NVAE: A Deep Hierarchical VAE

1. Intro: Variational Inference

2. 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. We want to train a directed generative deep latent variable model $p(x, z) = p(x|z)p(z)$. Since the true posterior $p(z|x)$ is in general intractable we introduce 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 $\epsilon \sim \mathbb{N}(0, I)$.

3. Why VAEs Are Not Great

VAEs have very different requirements from classification networks, but we share architectures. VAEs maximize mutual information between input and latent variables while classification networks discard information about the input.

VAEs respond differently to overparameterization.

1. Overparameterizing decoder hurts test log-likelihood because log likelihood only depends on the generative model.

2. Overparameterizing encoder helps reduce amortization gap.

Very deep VAEs have instabilities due to unbounded KL divergence (to the point that SOTA VAEs omit batch norm to combat randomness that could amplify instability).

4. Make Model Architecture a First Class Citizen

Work on VAEs has focused on addressing statistical challenges,

1. better approximations to true posterior
2. formulating tighter bounds
3. reducing gradient noise
4. extending vaes to discrete variables
5. solving posterior collapse

This paper instead focuses on architecture design.

5. Problems with VQVAE

VQVAE2 is not a VAE. The loss does not correspond to a lower bound on data log-likelihood

Prior uses PixelCNN which is slow autoregressive model to sample.

6. NVAE: Deep and Hierarchical

In deep hierarchical VAEs to increase expressiveness of the approximate posterior and the prior, latent variables are partitioned, $z = z_1, z_2, ..., z_L$ where L is the number of groups. The prior is,

$$p(z) = \prod_l p(z_l|z_{<l})$$

and the approximate posterior is

$$q(z|x) = \prod_l q(z_l|z_{<l}, x)$$

The variational lower bound is,

$$L(x) := \mathbb{E}_{q(z|x)}[\log p(x|z)] - KL(q(z_1|x)||p(z_1)) - \sum_{l=2}^{L} \mathbb{E}_{q(z_{<l}|x)}[KL(q(z_l|x, z_{<l})||p(z_l|z_{<l}))]$$

where $q(z_{<l}|x) := \prod_{i=1}^{l-1} q(z_i|x, z_{<i})$ is the approximate posterior up to $(l-1)^{th}$ group.

7. Modeling Long Range Correlations

Hierarchical modeling alone is not sufficient to model long range correlations. We also need to increase the receptive field of the conv layers./ We could increase the conv kernel size but this increases the parameters in the model. Instead use depthwise convolutions.

Idea: For image of depth m, use m depth 1 kernels (instead of 1 depth m kernel) then use a 1x1 kernel to merge.

A $k \times k$ classic convolution mapping a C-channel tensor to the same size has k^2C^2 parameters. A depthwise convolution operating in the same regime has k^2C parameters.

But this limits expressiveness. So before we apply these convolutions we can expand the number of channels with a 1x1 regular convolution then map back to original channel size after with another 1x1 convolution.

8. Residual Blocks

Batch norm is back. Its negative effects were in evaluation and not in training, this is because slow-moving running batch norm statistics shift the layer. All we need to do is modify the momentum parameter and regularize the norm of the parameters.

Use swish activation for empirical reasons.

Use squeeze and excitation blocks to model correlations between channels.

Generative Model Residual Cell: BN -> conv1x1 -> BN/Swish ->depthwise conv5x5 -> BN-Swish -> conv1x1 -> BN -> SE -> +

Encoder Residual Cell: BN/Swish -> conv3x3 -> BN/Swish -> conv3x3 -> SE -> +

9. Memory Hacks

Use APEX library to cast as many operations as possible (especially convolutions) to half-precision floats. Reduces GPU memory by 40 percent. Fuse BN and swish and use 'gradient check-pointing' to reduce GPU memory by another 18 percent (on cifar10).

10. Unbounded KL Divergence

Hierarchical VAEs have traditionally faced challenges since $KL(q(z_l|x, z_{<z})||p(z_l|z_{<l}))$ is unbounded.

Sharp gradient updates could push the model to an unstable region.

Use residual norm distributions:

$$p(z_l^i|z_{<l}) := N(\mu_i(z_{<l}), \sigma_i(z_{<l}))$$
$$q(z_l^i|z_{<l}, x) := N(\mu_i(z_{<l}) + \Delta\mu_i(z_{<l}, x), \sigma_i(z_{<l}) * \Delta\sigma_i(z_{<l}, x)))$$

where Δs are relative location and scale wrt prior. So when prior move the posterior moves accordingly.

Use spectral regularization: Ensure the encoder doesn't change too much when input changes, smoothness. Regularize the Lipschitz constant. Estimating Lipschitz constant is intractable so use spectral regularization to minimize Lipschitz constant for each layer,

$$L_{SR} = \lambda \sum_i s^{(i)}$$

where $s^{(i)}$ is the largest singular value.

Add inverse autoregressive flows to each group in $q(z|x)$.