Supplements: Infinite Variational autoencoder for Semi-supervised Learnings

M. Ehsan Abbasnejad Anthony Dick
Anton van den Hengel
The University of Adelaide
{ehsan.abbasnejad, anthony.dick, anton.vandenhengel}@adelaide.edu.au

1. Mathematical Details of the Infinite Variational Autoencoder

We have the following

$$\begin{aligned} p\left(\mathbf{c}, \boldsymbol{\theta}, \mathbf{x}_{1,\ldots,n}, \alpha\right) &= \int\int p_{\boldsymbol{\theta}}(\mathbf{x}_{1,\ldots,n}|\mathbf{c}, \mathbf{z}) p(\mathbf{z}) p(\mathbf{c}|\boldsymbol{\pi}) p(\boldsymbol{\pi}|\alpha) d\boldsymbol{\pi} d\mathbf{z} \\ &= \int p_{\boldsymbol{\theta}}(\mathbf{x}_{1,\ldots,n}|\mathbf{c}, \mathbf{z}) p(\mathbf{z}) \left[\int p(\mathbf{c}|\boldsymbol{\pi}) p(\boldsymbol{\pi}|\alpha) d\boldsymbol{\pi}\right] d\mathbf{z} \\ &= \int \left(\left[\prod_i^n p_{\boldsymbol{\theta}_{\mathbf{c}_i}}(\mathbf{x}_i|\mathbf{z}_{\mathbf{c}_i}) p(\mathbf{z}_{\mathbf{c}_i})\right] \left[\int p(\mathbf{c}|\boldsymbol{\pi}) p(\boldsymbol{\pi}|\alpha) d\boldsymbol{\pi}\right] \right) d\mathbf{z}_{\mathbf{c}_i}. \end{aligned}$$

To perform inference so that the distributions of the unknown parameters are known, we use blocked *Gibbs sampling* by iterating the following two steps:

1. Sample for the unknown density of the observations with parameter $\boldsymbol{\theta}$ (we drop αbecause it is conditionally independent $\mathbf{x}$):

$$\mathbf{x}_{1,\ldots,n}, \boldsymbol{\theta} \quad \sim \quad p(\mathbf{x}_{1,\ldots,n}, \boldsymbol{\theta}|\mathbf{c})$$

2. Sample the base VAE assignments:

$$\mathbf{c} \quad \sim \quad p(\mathbf{c}|\mathbf{x}_{1,\ldots,n}, \boldsymbol{\theta}, \alpha)$$

For the first step, we use variational inference to find joint probability of the input and its parameter $\boldsymbol{\theta}$ *conditioned* on current assignments. Using standard variational inference, we have,

$$\begin{aligned} p(\mathbf{x}_{1,\ldots,n}, \boldsymbol{\theta}|\mathbf{c}) &= \int \prod_i p_{\boldsymbol{\theta}_{\mathbf{c}_i}}(\mathbf{x}_i|\mathbf{z}_{\mathbf{c}_i}) p(\mathbf{z}_{\mathbf{c}_i}) d\mathbf{z}_{\mathbf{c}_i} \\ &= \int \prod_i \frac{p_{\boldsymbol{\theta}_{\mathbf{c}_i}}(\mathbf{x}_i|\mathbf{z}_{\mathbf{c}_i}) p_{\boldsymbol{\theta}}(\mathbf{z}_{\mathbf{c}_i})}{q_\phi(\mathbf{z}_{\mathbf{c}_i}|\mathbf{x}_i)} q_\phi(\mathbf{z}_{\mathbf{c}_i}|\mathbf{x}_i) d\mathbf{z}_{\mathbf{c}_i} \\ &= \int \prod_i p_{\boldsymbol{\theta}_{\mathbf{c}_i}}(\mathbf{x}_i|\mathbf{z}_{\mathbf{c}_i}) \frac{p_{\boldsymbol{\theta}}(\mathbf{z}_{\mathbf{c}_i})}{q_\phi(\mathbf{z}_{\mathbf{c}_i}|\mathbf{x}_i)} q_\phi(\mathbf{z}_{\mathbf{c}_i}|\mathbf{x}_i) d\mathbf{z}_{\mathbf{c}_i} \end{aligned}$$

