CS 533: Natural Language Processing

More Variational Autoencoders, Discrete Latent Variables

Karl Stratos

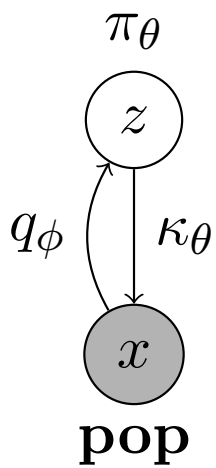
Rutgers University

Review: Variational Autoencoders (VAEs)

- **VAE.** Maximizing ELBO written as an autoencoding objective

$$\mathrm{ELBO}(\theta, \phi) = \underbrace{\mathop{\mathbf{E}}_{x \sim \mathbf{pop},\, z \sim q_\phi(\cdot|x)} [\log \kappa_\theta(x|z)]}_{\text{reconstruction}} - \underbrace{\mathop{\mathbf{E}}_{x \sim \mathbf{pop}} [D_{\mathrm{KL}}(q_\phi(\cdot|x)||\pi_\theta)]}_{\text{regularization}}$$

- Example: Gaussian VAE for Language Modeling



  - Population distribution $\mathbf{pop}$ over sentences x
  - $q_\phi(z|x) = \mathcal{N}(\mu_\phi(x), \mathrm{diag}(\sigma^2_\phi(x)))$ conditional density over $\mathbb{R}^d$ given x
  - $\kappa_\theta(x|z)$ any language model conditioning on z
  - $\pi_\theta(z) = \mathcal{N}(0_d, I_{d\times d})$ standard Gaussian prior

- Many tricks possible with Gaussian parameterization: (1) KL computed exactly, (2) reconstruction term estimated by differentiable sampling ("reparameterization trick")

Usage of Gaussian VAE

- Once Gaussian VAE is "well trained" (more later), we can generate sentences meaningfully from $\mathbb{R}^d$
- "Interpolation": Sample $z, z' \sim \mathcal{N}(0_d, I_{d\times d})$, greedy decoding from $\kappa_\theta(x|\alpha z + (1-\alpha)z')$ (examples from Li et al. (2019))

$$
\begin{aligned}
z &\Rightarrow \text{a man with a cane is walking down the street .} \\
0.8z + 0.2z' &\Rightarrow \text{a man with a cane is walking down the street .} \\
0.6z + 0.4z' &\Rightarrow \text{a man in a blue shirt is eating food .} \\
0.4z + 0.6z' &\Rightarrow \text{people are eating food .} \\
0.2z + 0.8z' &\Rightarrow \text{people walk in a city .} \\
z' &\Rightarrow \text{people are outside in a city}
\end{aligned}
$$

- Semi-supervised learning: Can use $\mu_\phi(x) \in \mathbb{R}^d$ as a pretrained embedding of x and train a downstream classifier
- Also a better language model: Perplexity slightly lower compared to vanilla LM (100 vs 96 on PTB). But how do we measure perplexity with this model?

ELBO vs MLL

- Recall: ELBO is a (possibly loose) lower bound on MLL (marginal log likelihood)

$$L(\theta) - \mathrm{ELBO}(\theta, \phi) = D_{\mathrm{KL}}(q_\phi || \omega_\theta) \geq 0$$

- Need an unbiased estimator of $L(\theta)$ to estimate it directly (e.g., for perplexity which is $\exp(-L(\theta)/N_{\text{tokens}})$)
- Solution: *Multi-sample* importance sampling

$$L_x(\theta) = \log \mathop{\mathbf{E}}_{z_1 \ldots z_K \sim q_\phi(\cdot|x)} \left[\frac{1}{K} \sum_{k=1}^{K} \frac{p_\theta(x, z_k)}{q_\phi(z_k|x)} \right]$$

$$\geq \mathop{\mathbf{E}}_{z_1 \ldots z_K \sim q_\phi(\cdot|x)} \left[\log \left(\frac{1}{K} \sum_{k=1}^{K} \frac{p_\theta(x, z_k)}{q_\phi(z_k|x)} \right) \right] = \mathrm{ELBO}_x^K(\theta, \phi)$$

- ELBO special case with $K = 1$, but $\mathrm{ELBO}_x^K(\theta, \phi) \to L_x(\theta)$ as $K \to \infty$. Can also be directly optimized: "importance weighted autoencoder" (Burda et al., 2016)

