Deep Latent-Variable Models of Natural Language

Yoon Kim, Sam Wiseman, Alexander Rush

harvardnlp

Tutorial 2018

https://github.com/harvardnlp/DeepLatentNLP

Maximizing the Evidence Lower Bound

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

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

Aggregate ELBO objective,

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

Maximizing the Evidence Lower Bound

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

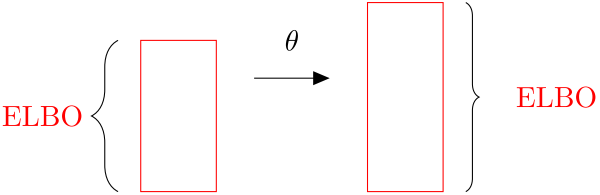
$$\arg\max_{\theta,\lambda} \mathrm{ELBO}(\theta, \lambda)$$

Aggregate ELBO objective,

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

Maximizing ELBO: Model Parameters

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



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

Model Estimation: Gradient Ascent on Model Parameters

Easy: Gradient respect to θ

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

- Since q not dependent on θ, ∇ 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

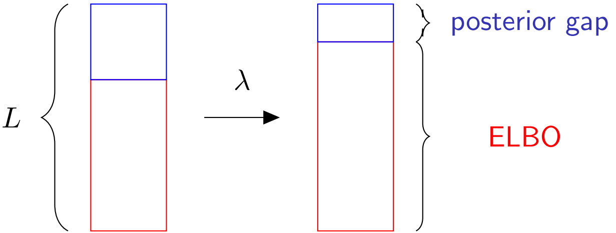
Easy: Gradient respect to θ

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

- Since q not dependent on θ, ∇ 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

$$
\begin{aligned}
&\arg\max_{\lambda} \mathrm{ELBO}(\theta, \lambda) \\
&\qquad = \arg\max_{\lambda} \log p(x;\, \theta) - \mathrm{KL}[q(z \mid x; \lambda) \,\|\, p(z \mid x;\, \theta)] \\
&\qquad = \arg\min_{\lambda} \mathrm{KL}[q(z \mid x; \lambda) \,\|\, p(z \mid x;\, \theta)]
\end{aligned}
$$



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 λ?

Hard: Gradient respect to λ

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

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

Model Inference: Gradient Ascent on λ?

Hard: Gradient respect to λ

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

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

Strategy 1: Exact Gradient

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

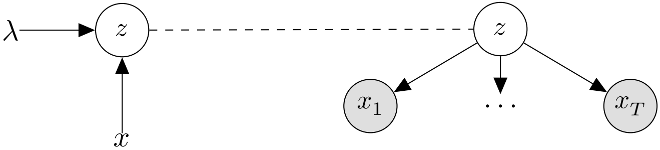
- Naive enumeration: Linear in $|\mathcal{Z}|$.
- Depending on structure of q and p, potentially faster with dynamic programming.
- Applicable mainly to Model 1 and 3 (Discrete and Structured), or Model 2 with point estimate.

Strategy 1: Exact Gradient

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

- 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



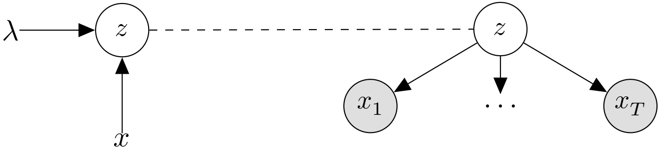
Let $q(z \mid x;\, \lambda) = \text{Cat}(\nu)$ where $\nu = \text{enc}(x; \lambda)$

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

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

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

Example: Model 1 - Naive Bayes



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

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

Strategy 2: Sampling

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

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

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

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

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

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

Strategy 2: Sampling

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

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

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

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

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

- Manipulate expression so we can move ∇_λ 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]

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

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]

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

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]

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

Strategy 2a: Sampling — Score Function Gradient Estimator

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]

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

Strategy 2a: Sampling — Score Function Gradient Estimator

Second term. Need additional identity:

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

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

Strategy 2a: Sampling — Score Function Gradient Estimator

Second term. Need additional identity:

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

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

$$= \sum_z \Big(\underbrace{\nabla_\lambda q(z \,|\, x;\, \lambda)}_{q \nabla \log q} \Big) \log q(z \,|\, x;\, \lambda) + q(z \,|\, x;\, \lambda) \Big(\underbrace{\nabla_\lambda \log q(z \,|\, x;\, \lambda)}_{\frac{\nabla q}{q}} \Big)$$

