategy 2a: Sampling — Score Function Gradient Estimator

Introduction

Models

Variational

Objective

Inference

Strategies

Exact Gradient

**Sampling**

Conjugacy

Advanced Topics

Case Studies

Conclusion

References

First term. Use basic identity:

$$\nabla \log q = \frac{\nabla q}{q} \Rightarrow \nabla q = q \nabla \log q$$

Policy-gradient style training [Williams 1992]

$$\begin{aligned} \nabla_{\lambda} \mathbb{E}_q \left[ \log p(x, z; \theta) \right] &= \sum_z \nabla_{\lambda} q(z | x; \lambda) \log p(x, z; \theta) \\ &= \sum_z q(z | x; \lambda) \nabla_{\lambda} \log q(z | x; \lambda) \log p(x, z; \theta) \end{aligned}$$

## Strategy 2a: Sampling — Score Function Gradient Estimator

Introduction

Models

Variational  
ObjectiveInference  
Strategies

Exact Gradient

**Sampling**

Conjugacy

Advanced Topics

Case Studies

Conclusion

References

First term. Use basic identity:

$$\nabla \log q = \frac{\nabla q}{q} \Rightarrow \nabla q = q \nabla \log q$$

Policy-gradient style training [Williams 1992]

$$\begin{aligned} \nabla_{\lambda} \mathbb{E}_q \left[ \log p(x, z; \theta) \right] &= \sum_z \nabla_{\lambda} q(z | x; \lambda) \log p(x, z; \theta) \\ &= \sum_z q(z | x; \lambda) \nabla_{\lambda} \log q(z | x; \lambda) \log p(x, z; \theta) \\ &= \mathbb{E}_q \left[ \log p(x, z; \theta) \nabla_{\lambda} \log q(z | x; \lambda) \right] \end{aligned}$$

## Strategy 2a: Sampling — Score Function Gradient Estimator

Second term. Need additional identity:

$$\sum \nabla q = \nabla \sum q = \nabla 1 = 0$$

$$\nabla_{\lambda} \mathbb{E}_q \left[ \log q(z | x; \lambda) \right] = \sum_z \nabla_{\lambda} \left( q(z | x; \lambda) \log q(z | x; \lambda) \right)$$

## Strategy 2a: Sampling — Score Function Gradient Estimator

Introduction

Models

Variational

Objective

Inference

Strategies

Exact Gradient

**Sampling**

Conjugacy

Advanced Topics

Case Studies

Conclusion

References

Second term. Need additional identity:

$$\sum \nabla q = \nabla \sum q = \nabla 1 = 0$$

$$\begin{aligned} \nabla_{\lambda} \mathbb{E}_q \left[ \log q(z | x; \lambda) \right] &= \sum_z \nabla_{\lambda} \left( q(z | x; \lambda) \log q(z | x; \lambda) \right) \\ &= \sum_z \left( \underbrace{\nabla_{\lambda} q(z | x; \lambda)}_{q \nabla \log q} \right) \log q(z | x; \lambda) + q(z | x; \lambda) \left( \underbrace{\nabla_{\lambda} \log q(z | x; \lambda)}_{\frac{\nabla q}{q}} \right) \end{aligned}$$

## Strategy 2a: Sampling — Score Function Gradient Estimator

Second term. Need additional identity:

$$\sum \nabla q = \nabla \sum q = \nabla 1 = 0$$

$$\begin{aligned} \nabla_{\lambda} \mathbb{E}_q \left[ \log q(z | x; \lambda) \right] &= \sum_z \nabla_{\lambda} \left( q(z | x; \lambda) \log q(z | x; \lambda) \right) \\ &= \sum_z \log q(z | x; \lambda) \nabla_{\lambda} q(z | x; \lambda) + \sum_z \nabla_{\lambda} q(z | x; \lambda) \end{aligned}$$



## Strategy 2a: Sampling — Score Function Gradient Estimator

Introduction

Models

Variational

Objective

Inference

Strategies

Exact Gradient

**Sampling**

Conjugacy

Advanced Topics

Case Studies

Conclusion

References

Second term. Need additional identity:

$$\sum \nabla q = \nabla \sum q = \nabla 1 = 0$$

$$\begin{aligned} \nabla_{\lambda} \mathbb{E}_q \left[ \log q(z | x; \lambda) \right] &= \sum_z \nabla_{\lambda} \left( q(z | x; \lambda) \log q(z | x; \lambda) \right) \\ &= \sum_z \log q(z | x; \lambda) \nabla_{\lambda} \log q(z | x; \lambda) + \underbrace{\sum_z \nabla_{\lambda} q(z | x; \lambda)}_{=\nabla \sum q = \nabla 1 = 0} \end{aligned}$$

## Strategy 2a: Sampling — Score Function Gradient Estimator

Second term. Need additional identity:

$$\sum \nabla q = \nabla \sum q = \nabla 1 = 0$$

$$\begin{aligned}\nabla_{\lambda} \mathbb{E}_q \left[ \log q(z | x; \lambda) \right] &= \sum_z \nabla_{\lambda} \left( q(z | x; \lambda) \log q(z | x; \lambda) \right) \\ &= \sum_z \log q(z | x; \lambda) \nabla_{\lambda} \log q(z | x; \lambda) + \sum_z \nabla_{\lambda} q(z | x; \lambda) \\ &= \mathbb{E}_q [\log q(z | x; \lambda) \nabla_{\lambda} q(z | x; \lambda)]\end{aligned}$$

## Strategy 2a: Sampling — Score Function Gradient Estimator

Putting these together,

$$\begin{aligned}\nabla_{\lambda} \text{ELBO}(\theta, \lambda; x) &= \nabla_{\lambda} \mathbb{E}_q \left[ \log \frac{p(x, z; \theta)}{q(z | x; \lambda)} \right] \\ &= \mathbb{E}_q \left[ \log \frac{p(x, z; \theta)}{q(z | x; \lambda)} \nabla_{\lambda} \log q(z | x; \lambda) \right] \\ &= \mathbb{E}_q \left[ R_{\theta, \lambda}(z) \nabla_{\lambda} \log q(z | x; \lambda) \right]\end{aligned}$$

## Strategy 2a: Sampling — Score Function Gradient Estimator

Introduction

Models

Variational  
ObjectiveInference  
Strategies

Exact Gradient

**Sampling**

Conjugacy

Advanced Topics

Case Studies

Conclusion

References

Estimate with samples,

$$z^{(1)}, \dots, z^{(J)} \sim q(z | x; \lambda)$$

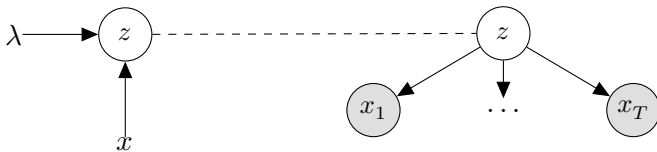
$$\begin{aligned} & \mathbb{E}_q \left[ R_{\theta, \lambda}(z) \nabla_{\lambda} \log q(z | x; \lambda) \right] \\ & \approx \frac{1}{J} \sum_{j=1}^J R_{\theta, \lambda}(z^{(j)}) \nabla_{\lambda} \log q(z^{(j)} | x; \lambda) \end{aligned}$$

Intuition: if a sample  $z^{(j)}$  has high reward  $R_{\theta, \lambda}(z^{(j)})$ , increase the probability of  $z^{(j)}$  by moving along the gradient  $\nabla_{\lambda} \log q(z^{(j)} | x; \lambda)$ .

## Strategy 2a: Sampling — Score Function Gradient Estimator

- Essentially reinforcement learning with reward  $R_{\theta,\lambda}(z)$
- Score function gradient is generally applicable regardless of what distribution  $q$  takes (only need to evaluate  $\nabla_{\lambda} \log q$ ).
- This generality comes at a cost, since the reward is “black-box”: unbiased estimator, but high variance.
- In practice, need variance-reducing **control variate**  $B$ . (More on this later).

## Example: Model 1 - Naive Bayes



Let  $q(z | x; \lambda) = \text{Cat}(\nu)$  where  $\nu = \text{enc}(x; \lambda)$

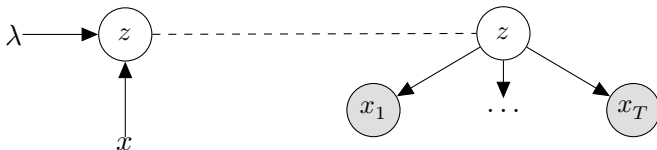
Sample  $z^{(1)}, \dots, z^{(J)} \sim q(z | x; \lambda)$

$$\nabla_{\lambda} \text{ELBO}(\theta, \lambda; x) = \mathbb{E}_q \left[ \log \frac{p(x, z; \theta)}{q(z | x; \lambda)} \nabla_{\lambda} \log q(z | x; \lambda) \right]$$

$$\approx \frac{1}{J} \sum_{j=1}^J \nu_{z^{(j)}} \log \frac{p(x, z^{(j)}; \theta)}{\nu_{z^{(j)}}} \nabla_{\lambda} \log \nu_{z^{(j)}}$$

Computational complexity:  $O(J)$  vs  $O(|\mathcal{Z}|)$

## Example: Model 1 - Naive Bayes



Let  $q(z | x; \lambda) = \text{Cat}(\nu)$  where  $\nu = \text{enc}(x; \lambda)$

Sample  $z^{(1)}, \dots, z^{(J)} \sim q(z | x; \lambda)$

$$\begin{aligned} \nabla_{\lambda} \text{ELBO}(\theta, \lambda; x) &= \mathbb{E}_q \left[ \log \frac{p(x, z; \theta)}{q(z | x; \lambda)} \nabla_{\lambda} \log q(z | x; \lambda) \right] \\ &\approx \frac{1}{J} \sum_{j=1}^J \nu_{z^{(j)}} \log \frac{p(x, z^{(j)}; \theta)}{\nu_{z^{(j)}}} \nabla_{\lambda} \log \nu_{z^{(j)}} \end{aligned}$$