Discrete Latent Variables

- Gaussian reparameterization trick allows us to "backpropagate through sampling" for continuous $z \in \mathbb{R}^d$

$$\frac{\partial}{\partial \phi}\left(\mathop{\mathbf{E}}_{z\sim q_\phi(\cdot|x)}\left[\log \kappa_\theta(x|z)\right]\right) = \mathop{\mathbf{E}}_{\epsilon\sim\mathcal{N}(0_d, I_{d\times d})}\left[\frac{\partial}{\partial \phi}\left(\log \kappa_\theta(x|\mu_\phi(x) + \sigma_\phi(x) \odot \epsilon)\right)\right]$$

- What if $z \in \{0,1\}^d$? Why do we care about discrete? Some reasons
  - (Possibly) Interpretable: 1st dim corresponding to sentiment, 2nd dim gender, etc.
  - Compression: Cheaper to store ints than floats.
- Typical inference network: "mean-field approximation"

$$q_\phi(z|x) = \prod_{i=1}^{d} q_\phi(z_i|x, i)$$

Easy to parametrize: $q_\phi(1|x, i) = \sigma(w_i \cdot \mathbf{enc}_\phi(x))$

Backpropation Through Discrete Sampling

- Marginalization intractable (2^d possible values of z). This is despite mean-field approx

$$\mathop{\mathbf{E}}_{z\sim q_\phi(\cdot|x)}[\log \kappa_\theta(x|z)] = \sum_{z_1\in\{0,1\}} q_\phi(z_1|x,1)\cdots \sum_{z_d\in\{0,1\}} q_\phi(z_d|x,d)\log \kappa_\theta(x|z)$$

- So we'd like to approximate by sampling z. Here it's fine to sample $z_i \sim q_\phi(\cdot|x,i)$ independently and use $z = (z_1 \dots z_d)$ to estimate $\log \kappa_\theta(x|z)$
- Exercise: Verify the reparameterization trick

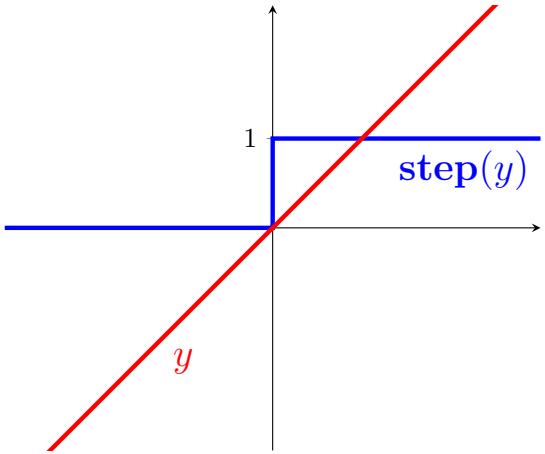
$$z_i \sim q_\phi(\cdot|x,i) \qquad \Leftrightarrow \qquad z_i = \textbf{step}(q_\phi(1|x,i) - \epsilon)$$

where $\epsilon \sim \mathrm{Unif}(0,1)$ and $\textbf{step}(y) = 1$ if $y \geq 0$ and 0 otherwise

- Unfortunately z_i still non-differentiable because of **step**

Straight-Through Gradient Estimator (Hinton, 2012)

- Idea: Approximate $\frac{\partial}{\partial y}\textbf{step}(y) \approx \frac{\partial}{\partial y} y = 1$
  - $f(y) = y$ linearization of $\textbf{step}(y)$ preserving the sign



- "Straight-through" gradient estimation of the step function

$$\frac{\partial \textbf{step}(f(\phi))}{\partial \phi} = \frac{\partial \textbf{step}(f(\phi))}{\cancel{\partial f(\phi)}} \times \frac{\partial f(\phi)}{\partial \phi} \approx \frac{\partial f(\phi)}{\partial \phi}$$

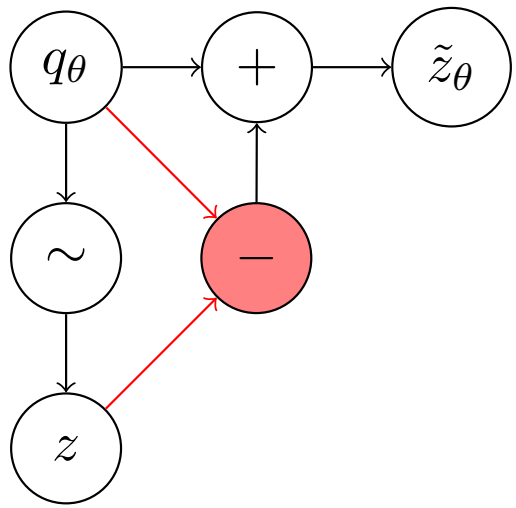
- With this approximation, we can sample $z_i \sim q_\phi(\cdot|x, i)$ in the forward pass and backpropage directly to $q_\phi(1|x, i)$ in the backward pass!