Strategy 2a: Sampling — Score Function Gradient Estimator

Second term. Need additional identity:

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

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

$$= \sum_z \log q(z \,|\, x;\, \lambda) \nabla_\lambda \log q(z \,|\, x;\, \lambda) + \sum_z \nabla_\lambda q(z \,|\, x;\, \lambda)$$

Strategy 2a: Sampling — Score Function Gradient Estimator

Second term. Need additional identity:

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

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

$$= \sum_z \log q(z \,|\, x;\, \lambda) \nabla_\lambda \log q(z \,|\, x;\, \lambda) + \underbrace{\sum_z \nabla_\lambda q(z \,|\, x;\, \lambda)}_{= \nabla \sum q = \nabla 1 = 0}$$

Strategy 2a: Sampling — Score Function Gradient Estimator

Second term. Need additional identity:

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

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

$$= \sum_z \log q(z \,|\, x;\, \lambda) \nabla_\lambda \log q(z \,|\, x;\, \lambda) + \sum_z \nabla_\lambda q(z \,|\, x;\, \lambda)$$

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

Strategy 2a: Sampling — Score Function Gradient Estimator

Putting these together,

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

Strategy 2a: Sampling — Score Function Gradient Estimator

Estimate with samples,

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

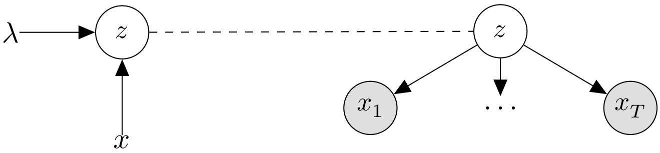
$$\begin{aligned}
&\mathbb{E}_q\Big[R_{\theta,\lambda}(z)\nabla_\lambda \log q(z \,|\, x;\, \lambda)\Big] \\
&\quad\approx \frac{1}{J}\sum_{j=1}^{J} R_{\theta,\lambda}(z^{(j)})\nabla_\lambda \log q(z^{(j)} \,|\, x;\, \lambda)
\end{aligned}$$

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

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 $q(z \mid x;\, \lambda) = \mathsf{Cat}(\nu)$ where $\nu = \mathrm{enc}(x; \lambda)$

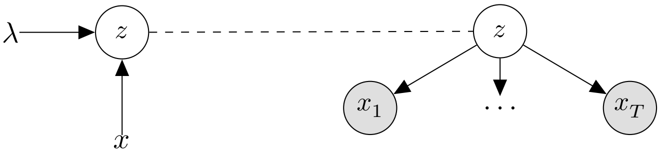
Sample $z^{(1)}, \dots, z^{(J)} \sim q(z \mid x; \lambda)$

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

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

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

Example: Model 1 - Naive Bayes



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

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

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

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

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

Strategy 2b: Sampling — Reparameterization

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

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

Gradient calculation (first term):

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

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,

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

Gradient calculation (first term):

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

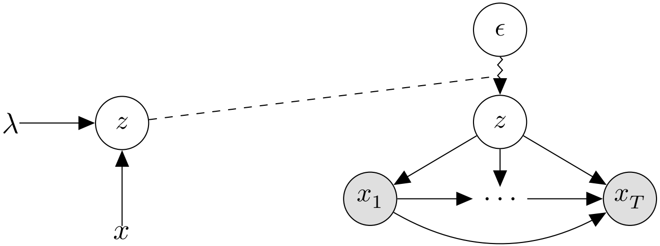
where

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

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

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

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

Strategy 2b: Sampling — Reparameterization

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

- Score function:

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

- Reparameterization:

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

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

Informally, reparameterization gradients differentiate through $R_{\theta,\lambda}()$ 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

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

exactly in closed-form.

Recall that

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

Strategy 3: Conjugacy

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

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

exactly in closed-form.

