VQVAE2

1. Intro: VAE

Consider a dataset consisting of iid samples of a continuous or discrete variable x. We assume that the data are generated by a random process involving an unobserved latent variable z. Consider a directed deep latent variable model $p(x, z) = p(x|z)p(z)$. By introducing a variational distribution (encoder) q_ϕ over z that approximates the intractable true posterior $p_\theta(z|x)$, we can optimize a lower bound on the marginal log likelihood (even when the integral of the marginal likelihood is intractable or when the true posterior density is intractable),

$$\log p_\theta(x^{(i)}) \geq L(\theta, \phi; x^{(i)}) = \mathbb{E}_{q_\phi(z|x)}[-\log q_\phi(z|x) + \log p_\theta(x, z)]$$

$$= -D_{KL}(q_\phi(z|x^{(i)})||p_\theta(z)) + \mathbb{E}_{q_\phi(z|x^{(i)})}\Big[\log p_\theta(x^{(i)}|z)\Big]$$

We can think of this as selecting the element of the family of distributions parameterized by a nn $q(z|x)$ that most closely approximates the true posterior distribution $p(z|x)$.

We learn the parameters ϕ and θ jointly using an encoder/decoder framework.

We want this to scale to large datasets. The naive Monte Carlo gradient estimator has high variance but by the reparameterization trick we can derive a low variance gradient estimator. Now let $\log q_\phi(z|x^{(i)}) = \log \mathbb{N}(z; \mu^{(i)}, \sigma^{2(i)}I)$ a multivariate Gaussian with diagonal covariance and we have the classic VAE. Here the reparameterization trick lets us sample using $z = \mu + \sigma\epsilon$ where $\epsilon \sim \mathbb{N}(0, I)$.

But VAEs have many problems. Poor posterior estimates and in the worst case posterior collapse. This prevents us from using, for example, autoregressive decoders. You can even intentionally remove things from z by using a decoder network that is particularly good at representing certain features.

2. VQVAE

Introduced in Neural Discrete Representation Learning by Aaron van den Oord and Oriol Vinyals.

Here we consider discrete latent variables and vector quantize our latent representation. Define a latent embedding space $e \in R^{K \times D}$ where K the size of the discrete latent space and D the dimension of each embedding vector e_i. Encode input $z_e(x)$ then z is the nearest neighbor in the embedding space e. This is like a nonlinear map from latents to a 1-of-K embedding. So the posterior is a categorical distribution, $q(z = k|x) = 1$ for $k =_j ||z_e(x) - e_j||_2$. Define a uniform prior over z and we get a KL divergence $(\log K)$.

Important note: There is no gradient in the embedding space, we copy gradients from $z_q(x)$ decoder input to $z_e(x)$ encoder output.

To optimize this we need to play some games with the objective function.

$$L = \log p(x|z_q(x)) + ||sg[z_e(x)] - e||_2^2 + \beta||z_e(x) - sg[e]||_2^2$$

where sg is a stop gradient operator which is identity at forward pass. The first loss term is reconstruction loss and optimized by the decoder. Since the embeddings receive no gradients we learn the embedding space with a dictionary learning algorithm called vector quantisation and use l_2 error to move the embedding vectors e_i towards encoder outputs $z_e(x)$. This is the second term. If the embeddings don't train as fast as the encoder parameters the embedding space can grow arbitrarily, so to force the model to commit to an embedding we add a 'commitment loss' which is the third term.

To generate fit an autoregressive distribution over z so that we can generate x by ancestral sampling. In this case use PixelCNN over discrete latents for images and WaveNet for audio.

PixelCNN models the joint distribution of pixels over an image as the product of conditional distributions,

$$p(x) = \prod_{i=1}^{n^2} p(x_i|x_1, ..., x_{i-1})$$

ordering the pixel dependencies in raster scan order so each pixel depends on the pixels above and to the left. Every conditional distribution is a convnet.

3. VQVAE2: Model OV

The core of the model is in two stages:
1. train a hierarchical VQVAE 2. fit PixelCNN (PixelSnail) over the induced discrete latent space

4. VQVAE2: Codebook Loss -> Exponential Moving Average

Let $z_{i,1}, z_{i,2}, ...$ be the set of n_i outputs from the encoder that are closest to the dictionary item e_i so that the second term of the loss is,

$$\sum_{j}^{n_i} ||z_{i,j} - e_i||_2^2$$

then the optimal value for e_i is simply the average of elements in the set,

$$e_i = \frac{1}{n_i} \sum_{j}^{n_i} z_{i,j}$$

and we see this used as the update in algorithms like K-means. But we can't get this update directly when working with minibatches so we use an exponential moving average as an online version of this update,

$$N_i^{(t)} := N_i^{(t-1)} * \gamma + n_i^{(t)}(1 - \gamma)$$

$$m_i^{(t)} := m_i^{(t-1)} * \gamma + \sum_{j}^{n_i^{(t)}} E(x)_{i,j}^{(t)}(1 - \gamma)$$

$$e_i^{(t)} := \frac{m_i^{(t)}}{N_i^{(t)}}$$

where $n_i^{(t)}$ is the number of vectors in $E(x)$ that will be quantized to codebook item e_i and γ is a decay parameter $\in [0, 1]$. Paper lets $\gamma = 0.99$.

5. VQVAE2: Hierarchical Model

We allow for a hierarchy of vector quantized representations so that local information and global information are modeled separately. For 256x256 images they use two levels. Encoder downsamples (by residual blocks) to a 64x64 representation that becomes our 'bottom level' latent map then downsamples to a 32x32 representation that becomes our 'top level' latent map.

6. VQVAE2: Priors

The prior over the top latent map has multi-headed self-attention so that it has a larger receptive field and can capture correlations in distance spatial locations. Residual gated convolution layers from PixelCNN are interspersed with causal multi-headed attention every five layers (aka PixelSNAIL). Causal convolutions allow high bandwidth access over a finite context size, attention allows access over a large context so the idea is to interleave these. The convolutions can be thought of as aggregating information to build the context over which to perform attentive lookup. Then [hacks] add dropout and 1x1 convolutions on top of PixelCNN stack to improve likelihood.

The conditional prior over the bottom latent map does not have self-attention (because of memory constraints) but instead uses large conditioning stakcs from the top prior. Training each prior separately lets us train larger models.