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
Introduction

Goals

Background

Models

Variational Objective

Inference Strategies

Advanced Topics

Case Studies

Conclusion

References
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

- 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

- 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

7 Conclusion
What are deep networks?

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

Important examples: The multilayer perceptron,

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

The recurrent neural network, which maps a sequence of inputs $z_{1:T}$ into a sequence of features $h_{1:T}$,

$$h_t = \text{RNN}(h_{t-1}, z_t; \pi) = \sigma(Uz_t + Vh_{t-1} + b) \quad \pi = \{V, U, b\}.$$
What are deep networks?

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

Important examples: The multilayer perceptron,

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

The recurrent neural network, which maps a sequence of inputs $z_{1:T}$ into a sequence of features $h_{1:T}$,

$$ h_t = \text{RNN}(h_{t-1}, z_t; \pi) = \sigma(Uz_t + Vh_{t-1} + b) \quad \pi = \{V, U, b\}.$$
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,

\[ p(x^{(1:N)}, z^{(1:N)}; \theta) = \prod_{n=1}^{N} p(x^{(n)} | z^{(n)}; \theta) p(z^{(n)}; \theta). \]
A directed PGM shows the conditional independence structure. By chain rule, latent variable model over observations can be represented as,

\[ p(x^{(1:N)}, z^{(1:N)}; \theta) = \prod_{n=1}^{N} p(x^{(n)} \mid z^{(n)}; \theta)p(z^{(n)}; \theta) \]

Specific models may factor further.
Posterior Inference

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

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

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 $p(x^{(n)} | z^{(n)}; \theta)$.  

Posterior Inference

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

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

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 $p(x^{(n)} \mid z^{(n)}; \theta)$. 


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.
Introduction

Models

- Discrete Models
- Continuous Models
- Structured Models

Variational Objective

Inference Strategies

Advanced Topics

Case Studies

Conclusion

References
A Language Model

Our goal is to model a sentence, \( x_1 \ldots x_T \).

Context: RNN language models are remarkable at this task,

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

Defined as,

\[
p(x_{1:T}) = \prod_{t=1}^{T} p(x_t | x_{<t}) = \prod_{t=1}^{T} \text{softmax}(Wh_t)x_t
\]

where \( h_t = \text{RNN}(h_{t-1}, x_{t-1}; \theta) \)
A Language Model

Our goal is to model a sentence, \( x_1 \ldots x_T \).

Context: RNN language models are remarkable at this task,

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

Defined as,

\[
p(x_{1:T}) = \prod_{t=1}^{T} p(x_t \mid x_{<t}) = \prod_{t=1}^{T} \text{softmax}(W h_t) x_t
\]

where \( h_t = \text{RNN}(h_{t-1}, x_{t-1}; \theta) \)
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*)
Model 1: Discrete Clustering

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

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

Generative process:

1. Draw cluster $z \in \{1, \ldots, K\}$ 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:

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

$$= \mu_z \times \prod_{t=1}^{T} \pi_{z,x_t}$$
Model 1: Graphical Model View

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

\[
= \prod_{n=1}^{N} \mu_{z^{(n)}} \times \prod_{t=1}^{T} \pi_{z^{(n)}, x_{t}^{(n)}}
\]
Deep Model 1: Discrete - Mixture of RNNs

Generative process:

1. Draw cluster $z \in \{1, \ldots, K\}$ 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:

\[
p(z \mid x; \theta) = \frac{p(z) \times p(x \mid z)}{p(x)}
\]

\[
= \frac{p(z) \times p(x \mid z)}{K \sum_{k=1}^{K} p(z=k) \times p(x \mid z=k)}
\]

- 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:

\[
p(z \mid x; \theta) = \frac{p(z) \times p(x \mid z)}{p(x)} = \frac{p(z) \times p(x \mid z)}{K \sum_{k=1}^{K} p(z=k) \times p(x \mid z=k)}
\]

- 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:

\[
p(z \mid x; \theta) = \frac{p(z) \times p(x \mid z)}{p(x)} \]

\[
= \frac{p(z) \times p(x \mid z)}{\sum_{k=1}^{K} p(z=k) \times p(x \mid z=k)}
\]

- 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
Model 2: Continuous / Dimensionality Reduction

Inference Process:

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

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 $z$ from Normal with param $\mu$.
2. For each $t$, draw word $x_t$ from categorical with param $\text{softmax}(Wz)$.

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

Intuition: $\mu$ is a global distribution, $z$ 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 $\mathbf{z} \sim \mathcal{N}(\mathbf{\mu}, \mathbf{I})$.
2. Draw each token $x_t$ from a conditional RNNLM.