Implemention Trick for Straight-Through

- PyTorch style code

$$z = \texttt{bernoulli}(q_\phi) \qquad \# \text{ Not differentiable}$$
$$\tilde{z}_\phi = q_\phi + (z - q_\phi).\texttt{detach}() \qquad \# \text{ Differentiable } (\tilde{z}_\phi.\texttt{val} = z.\texttt{val})$$

- Corresponding computation graph



- Generally useful trick. Another example: gradient reversal layer (Ganin and Lempitsky, 2014)

$$\tilde{x} = -x + (x + x).\texttt{detach}()$$

Categorical VAE

- What if $z \in \{1 \dots K\}^d$?
  - Again mean-field approx $q_\phi(z|x) = \prod_{i=1}^d q_\phi(z_i|x, i)$
  - Easy to parameterize: $q_\phi(\cdot|x, i) = \text{softmax}(\textbf{enc}_\phi^{(i)}(x))$
- **Gumbel-max trick**

$$z_i \sim q_\phi(\cdot|x, i) \quad \Leftrightarrow \quad z_i = \arg\max_{k=1}^{K} [\textbf{enc}_\phi^{(i)}(x)]_k + \epsilon_k$$

  where $\epsilon_1 \dots \epsilon_K \overset{iid}{\sim} \text{Gumbel}(0, 1)$. Unfortunately z_i still non-differentiable because of $\arg\max$
- **Differentiable relaxation.** WLOG assume one-hot representation: $z_i = e_k \in \{0, 1\}^K$ means $z_i = k$. Then

$$z_i \sim q_\phi(\cdot|x, i) \quad \overset{\tau \to 0^+}{\Leftrightarrow} \quad z_i = \text{softmax}\left(\frac{\textbf{enc}_\phi^{(i)}(x) + \epsilon}{\tau}\right)$$

  where $\epsilon_1 \dots \epsilon_K \overset{iid}{\sim} \text{Gumbel}(0, 1)$. Now z_i differentiable

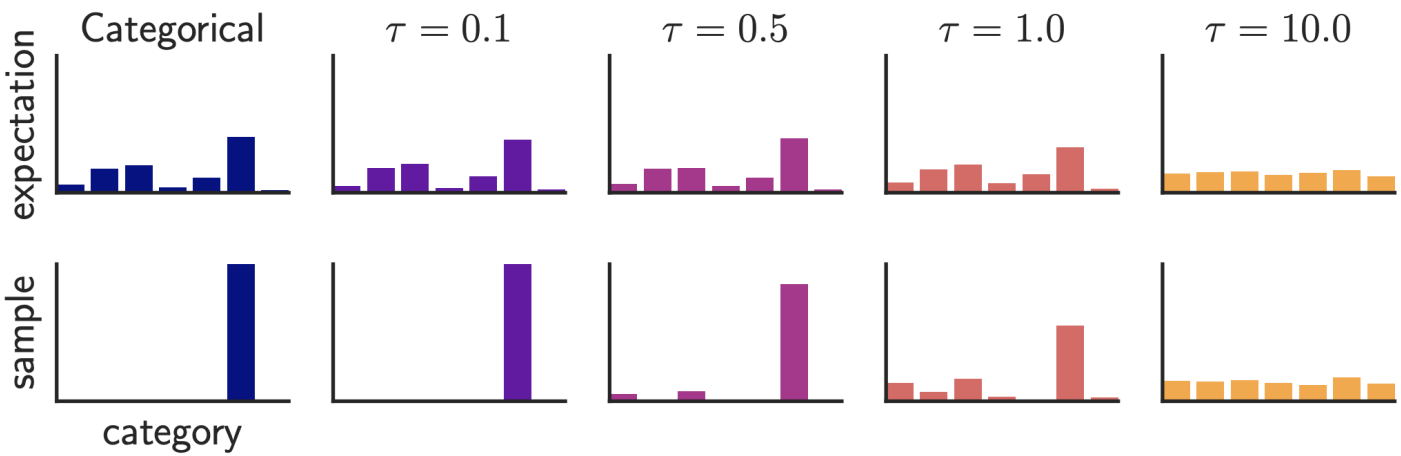
Gumbel-Softmax (Jang et al., 2016)