Recall that

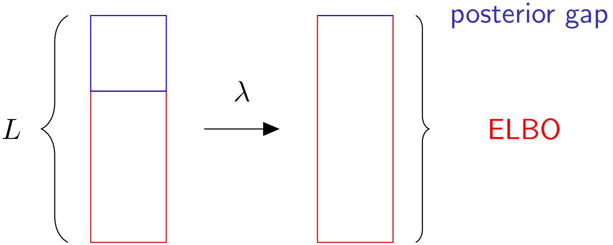
$$\arg\max_{\lambda} \mathrm{ELBO}(\theta, \lambda; x) = \arg\min_{\lambda} \mathrm{KL}[q(z \mid x;\ \lambda) \| p(z \mid x;\ \theta)]$$

Strategy 3a: Conjugacy — Tractable Posterior Inference

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

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

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

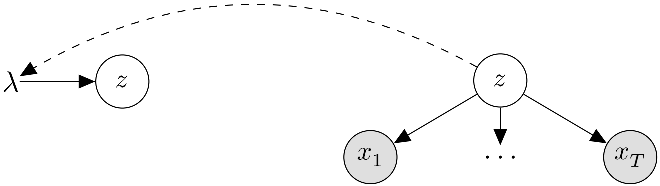


Example: Model 2 - Dirichlet-Multinomial

- $q(z;\, x; \lambda) = \text{Dir}(\lambda)$
- $p(x, z;\, \theta)$ is given by

$$z \sim \text{Dir}(\alpha)$$

$$x_t \,|\, z \sim \text{Cat}(z) \;\; \text{for } t = 1, \dots, T$$



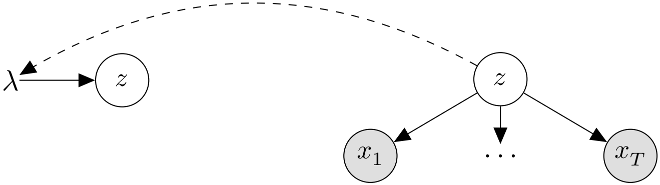
$$p(z \,|\, x; \theta) = \text{Dir}(z; \alpha + \sum_{t=1}^{T} x_t) \Rightarrow \qquad \lambda = \alpha + \sum_{t=1}^{T} x_t$$

Example: Model 2 - Dirichlet-Multinomial

- $q(z;\, x; \lambda) = \text{Dir}(\lambda)$
- $p(x, z;\, \theta)$ is given by

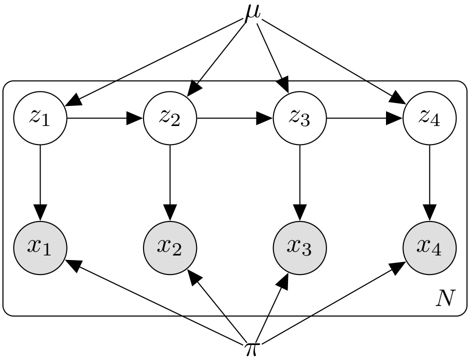
$$z \sim \text{Dir}(\alpha)$$

$$x_t \,|\, z \sim \text{Cat}(z) \;\text{ for } t = 1, \dots, T$$



$$p(z \,|\, x; \theta) = \text{Dir}(z; \alpha + \sum_{t=1}^{T} x_t) \Rightarrow \qquad \lambda = \alpha + \sum_{t=1}^{T} x_t$$

Reminder: Model 3 — HMM



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

Example: Model 3 — HMM

Run forward/backward dynamic programming to calculate posterior marginals,

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

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

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

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

Example: Model 3 — HMM

Run forward/backward dynamic programming to calculate posterior marginals,

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

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

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

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

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

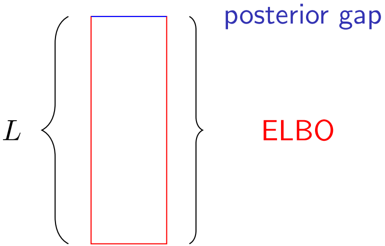
Connection: Gradient Ascent on Log Marginal Likelihood

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

- 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

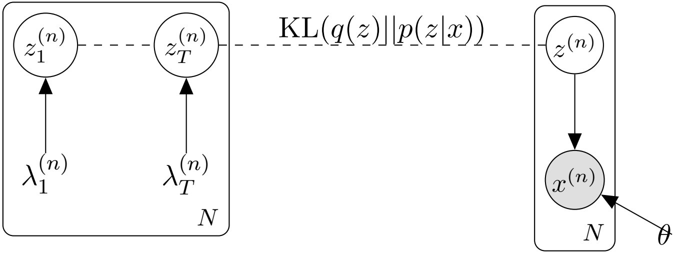
- Let $p(z \mid x; \theta)$ be intractable, but suppose $p(x, z; \theta)$ is **conditionally conjugate**, meaning $p(z_t \mid x, z_{-t}; \theta)$ is exponential family.
- Restrict the family of distributions q so that it factorizes over z_t, i.e.