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

$$p(x, \mathbf{z}; \pi, \mathbf{\mu}, \mathbf{I}) = p(\mathbf{z}; \mathbf{\mu}, \mathbf{I}) \times p(x | \mathbf{z}; \pi)$$

$$= \mathcal{N}(\mathbf{z}; \mathbf{\mu}, \mathbf{I}) \times \text{CRNNLM}(x_{1:T}; \pi, \mathbf{z})$$

where

$$\text{CRNNLM}(x_{1:T}; \pi, \mathbf{z}) = \prod_{t=1}^{T} \text{softmax}(W h_t)_{x_t}$$

$$h_t = \text{RNN}(h_{t-1}, [x_{t-1}; \mathbf{z}]; \pi)$$
Deep Model 2: Continuous "Mixture" of RNNs

Generative Process:

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

RNN is also conditioned on latent $z$,

$$ p(x, z; \pi, \mu, I) = p(z; \mu, I) \times p(x | z; \pi) $$

$$ = \mathcal{N}(z; \mu, I) \times \text{CRNNLM}(x_{1:T}; \pi, z) $$

where

$$ \text{CRNNLM}(x_{1:T}; \pi, z) = \prod_{t=1}^{T} \text{softmax}(W h_t) x_t $$

$$ h_t = \text{RNN}(h_{t-1}, [x_{t-1}; z]; \pi) $$
Deep Model 2: Continuous "Mixture" of RNNs

Generative Process:

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

RNN is also conditioned on latent $z$,

$$p(x, z; \pi, \mu, I) = p(z; \mu, I) \times p(x | z; \pi)$$

$$= \mathcal{N}(z; \mu, I) \times \text{CRNNLM}(x_{1:T}; \pi, z)$$

where

$$\text{CRNNLM}(x_{1:T}; \pi, z) = \prod_{t=1}^{T} \text{softmax}(W h_t) x_t$$

$$h_t = \text{RNN}(h_{t-1}, [x_{t-1}; z]; \pi)$$
Graphical Model View
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,

\[
p(z \mid x; \theta) = \frac{p(z; \mu) \times p(x \mid z; \pi)}{\int_{z} p(z; \mu) \times p(x \mid z; \pi) \, dz}
\]

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

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

Gives rise to the joint distribution:

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

$$= \prod_{t=1}^{T} \mu_{z_{t-1}, z_t} \times \prod_{t=1}^{T} \pi_{z_t, x_t}$$
Model 3: Structured - Hidden Markov Model

Generative Process:

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

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

Gives rise to the joint distribution:

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

$$= \prod_{t=1}^{T} \mu_{z_{t-1}, z_t} \times \prod_{t=1}^{T} \pi_{z_t, x_t}$$
Graphical Model View

\[
p(x, z; \theta) = \prod_{t=1}^{T} p(z_t | z_{t-1}; \mu_{z_{t-1}}) \times \prod_{t=1}^{T} p(x_t | z_t; \pi_{z_t})
\]

\[
= \prod_{t=1}^{T} \mu_{z_{t-1}, z_t} \times \prod_{t=1}^{T} \pi_{z_t, x_t}
\]
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

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

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

\[
p(x, z; \theta) = \prod_{t=1}^{T} p(z_t \mid z_{t-1}; \mu_{z_{t-1}}) \times \prod_{t=1}^{T} p(x_t \mid z_t; \pi_{z_t}) = \prod_{t=1}^{T} \mu_{z_{t-1},z_t} \times \prod_{t=1}^{T} \pi_{z_t,x_t}
\]
Posterior Inference

For structured models, Bayes’ rule may tractable,

\[
p(z \mid x; \theta) = \frac{p(z; \mu) \times p(x \mid z; \pi)}{\sum_{z'} p(z'; \mu) \times p(x \mid z'; \pi)}
\]

- 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

For structured models, Bayes’ rule may tractable,

\[ p(z \mid x; \theta) = \frac{p(z; \mu) \times p(x \mid z; \pi)}{\sum_{z'} p(z'; \mu) \times p(x \mid z'; \pi)} \]

- 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.
Introduction

Models

Variational Objective

Maximum Likelihood

ELBO

Inference Strategies

Advanced Topics

Case Studies

Conclusion

References
1 Introduction

2 Models

3 Variational Objective
   Maximum Likelihood
   ELBO

4 Inference Strategies

5 Advanced Topics

6 Case Studies

7 Conclusion
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,

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

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,

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

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

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

Models

Variational Objective

Maximum Likelihood

ELBO

Inference Strategies

Advanced Topics

Case Studies

Conclusion

References
Variational Inference

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

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

Provides framework for deriving a rich set of optimization algorithms.
Marginal Likelihood: Variational Decomposition

For any\(^1\) distribution \(q(z \mid x; \lambda)\) over \(z\),

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

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

\(^1\)Technical condition: \(\text{supp}(q(z)) \subset \text{supp}(p(z \mid x; \theta))\)
Evidence Lower Bound: Proof

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

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

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

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

\[
= \mathbb{E}_q \log \frac{p(x, z; \theta)}{q(z | x; \lambda)} + \text{KL}[q(z | x; \lambda) || p(z | x; \theta)]
\]
Evidence Lower Bound: Proof