- Let $d = 1$ for simplicity, $q_\phi(\cdot|x) = \text{softmax}(\textbf{enc}_\phi(x))$

$$\frac{\partial}{\partial \phi}\left(\mathop{\mathbf{E}}_{z\sim q_\phi(\cdot|x)}\left[\log \kappa_\theta(x|z)\right]\right) \approx \mathop{\mathbf{E}}_{\epsilon\sim \text{Gumbel}^K(0,1)}\left(\frac{\partial}{\partial \phi}\left(\log \kappa_\theta\left(x\middle|\underbrace{\text{softmax}\left(\frac{\textbf{enc}_\phi(x)+\epsilon}{\tau}\right)}_{[0,1]^K}\right)\right)\right)$$

Note κ_θ must handle vector representation of z

- In practice use fixed τ (e.g., 0.9)



Striaght-Through Gumbel-Softmax

- For a fixed temperature the actual input to κ_θ is never sparse
- Idea: Enforce sparsity by sampling and use straight-through
  - More aligned with test time (when we actually sample), some applications need sparse input (reinforcement learning)
- **Forward pass**

$$\begin{aligned}
\epsilon &\sim \text{Gumbel}^K(0,1) \\
\delta_\tau &= \text{softmax}\left(\frac{\textbf{enc}_\phi(x)+\epsilon}{\tau}\right) \\
z &\sim \text{Cat}(\delta_\tau) \\
\tilde{z} &= \delta_\tau + (\text{one-hot}(z) - \delta_\tau).\texttt{detach}() \\
J_{\text{recon}} &= \log \kappa_\theta(x|\tilde{z})
\end{aligned}$$

- Can be seen as approximating the gradient of a "snap" function with a linearization

Example: Bernoulli Mixture-Prior Model (Dong et al., 2019)

- $x \in \mathbb{R}^V$ raw document vector (V vocab size): binary if bag-of-words, real-valued if TFIDF
- Latent: $z \in \{0,1\}^m$ "hash code" of x ($m \ll V$), $c \in \{1 \dots C\}$ mixture component
- LVGM

$$p_\theta(x, c, z) = \pi_\theta^C(c) \times \pi_\theta^{Z|C}(z|c) \times \kappa_\theta(x|z)$$

- Generative story
  - Draw a component from $\pi_\theta^C(\cdot) = \mathrm{softmax}_c(v)$ ($v \in \mathbb{R}^C$ learnable)
  - Given $c \in \{1 \dots C\}$, draw a hash code from $\pi_\theta^{Z|C}(\cdot|c) = \sigma(u_c)$ where $u_c \in \mathbb{R}^d$ (learnable) and σ is element-wise sigmoid
  - Given $z \in \{0,1\}^m$, "draw" a document from $\kappa_\theta(\cdot|z)$ where $\kappa_\theta(x|z) = \prod_{i:x_i>0} \mathrm{softmax}_i(Ez)$ ($E \in \mathbb{R}^{V \times m}$ learnable)
- Will never really generate, so it's fine κ_θ is weird

(Example Cont.) Inference Network and KL

- Approximate posterior: For any $x \in \mathbb{R}^V$, $c \in \{1 \dots C\}$, $z \in \{0,1\}^m$

$$q_\phi(c, z|x) = q_\phi^{C|X}(c|x) \times q_\phi^{Z|X}(z|x)$$

  Similar parametrization: $q_\phi^{C|X}(\cdot|x) = \text{softmax}(\textbf{enc}_\phi(x))$ and $q_\phi^{Z|X}(\cdot|x) = \sigma(\textbf{enc}'_\phi(x))$