Computational complexity:  $O(J)$  vs  $O(|\mathcal{Z}|)$

## Strategy 2b: Sampling — Reparameterization

Suppose we can sample from  $q$  by applying a deterministic, differentiable transformation  $g$  to a base noise density,

$$\epsilon \sim \mathcal{U} \quad z = g(\epsilon, \lambda)$$

Gradient calculation (first term):

$$\begin{aligned} \nabla_{\lambda} \mathbb{E}_{z \sim q(z|x; \lambda)} \left[ \log p(x, z; \theta) \right] &= \nabla_{\lambda} \mathbb{E}_{\epsilon \sim \mathcal{U}} \left[ \log p(x, g(\epsilon, \lambda); \theta) \right] \\ &= \mathbb{E}_{\epsilon \sim \mathcal{U}} \left[ \nabla_{\lambda} \log p(x, g(\epsilon, \lambda); \theta) \right] \\ &\approx \frac{1}{J} \sum_{j=1}^J \nabla_{\lambda} \log p(x, g(\epsilon^{(j)}, \lambda); \theta) \end{aligned}$$

where

$$\epsilon^{(1)}, \dots, \epsilon^{(J)} \sim \mathcal{U}$$



## Strategy 2b: Sampling — Reparameterization

Suppose we can sample from  $q$  by applying a deterministic, differentiable transformation  $g$  to a base noise density,

$$\epsilon \sim \mathcal{U} \quad z = g(\epsilon, \lambda)$$

Gradient calculation (**first term**):

$$\begin{aligned} \nabla_{\lambda} \mathbb{E}_{z \sim q(z|x; \lambda)} \left[ \log p(x, z; \theta) \right] &= \nabla_{\lambda} \mathbb{E}_{\epsilon \sim \mathcal{U}} \left[ \log p(x, g(\epsilon, \lambda); \theta) \right] \\ &= \mathbb{E}_{\epsilon \sim \mathcal{U}} \left[ \nabla_{\lambda} \log p(x, g(\epsilon, \lambda); \theta) \right] \\ &\approx \frac{1}{J} \sum_{j=1}^J \nabla_{\lambda} \log p(x, g(\epsilon^{(j)}, \lambda); \theta) \end{aligned}$$

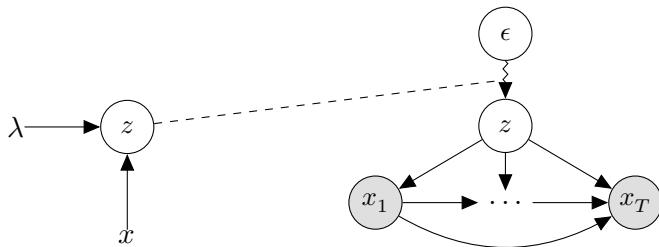
where

$$\epsilon^{(1)}, \dots, \epsilon^{(J)} \sim \mathcal{U}$$

## Strategy 2b: Sampling — Reparameterization

- Unbiased-like score function gradient estimator, but empirically lower variance.
- In practice, single sample is often sufficient.
- Cannot be used out-of-the-box for discrete  $z$ .

## Strategy 2: Continuous Latent Variable RNN



Choose variational family to be an amortized diagonal Gaussian

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

$$\mu, \sigma^2 = \text{enc}(x; \lambda)$$

## Strategy 2b: Sampling — Reparameterization

Introduction

Models

Variational  
ObjectiveInference  
Strategies

Exact Gradient

**Sampling**

Conjugacy

Advanced Topics

Case Studies

Conclusion

References

(Recall  $R_{\theta,\lambda}(z) = \log \frac{p(x,z;\theta)}{q(z|x;\lambda)}$ )

- Score function:

$$\nabla_{\lambda} \text{ELBO}(\theta, \lambda; x) = \mathbb{E}_{z \sim q}[R_{\theta,\lambda}(z) \nabla_{\lambda} \log q(z | x; \lambda)]$$

- Reparameterization:

$$\nabla_{\lambda} \text{ELBO}(\theta, \lambda; x) = \mathbb{E}_{\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})}[\nabla_{\lambda} R_{\theta,\lambda}(g(\epsilon, \lambda; x))]$$

where  $g(\epsilon, \lambda; x) = \mu + \sigma\epsilon$ .

Informally, reparameterization gradients differentiate through  $R_{\theta,\lambda}(\cdot)$  and thus has “more knowledge” about the structure of the objective function.

## ① Introduction

## ② Models

## ③ Variational Objective

## ④ Inference Strategies

Exact Gradient

Sampling

Conjugacy

## ⑤ Advanced Topics

## ⑥ Case Studies

## Strategy 3: Conjugacy

For certain choices for  $p$  and  $q$ , we can compute parts of

$$\arg \max_{\lambda} \text{ELBO}(\theta, \lambda; x)$$

exactly in closed-form.

Recall that

$$\arg \max_{\lambda} \text{ELBO}(\theta, \lambda; x) = \arg \min_{\lambda} \text{KL}[q(z | x; \lambda) \| p(z | x; \theta)]$$

## Strategy 3: Conjugacy

For certain choices for  $p$  and  $q$ , we can compute parts of

$$\arg \max_{\lambda} \text{ELBO}(\theta, \lambda; x)$$

exactly in closed-form.

Recall that

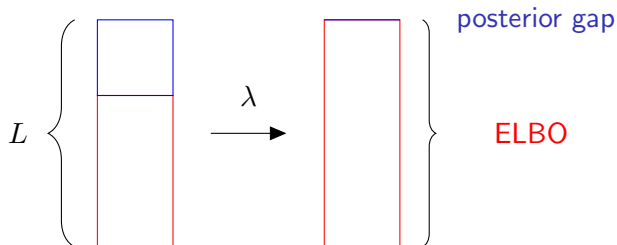
$$\arg \max_{\lambda} \text{ELBO}(\theta, \lambda; x) = \arg \min_{\lambda} \text{KL}[q(z | x; \lambda) \| p(z | x; \theta)]$$

## Strategy 3a: Conjugacy — Tractable Posterior Inference

Suppose we can tractably calculate  $p(z | x; \theta)$ . Then  $\text{KL}[q(z | x; \lambda) || p(z | x; \theta)]$  is minimized when,

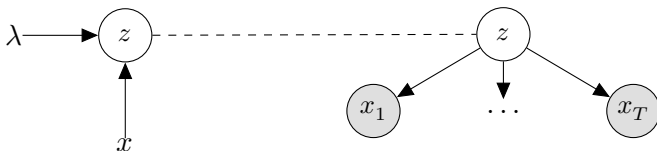
$$q(z | x; \lambda) = p(z | x; \theta)$$

- The E-step in Expectation Maximization algorithm [Dempster et al. 1977]





## Example: Model 1 - Naive Bayes

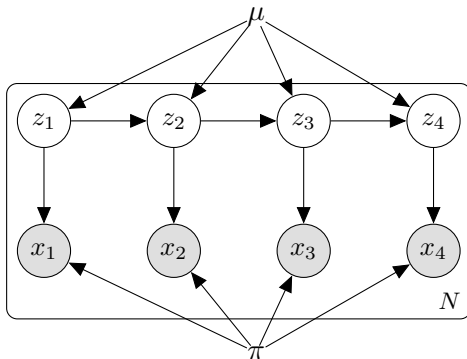


$$p(z \mid x; \theta) = \frac{p(x, z; \theta)}{\sum_{z'=1}^K p(x, z'; \theta)}$$

So  $\lambda$  is given by the parameters of the categorical distribution, i.e.

$$\lambda = [p(z = 1 \mid x; \theta), \dots, p(z = K \mid x; \theta)]$$

## Reminder: Model 3 — HMM



$$p(x, z; \theta) = p(z_0) \prod_{t=1}^T p(z_t | z_{t-1}; \mu) p(x_t | z_t; \pi)$$

## Example: Model 3 — HMM

Run forward/backward dynamic programming to calculate posterior marginals,

$$p(z_t, z_{t+1} \mid x; \theta)$$

variational parameters  $\lambda \in \mathbb{R}^{TK^2}$  store edge marginals. These are enough to calculate

$$q(z; \lambda) = p(z \mid x; \theta)$$

(i.e. the exact posterior) over any sequence  $z$ .

## Example: Model 3 — HMM

Run forward/backward dynamic programming to calculate posterior marginals,

$$p(z_t, z_{t+1} \mid x; \theta)$$

variational parameters  $\lambda \in \mathbb{R}^{TK^2}$  store edge marginals. These are enough to calculate

$$q(z; \lambda) = p(z \mid x; \theta)$$

(i.e. the exact posterior) over any sequence  $z$ .

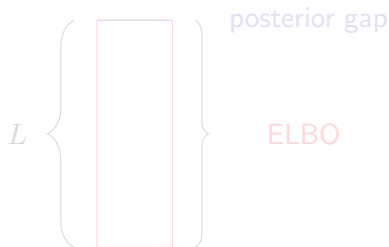
## Connection: Gradient Ascent on Log Marginal Likelihood

Why not perform gradient ascent directly on log marginal likelihood?

$$\log p(x; \theta) = \log \sum_z p(x, z; \theta)$$

Same as optimizing ELBO with posterior inference (i.e EM). Gradients of model parameters given by (where  $q(z | x; \lambda) = p(z | x; \theta)$ ):

$$\nabla_{\theta} \log p(x; \theta) = \mathbb{E}_{q(z | x; \lambda)} [\nabla_{\theta} \log p(x, z; \theta)]$$



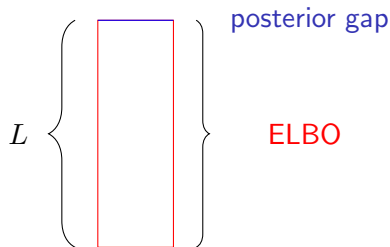
## Connection: Gradient Ascent on Log Marginal Likelihood

Why not perform gradient ascent directly on log marginal likelihood?

$$\log p(x; \theta) = \log \sum_z p(x, z; \theta)$$

Same as optimizing ELBO with posterior inference (i.e EM). Gradients of model parameters given by (where  $q(z | x; \lambda) = p(z | x; \theta)$ ):

$$\nabla_{\theta} \log p(x; \theta) = \mathbb{E}_{q(z | x; \lambda)} [\nabla_{\theta} \log p(x, z; \theta)]$$



## Connection: Gradient Ascent on Log Marginal Likelihood

Introduction

Models

Variational  
ObjectiveInference  
StrategiesExact Gradient  
Sampling  
Conjugacy

Advanced Topics

Case Studies

Conclusion

References

- Practically, this means we don't have to manually perform posterior inference in the E-step. Can just calculate  $\log p(x; \theta)$  and call backpropagation.
- Example: in deep HMM, just implement forward algorithm to calculate  $\log p(x; \theta)$  and backpropagate using autodiff. No need to implement backward algorithm. (Or vice versa).