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

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

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

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

\[ = \mathbb{E}_q \log \frac{p(x, z; \theta)}{q(z | x; \lambda)} + \text{KL}[q(z | x; \lambda) \| p(z | x; \theta)] \]
Evidence Lower Bound: Proof

\[
\log p(x; \theta) = \mathbb{E}_q \log p(x) \quad \text{(Expectation over } z) \\
= \mathbb{E}_q \log \frac{p(x, z)}{p(z \mid x)} \quad \text{(Mult/div by } p(z \mid x), \text{ combine numerator)} \\
= \mathbb{E}_q \log \left( \frac{p(x, z)}{q(z \mid x) p(z \mid x)} \frac{q(z \mid x)}{q(z \mid x) p(z \mid x)} \right) \quad \text{(Mult/div by } q(z \mid x)) \\
= \mathbb{E}_q \log \frac{p(x, z)}{q(z \mid x)} + \mathbb{E}_q \log \frac{q(z \mid x)}{p(z \mid x)} \quad \text{(Split Log)} \\
= \mathbb{E}_q \log \frac{p(x, z; \theta)}{q(z \mid x; \lambda)} + \text{KL}[q(z \mid x; \lambda) \parallel p(z \mid x; \theta)]
\]
Evidence Lower Bound: Proof

\[
\log p(x; \theta) = \mathbb{E}_q \log p(x) \quad \text{(Expectation over } z) \\
= \mathbb{E}_q \log \frac{p(x, z)}{p(z | x)} \quad \text{(Mult/div by } p(z | x), \text{ combine numerator)} \\
= \mathbb{E}_q \log \left( \frac{p(x, z) q(z | x)}{q(z | x) p(z | x)} \right) \quad \text{(Mult/div by } q(z | x)) \\
= \mathbb{E}_q \log \frac{p(x, z)}{q(z | x)} + \mathbb{E}_q \log \frac{q(z | x)}{p(z | x)} \quad \text{(Split Log)} \\
= \mathbb{E}_q \log \frac{p(x, z; \theta)}{q(z | x; \lambda)} + \text{KL}[q(z | x; \lambda) \parallel p(z | x; \theta)]
\]
Evidence Lower Bound: Proof

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

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

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

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

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

$$\sum_{n=1}^{N} \log p(x^{(n)}; \theta) \geq \sum_{n=1}^{N} \text{ELBO}(\theta, \lambda; x^{(n)})$$

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

$$= \text{ELBO}(\theta, \lambda; x^{(1:N)}) = \text{ELBO}(\theta, \lambda)$$
Setup: Selecting Variational Family

- 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

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

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

$$\arg \max_{\theta, \lambda} \text{ELBO}(\theta, \lambda) = \arg \max_{\theta, \lambda} \sum_{n=1}^{N} \text{ELBO}(\theta, \lambda; x^{(n)})$$

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

\[
\arg \max_{\theta, \lambda} \text{ELBO}(\theta, \lambda) = \arg \max_{\theta, \lambda} \sum_{n=1}^{N} \text{ELBO}(\theta, \lambda; x^{(n)})
\]

\[
= \arg \max_{\theta, \lambda} \sum_{n=1}^{N} \mathbb{E}_{q} \left[ \log \frac{p(x^{(n)}, z^{(n)}; \theta)}{q(z^{(n)} | x^{(n)}; \lambda)} \right]
\]
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

Easy: Gradient respect to $\theta$

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

- 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

Easy: Gradient respect to $\theta$

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

- 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

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

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

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

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 $\lambda$

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

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

- 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
Tutorial: 
Deep Latent NLP 
(bit.do/lvnlp)

Introduction
Models
Variational 
Objective
Inference 
Strategies
Exact Gradient 
Sampling 
Conjugacy
Advanced Topics
Case Studies
Conclusion
References

Model Inference: Gradient Ascent on $\lambda$?

Hard: Gradient respect to $\lambda$

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

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

• 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
Strategy 1: Exact Gradient

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

- 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

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

- Naive enumeration: Linear in \(|Z|\).
- 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 \( q(z \mid x; \lambda) = \text{Cat}(\nu) \) where \( \nu = \text{enc}(x; \lambda) \)

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

= \nabla_\lambda \left( \sum_{z \in \mathcal{Z}} q(z \mid x; \lambda) \log \frac{p(x, z; \theta)}{q(z \mid x; \lambda)} \right)

= \nabla_\lambda \left( \sum_{z \in \mathcal{Z}} \nu_z \log \frac{p(x, z; \theta)}{\nu_z} \right)
\]
Example: Model 1 - Naive Bayes

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

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

$$= \nabla_\lambda \left( \sum_{z \in \mathcal{Z}} q(z \mid x; \lambda) \log \frac{p(x, z; \theta)}{q(z \mid x; \lambda)} \right)$$

$$= \nabla_\lambda \left( \sum_{z \in \mathcal{Z}} \nu_z \log \frac{p(x, z; \theta)}{\nu_z} \right)$$
1 Introduction

