Discrete VAE's

John Thickstun

Previously, we've considered VAE's with continuous latent code spaces $\mathcal{Z}$. We've also considered GMM's over finite code spaces. We'll now take a look at models where the latent code space is finite, but $|\mathcal{Z}| = K$ is so large that is not amenable to the exact techniques we applied to GMM. Why would we be interested in learning discrete latent codes? One perspective on latent variable models is that we are want to learn a compressed representation of our data: i.e. $\mathcal{Z}$ is a "small" space that characterizes interesting information about observations in $\mathcal{X}$. One way to interpret this idea of compression is in the sense of dimensionality: we often construct VAE's using a latent code space that is much lower dimensional than $\mathcal{X}$. But this is an information theoretically poor interpretation of compression; we can store arbitrary amounts of information (in the sense of Shannon) in to a single latent scalar value. In contrast, if $\mathcal{Z}$ is a finite space, then we have a very precise notion of compression: the bits of information represented by $\mathcal{Z}$ is upper bounded by $\log_2(K)$.

A Discrete Variational Lower Bound

Recall that for GMM we computed marginal likelihood by explicitly marginalizing over the latent variable:

$$p(x) = \int_{\mathcal{Z}} p(x|z)p(z)\,dz = \sum_{k=1}^{K} \pi_k \mathcal{N}(x; \mu_k, \Sigma_k). \tag{1}$$

In the continuous domain, directly computing this integral became intractable and we resorted on approximations based on importance sampling to train our a model. When $\mathcal{Z}$ is finite, but $|\mathcal{Z}| = K$ is very large we find ourselves in a similar situation, so we will revisit the same importance sampling techniques that lead us to the ELBO and VAE.

Recall the evidence lower-bound that we derived previously:

$$\hat{\theta}_{\text{MLE}} = \arg\max_{\theta} \sup_{q} \mathop{\mathbb{E}}_{\substack{x\sim p \\ z\sim q_\varphi(\cdot|x)}} \big[\log p_\theta(x|z) - D(q_\varphi(z|x) \parallel r(z))\big]. \tag{2}$$

Nothing about that derivation required that $\mathcal{Z}$ be continuous, so we can use this variational objective to optimize a discrete VAE. If $\mathcal{Z}$ is discrete, it's natural to consider a uniform prior $r(z)$. The divergence term in the evidence lower-bound (2) is just the entropy

$$D(q_\varphi(z|x) \parallel r(z)) = \log(K) - H(q_\varphi(z|x)). \tag{3}$$

The likelihood term remains unchanged, and we can write the combined variational bound as

$$\mathcal{L}(x, \theta, \varphi) = \mathop{\mathbb{E}}_{z\sim q_\varphi(\cdot|x)} [\log p_\theta(x|z)] + H(q_\varphi(z|x)) - \log(K) \tag{4}$$

However, we cannot directly get gradients of (4) with respect to φ and, unlike the case for continuous codes, we cannot fix this with the "reparameterization trick."

The Policy Gradient Theorem

One way to construct a monte carlo gradient estimator for the likelihood term in (4) makes use of the Policy Gradient Theorem, aka REINFORCE [Williams, 1992]:

$$
\begin{aligned}
\nabla_\varphi \mathop{\mathbb{E}}_{z\sim q_\varphi(\cdot|x)} \big[\log p_\theta(x|z)\big] &= \sum_{z\in\mathcal{Z}} \log p_\theta(x|z)\nabla_\varphi q_\varphi(z|x) \\
&= \sum_{z\in\mathcal{Z}} \log p_\theta(x|z) q_\varphi(z|x)\nabla_\varphi \log q_\varphi(z|x) \\
&= \mathop{\mathbb{E}}_{z\sim q_\varphi(\cdot|x)} \big[\log p_\theta(x|z)\nabla_\varphi \log q_\varphi(z|x)\big] && (5) \\
&= \mathop{\mathbb{E}}_{z\sim q_\varphi(\cdot|x)} \left[\frac{\log p_\theta(x|z)}{q_\varphi(z|x)}\nabla_\varphi q_\varphi(z|x)\right]. && (6)
\end{aligned}
$$

This gradient estimator applies equally well in the continuous case, and is discussed in the original VAE paper [Kingma and Welling, 2014] where it is dismissed due to its high variance in comparison to the path estimator constructed via reparameterization. The variance of this estimator is a serious problem in the discrete setting as well.