(See Eisner [2016]: “Inside-Outside and Forward-Backward Algorithms Are Just Backprop”)

## Strategy 3b: Conditional Conjugacy

Introduction

Models

Variational  
ObjectiveInference  
Strategies

Exact Gradient

Sampling

Conjugacy

Advanced Topics

Case Studies

Conclusion

References

- Let  $p(z | x; \theta)$  be intractable, but suppose  $p(x, z; \theta)$  is **conditionally conjugate**, meaning  $p(z_t | x, z_{-t}; \theta)$  is exponential family.
- Restrict the family of distributions  $q$  so that it factorizes over  $z_t$ , i.e.

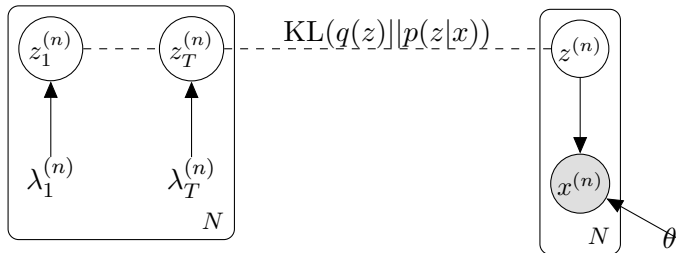
$$q(z; \lambda) = \prod_{t=1}^T q(z_t; \lambda_t)$$

(**mean field** family)

- Further choose  $q(z_t; \lambda_t)$  so that it is in the same family as  $p(z_t | x, z_{-t}; \theta)$ .



## Strategy 3b: Conditional Conjugacy



$$q(z; \lambda) = \prod_{t=1}^T q(z_t; \lambda_t)$$

## Mean Field Family

Introduction

Models

Variational  
ObjectiveInference  
Strategies

Exact Gradient

Sampling

Conjugacy

Advanced Topics

Case Studies

Conclusion

References

- Optimize ELBO via coordinate ascent, i.e. iterate for  $\lambda_1, \dots, \lambda_T$

$$\arg \max_{\lambda_t} \text{KL} \left[ \prod_{t=1}^T q(z_t; \lambda_t) \parallel p(z \mid x; \theta) \right]$$

- Coordinate ascent updates will take the form

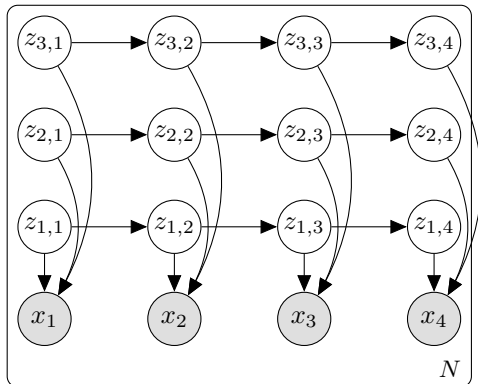
$$q(z_t; \lambda_t) \propto \exp \left( \mathbb{E}_{q(z_{-t}; \lambda_{-t})} [\log p(x, z; \theta)] \right)$$

where

$$\mathbb{E}_{q(z_{-t}; \lambda_{-t})} [\log p(x, z; \theta)] = \sum_{j \neq t} \prod_{j \neq t} q(z_j; \lambda_j) \log p(x, z; \theta)$$

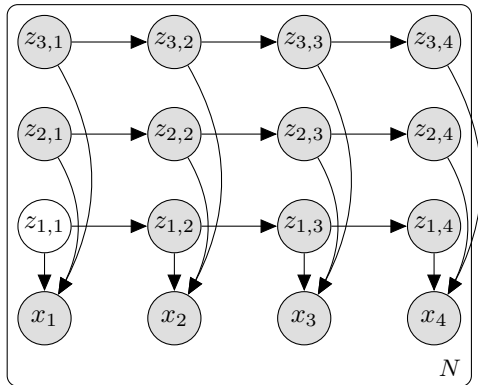
- Since  $p(z_t \mid x, z_{-t})$  was assumed to be in the exponential family, above updates can be derived in closed form.

## Example: Model 3 — Factorial HMM



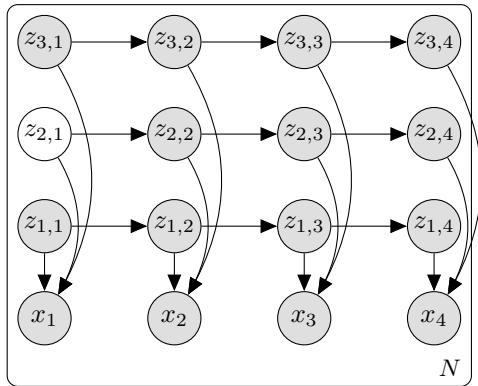
$$p(x, z; \theta) = \prod_{l=1}^L \prod_{t=1}^T p(z_{l,t} | z_{l,t-1}; \theta) p(x_t | z_{l,t}; \theta)$$

## Example: Model 3 — Factorial HMM



$$q(z_{1,1}; \lambda_{1,1}) \propto \exp \left( \mathbb{E}_{q(z_{-(1,1)}; \lambda_{-(1,1)})} [\log p(x, z; \theta)] \right)$$

## Example: Model 3 — Factorial HMM



$$q(z_{2,1}; \lambda_{2,1}) \propto \exp \left( \mathbb{E}_{q(z_{-(2,1)}; \lambda_{-(2,1)})} [\log p(x, z; \theta)] \right)$$

## Example: Model 3 — Factorial HMM

Introduction

Models

Variational  
ObjectiveInference  
Strategies

Exact Gradient

Sampling

Conjugacy

Advanced Topics

Case Studies

Conclusion

References

Exact Inference:

- Naive:  $K$  states,  $L$  levels  $\implies$  HMM with  $K^L$  states  $\implies O(TK^{2L})$
- Smarter:  $O(TLK^{L+1})$

Mean Field:

- Gaussian emissions:  $O(TLK^2)$  [Ghahramani and Jordan 1996].
- Categorical emission: need more variational approximations, but ultimately  $O(LKVT)$  [Nepal and Yates 2013].

## Example: Model 3 — Factorial HMM

Introduction

Models

Variational  
ObjectiveInference  
Strategies

Exact Gradient

Sampling

Conjugacy

Advanced Topics

Case Studies

Conclusion

References

Exact Inference:

- Naive:  $K$  states,  $L$  levels  $\implies$  HMM with  $K^L$  states  $\implies O(TK^{2L})$
- Smarter:  $O(TLK^{L+1})$

Mean Field:

- Gaussian emissions:  $O(TLK^2)$  [Ghahramani and Jordan 1996].
- Categorical emission: need more variational approximations, but ultimately  $O(LKVT)$  [Nepal and Yates 2013].

## ① Introduction

## ② Models

## ③ Variational Objective

## ④ Inference Strategies

## ⑤ Advanced Topics

Gumbel-Softmax

Flows

IWAE

## ⑥ Case Studies



## Advanced Topics

- 1 Gumbel-Softmax: Extend reparameterization to discrete variables.
- 2 Flows: Optimize a tighter bound by making the variational family  $q$  more flexible.
- 3 Importance Weighting: Optimize a tighter bound through importance sampling.

- 1 Introduction
- 2 Models
- 3 Variational Objective
- 4 Inference Strategies
- 5 **Advanced Topics**
  - Gumbel-Softmax
  - Flows
  - IWAE
- 6 Case Studies

## Challenges of Discrete Variables

## Introduction

## Models

Variational  
ObjectiveInference  
Strategies

## Advanced Topics

## Gumbel-Softmax

## Flows

## IWAE

## Case Studies

## Conclusion

## References

Review: we can always use score function estimator

$$\begin{aligned}\nabla_{\lambda} \text{ELBO}(x, \theta, \lambda) &= \mathbb{E}_q \left[ \log \frac{p(x, z; \theta)}{q(z | x; \lambda)} \nabla_{\lambda} \log q(z | x; \lambda) \right] \\ &= \mathbb{E}_q \left[ \left( \log \frac{p(x, z; \theta)}{q(z | x; \lambda)} - B \right) \nabla_{\lambda} \log q(z | x; \lambda) \right]\end{aligned}$$

- $\mathbb{E}_q[B \nabla_{\lambda} \log q(z | x; \lambda)] = 0$
- Control variate  $B$  (not dependent on  $z$ , but can depend on  $x$ ).
- Estimate this quantity with another neural net [Mnih and Gregor 2014]

$$\left( B(x; \psi) - \log \frac{p(x, z; \theta)}{q(z | x; \lambda)} \right)^2$$

## Challenges of Discrete Variables

## Introduction

## Models

Variational  
ObjectiveInference  
Strategies

## Advanced Topics

## Gumbel-Softmax

## Flows

## IWAE

## Case Studies

## Conclusion

## References

Review: we can always use score function estimator

$$\begin{aligned}\nabla_{\lambda} \text{ELBO}(x, \theta, \lambda) &= \mathbb{E}_q \left[ \log \frac{p(x, z; \theta)}{q(z | x; \lambda)} \nabla_{\lambda} \log q(z | x; \lambda) \right] \\ &= \mathbb{E}_q \left[ \left( \log \frac{p(x, z; \theta)}{q(z | x; \lambda)} - B \right) \nabla_{\lambda} \log q(z | x; \lambda) \right]\end{aligned}$$

- $\mathbb{E}_q[B \nabla_{\lambda} \log q(z | x; \lambda)] = 0$
- Control variate  $B$  (not dependent on  $z$ , but can depend on  $x$ ).
- Estimate this quantity with another neural net [Mnih and Gregor 2014]

$$\left( B(x; \psi) - \log \frac{p(x, z; \theta)}{q(z | x; \lambda)} \right)^2$$

## Gumbel-Softmax: Discrete Reparameterization [Jang et al. 2017; Maddison et al. 2017]

The “Gumbel-Max” trick [Papandreou and Yuille 2011]

$$p(z_k = 1; \alpha) = \frac{\alpha_k}{\sum_{j=1}^K \alpha_j}$$

where  $z = [0, 0, \dots, 1, \dots, 0]$  is a one-hot vector.