2 Models

3 Variational Objective

4 Inference Strategies
   - Exact Gradient
   - Sampling
   - Conjugacy

5 Advanced Topics

6 Case Studies
Strategy 2: Sampling

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

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

\[ z^{(1)}, \ldots, z^{(J)} \sim q(z \mid x; \lambda) \]

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

- Manipulate expression so we can move \( \nabla_\lambda \) inside \( \mathbb{E}_q \) before sampling.
Strategy 2: Sampling

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

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

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

\[ z^{(1)}, \ldots, z^{(J)} \sim q(z | x; \lambda) \]

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

- 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]

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

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

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

$$= \sum_z q(z | x; \lambda) \nabla_\lambda \log q(z | x; \lambda) \log p(x, z; \theta)$$
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]

\[
\nabla_{\lambda} \mathbb{E}_q \left[ \log p(x, z; \theta) \right] = \sum_z \nabla_{\lambda} q(z | x; \lambda) \log p(x, z; \theta) \\
= \sum_z q(z | x; \lambda) \nabla_{\lambda} \log q(z | x; \lambda) \log p(x, z; \theta) \\
= \mathbb{E}_q \left[ \log p(x, z; \theta) \nabla_{\lambda} \log q(z | x; \lambda) \right]
\]
Strategy 2a: Sampling — Score Function Gradient Estimator

Second term. Need additional identity:

\[ \sum \nabla q = \nabla \sum q = \nabla 1 = 0 \]

\[ \nabla \lambda E_q \left[ \log q(z \mid x; \lambda) \right] = \sum_z \nabla_\lambda \left( q(z \mid x; \lambda) \log q(z \mid x; \lambda) \right) \]
Strategy 2a: Sampling — Score Function Gradient Estimator

Second term. Need additional identity:

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

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

$$= \sum_{z} \left( \frac{\nabla_{\lambda} q(z | x; \lambda)}{q \nabla \log q} \right) \log q(z | x; \lambda) + q(z | x; \lambda) \left( \frac{\nabla_{\lambda} \log q(z | x; \lambda)}{\frac{\nabla q}{q}} \right)$$
Strategy 2a: Sampling — Score Function Gradient Estimator

Second term. Need additional identity:

\[ \sum \nabla q = \nabla \sum q = \nabla 1 = 0 \]

\[
\nabla_\lambda \mathbb{E}_q \left[ \log q(z \mid x; \lambda) \right] = \sum_z \nabla_\lambda \left( q(z \mid x; \lambda) \log q(z \mid x; \lambda) \right) \\
= \sum_z \log q(z \mid x; \lambda) q(z \mid x; \lambda) \nabla_\lambda \log q(z \mid x; \lambda) + \sum_z \nabla_\lambda q(z \mid x; \lambda)
\]
Strategy 2a: Sampling — Score Function Gradient Estimator

Second term. Need additional identity:

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

$$\nabla_{\lambda} \mathbb{E}_{q} \left[ \log q(z \mid x; \lambda) \right] = \sum_{z} \nabla_{\lambda} \left( q(z \mid x; \lambda) \log q(z \mid x; \lambda) \right)$$

$$= \sum_{z} \log q(z \mid x; \lambda) q(z \mid x; \lambda) \nabla_{\lambda} \log q(z \mid x; \lambda) + \sum_{z} \nabla_{\lambda} q(z \mid x; \lambda)$$

$$= \nabla \sum q = \nabla 1 = 0$$
Strategy 2a: Sampling — Score Function Gradient Estimator

Second term. Need additional identity:

\[ \sum \nabla q = \nabla \sum q = \nabla 1 = 0 \]

\[
\nabla_\lambda \mathbb{E}_q \left[ \log q(z \mid x; \lambda) \right] = \sum_z \nabla_\lambda \left( q(z \mid x; \lambda) \log q(z \mid x; \lambda) \right) \\
= \sum_z \log q(z \mid x; \lambda) q(z \mid x; \lambda) \nabla_\lambda \log q(z \mid x; \lambda) + \sum_z \nabla_\lambda q(z \mid x; \lambda) \\
= \mathbb{E}_q [\log q(z \mid x; \lambda) \nabla_\lambda q(z \mid x; \lambda)]
\]
Strategy 2a: Sampling — Score Function Gradient Estimator

Putting these together,

\[
\nabla_\lambda \text{ELBO}(\theta, \lambda; x) = \nabla_\lambda \mathbb{E}_q \left[ \log \frac{p(x, z; \theta)}{q(z \mid x; \lambda)} \right] \\
= \mathbb{E}_q \left[ \log \frac{p(x, z; \theta)}{q(z \mid x; \lambda)} \nabla_\lambda \log q(z \mid x; \lambda) \right] \\
= \mathbb{E}_q \left[ R_{\theta, \lambda}(z) \nabla_\lambda \log q(z \mid x; \lambda) \right]
\]
Strategy 2a: Sampling — Score Function Gradient Estimator

Estimate with samples,

\[ z^{(1)}, \ldots, z^{(J)} \sim q(z \mid x; \lambda) \]