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

(**mean field** family)

- Further choose $q(z_t; x\lambda_t)$ 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 $\lambda_1, \dots, \lambda_T$

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

- Coordinate ascent updates will take the form

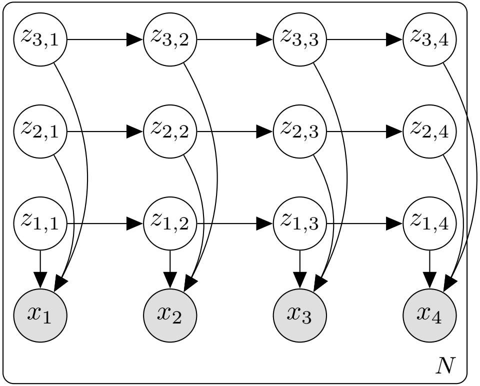
$$q(z_t;\, \lambda_t) \propto \exp\left(\mathbb{E}_{q(z_{-t};\, \lambda_{-t})}[\log p(x, z;\, \theta)]\right)$$

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

Example: Model 3 — Factorial HMM



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

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].

Sentence VAE Example [Bowman et al. 2016]

Generative Model (Model 2):

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

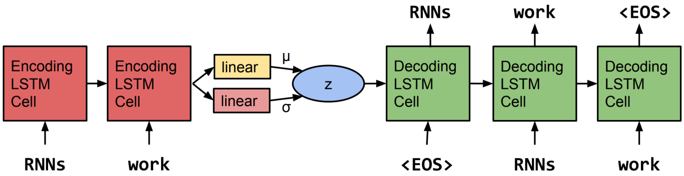
Variational Model (Amortized): Deep Diagonal Gaussians,

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

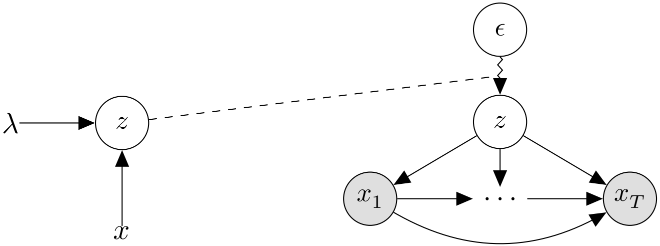
$$\tilde{\boldsymbol{h}}_T = \mathrm{RNN}(x; \psi)$$

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

Sentence VAE Example [Bowman et al. 2016]



(from Bowman et al. [2016])



Issue 1: Posterior Collapse

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

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

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:

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

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

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

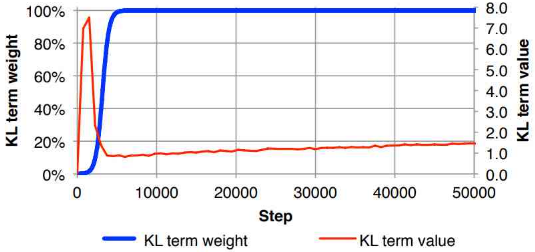
(On Yahoo Corpus from Yang et al. [2017])

Mitigating Posterior Collapse

Gradually anneal multiplier on KL term, i.e.

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

β 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,

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

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

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

Evaluation

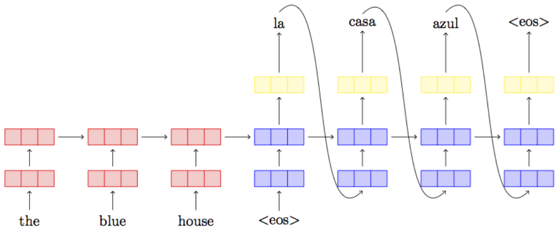
Qualitative evaluation

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

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])

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



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

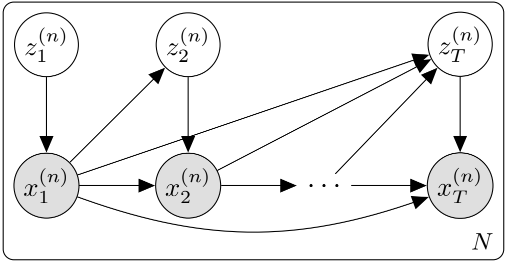
Generative process:

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

Latent, Per-token Experts [Yang et al. 2018]