Can sample from  $p(z; \alpha)$  by

- 1 Drawing independent Gumbel noise  $\epsilon = \epsilon_1, \dots, \epsilon_K$

$$\epsilon_k = -\log(-\log u_k) \quad u_k \sim \mathcal{U}(0, 1)$$

- 2 Adding  $\epsilon_k$  to  $\log \alpha_k$ , finding  $\arg\max$ , i.e.

$$i = \arg \max_k [\log \alpha_k + \epsilon_k] \quad z_i = 1$$

## Gumbel-Softmax: Discrete Reparameterization [Jang et al. 2017; Maddison et al. 2017]

The “Gumbel-Max” trick [Papandreou and Yuille 2011]

$$p(z_k = 1; \alpha) = \frac{\alpha_k}{\sum_{j=1}^K \alpha_j}$$

where  $z = [0, 0, \dots, 1, \dots, 0]$  is a one-hot vector.

Can sample from  $p(z; \alpha)$  by

- 1 Drawing independent Gumbel noise  $\epsilon = \epsilon_1, \dots, \epsilon_K$

$$\epsilon_k = -\log(-\log u_k) \quad u_k \sim \mathcal{U}(0, 1)$$

- 2 Adding  $\epsilon_k$  to  $\log \alpha_k$ , finding  $\operatorname{argmax}$ , i.e.

$$i = \arg \max_k [\log \alpha_k + \epsilon_k] \quad z_i = 1$$

## Gumbel-Softmax: Discrete Reparameterization [Jang et al. 2017; Maddison et al. 2017]

The “Gumbel-Max” trick [Papandreou and Yuille 2011]

$$p(z_k = 1; \alpha) = \frac{\alpha_k}{\sum_{j=1}^K \alpha_j}$$

where  $z = [0, 0, \dots, 1, \dots, 0]$  is a one-hot vector.

Can sample from  $p(z; \alpha)$  by

- 1 Drawing independent Gumbel noise  $\epsilon = \epsilon_1, \dots, \epsilon_K$

$$\epsilon_k = -\log(-\log u_k) \quad u_k \sim \mathcal{U}(0, 1)$$

- 2 Adding  $\epsilon_k$  to  $\log \alpha_k$ , finding  $\operatorname{argmax}$ , i.e.

$$i = \arg \max_k [\log \alpha_k + \epsilon_k] \quad z_i = 1$$

## Gumbel-Softmax: Discrete Reparameterization [Jang et al. 2017; Maddison et al. 2017]

Reparameterization:

$$z = \arg \max_{s \in \Delta^{K-1}} (\log \alpha + \epsilon)^\top s = g(\epsilon, \alpha)$$

$z = g(\epsilon, \alpha)$  is a deterministic function applied to stochastic noise. Let's try applying this:

$$q(z_k = 1 \mid x; \lambda) = \frac{\alpha_k}{\sum_{j=1}^K \alpha_j} \quad \alpha = \text{enc}(x; \lambda)$$

(Recalling  $R_{\theta, \lambda}(z) = \log \frac{p(x, z; \theta)}{q(z \mid x; \lambda)}$ ),

$$\begin{aligned} \nabla_\lambda \mathbb{E}_{q(z \mid x; \lambda)} [R_{\theta, \lambda}(z)] &= \nabla_\lambda \mathbb{E}_{\epsilon \sim \text{Gumbel}} [R_{\theta, \lambda}(g(\epsilon, \alpha))] \\ &= \mathbb{E}_{\epsilon \sim \text{Gumbel}} [\nabla_\lambda R_{\theta, \lambda}(z)] \end{aligned}$$



## Gumbel-Softmax: Discrete Reparameterization [Jang et al. 2017; Maddison et al. 2017]

Reparameterization:

$$z = \arg \max_{s \in \Delta^{K-1}} (\log \alpha + \epsilon)^\top s = g(\epsilon, \alpha)$$

$z = g(\epsilon, \alpha)$  is a deterministic function applied to stochastic noise. Let's try applying this:

$$q(z_k = 1 \mid x; \lambda) = \frac{\alpha_k}{\sum_{j=1}^K \alpha_j} \quad \alpha = \text{enc}(x; \lambda)$$

(Recalling  $R_{\theta, \lambda}(z) = \log \frac{p(x, z; \theta)}{q(z \mid x; \lambda)}$ ),

$$\begin{aligned} \nabla_\lambda \mathbb{E}_{q(z \mid x; \lambda)}[R_{\theta, \lambda}(z)] &= \nabla_\lambda \mathbb{E}_{\epsilon \sim \text{Gumbel}}[R_{\theta, \lambda}(g(\epsilon, \alpha))] \\ &= \mathbb{E}_{\epsilon \sim \text{Gumbel}}[\nabla_\lambda R_{\theta, \lambda}(z)] \end{aligned}$$

## Gumbel-Softmax: Discrete Reparameterization [Jang et al. 2017; Maddison et al. 2017]

But this won't work, because zero gradients (almost everywhere)

$$z = g(\epsilon, \alpha) = \arg \max_{s \in \Delta^{K-1}} (\log \alpha + \epsilon)^\top s \implies \nabla_\lambda R_{\theta, \lambda}(z) = 0$$

Gumbel-Softmax trick: replace  $\arg \max$  with softmax

$$z = \text{softmax} \left( \frac{\log \alpha + \epsilon}{\tau} \right) \quad z_k = \frac{\exp((\log \alpha_k + \epsilon_k)/\tau)}{\sum_{j=1}^K \exp((\log \alpha_j + \epsilon_j)/\tau)}$$

$$\nabla_\lambda \mathbb{E}_{q(z|x; \lambda)} [R_{\theta, \lambda}(z)] \approx \mathbb{E}_{\epsilon \sim \text{Gumbel}} \left[ \nabla_\lambda R_{\theta, \lambda} \left( \text{softmax} \left( \frac{\log \alpha + \epsilon}{\tau} \right) \right) \right]$$

where  $\tau$  is a temperature term.

## Gumbel-Softmax: Discrete Reparameterization [Jang et al. 2017; Maddison et al. 2017]

But this won't work, because zero gradients (almost everywhere)

$$z = g(\epsilon, \alpha) = \arg \max_{s \in \Delta^{K-1}} (\log \alpha + \epsilon)^\top s \implies \nabla_\lambda R_{\theta, \lambda}(z) = 0$$

Gumbel-Softmax trick: replace arg max with softmax

$$z = \text{softmax} \left( \frac{\log \alpha + \epsilon}{\tau} \right) \quad z_k = \frac{\exp((\log \alpha_k + \epsilon_k)/\tau)}{\sum_{j=1}^K \exp((\log \alpha_j + \epsilon_j)/\tau)}$$

$$\nabla_\lambda \mathbb{E}_{q(z|x; \lambda)} [R_{\theta, \lambda}(z)] \approx \mathbb{E}_{\epsilon \sim \text{Gumbel}} \left[ \nabla_\lambda R_{\theta, \lambda} \left( \text{softmax} \left( \frac{\log \alpha + \epsilon}{\tau} \right) \right) \right]$$

where  $\tau$  is a temperature term.

## Gumbel-Softmax: Discrete Reparameterization [Jang et al. 2017; Maddison et al. 2017]

But this won't work, because zero gradients (almost everywhere)

$$z = g(\epsilon, \alpha) = \arg \max_{s \in \Delta^{K-1}} (\log \alpha + \epsilon)^\top s \implies \nabla_\lambda R_{\theta, \lambda}(z) = 0$$

Gumbel-Softmax trick: replace arg max with softmax

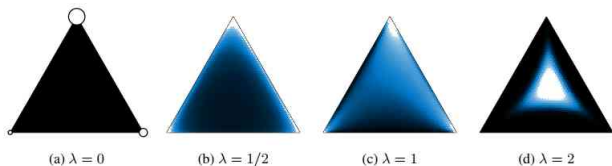
$$z = \text{softmax} \left( \frac{\log \alpha + \epsilon}{\tau} \right) \quad z_k = \frac{\exp((\log \alpha_k + \epsilon_k)/\tau)}{\sum_{j=1}^K \exp((\log \alpha_j + \epsilon_j)/\tau)}$$

$$\nabla_\lambda \mathbb{E}_{q(z|x; \lambda)} [R_{\theta, \lambda}(z)] \approx \mathbb{E}_{\epsilon \sim \text{Gumbel}} \left[ \nabla_\lambda R_{\theta, \lambda} \left( \text{softmax} \left( \frac{\log \alpha + \epsilon}{\tau} \right) \right) \right]$$

where  $\tau$  is a temperature term.

## Gumbel-Softmax: Discrete Reparameterization [Jang et al. 2017; Maddison et al. 2017]

- Approaches a discrete distribution as  $\tau \rightarrow 0$  (anneal  $\tau$  during training).
- Reparameterizable by construction
- Differentiable and has non-zero gradients



(from Maddison et al. [2017])

## Gumbel-Softmax: Discrete Reparameterization [Jang et al. 2017; Maddison et al. 2017]

- See Maddison et al. [2017] on whether we can use the original categorical densities  $p(z), q(z)$ , or need to use relaxed densities  $p_{\text{GS}}(z), q_{\text{GS}}(z)$ .
- Requires that  $p(x | z; \theta)$  “makes sense” for non-discrete  $z$  (e.g. attention).
- Lower-variance, but biased gradient estimator. Variance  $\rightarrow \infty$  as  $\tau \rightarrow 0$ .

## ① Introduction

## ② Models

## ③ Variational Objective

## ④ Inference Strategies

## ⑤ Advanced Topics

Gumbel-Softmax

Flows

IWAE

## ⑥ Case Studies

## Flows [Rezende and Mohamed 2015; Kingma et al. 2016]

## Recall

$$\log p(x; \theta) = \text{ELBO}(\theta, \lambda; x) - \text{KL}[q(z | x; \lambda) \| p(z | x; \theta)]$$

Bound is tight when variational posterior equals true posterior

$$q(z | x; \lambda) = p(z | x; \theta) \implies \log p(x; \theta) = \text{ELBO}(\theta, \lambda; x)$$

We want to make  $q(z | x; \lambda)$  as flexible as possible: can we do better than just Gaussian?