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

Intuition: if a sample \( z^{(j)} \) is has high reward \( R_{\theta,\lambda}(z^{(j)}) \), increase the probability of \( z^{(j)} \) by moving along the gradient \( \nabla_{\lambda} \log q(z^{(j)} \mid x; \lambda) \).
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 \( q(z \mid x; \lambda) = \text{Cat}(\nu) \) where \( \nu = \text{enc}(x; \lambda) \)

Sample \( z^{(1)}, \ldots, z^{(J)} \sim q(z \mid x; \lambda) \)

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

Computational complexity: \( O(J) \) vs \( O(|Z|) \)
Example: Model 1 - Naive Bayes

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

Sample \( z^{(1)}, \ldots, z^{(J)} \sim q(z \mid x; \lambda) \)

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

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

Computational complexity: \( O(J) \) vs \( O(|Z|) \)
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):

$$
\nabla_\lambda \mathbb{E}_{z \sim q(z \mid x; \lambda)} \left[ \log p(x, z; \theta) \right] = \nabla_\lambda \mathbb{E}_{\epsilon \sim \mathcal{U}} \left[ \log p(x, g(\epsilon, \lambda); \theta) \right]
$$

$$
= \mathbb{E}_{\epsilon \sim \mathcal{U}} \left[ \nabla_\lambda \log p(x, g(\epsilon, \lambda); \theta) \right]
$$

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

where

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

$$
\nabla_\lambda \mathbb{E}_{z \sim q(z \mid x; \lambda)} \left[ \log p(x, z; \theta) \right] = \nabla_\lambda \mathbb{E}_{\epsilon \sim \mathcal{U}} \left[ \log p(x, g(\epsilon, \lambda); \theta) \right]
$$

$$
= \mathbb{E}_{\epsilon \sim \mathcal{U}} \left[ \nabla_\lambda \log p(x, g(\epsilon, \lambda); \theta) \right]
$$

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

where

$$
\epsilon^{(1)}, \ldots, \epsilon^{(J)} \sim \mathcal{U}
$$
Strategy 2b: Sampling — Reparameterization

- Unbiased, like the 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

\[ q(z \mid x; \lambda) = \mathcal{N}(\mu, \sigma^2) \]

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

Then we can sample from \( q(z \mid x; \lambda) \) by

\[ \epsilon \sim \mathcal{N}(0, I) \quad z = \mu + \sigma \epsilon \]
Strategy 2b: Sampling — Reparameterization

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

$$\arg \max_{\lambda} \text{ELBO}(\theta, \lambda; x) = \arg \min_{\lambda} \text{KL}[q(z \mid x; \lambda) \| p(z \mid x; \theta)]$$
Strategy 3a: Conjugacy — Tractable Posterior Inference

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

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

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

$$L \{ \begin{array}{c} \text{posterior gap} \end{array} \ \lambda \ \text{ELBO} \}$$
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.

\[ \lambda = [p(z = 1 \mid x; \theta), \ldots, p(z = K \mid x; \theta)] \]
Example: 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 \(q(z \mid x; \lambda) = p(z \mid x; \theta)\)):

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

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

\[
\nabla_{\theta} \log p(x; \theta) = \mathbb{E}_{q(z \mid x; \lambda)}[\nabla_{\theta} \log p(x, z; \theta)]
\]
Connection: Gradient Ascent on Log Marginal Likelihood

- 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

- Let $p(z \mid x; \theta)$ be intractable, but suppose $p(x, z; \theta)$ is **conditionally conjugate**, meaning $p(z_t \mid x, z_{-t}; \theta)$ 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 $p(z_t \mid x, z_{-t}; \theta)$. 
Strategy 3b: Conditional Conjugacy

\[ q(z; \lambda) = \prod_{t=1}^{T} q(z_t; \lambda_t) \]
Mean Field Family

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

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

- Coordinate ascent updates will take the form

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

where

\[
E_{q(z_{-t}; \lambda_{-t})} [\log p(x, z; \theta)] = \sum_j \prod_{j \neq t} q(z_j; \lambda_j) \log p(x, z; \theta)
\]

- 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

\[ q(z_{2,1}; \lambda_{2,1}) \propto \exp \left( \mathbb{E}_{q(z_{-2,1}; \lambda_{-2,1})} \left[ \log p(x, z; \theta) \right] \right) \]
Example: Model 3 — Factorial HMM

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

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].
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.
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
</tr>
<tr>
<td>2</td>
<td>Models</td>
</tr>
<tr>
<td>3</td>
<td>Variational Objective</td>
</tr>
<tr>
<td>4</td>
<td>Inference Strategies</td>
</tr>
<tr>
<td>5</td>
<td>Advanced Topics</td>
</tr>
<tr>
<td>6</td>
<td>Case Studies</td>
</tr>
<tr>
<td>7</td>
<td>Conclusion</td>
</tr>
<tr>
<td></td>
<td>References</td>
</tr>
<tr>
<td></td>
<td>Gumbel-Softmax</td>
</tr>
<tr>
<td></td>
<td>Flows</td>
</tr>
<tr>
<td></td>
<td>IWAE</td>
</tr>
<tr>
<td></td>
<td>Flows</td>
</tr>
<tr>
<td></td>
<td>IWAE</td>
</tr>
</tbody>
</table>
Challenges of Discrete Variables