The divergence term (3) is also troublesome in this setting; unlike the Gaussian case, there is no closed-form solution so we will need to use monte carlo estimators. Observe that

$$
H(q_\varphi(z|x)) = \mathop{\mathbb{E}}_{z\sim q_\varphi(\cdot|x)} [-\log q_\varphi(z|x)] = -\sum_{z\in\mathcal{Z}} q_\varphi(z|x)\log q_\varphi(z|x). \tag{7}
$$

The sum over the latent space $\mathcal{Z}$ in (7) is a big problem; we'll want to approximate this term with importance sampling. We can get gradients via REINFORCE, reiterating the previous caveat that the variance of this gradient estimator will be very high:

$$
\nabla_\varphi \mathop{\mathbb{E}}_{z\sim q_\varphi(\cdot|x)} [-\log q_\varphi(z|x)] = -\mathop{\mathbb{E}}_{z\sim q_\varphi(\cdot|x)} \left[\frac{(1+\log q_\varphi(z|x))}{q_\varphi(z|x)}\nabla_\varphi q_\varphi(z|x)\right]. \tag{8}
$$

Variance reduction techniques are considered in Tucker et al. [2017].

Gumbel Softmax

Another gradient estimator that avoids the variance issues with REINFORCE is Gumbel-softmax estimation [Jang et al., 2016, Maddison et al., 2016]. This estimator can sometimes work quite well, especially when K is not too large, but the caveat is that it is a biased estimator and can be unreliable. Suppose we want to draw samples from a categorical distribution $q = \text{Categorical}(k)$ with class probabilities $q_1, \dots, q_k$. If $g_1, \dots, g_k \sim \text{Gumbel}(0,1)$ then $z \sim q$ where

$$
z = \arg\max_i \big[g_i + \log q_i\big].
$$

This procedure for sampling from q lends itself to the "reparameterization trick":

$$
\mathop{\mathbb{E}}_{z\sim q} \big[f(z)\big] = \mathop{\mathbb{E}}_{g_i\sim\text{Gumbel}(0,1)} \left[f\left(\arg\max_i\, [g_i + \log q_i]\right)\right].
$$

However, we still cannot differentiate through the argmax.

The Gumbel-softmax estimator relaxes the argmax operation to a softmax with a temperature parameter τ:

$$\mathop{\mathbb{E}}_{g_i \sim \text{Gumbel}(0,1)} \left[f \left(\frac{\exp((g_i + \log q_i)/\tau)}{\sum_{j=1}^{k} \exp((g_j + \log q_j)/\tau)} \right) \right].$$

As $\tau \to 0$ the relaxation recovers the argmax, but gradient estimates blow up, so there is a tradeoff between the bias of the estimator and our ability to use it to optimize.

The Straight-Through Estimator

The straight-through estimator was initially proposed by Bengio et al. [2013] (with credit to earlier lecture notes by Hinton) as a means for differentiating through thresholding operations. Consider a threshold function $\text{th} : \mathbb{R} \to \mathbb{R}$ s.t. $\text{th}(x) = \mathbf{1}(x \geq 0)$ and consider a composition $f = g \circ \text{th} \circ h$ for differentiable real functions g, h. Observe that $\nabla_x \text{th} = 0$ and therefore $\nabla_x f(x) = 0$:

$$\begin{aligned} \nabla_x f(x) &= \left[(\nabla_x f)(\text{th} \circ h(x)) \right] \nabla_x \text{th} \circ h(x) \\ &= \left[(\nabla_x f)(\text{th} \circ h(x)) \right] \left[(\nabla_x \text{th})(h(x)) \right] \nabla_x h(x) = 0. \end{aligned}$$

The straight-through estimator $\tilde{\nabla}_x$ replaces $\nabla_x \text{th}$ with the constant function $\mathbf{1}$:

$$\tilde{\nabla}_x f(x) = \left[(\nabla_x f)(\text{th} \circ h(x)) \right] \nabla_x h(x).$$

I.e. we compute gradients of th as if it were the identity function. Another way of stating this operation is that we "copy gradients" across the non-differentiable operation th; i.e.