## Flows [Rezende and Mohamed 2015; Kingma et al. 2016]

Recall

$$\log p(x; \theta) = \text{ELBO}(\theta, \lambda; x) - \text{KL}[q(z | x; \lambda) \| p(z | x; \theta)]$$

Bound is tight when variational posterior equals true posterior

$$q(z | x; \lambda) = p(z | x; \theta) \implies \log p(x; \theta) = \text{ELBO}(\theta, \lambda; x)$$

We want to make  $q(z | x; \lambda)$  as flexible as possible: can we do better than just Gaussian?

## Flows [Rezende and Mohamed 2015; Kingma et al. 2016]

Introduction

Models

Variational  
ObjectiveInference  
Strategies

Advanced Topics

Gumbel-Softmax

**Flows**

IWAE

Case Studies

Conclusion

References

**Idea:** transform a sample from a simple initial variational distribution,

$$z_0 \sim q(z \mid x; \lambda) = \mathcal{N}(\mu, \sigma^2) \quad \mu, \sigma^2 = \text{enc}(x; \lambda)$$

into a more complex one

$$z_K = f_K \circ \cdots \circ f_2 \circ f_1(z_0; \lambda)$$

where  $f_i(z_{i-1}; \lambda)$ 's are **invertible** transformations (whose parameters are absorbed by  $\lambda$ ).

## Flows [Rezende and Mohamed 2015; Kingma et al. 2016]

## Introduction

## Models

## Variational

## Objective

## Inference

## Strategies

## Advanced Topics

## Gumbel-Softmax

## Flows

## IWAE

## Case Studies

## Conclusion

## References

**Idea:** transform a sample from a simple initial variational distribution,

$$z_0 \sim q(z | x; \lambda) = \mathcal{N}(\mu, \sigma^2) \quad \mu, \sigma^2 = \text{enc}(x; \lambda)$$

into a more complex one

$$z_K = f_K \circ \cdots \circ f_2 \circ f_1(z_0; \lambda)$$

where  $f_i(z_{i-1}; \lambda)$ 's are **invertible** transformations (whose parameters are absorbed by  $\lambda$ ).

## Flows [Rezende and Mohamed 2015; Kingma et al. 2016]

Sample from final variational posterior is given by  $z_K$ . Density is given by the change of variables formula:

$$\begin{aligned}\log q_K(z_K | x; \lambda) &= \log q(z_0 | x; \lambda) + \sum_{k=1}^K \log \left| \frac{\partial f_k^{-1}}{\partial z_k} \right| \\ &= \underbrace{\log q(z_0 | x; \lambda)}_{\text{log density of Gaussian}} - \sum_{k=1}^K \underbrace{\log \left| \frac{\partial f_k}{\partial z_{k-1}} \right|}_{\text{log determinant of Jacobian}}\end{aligned}$$

Determinant calculation is  $O(N^3)$  in general, but can be made faster depending on parameterization of  $f_k$

## Flows [Rezende and Mohamed 2015; Kingma et al. 2016]

Introduction

Models

Variational  
ObjectiveInference  
Strategies

Advanced Topics

Gumbel-Softmax

**Flows**

IWAE

Case Studies

Conclusion

References

Can still use reparameterization to obtain gradients. Letting

$$F(z) = f_K \circ \cdots \circ f_1(z),$$

$$\begin{aligned}\text{ELBO}(\theta, \lambda; x) &= \nabla_{\lambda} \mathbb{E}_{q_K(z_K | x; \lambda)} \left[ \log \frac{p(x, z; \theta)}{q_K(z_K | x; \lambda)} \right] \\ &= \nabla_{\lambda} \mathbb{E}_{q(z_0 | x; \lambda)} \left[ \log \frac{p(x, F(z_0); \theta)}{q(z_0 | x; \lambda)} - \log \left| \frac{\partial F}{\partial z_0} \right| \right] \\ &= \mathbb{E}_{\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[ \nabla_{\lambda} \left( \log \frac{p(x, F(z_0); \theta)}{q(z_0 | x; \lambda)} - \log \left| \frac{\partial F}{\partial z_0} \right| \right) \right]\end{aligned}$$

## Flows [Rezende and Mohamed 2015; Kingma et al. 2016]

## Introduction

## Models

Variational  
ObjectiveInference  
Strategies

## Advanced Topics

## Gumbel-Softmax

**Flows**

## IWAE

## Case Studies

## Conclusion

## References

Examples of  $f_k(z_{k-1}; \lambda)$ 

- Normalizing Flows [Rezende and Mohamed 2015]

$$f_k(z_{k-1}) = z_{k-1} + u_k h(w_k^\top z_{k-1} + b_k)$$

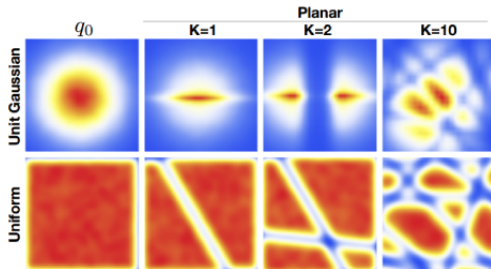
- Inverse Autoregressive Flows [Kingma et al. 2016]

$$f_k(z_{k-1}) = z_{k-1} \odot \sigma_k + \mu_k$$

$$\sigma_{k,d} = \text{sigmoid}(\text{NN}(z_{k-1,<d})) \quad \mu_{k,d} = \text{NN}(z_{k-1,<d})$$

(In this case the Jacobian is upper triangular, so determinant is just the product of diagonals)

## Flows [Rezende and Mohamed 2015; Kingma et al. 2016]



(from Rezende and Mohamed [2015])

- 1 Introduction
- 2 Models
- 3 Variational Objective
- 4 Inference Strategies
- 5 Advanced Topics**
  - Gumbel-Softmax
  - Flows
  - IWAE
- 6 Case Studies



## Importance Weighted Autoencoder (IWAE) [Burda et al. 2015]

- Flows are a way of tightening the ELBO by making the variational family more flexible.
- Not the only way: can obtain a tighter lower bound on  $\log p(x; \theta)$  by using multiple importance samples.

Consider:

$$I_K = \frac{1}{K} \sum_{k=1}^K \frac{p(x, z^{(k)}; \theta)}{q(z^{(k)} | x; \lambda)},$$

where each  $z^{(k)} \sim q(z | x; \lambda)$ .

Note that  $I_K$  is an unbiased estimator of  $p(x; \theta)$ :

$$\mathbb{E}_{q(z^{(1:K)} | x; \lambda)} [I_K] = p(x; \theta).$$

## Importance Weighted Autoencoder (IWAE) [Burda et al. 2015]

- Flows are a way of tightening the ELBO by making the variational family more flexible.
- Not the only way: can obtain a tighter lower bound on  $\log p(x; \theta)$  by using multiple importance samples.

Consider:

$$I_K = \frac{1}{K} \sum_{k=1}^K \frac{p(x, z^{(k)}; \theta)}{q(z^{(k)} | x; \lambda)},$$

where each  $z^{(k)} \sim q(z | x; \lambda)$ .

Note that  $I_K$  is an unbiased estimator of  $p(x; \theta)$ :

$$\mathbb{E}_{q(z^{(1:K)} | x; \lambda)} [I_K] = p(x; \theta).$$

## Importance Weighted Autoencoder (IWAE) [Burda et al. 2015]

## Introduction

## Models

Variational  
ObjectiveInference  
Strategies

## Advanced Topics

## Gumbel-Softmax

## Flows

## IWAE

## Case Studies

## Conclusion

## References

Any unbiased estimator of  $p(x; \theta)$  can be used to obtain a lower bound, using Jensen's inequality:

$$\begin{aligned} p(x; \theta) &= \mathbb{E}_{q(z^{(1:K)} | x; \lambda)} [I_K] \\ \implies \log p(x; \theta) &\geq \mathbb{E}_{q(z^{(1:K)} | x; \lambda)} [\log I_K] \\ &= \mathbb{E}_{q(z^{(1:K)} | x; \lambda)} \left[ \log \frac{1}{K} \sum_{k=1}^K \frac{p(x, z^{(k)}; \theta)}{q(z^{(k)} | x; \lambda)} \right] \end{aligned}$$

However, can also show [Burda et al. 2015]:

- $\log p(x; \theta) \geq \mathbb{E} [\log I_K] \geq \mathbb{E} [\log I_{K-1}]$
- $\lim_{K \rightarrow \infty} \mathbb{E} [\log I_K] = \log p(x; \theta)$  under mild conditions

## Importance Weighted Autoencoder (IWAE) [Burda et al. 2015]

Introduction

Models

Variational  
ObjectiveInference  
Strategies

Advanced Topics

Gumbel-Softmax

Flows

IWAE

Case Studies

Conclusion

References

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

- Note that with  $K = 1$ , we recover the ELBO.
- Can interpret  $\frac{p(x, z^{(k)}; \theta)}{q(z^{(k)} | x; \lambda)}$  as importance weights.
- If  $q(z | x; \lambda)$  is reparameterizable, we can use the reparameterization trick to optimize  $\mathbb{E} [\log I_K]$  directly.
- Otherwise, need score function gradient estimators [Mnih and Rezende 2016].