Review: we can always use score function estimator

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

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

- \( \mathbb{E}_q[B \nabla_\lambda \log q(z | x; \lambda)] = 0 \) (since \( \mathbb{E}[\nabla \log q] = \sum q \nabla \log q = \sum \nabla q = 0 \))
- 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

Review: we can always use score function estimator

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

- \[ \mathbb{E}_q [B \nabla_\lambda \log q(z | x; \lambda)] = 0 \text{ (since } \mathbb{E} [\nabla \log q] = \sum q \nabla \log q = \sum \nabla q = 0) \]
- 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 \( z = [0, 0, \ldots, 1, \ldots, 0] \) is a one-hot vector.

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

1. **Drawing independent Gumbel noise** \( \epsilon = \epsilon_1, \ldots, \epsilon_K \)

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

2. **Adding** \( \epsilon_k \) **to** \( \log \alpha_k \), **finding 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 \( z = [0, 0, \ldots, 1, \ldots, 0] \) is a one-hot vector.

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

1. Drawing independent Gumbel noise \( \epsilon = \epsilon_1, \ldots, \epsilon_K \)

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

2. Adding \( \epsilon_k \) to \( \log \alpha_k \), finding 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 \( z = [0, 0, \ldots, 1, \ldots, 0] \) is a one-hot vector.

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

1. Drawing independent Gumbel noise \( \epsilon = \epsilon_1, \ldots, \epsilon_K \)

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

2. Adding \( \epsilon_k \) to \( \log \alpha_k \), finding 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:

\[ q(z_k = 1 \mid x; \lambda) = \frac{\alpha_k}{\sum_{j=1}^{K} \alpha_j} \]

\[ \alpha = \text{enc}(x; \lambda) \]

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

\[ \nabla_{\lambda} \mathbb{E}_{q(z \mid x; \lambda)}[R_{\theta, \lambda}(z)] = \nabla_{\lambda} \mathbb{E}_{\epsilon \sim \text{Gumbel}}[R_{\theta, \lambda}(g(\epsilon, \alpha))] = \mathbb{E}_{\epsilon \sim \text{Gumbel}}[\nabla_{\lambda} R_{\theta, \lambda}(g(\epsilon, \alpha))] \]
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 \(R_{\theta, \lambda}(z) = \log \frac{p(x,z; \theta)}{q(z \mid x; \lambda)}\),

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

\[=
\mathbb{E}_{\epsilon \sim \text{Gumbel}}[\nabla_{\lambda} R_{\theta, \lambda}(g(\epsilon, \alpha))]\]
**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 \(R_{\theta,\lambda}(z) = \log \frac{p(x,z;\theta)}{q(z \mid x; \lambda)}\),

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

\[
= \mathbb{E}_{\epsilon \sim \text{Gumbel}}[\nabla_\lambda R_{\theta,\lambda}(g(\epsilon, \alpha))]
\]
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

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

(where \(\tau\) is a temperature term.)

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

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

(where \( \tau \) is a temperature term.)

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

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

(where \( \tau \) is a temperature term.)

\[ \nabla_{\lambda} \mathbb{E}_{q(z \mid x; \lambda)}[R_{\theta, \lambda}(z)] \approx \mathbb{E}_{\epsilon \sim \text{Gumbel}} \left[ \nabla_{\lambda} R_{\theta, \lambda}\left( \text{softmax}\left(\frac{\log \alpha + \epsilon}{\tau}\right) \right) \right] \]
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 $p_{GS}(z), q_{GS}(z)$.
• Requires that $p(x | z; \theta)$ “makes sense” for non-discrete $z$ (e.g. attention).
• Lower-variance, but biased gradient estimator. Variance $\to \infty$ as $\tau \to 0$. 
1 Introduction

2 Models

3 Variational Objective

4 Inference Strategies

5 Advanced Topics

6 Case Studies
Flows [Rezende and Mohamed 2015; Kingma et al. 2016]

Recall

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

Bound is tight when variational posterior equals true posterior

\[ q(z \mid x; \lambda) = p(z \mid x; \theta) \implies \log p(x; \theta) = \text{ELBO}(\theta, \lambda; x) \]

We want to make \( q(z \mid x; \lambda) \) as flexible as possible: can we do better than just Gaussian?
Flows [Rezende and Mohamed 2015; Kingma et al. 2016]

Recall

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

Bound is tight when variational posterior equals true posterior

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

We want to make $q(z \mid x; \lambda)$ as flexible as possible: can we do better than just Gaussian?
Flows [Rezende and Mohamed 2015; Kingma et al. 2016]

**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]

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]