Taking the $\log$ from both sides and using Jensen's inequality, we have the following lower bound for the joint distribution of the observations conditioned on the latent variable assignments (VAE assignments in the infinite mixture):

$$\log\left(p(\mathbf{x}_{1,\ldots,n}, \boldsymbol{\theta}|\mathbf{c})\right) \geq \sum_i -\mathrm{KL}\left(q_\phi(\mathbf{z}_{\mathbf{c}_i}|\mathbf{x}_i) \| p_{\boldsymbol{\theta}}(\mathbf{z}_{\mathbf{c}_i})\right) + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}_i)}[\log p_{\boldsymbol{\theta}_{\mathbf{c}_i}}(\mathbf{x}_i|\mathbf{z}_{\mathbf{c}_i})]. \tag{1}$$

$$= \sum_i \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}_i)}[\log p_{\boldsymbol{\theta}_{\mathbf{c}_i}}(\mathbf{x}_i, \mathbf{z}_{\mathbf{c}_i}) - \log q_\phi(\mathbf{z}_{\mathbf{c}_i}|\mathbf{x}_i)] \tag{2}$$

Here, $\boldsymbol{\theta}$ denotes all the parameters in the decoder network and ϕ all the parameters in the encoder. Now, to compute the expectations in both the KL-divergence and the conditional likelihood of the second term, we use the sampling with the reparameterization trick. Thus, Equation 1 is rewritten as

$$\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}_i)}[\log p_{\boldsymbol{\theta}_{\mathbf{c}_i}}(\mathbf{x}_i, \mathbf{z}_{\mathbf{c}_i}) - \log q_\phi(\mathbf{z}_{\mathbf{c}_i}|\mathbf{x}_i)] \quad \approx \quad \frac{1}{L}\sum_{\ell=1}^{L} \log p_{\boldsymbol{\theta}_{\mathbf{c}_i}}(\mathbf{x}_i, \mathbf{z}^{\ell}_{\mathbf{c}_i}) - \log q_\phi(\mathbf{z}^{\ell}_{\mathbf{c}_i}|\mathbf{x}_i)$$

where $\mathbf{z}$ is taken from a differentiable function that performs a random transformation of $\mathbf{x}$. This differentiable transformation function allows for using stochastic gradient descent in backpropagation algorithm. We use $L = 2$ in our experiments.

For sampling the base VAE assignment $\mathbf{c}$, we know that

$$p(\mathbf{c}|\mathbf{x}_{1,\dots,n}, \boldsymbol{\theta}, \alpha) \quad = \quad \int \underbrace{p(\mathbf{c}|\mathbf{x}_{1,\dots,n}, \boldsymbol{\theta}, \boldsymbol{\pi})}_{\text{Multinomial distribution}} \quad \underbrace{p(\boldsymbol{\pi}|\alpha)}_{\text{Dirichlet distribution}} \quad d\boldsymbol{\pi}$$

This integral corresponds to a Multinomial distribution with Dirichlet prior where the number of components C is a constant. We have,

$$p(\mathbf{c}_1 \dots, \mathbf{c}_C|\mathbf{x}_{1,\dots,n}, \boldsymbol{\theta}, \boldsymbol{\pi}) \quad = \quad \prod_{j}^{C} \boldsymbol{\pi}_j^{n_j}, \qquad\qquad n_j = p_{\boldsymbol{\theta}_j}(\mathbf{c}_i = j|\mathbf{x}_i) \times \sum_{i=1}^{n} \mathbb{I}[\mathbf{c}_i = j],$$