$$\tilde{\nabla}_x \text{th} \circ h(x) = \nabla_x h(x).$$

This idea can be adapted to differentiate through a discrete sampling operation. Suppose we have a categorical distribution $q_\varphi \in \Delta^k$ and we want to estimate

$$\nabla_\varphi \mathop{\mathbb{E}}_{z \sim q_\varphi} [f(z)].$$

The straight-through estimator ignores the sampling operation (i.e. the expectation) and formally defines the gradient of a sampled value $z \sim q_\varphi$ to be $\tilde{\nabla}_\varphi z \equiv \nabla_\varphi q_\varphi(z)$. We then compute a gradient of an expression containing sampled values z by application of the chain rule:

$$\tilde{\nabla}_\varphi f(z) = (\nabla_z f(z)) \tilde{\nabla}_\varphi z = (\nabla_z f(z)) \nabla_\varphi q_\varphi(z).$$

I.e. we take the gradient of the sampled element z to be the gradient of the categorical distribution's z'th component. More formally, the straight-through estimator is

$$\tilde{\nabla}_\varphi \mathop{\mathbb{E}}_{z \sim q_\varphi} [f(z)] \equiv \mathop{\mathbb{E}}_{z \sim q_\varphi} \left[(\nabla_z f(z)) \nabla_\varphi q_\varphi(z) \right].$$

It's not entirely clear to me why this should work. But it's been used effectively in recent formulations of discrete VAE's [Van Den Oord et al., 2017, Razavi et al., 2019].

Automatic Differentiation

We'll conclude the discussion of discrete VAE's with a brief comment on efficient implementation of REINFORCE-style algorithms using an autodiff system. This trick is well-known in the reinforcement learning community, but I'm not sure how broadly people are aware of it. Writing out the gradient estimates proposed by REINFORCE naively leads us to construct estimators like these:

$$\begin{aligned}
&\nabla_\theta \mathcal{L}(x,z,\varphi,\theta) \approx \nabla_\theta \log p_\theta(x|z),\\
&\nabla_\varphi \mathcal{L}(x,z,\varphi,\theta) \approx \left(\log p_\theta(x|z) - 1 - \log q_\varphi(z|x)\right)\nabla_\varphi \log q_\varphi(z|x).
\end{aligned} \tag{9}$$

This approach has some computational downsides: we need to build separate proxy expressions to compute the gradients with respect to θ and φ, and we will waste time recomputing the forward pass of common subexpressions. We'll typically also want to make a separate forward pass on the full objective to evaluate the loss itself. But if we're clever, we can write down the loss in a form that directly allows us to get all the required gradients with a single autodiff operation.

Let's start with the full discrete VAE formulation with gradients given by Equation (9). Approximating the loss $\mathcal{L}(x,\varphi,\psi)$ given by Equation (4) using a sample $z \sim q_\varphi(\cdot|x)$, we have the objective

$$\begin{aligned}
\widehat{\mathcal{L}}(x,z,\varphi,\psi) &= \log p_\theta(x|z) - \log(|\mathcal{Z}|) - \log q_\varphi(z|x)\\
&= \frac{q_\varphi(z|x)\log p_\theta(x|z)}{\{q_\varphi(z|x)\}} && \text{(likelihood)}\\
&\quad + 1 - \log(|\mathcal{Z}|) - \frac{q_\varphi(z|x)\left(1 + \{\log q_\varphi(z|x)\}\right)}{\{q_\varphi(z|x)\}} && \text{(divergence)}.
\end{aligned}$$

Ignoring the curly brackets, observe that this truly is a simple algebraic identity. The curly brackets indicate that when we take gradients, we will treat the terms inside as constants (in the language of e.g., PyTorch, we "detach" this subexpressions from the graph). Following this rule, the gradients of $\widehat{\mathcal{L}}$ match the gradients derived in Equation (9).

References

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013. (document)

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016. (document)

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *International Conference on Learning Representations (ICLR)*, 2014. (document)

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016. (document)

Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. In *Advances in Neural Information Processing Systems*, pages 14866–14876, 2019. (document)

George Tucker, Andriy Mnih, Chris J Maddison, John Lawson, and Jascha Sohl-Dickstein. Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models. In *Advances in Neural Information Processing Systems*, 2017. (document)

Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, 2017. (document)

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 1992. (document)