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

$$\log q_K(z_K | x; \lambda) = \log q(z_0 | x; \lambda) + \sum_{k=1}^{K} \log \left| \frac{\partial f_{k-1}^{-1}}{\partial z_k} \right|$$

$$= \log q(z_0 | x; \lambda) - \sum_{k=1}^{K} \log \left| \frac{\partial f_k}{\partial z_{k-1}} \right|$$

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]

Can still use reparameterization to obtain gradients. Letting

\[ F(z) = f_K \circ \cdots \circ f_1(z), \]

\[
\text{ELBO}(\theta, \lambda; x) = \nabla_{\lambda} \mathbb{E}_{q_K(z_K \mid x; \lambda)} \left[ \log \frac{p(x, z; \theta)}{q_K(z_K \mid x; \lambda)} \right] \\
= \nabla_{\lambda} \mathbb{E}_{q(z_0 \mid x; \lambda)} \left[ \log \frac{p(x, F(z_0); \theta)}{q(z_0 \mid x; \lambda)} - \log \left| \frac{\partial F}{\partial z_0} \right| \right] \\
= \mathbb{E}_{\epsilon \sim \mathcal{N}(0, I)} \left[ \nabla_{\lambda} \left( \log \frac{p(x, F(z_0); \theta)}{q(z_0 \mid x; \lambda)} - \log \left| \frac{\partial F}{\partial z_0} \right| \right) \right]
\]
Flows [Rezende and Mohamed 2015; Kingma et al. 2016]

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

- **Normalizing Flows** [Rezende and Mohamed 2015]

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

- **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])
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 $z^{(1:K)} \sim \prod_{k=1}^{K} q(z^{(k)} | x; \lambda)$.

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 $z^{(1:K)} \sim \prod_{k=1}^{K} q(z^{(k)} | x; \lambda)$.

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]

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

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

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

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

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

- $\log p(x; \theta) \geq \mathbb{E} [\log I_K] \geq \mathbb{E} [\log I_{K-1}]$

- $\lim_{K \to \infty} \mathbb{E} [\log I_K] = \log p(x; \theta)$ under mild conditions
Importance Weighted Autoencoder (IWAE) [Burda et al. 2015]

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

- Note that with \( K = 1 \), we recover the ELBO.
- Can interpret \( \frac{p(x, z^{(k)}; \theta)}{q(z^{(k)} \mid x; \lambda)} \) as importance weights.
- If \( q(z \mid x; \lambda) \) 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].
Introduction

Models

Variational Objective

Inference Strategies

Advanced Topics

Case Studies

Sentence VAE
Encoder/Decoder with Latent Variables
Latent Summaries and Topics

Conclusion

References
Sentence VAE Example [Bowman et al. 2016]

Generative Model (Model 2):

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

Variational Model (Amortized): Deep Diagonal Gaussians,

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

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

$$\mu = W_1 \tilde{h}_T \quad \sigma^2 = \exp(W_2 \tilde{h}_T) \quad \lambda = \{W_1, W_2, \psi\}$$
Sentence VAE Example [Bowman et al. 2016]

(from Bowman et al. [2016])
Issue 1: Posterior Collapse

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

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

<table>
<thead>
<tr>
<th>Model</th>
<th>L/ELBO</th>
<th>Reconstruction</th>
<th>KL</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNN LM</td>
<td>-329.10</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RNN VAE</td>
<td>-330.20</td>
<td>-330.19</td>
<td>0.01</td>
</tr>
</tbody>
</table>

(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:

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

$$\text{KL}[q(z \mid x; \lambda) \parallel p(z)] = 0$$
Mitigating Posterior Collapse

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

<table>
<thead>
<tr>
<th>Model</th>
<th>LL/ELBO</th>
<th>Reconstruction</th>
<th>KL</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNN LM</td>
<td>-329.1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RNN VAE</td>
<td>-330.2</td>
<td>-330.2</td>
<td>0.01</td>
</tr>
<tr>
<td>+ Word Drop</td>
<td>-334.2</td>
<td>-332.8</td>
<td>1.44</td>
</tr>
<tr>
<td>CNN VAE</td>
<td>-332.1</td>
<td>-322.1</td>
<td>10.0</td>
</tr>
</tbody>
</table>

(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

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

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

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

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

Qualitative evaluation

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

(from Bowman et al. [2016])
Encoder/Decoder  [Sutskever et al. 2014; Cho et al. 2014]

Given: Source information \( s = s_1, \ldots, s_M \).

Generative process:

- Draw \( x_{1:T} | s \sim \text{CRNNLM}(\theta, \text{enc}(s)) \).
Latent, Per-token Experts [Yang et al. 2018]

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

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

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]

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):

<table>
<thead>
<tr>
<th>Model</th>
<th>PPL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Merity et al. [2018]</td>
<td>57.30</td>
</tr>
<tr>
<td>Softmax-mixture [Yang et al. 2018]</td>
<td>54.44</td>
</tr>
</tbody>
</table>

Dialogue generation results ($s$ is context):

<table>
<thead>
<tr>
<th>Model</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Prec</td>
</tr>
<tr>
<td>No mixture</td>
<td>14.1</td>
</tr>
<tr>
<td>Softmax-mixture [Yang et al. 2018]</td>
<td>15.7</td>
</tr>
</tbody>
</table>
Decoding with an attention mechanism:

$$x_t | x_{<t}, s \sim \text{softmax}(W[h_t, \sum_{m=1}^{M} \alpha_{t,m} \text{enc}(s)_m]).$$
Copy Attention [Gu et al. 2016; Gulcehre et al. 2016]

Copy attention models copying words directly from $s$.