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

- Draw $z_t \mid x_{<t}, s \sim \mathrm{softmax}(\boldsymbol{U}\boldsymbol{h}_{t-1})$.
- Draw $x_t \mid z_t, x_{<t}, s \sim \mathrm{softmax}(\boldsymbol{W} \tanh(\boldsymbol{Q}_{z_t}\boldsymbol{h}_{t-1}); \theta)$



If $\boldsymbol{U} \in \mathbb{R}^{K \times d}$, 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]

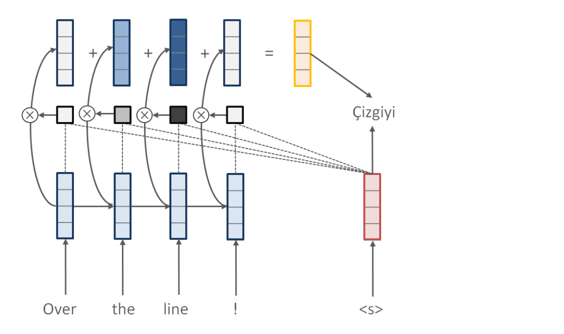
PTB language modeling results (s is constant):

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

Attention [Bahdanau et al. 2015]



Decoding with an attention mechanism:

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

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

Copy attention models copying words directly from s.

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

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

Copy Attention

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

$$\max_{\theta} \log p(x_1, \ldots, x_T \mid s;\, \theta)$$

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

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 $t = 1, \ldots, T$,

- Set $\boldsymbol{\alpha}_t$ to be attention weights.
- Draw $z_t \,|\, x_{<t}, s \sim \mathrm{Cat}(\boldsymbol{\alpha}_t)$.
- Draw $x_t \,|\, z_t, x_{<t}, s \sim \mathrm{softmax}(\boldsymbol{W}[\boldsymbol{h}_{t-1}, \mathbf{enc}(s_{z_t})]; \theta)$.

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

Marginal likelihood under latent attention model:

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

Standard attention likelihood:

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

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:

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

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

Test-time: Calculate $p(x_t \mid x_{<t}, s; \theta)$ 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 $z_{1:S}$.

Summary as a Latent Variable [Miao and Blunsom 2016]

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

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

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

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:

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

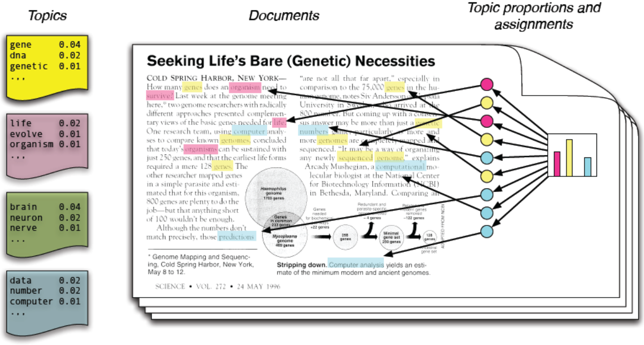
- $q(z_{1:M};\,\lambda)$ approximates $p(z_{1:M} \mid x_{1:T};\,\theta)$; also implemented with encoder/decoder RNNs.
- $q(z_{1:M};\,\lambda)$ not reparameterizable, so gradients use REINFORCE + baselines.

Summary as a Latent Variable [Miao and Blunsom 2016]

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

- Train $q(z_{1:M};\lambda) \approx p(z_{1:M} \mid x_{1:T};\theta)$ with labeled examples.
- Infer summary z for an *unlabeled* document with q.
- Use inferred z to improve model $p(x_{1:T} \mid z_{1:M};\theta)$.
- 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 $\mathbf{z}_{top}^{(n)} \sim Dir(\boldsymbol{\alpha})$
- For $t = 1, \dots, T$:
  - Draw topic $z_t^{(n)} \sim Cat(\mathbf{z}_{top}^{(n)})$
  - Draw $x_t \sim Cat(\boldsymbol{\beta}_{z_t^{(n)}})$

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 $\mathbf{z}_{top}^{(n)} = \mathrm{softmax}(\boldsymbol{W}\mathbf{z}_0^{(n)})$.
- Use analogous transformation when drawing from $q(\mathbf{z}_{top}^{(n)}; \lambda)$.

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

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

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

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

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

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

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*.

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 Merrienboer, 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*.

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

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*.

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.

Yoon Kim, Sam Wiseman, Andrew C. Miller, David Sontag, and Alexander M. Rush. 2018. Semi-Amortized Variational Autoencoders. In *Proceedings of ICML*.

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

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*.

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

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

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*.

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*.