## ① Introduction

## ② Models

## ③ Variational Objective

## ④ Inference Strategies

## ⑤ Advanced Topics

## ⑥ Case Studies

Sentence VAE

Encoder/Decoder with Latent Variables

Latent Summaries and Topics

## ① Introduction

## ② Models

## ③ Variational Objective

## ④ Inference Strategies

## ⑤ Advanced Topics

## ⑥ Case Studies

### Sentence VAE

Encoder/Decoder with Latent Variables

Latent Summaries and Topics

## Sentence VAE Example [Bowman et al. 2016]

## Introduction

## Models

Variational  
ObjectiveInference  
Strategies

## Advanced Topics

## Case Studies

## Sentence VAE

Encoder/Decoder  
with Latent VariablesLatent Summaries  
and Topics

## Conclusion

## References

Generative Model (Model 2):

- Draw  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- Draw  $x_t \mid \mathbf{z} \sim \text{CRNNLM}(\theta, \mathbf{z})$

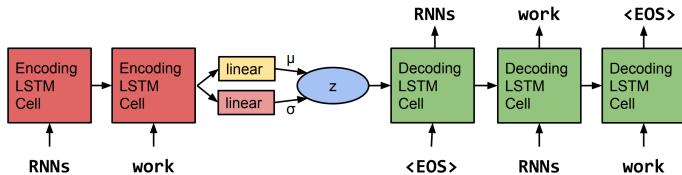
Variational Model (Amortized): Deep Diagonal Gaussians,

$$q(\mathbf{z} \mid x; \lambda) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$$

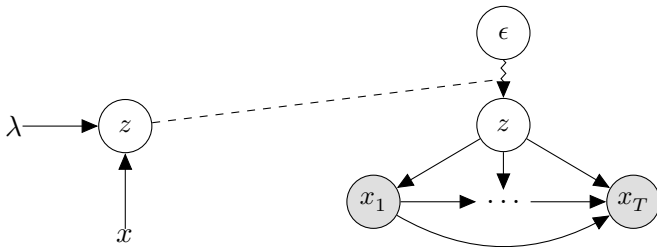
$$\tilde{\mathbf{h}}_T = \text{RNN}(x; \psi)$$

$$\boldsymbol{\mu} = \mathbf{W}_1 \tilde{\mathbf{h}}_T \quad \boldsymbol{\sigma}^2 = \exp(\mathbf{W}_2 \tilde{\mathbf{h}}_T) \quad \lambda = \{\mathbf{W}_1, \mathbf{W}_2, \psi\}$$

## Sentence VAE Example [Bowman et al. 2016]



(from Bowman et al. [2016])





## Issue 1: Posterior Collapse

$$\text{ELBO}(\theta, \lambda) = \mathbb{E}_{q(z|x; \lambda)} \left[ \log \frac{p(x, z; \theta)}{q(z|x; \lambda)} \right]$$

$$= \underbrace{\mathbb{E}_{q(z|x; \lambda)} [\log p(x|z; \theta)]}_{\text{Reconstruction likelihood}} - \underbrace{\text{KL}[q(z|x; \lambda) \| p(z)]}_{\text{Regularizer}}$$

Model	L/ELBO	Reconstruction	KL
RNN LM	-329.10	-	-
RNN VAE	-330.20	-330.19	0.01

(On Yahoo Corpus from Yang et al. [2017])

## Issue 1: Posterior Collapse

- $x$  and  $z$  become independent, and  $p(x, z; \theta)$  reduces to a non-LV language model.
- Chen et al. [2017]: If it's possible to model  $p_{\star}(x)$  without making use of  $z$ , then ELBO optimum is at:

$$p_{\star}(x) = p(x | z; \theta) = p(x; \theta) \quad q(z | x; \lambda) = p(z)$$

$$\text{KL}[q(z | x; \lambda) || p(z)] = 0$$

## Mitigating Posterior Collapse

Use less powerful likelihood models [Miao et al. 2016; Yang et al. 2017], or “word dropout” [Bowman et al. 2016].

Model	LL/ELBO	Reconstruction	KL
RNN LM	-329.1	-	-
RNN VAE	-330.2	-330.2	0.01
+ Word Drop	-334.2	-332.8	1.44
CNN VAE	-332.1	-322.1	10.0

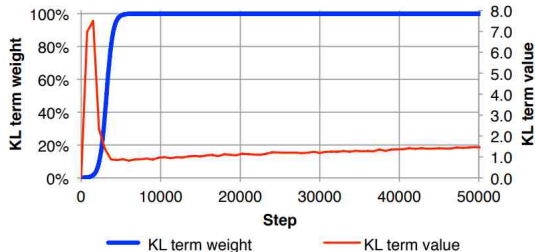
(On Yahoo Corpus from Yang et al. [2017])

## Mitigating Posterior Collapse

Gradually anneal multiplier on KL term, i.e.

$$\mathbb{E}_{q(z|x;\lambda)}[\log p(x|z;\theta)] - \beta \text{KL}[q(z|x;\lambda)||p(z)]$$

$\beta$  goes from 0 to 1 as training progresses



(from Bowman et al. [2016])

## Mitigating Posterior Collapse

## Introduction

## Models

Variational  
ObjectiveInference  
Strategies

## Advanced Topics

## Case Studies

## Sentence VAE

Encoder/Decoder  
with Latent VariablesLatent Summaries  
and Topics

## Conclusion

## References

Other approaches:

- Use auxiliary losses (e.g. train  $z$  as part of a topic model) [Dieng et al. 2017; Wang et al. 2018]
- Use von Mises–Fisher distribution with a fixed concentration parameter [Guu et al. 2017; Xu and Durrett 2018]
- Combine stochastic/amortized variational inference [Kim et al. 2018]
- Add skip connections [Dieng et al. 2018]

In practice, often necessary to combine various methods.

## Issue 2: Evaluation

- ELBO always lower bounds  $\log p(x; \theta)$ , so can calculate an upper bound on PPL efficiently.
- When reporting ELBO, should also separately report,

$$\text{KL}[q(z | x; \lambda) || p(z)]$$

to give an indication of how much the latent variable is being “used”.

## Issue 2: Evaluation

Also can evaluate  $\log p(x; \theta)$  with importance sampling

$$\begin{aligned} p(x; \theta) &= \mathbb{E}_{q(z|x; \lambda)} \left[ \frac{p(x|z; \theta)p(z)}{q(z|x; \lambda)} \right] \\ &\approx \frac{1}{K} \sum_{k=1}^K \frac{p(x|z^{(k)}; \theta)p(z^{(k)})}{q(z^{(k)}|x; \lambda)} \end{aligned}$$

So

$$\implies \log p(x; \theta) \approx \log \frac{1}{K} \sum_{k=1}^K \frac{p(x|z^{(k)}; \theta)p(z^{(k)})}{q(z^{(k)}|x; \lambda)}$$

## Evaluation

## Introduction

## Models

Variational  
ObjectiveInference  
Strategies

## Advanced Topics

## Case Studies

## Sentence VAE

Encoder/Decoder  
with Latent VariablesLatent Summaries  
and Topics

## Conclusion

## References

## Qualitative evaluation

- Evaluate samples from prior/variational posterior.
- Interpolation in latent space.

---

**i went to the store to buy some groceries .**  
*i store to buy some groceries .*  
*i were to buy any groceries .*  
*horses are to buy any groceries .*  
*horses are to buy any animal .*  
*horses the favorite any animal .*  
*horses the favorite favorite animal .*  
**horses are my favorite animal .**

---

(from Bowman et al. [2016])



## ① Introduction

## ② Models

## ③ Variational Objective

## ④ Inference Strategies

## ⑤ Advanced Topics

## ⑥ Case Studies

Sentence VAE

Encoder/Decoder with Latent Variables

Latent Summaries and Topics

## Encoder/Decoder [Sutskever et al. 2014; Cho et al. 2014]

### Introduction

### Models

### Variational Objective

### Inference Strategies

### Advanced Topics

### Case Studies

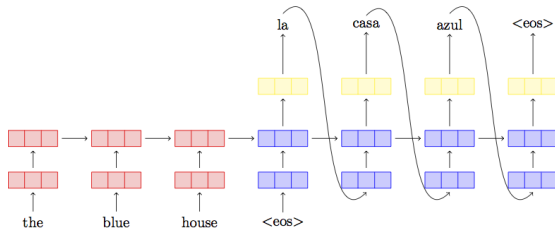
#### Sentence VAE

#### Encoder/Decoder with Latent Variables

#### Latent Summaries and Topics

### Conclusion

### References



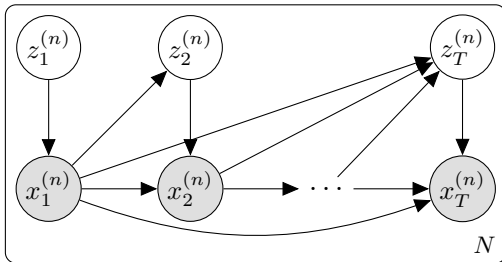
**Given:** Source information  $s = s_1, \dots, s_M$ .

**Generative process:**

- Draw  $x_{1:T} \mid s \sim \text{CRNNLM}(\theta, \text{enc}(s))$ .

**Generative process:** For  $t = 1, \dots, T$ ,

- Draw  $z_t \mid x_{<t}, s \sim \text{softmax}(U h_t)$ .
- Draw  $x_t \mid z_t, x_{<t}, s \sim \text{softmax}(W \tanh(Q_{z_t} h_t); \theta)$



If  $U \in \mathbb{R}^{K \times d}$ , used  $K$  experts; increases the flexibility of per-token distribution.

## Case-Study: Latent Per-token Experts [Yang et al. 2018]

## Introduction

## Models

Variational  
ObjectiveInference  
Strategies

## Advanced Topics

## Case Studies

## Sentence VAE

Encoder/Decoder  
with Latent VariablesLatent Summaries  
and Topics

## Conclusion

## References

**Learning:**  $z_t$  are independent given  $x_{<t}$ , so we can marginalize at each time-step (Method 3: Conjugacy).

$$\arg \max_{\theta} \log p(x \mid s; \theta) =$$

$$\arg \max_{\theta} \log \prod_{t=1}^T \sum_{k=1}^K p(z_t=k \mid s, x_{<t}; \theta) p(x_t \mid z_t=k, x_{<t}, s; \theta).$$

**Test-time:**

$$\arg \max_{x_{1:T}} \prod_{t=1}^T \sum_{k=1}^K p(z_t=k \mid s, x_{<t}; \theta) p(x_t \mid z_t=k, x_{<t}, s; \theta).$$

## Case-Study: Latent, Per-token Experts [Yang et al. 2018]

PTB language modeling results ( $s$  is constant):

Model	PPL
Merity et al. [2018]	57.30
Softmax-mixture [Yang et al. 2018]	54.44

Dialogue generation results ( $s$  is context):

Model	BLEU	
	Prec	Rec
No mixture	14.1	11.1
Softmax-mixture [Yang et al. 2018]	15.7	12.3

Introduction

Models

Variational  
ObjectiveInference  
Strategies

Advanced Topics

Case Studies

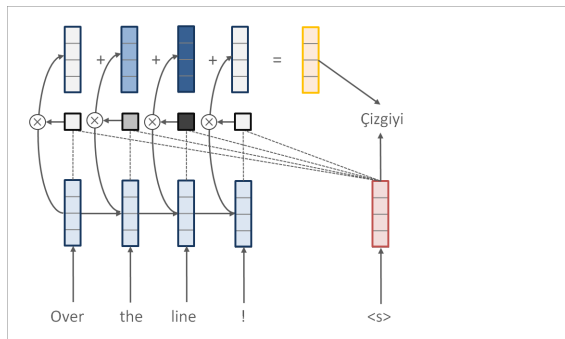
Sentence VAE

**Encoder/Decoder  
with Latent Variables**Latent Summaries  
and Topics

Conclusion

References

## Attention [Bahdanau et al. 2015]



Decoding with an attention mechanism:

$$x_t \mid x_{<t}, s \sim \text{softmax}(\mathbf{W}[\mathbf{h}_t, \sum_{m=1}^M \alpha_{t,m} \mathbf{enc}(s)_m]).$$

## Copy Attention [Gu et al. 2016; Gulcehre et al. 2016]

## Introduction

## Models

Variational  
ObjectiveInference  
Strategies

## Advanced Topics

## Case Studies

## Sentence VAE

Encoder/Decoder  
with Latent VariablesLatent Summaries  
and Topics

## Conclusion

## References

Copy attention models copying words directly from  $s$ .