- Thanks to independence assumptions, we can compute KL directly

$$D_{\text{KL}}(q_\phi(\cdot|x)||\pi_\theta)$$

$$= \underbrace{D_{\text{KL}}(q_\phi^{C|X}(\cdot|x)||\pi_\theta^{C}(\cdot))}_{\sum_c q_\phi^{C|X}(c|x) \log \frac{q_\phi^{C|X}(c|x)}{\pi_\theta^{C}(c)}} + \sum_{c=1}^{C} q^{C|X}(c|x) \underbrace{D_{\text{KL}}(q_\phi^{Z|X}(\cdot|x)||\pi_\theta^{Z|C}(\cdot|c))}_{\sum_{i=1}^m \sum_{z_i=0,1} q_\phi^{Z_i|X}(z_i|x) \log \frac{q_\phi^{Z_i|X}(z_i|x)}{\pi_\theta^{Z_i|C}(z_i|c)}}$$

(Example Cont.) Straight-Through

- Reconstruction term (given x)

$$\mathop{\mathbf{E}}_{(c,z)\sim q_\phi(\cdot|x)}[\log \kappa_\theta(x|z)] = \mathop{\mathbf{E}}_{z\sim q_\phi^{Z|X}(\cdot|x)}[\log \kappa_\theta(x|z)]$$

- Forward pass with straight-through and "data-dependent noise" given by $\sigma_\phi : \{0,1\}^V \to \mathbb{R}$ (control variance of z adaptively for input)

$$\begin{aligned}
\delta &= \sigma(\mathbf{enc}'_\phi(x)) \\
z &\sim \delta && \text{(i.e., } z_i \sim \text{Bernoulli}(\sigma([\mathbf{enc}'_\phi(x)]_i))\text{)} \\
\tilde{z} &= \delta + (z - \delta).\texttt{detach}() \\
\epsilon &\sim \mathcal{N}(0_m, I_{m\times m}) \\
\bar{z} &= \tilde{z} + \sigma_\phi(x) \odot \epsilon && \text{(i.e., } \bar{z} \sim \mathcal{N}(\tilde{z}, \text{diag}(\sigma^2_\phi(x)))\text{)} \\
J_{\text{recon}} &= \log \kappa_\theta(x|\bar{z})
\end{aligned}$$

- Once the model is learned, use $q_\phi^{Z|X}$ to hash documents

Posterior Collapse in VAEs

- What's one undesirable strategy to maximize

$$\mathrm{ELBO}(\theta, \phi) = \mathop{\mathbf{E}}_{x \sim \mathbf{pop},\, z \sim q_\phi(\cdot|x)} [\log \kappa_\theta(x|z)] - \mathop{\mathbf{E}}_{x \sim \mathbf{pop}} [D_{\mathrm{KL}}(q_\phi(\cdot|x)||\pi_\theta)]$$

Posterior Collapse in VAEs

- What's one undesirable strategy to maximize

$$\mathrm{ELBO}(\theta, \phi) = \mathop{\mathbf{E}}_{x \sim \mathbf{pop},\, z \sim q_\phi(\cdot|x)} [\log \kappa_\theta(x|z)] - \mathop{\mathbf{E}}_{x \sim \mathbf{pop}} [D_{\mathrm{KL}}(q_\phi(\cdot|x)||\pi_\theta)]$$

- Annihilate the KL term by setting $q_\phi(z|x) = \pi_\theta(z)$!
- Decoder κ_θ will then learn to ignore z
  - Degenerates to unconditional generative model
  - Known as **posterior collapse**
- Many strategies to alleviate
  - Annealing: Gradually increase KL weight β during training
  - Free bits (Kingma et al., 2016): Don't penalize KL if it's less than λ bits. Per-dimension version:

$$\sum_{i=1}^{d} \max \left\{ \lambda, D_{\mathrm{KL}}(q_\phi^{Z_i|X} || \pi_\theta^{Z_i}) \right\}$$

- Current best practice (Li et al., 2019): Do both with encoder pretraining (pretrain without KL term, reset decoder, train with annealing on the free-bits KL term)

Quantities to Monitor During VAE Training

- Marginalized log likelihood (not equal to ELBO)
- ELBO, consisting of two terms
  - Reconstruction term: Are we doing a good job of reconstructing the input? Always better in pure autoencoding
  - KL term: If this is zero, we have posterior collapse
- Mutual information between x and z
- Number of active units (Burda et al., 2016): z_i is ϵ-active if

$$\underset{\substack{x \sim \mathbf{pop} \\ z_i \sim q_\phi^{z_i|X}(\cdot|x)}}{\mathrm{Cov}} (x, z_i) > \epsilon$$