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

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

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

Copy attention models copying words directly from $s$.

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

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

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

\[
\max_{\theta} \log p(x_1, \ldots, x_T \mid s; \theta)
= \max_{\theta} \log \prod_{t=1}^{T} \sum_{z'_t \in \{0,1\}} p(z_t = z'_t \mid s, x_{<t}; \theta) p(x_t \mid z'_t, x_{<t}, x; \theta).
\]

**Test-time:**

\[
\arg \max_{x_{1:T}} \prod_{t=1}^{T} \sum_{z'_t \in \{0,1\}} p(z_t = z'_t \mid s, x_{<t}; \theta) p(x_t \mid z'_t, x_{<t}, s; \theta).
\]
Attention as a Latent Variable [Deng et al. 2018]

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

- Set $\alpha_t$ to be attention weights.
- Draw $z_t \mid x_{<t}, s \sim \text{Cat}(\alpha_t)$.
- Draw $x_t \mid z_t, x_{<t}, s \sim \text{softmax}(W[h_t, \text{enc}(s_{z_t})]; \theta)$. 
Attention as a Latent Variable [Deng et al. 2018]

Marginal likelihood under latent attention model:

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

Standard attention likelihood:

$$p(x_{1:T} | s; \theta) = \prod_{t=1}^{T} \text{softmax}(W[h_t, \sum_{m=1}^{M} \alpha_{t,m} \text{enc}(s_m); \theta])_{x_t}.$$
Attention as a Latent Variable [Deng et al. 2018]

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

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

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

- \( q(z_t | x; \lambda) \) 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]

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

MT Results on IWSLT-2014:

<table>
<thead>
<tr>
<th>Model</th>
<th>PPL</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard Attn</td>
<td>7.03</td>
<td>32.31</td>
</tr>
<tr>
<td>Latent Attn (marginal)</td>
<td>6.33</td>
<td>33.08</td>
</tr>
<tr>
<td>Latent Attn (ELBO)</td>
<td>6.13</td>
<td>33.09</td>
</tr>
</tbody>
</table>
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 $z_{1:S}$. 
Summary as a Latent Variable [Miao and Blunsom 2016]

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

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

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 $x = x_1, \ldots, x_T$:

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

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:

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

- $q(z_{1:M}; \lambda)$ 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!
Generative process: for each document $x^{(n)} = x_1^{(n)}, \ldots, x_T^{(n)}$,

- **Draw topic distribution** $z_{top}^{(n)} \sim Dir(\alpha)$
- **For** $t = 1, \ldots, T$:
  - **Draw topic** $z_t^{(n)} \sim Cat(z_{top}^{(n)})$
  - **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 \( q(z_{top}^{(n)}; \lambda) \) is reparameterizable.

**Idea:** draw \( z_{top}^{(n)} \) from a transformation of a Gaussian.

- Draw \( z_0^{(n)} \sim N(\mu_0, \sigma_0^2) \)
- Set \( z_{top}^{(n)} = \text{softmax}(Wz_0^{(n)}) \).
- Use analogous transformation when drawing from \( q(z_{top}^{(n)}; \lambda) \).
Simple, Deep Topic Models [Miao et al. 2017]

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

\[
p\left( \{x^{(n)}\}_{n=1}^{N}, \{z_{top}^{(n)}\}_{n=1}^{N}; \theta \right) = \prod_{n=1}^{N} p(z_{top}^{(n)} | \theta) \prod_{t=1}^{T} \sum_{k=1}^{K} z_{top,k}^{(n)} \beta_{k,x_t^{(n)}}
\]

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

\[
\max_{\lambda, \theta} \mathbb{E}_{q(z_{top}^{(n)}; \lambda)} \left[ \log p(x^{(n)} | z_{top}^{(n)}; \theta) \right] - KL[\mathcal{N}(z_0^{(n)}; \lambda) \| \mathcal{N}(z_0^{(n)}; \mu_0, \sigma_0^2)]
\]
Simple, Deep Topic Models [Miao et al. 2017]

Perplexities on held-out documents, for three datasets:

<table>
<thead>
<tr>
<th>Model</th>
<th>MXM</th>
<th>20News</th>
<th>RCV1</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnlineLDA</td>
<td>342</td>
<td>1015</td>
<td>1058</td>
</tr>
<tr>
<td>AVI-LDA</td>
<td>272</td>
<td>830</td>
<td>602</td>
</tr>
</tbody>
</table>
Deep Latent-Variable NLP: Two Views

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

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


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