**Generative process:** For  $t = 1, \dots, T$ ,

- Set  $\alpha_t$  to be attention weights.
- Draw  $z_t \mid x_{<t}, s \sim \text{Bern}(\text{MLP}([\mathbf{h}_t, \text{enc}(s)]))$ .
- If  $z_t = 0$ 
  - Draw  $x_t \mid z_t, x_{<t}, s \sim \text{softmax}(\mathbf{W}\mathbf{h}_t)$ .
- Else
  - Draw  $x_t \in \{s_1, \dots, s_M\} \mid z_t, x_{<t}, s \sim \text{Cat}(\alpha_t)$ .

## Copy Attention

## Introduction

## Models

Variational  
ObjectiveInference  
Strategies

## Advanced Topics

## Case Studies

## Sentence VAE

Encoder/Decoder  
with Latent VariablesLatent Summaries  
and Topics

## Conclusion

## References

**Learning:** Can maximize the log per-token marginal [Gu et al. 2016], as with per-token experts:

$$\begin{aligned} & \max_{\theta} \log p(x_1, \dots, x_T \mid s; \theta) \\ &= \max_{\theta} \log \prod_{t=1}^T \sum_{z' \in \{0,1\}} p(z_t = z' \mid s, x_{<t}; \theta) p(x_t \mid z', x_{<t}, s; \theta). \end{aligned}$$

**Test-time:**

$$\arg \max_{x_{1:T}} \prod_{t=1}^T \sum_{z' \in \{0,1\}} p(z_t = z' \mid s, x_{<t}; \theta) p(x_t \mid z', x_{<t}, s; \theta).$$



## Attention as a Latent Variable [Deng et al. 2018]

**Generative process:** For  $t = 1, \dots, T$ ,

- Set  $\alpha_t$  to be attention weights.
- Draw  $z_t \mid x_{<t}, s \sim \text{Cat}(\alpha_t)$ .
- Draw  $x_t \mid z_t, x_{<t}, s \sim \text{softmax}(\mathbf{W}[\mathbf{h}_t, \text{enc}(s_{z_t})]; \theta)$ .

## Attention as a Latent Variable [Deng et al. 2018]

## Introduction

## Models

Variational  
ObjectiveInference  
Strategies

## Advanced Topics

## Case Studies

## Sentence VAE

**Encoder/Decoder  
with Latent Variables**Latent Summaries  
and Topics

## Conclusion

## References

Marginal likelihood under latent attention model:

$$p(x_{1:T} \mid s; \theta) = \prod_{t=1}^T \sum_{m=1}^M \alpha_{t,m} \text{softmax}(\mathbf{W}[\mathbf{h}_t, \mathbf{enc}(s_m)]; \theta)_{x_t}.$$

Standard attention likelihood:

$$p(x_{1:T} \mid s; \theta) = \prod_{t=1}^T \text{softmax}(\mathbf{W}[\mathbf{h}_t, \sum_{m=1}^M \alpha_{t,m} \mathbf{enc}(s_m)]; \theta)_{x_t}.$$

## Attention as a Latent Variable [Deng et al. 2018]

## Introduction

## Models

Variational  
ObjectiveInference  
Strategies

## Advanced Topics

## Case Studies

## Sentence VAE

Encoder/Decoder  
with Latent VariablesLatent Summaries  
and Topics

## Conclusion

## References

**Learning Strategy #1:** Maximize the log marginal via enumeration as above.

**Learning Strategy #2:** Maximize the ELBO with AVI:

$$\max_{\lambda, \theta} \mathbb{E}_{q(z_t; \lambda)} [\log p(x_t | x_{<t}, z_t, s)] - \text{KL}[q(z_t; \lambda) \| p(z_t | x_{<t}, s)].$$

- $q(z_t | x; \lambda)$  approximates  $p(z_t | x_{1:T}, s; \theta)$ ; implemented with a BLSTM.
- $q$  isn't reparameterizable, so gradients obtained using REINFORCE + baseline.

## Attention as a Latent Variable [Deng et al. 2018]

## Introduction

## Models

Variational  
ObjectiveInference  
Strategies

## Advanced Topics

## Case Studies

## Sentence VAE

Encoder/Decoder  
with Latent VariablesLatent Summaries  
and Topics

## Conclusion

## References

**Test-time:** Calculate  $p(x_t | x_{<t}, s; \theta)$  by summing out  $z_t$ .

MT Results on IWSLT-2014:

Model	PPL	BLEU
Standard Attn	7.03	32.31
Latent Attn (marginal)	6.33	33.08
Latent Attn (ELBO)	6.13	33.09

## Encoder/Decoder with Structured Latent Variables

At least two EMNLP 2018 papers augment encoder/decoder text generation models with *structured* latent variables:

- 1 Lee et al. [2018] generate  $x_{1:T}$  by iteratively refining sequences of words  $z_{1:T}$ .
- 2 Wiseman et al. [2018] generate  $x_{1:T}$  conditioned on a latent template or plan  $z_{1:S}$ .

## ① Introduction

## ② Models

## ③ Variational Objective

## ④ Inference Strategies

## ⑤ Advanced Topics

## ⑥ Case Studies

Sentence VAE

Encoder/Decoder with Latent Variables

Latent Summaries and Topics

## Summary as a Latent Variable [Miao and Blunsom 2016]

Generative process for a document  $x = x_1, \dots, x_T$ :

- Draw a latent summary  $z_1, \dots, z_M \sim \text{RNNLM}(\theta)$
- Draw  $x_1, \dots, x_T \mid z_{1:M} \sim \text{CRNNLM}(\theta, z)$

Posterior Inference:

$$p(z_{1:M} \mid x_{1:T}; \theta) = p(\text{summary} \mid \text{document}; \theta).$$

## Summary as a Latent Variable [Miao and Blunsom 2016]

Generative process for a document  $x = x_1, \dots, x_T$ :

- Draw a latent summary  $z_1, \dots, z_M \sim \text{RNNLM}(\theta)$
- Draw  $x_1, \dots, x_T \mid z_{1:M} \sim \text{CRNNLM}(\theta, z)$

Posterior Inference:

$$p(z_{1:M} \mid x_{1:T}; \theta) = p(\text{summary} \mid \text{document}; \theta).$$



## Summary as a Latent Variable [Miao and Blunsom 2016]

**Learning:** Maximize the ELBO with amortized family:

$$\max_{\lambda, \theta} \mathbb{E}_{q(z_{1:M}; \lambda)} [\log p(x_{1:T} | z_{1:M}; \theta)] - \text{KL}[q(z_{1:M}; \lambda) || p(z_{1:M}; \theta)]$$

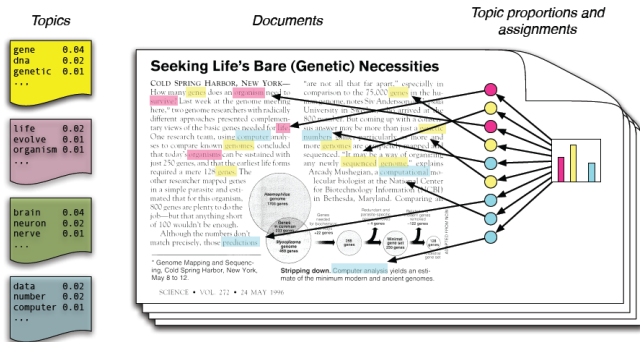
- $q(z_{1:M}; \lambda)$  approximates  $p(z_{1:M} | x_{1:T}; \theta)$ ; also implemented with encoder/decoder RNNs.
- $q(z_{1:M}; \lambda)$  not reparameterizable, so gradients use REINFORCE + baselines.

## Summary as a Latent Variable [Miao and Blunsom 2016]

**Semi-supervised Training:** Can also use documents *without* corresponding summaries in training.

- Train  $q(z_{1:M}; \lambda) \approx p(z_{1:M} | x_{1:T}; \theta)$  with labeled examples.
- Infer summary  $z$  for an *unlabeled* document with  $q$ .
- Use inferred  $z$  to improve model  $p(x_{1:T} | z_{1:M}; \theta)$ .
- Allows for outperforming strictly supervised models!

## Topic Models [Blei et al. 2003]



Generative process: for each document  $x^{(n)} = x_1^{(n)}, \dots, x_T^{(n)}$ ,

- Draw topic distribution  $\mathbf{z}_{top}^{(n)} \sim Dir(\alpha)$
- For  $t = 1, \dots, T$ :
  - Draw topic  $z_t^{(n)} \sim Cat(\mathbf{z}_{top}^{(n)})$
  - Draw  $x_t \sim Cat(\beta_{z_t^{(n)}})$

## Simple, Deep Topic Models [Miao et al. 2017]

**Motivation:** easy to learn deep topic models with VI if  $q(\mathbf{z}_{top}^{(n)}; \lambda)$  is reparameterizable.

**Idea:** draw  $\mathbf{z}_{top}^{(n)}$  from a transformation of a Gaussian.

- Draw  $\mathbf{z}_0^{(n)} \sim \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\sigma}_0^2)$
- Set  $\mathbf{z}_{top}^{(n)} = \text{softmax}(\mathbf{W}\mathbf{z}_0^{(n)})$ .
- Use analogous transformation when drawing from  $q(\mathbf{z}_{top}^{(n)}; \lambda)$ .