Using the standard Dirichlet integration, we have

$$\begin{aligned} p(\mathbf{c}_1 \dots, \mathbf{c}_C|\mathbf{x}_{1,\dots,n}, \boldsymbol{\theta}) \quad &= \quad \int p(\mathbf{c}_1 \dots, \mathbf{c}_C|\mathbf{x}_{1,\dots,n}, \boldsymbol{\theta}, \boldsymbol{\pi}_1, \dots, \boldsymbol{\pi}_C) p(\boldsymbol{\pi}_1, \dots, \boldsymbol{\pi}_C) d\boldsymbol{\pi}_1, \dots, \boldsymbol{\pi}_C \\ &= \quad \frac{\Gamma(\alpha)}{\Gamma(\alpha+n)} \prod_{j=1}^{C} \frac{\Gamma(n_j + \alpha/C)}{\Gamma(\alpha/C)} \end{aligned}$$

where we can draw samples from the conditional probabilities as

$$p(\mathbf{c}_i|\mathbf{c}_{i-1}, \mathbf{c}_{i+1} \dots, \mathbf{c}_C, \mathbf{x}_{1,\dots,n}, \boldsymbol{\theta}) \quad = \quad \frac{\eta_j(\mathbf{x}_i) + \alpha/C}{n - 1 + \alpha}.$$

When taking the number of components to approach infinity, $C \to \infty$ it is easy to see that the results in the paper is obtained.

2. Base Variational Autoencoder's Architecture

The base autoencoder contains the following layers:

1. Input layer: Depending on the type of the input it's dimensions are different.

2. A fully connected layer with the number of hidden dimensions $\mathbf{h}$. This number of hidden dimensions is what is changed during training and infinite mixture uses infinite hidden dimensions. We use batch normalization in the input of this layer that according to our experiments helps with the convergence and better performance. The output of the batch normalization is used in $\tanh$ nonlinearity units. The original VAE paper did not use batch normalization.

3. The output of the last hidden layer is used in another fully connected layer with linear units to estimate the mean of the density layer μ. This is the mean of the Gaussian density of the latent variable $\mathbf{z}$. Since this density is multivariate, we found 10 percent of the hidden dimensions to performing the best. For the Dogs experiment in the paper, we used 2-dimensional latent space.

4. Similar to the mean layer μ, we have another layer with the same dimensions for estimating the diagonal entries of the density of the latent space σ.

5. For the decoder, we need to sample from the density of the latent variable $\mathbf{z}$ to compute the reconstruction. We use two samples though-out our experiments to estimate the expected reconstruction following below steps for the decoder:

(a) Sample from the latent multivariate Gaussian distribution with mean μ and variance σ.

(b) The sample is used in another fully connected layer with hidden dimensions $\mathbf{h}$. We use batch normalization in the input of this layer too. Batch normalization helps with searching for a latent space in a lower dimensions. The output of the batch normalization is used in $\tanh$ nonlinearity units.

(c) The output of the batch normalized latent space is used in another fully connected layer with sigmoid nonlinearity unit to reconstruct the input.

It should be noted that this non-symmetric autoencoder corresponds to the binary VAE described in the original paper (with minor changes that helped with its convergence stability and performance). We found this architecture to perform better than its alternative symmetric one for the semi-supervised learning application of ours for CIFAR-10 and MNIST. In evaluation of the model for computing the loss, we use the cross-entropy measure for $p(\mathbf{x}|\mathbf{z})$ since the variables are considered binary.

For 3D ModelNet and Dogs dataset, we use a symmetric variant that showed to be more effective. In the symmetric version, we changed all the $\tanh$ units to softplus ($\log(1 + \exp(x))$). The final step 5c is changed to the following: We use the hidden layer $\mathbf{h}$ to feed into two fully connected layers for the mean and variance of the decoder μ_{dec}, σ_{dec}. We then sample from this decoding density for the reconstruction of the input. For computing the loss, we just use the log-likelihood of the reconstruction.

All the code is implemented in Python using Lasagne[1] and Theano[2]. We will release the code for the submission amongst with the dogs dataset.

[1]https://github.com/Lasagne/Lasagne

[2]http://deeplearning.net/software/theano/