## Simple, Deep Topic Models [Miao et al. 2017]

## Introduction

## Models

Variational  
ObjectiveInference  
Strategies

## Advanced Topics

## Case Studies

## Sentence VAE

Encoder/Decoder  
with Latent VariablesLatent Summaries  
and Topics

## Conclusion

## References

**Learning Step #1:** Marginalize out per-word latents  $z_t^{(n)}$ .

$$p(\{x^{(n)}\}_{n=1}^N, \{\mathbf{z}_{top}^{(n)}\}_{n=1}^N; \theta) = \prod_{n=1}^N p(\mathbf{z}_{top}^{(n)} | \theta) \prod_{t=1}^T \sum_{k=1}^K z_{top,k}^{(n)} \beta_{k,x_t^{(n)}}$$

**Learning Step #2:** Use AVI to optimize resulting ELBO.

$$\max_{\lambda, \theta} \mathbb{E}_{q(\mathbf{z}_{top}^{(n)}; \lambda)} \left[ \log p(x^{(n)} | \mathbf{z}_{top}^{(n)}; \theta) \right] - \text{KL}[\mathcal{N}(\mathbf{z}_0^{(n)}; \lambda) \| \mathcal{N}(\mathbf{z}_0^{(n)}; \boldsymbol{\mu}_0, \boldsymbol{\sigma}_0^2)]$$

## Simple, Deep Topic Models [Miao et al. 2017]

Perplexities on held-out documents, for three datasets:

Model	MXM	20News	RCV1
OnlineLDA [Hoffman et al. 2010]	342	1015	1058
AVI-LDA [Miao et al. 2017]	272	830	602

① Introduction

② Models

③ Variational Objective

④ Inference Strategies

⑤ Advanced Topics

⑥ Case Studies

**⑦ Conclusion**

## Deep Latent-Variable NLP: Two Views

Introduction

Models

Variational  
Objective

Inference  
Strategies

Advanced Topics

Case Studies

Conclusion

References

Deep Models & LV Models are naturally **complementary**:

- Rich set of model choices: discrete, continuous, and structured.
- Real applications across NLP including some state-of-the-art models.

Deep Models & LV Models are frustratingly **incompatible**:

- Many interesting approaches to the problem: reparameterization, score-function, and more.
- Lots of area for research into improved approaches.



## Deep Latent-Variable NLP: Two Views

Introduction

Models

Variational  
Objective

Inference  
Strategies

Advanced Topics

Case Studies

Conclusion

References

Deep Models & LV Models are naturally **complementary**:

- Rich set of model choices: discrete, continuous, and structured.
- Real applications across NLP including some state-of-the-art models.

Deep Models & LV Models are frustratingly **incompatible**:

- Many interesting approaches to the problem: reparameterization, score-function, and more.
- Lots of area for research into improved approaches.

## Implementation

- Modern toolkits make it easy to implement these models.
- Combine the flexibility of auto-differentiation for optimization (PyTorch) with distribution and VI libraries (Pyro).

In fact, we have implemented this entire tutorial. See website link:

<http://bit.do/lvnlp>

## Implementation

- Modern toolkits make it easy to implement these models.
- Combine the flexibility of auto-differentiation for optimization (PyTorch) with distribution and VI libraries (Pyro).

In fact, we have implemented this entire tutorial. See website link:

<http://bit.do/lvnlp>

Charu C. Aggarwal and ChengXiang Zhai. 2012. A Survey of Text Clustering Algorithms. In *Mining Text Data*, pages 77–128. Springer.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *Proceedings of ICLR*.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyal, Andrew M. Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating Sentences from a Continuous Space. In *Proceedings of CoNLL*.

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based N-gram Models of Natural Language. *Computational linguistics*, 18(4):467–479.

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. 2015. Importance Weighted Autoencoders. In *Proceedings of ICLR*.

Xi Chen, Diederik P. Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2017. Variational Lossy Autoencoder. In *Proceedings of ICLR*.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. In *Proceedings of the Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. 1977. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38.

Yuntian Deng, Yoon Kim, Justin Chiu, Demi Guo, and Alexander M. Rush. 2018. Latent Alignment and Variational Attention. In *Proceedings of NIPS*.

Adji B. Dieng, Yoon Kim, Alexander M. Rush, and David M. Blei. 2018. Avoiding Latent Variable Collapse with Generative Skip Models. In *Proceedings of the ICML Workshop on Theoretical Foundations and Applications of Deep Generative Models*.

Adji B. Dieng, Chong Wang, Jianfeng Gao, and John Paisley. 2017. TopicRNN: A Recurrent Neural Network With Long-Range Semantic Dependency. In *Proceedings of ICLR*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12.

Jason Eisner. 2016. Inside-Outside and Forward-Backward Algorithms Are Just Backprop (tutorial paper). In *Proceedings of the Workshop on Structured Prediction for NLP*.

Ekaterina Garmash and Christof Monz. 2016. Ensemble learning for multi-source neural machine translation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1409–1418.

Zoubin Ghahramani and Michael I. Jordan. 1996. Factorial Hidden Markov Models. In *Proceedings of NIPS*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1631–1640.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 140–149.

Kelvin Guu, Tatsunori B. Hashimoto, Yonatan Oren, and Percy Liang. 2017. Generating Sentences by Editing Prototypes. *arXiv:1709.08878*.

William P Headden III, Mark Johnson, and David McClosky. 2009. Improving Unsupervised Dependency Parsing with Richer Contexts and Smoothing. In *Proceedings of NAACL*.

Marti A Hearst. 1997. Texttiling: Segmenting text into multi-paragraph subtopic passages. *Computational linguistics*, 23(1):33–64.

Matthew Hoffman, Francis R Bach, and David M Blei. 2010. Online learning for latent dirichlet allocation. In *advances in neural information processing systems*, pages 856–864.

- Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward Controlled Generation of Text. In *Proceedings of ICML*.
- Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. 1991. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87.
- Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical Reparameterization with Gumbel-Softmax. In *Proceedings of ICLR*.
- Yoon Kim, Sam Wiseman, Andrew C. Miller, David Sontag, and Alexander M. Rush. 2018. Semi-Amortized Variational Autoencoders. In *Proceedings of ICML*.
- Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *Proceedings of ICLR*.
- Diederik P. Kingma, Tim Salimans, and Max Welling. 2016. Improving Variational Inference with Autoregressive Flow. *arXiv:1606.04934*.
- Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *Proceedings of ICLR*.
- Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought Vectors. In *Proceedings of NIPS*.
- Dan Klein and Christopher D Manning. 2004. Corpus-based Induction of Syntactic Structure: Models of Dependency and Constituency. In *Proceedings of ACL*.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic Non-Autoregressive Neural Sequence Modeling by Iterative Refinement. In *Proceedings of EMNLP*.

Stefan Lee, Senthil Purushwalkam Shiva Prakash, Michael Cogswell, Viresh Ranjan, David Crandall, and Dhruv Batra. 2016. Stochastic Multiple Choice Learning for Training Diverse Deep Ensembles. In *Proceedings of NIPS*.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. 2017. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. In *Proceedings of ICLR*.

Bernard Merialdo. 1994. Tagging english text with a probabilistic model. *Computational linguistics*, 20(2):155–171.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. Regularizing and optimizing LSTM language models. In *International Conference on Learning Representations*.

Yishu Miao and Phil Blunsom. 2016. Language as a Latent Variable: Discrete Generative Models for Sentence Compression. In *Proceedings of EMNLP*.

Yishu Miao, Edward Grefenstette, and Phil Blunsom. 2017. Discovering Discrete Latent Topics with Neural Variational Inference. In *Proceedings of ICML*.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural Variational Inference for Text Processing. In *Proceedings of ICML*.



Andriy Mnih and Danilo J. Rezende. 2016. Variational Inference for Monte Carlo Objectives. In *Proceedings of ICML*.

Andriy Mnih and Karol Gregor. 2014. Neural Variational Inference and Learning in Belief Networks. In *Proceedings of ICML*.

Anjan Nepal and Alexander Yates. 2013. Factorial Hidden Markov Models for Learning Representations of Natural Language. *arXiv:arXiv:1312.6168*.

George Papandreou and Alan L. Yuille. 2011. Perturb-and-Map Random Fields: Using Discrete Optimization to Learn and Sample from Energy Models. In *Proceedings of ICCV*.

Danilo J. Rezende and Shakir Mohamed. 2015. Variational Inference with Normalizing Flows. In *Proceedings of ICML*.

Noah A Smith and Jason Eisner. 2005. Contrastive Estimation: Training log-linear models on unlabeled data. In *Proceedings of ACL*, pages 354–362. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Proceedings of NIPS*.

Ke Tran, Yonatan Bisk, Ashish Vaswani, Daniel Marcu, and Kevin Knight. 2016. Unsupervised Neural Hidden Markov Models. In *Proceedings of the Workshop on Structured Prediction for NLP*.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. Hmm-based word alignment in statistical translation. In *Proceedings of the 16th conference on Computational linguistics-Volume 2*, pages 836–841. Association for Computational Linguistics.

Wenlin Wang, Zhe Gan, Wenqi Wang, Dinghan Shen, Jiaji Huang, Wei Ping, Sanjeev Satheesh, and Lawrence Carin. 2018. Topic Compositional Neural Language Model. In *Proceedings of AISTATS*.

Peter Willett. 1988. Recent Trends in Hierarchic Document Clustering: A Critical Review. *Information Processing & Management*, 24(5):577–597.

Ronald J. Williams. 1992. Simple Statistical Gradient-following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*, 8.

Sam Wiseman, Stuart M Shieber, and Alexander M Rush. 2018. Learning neural templates for text generation. In *Proceedings of EMNLP*.

Jiacheng Xu and Greg Durrett. 2018. Spherical Latent Spaces for Stable Variational Autoencoders. In *Proceedings of EMNLP*.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W. Cohen. 2018. Breaking the Softmax Bottleneck: A High-Rank RNN Language Model. In *Proceedings of ICLR*.

Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. 2017. Improved Variational Autoencoders for Text Modeling using Dilated Convolutions. In *Proceedings of ICML*.

Introduction

Models

Variational  
Objective

Inference  
Strategies

Advanced Topics

Case Studies

Conclusion

References