Guidelines to implement the code

Md Mahmudul Hasan

November 25, 2022

1 Background

β-VAE proposed in [1] is meant to compress the input information into a constrained multivariate latent distribution (encoding) to reconstruct it as accurately as possible (decoding). Essentially, β-VAE is a more general version of VAE [2] and has similar structure and operation as VAE. β-VAE has the same bi-chambered structure as that of VAE. Consequently, the functionality of the corresponding parts of the network i.e. the encoder and the decoder are also the same as VAE.

The major difference between β-VAE and VAE is that β-VAE has a factor β multiplied with the KL divergence component of the cost function. β-VAE minimizes the following objective function given by (1).

$$\min_{\phi,\theta} \quad E_{q(x)}\left[\beta D_{KL}(q_\phi(z|x)||p(z)) - E_{q_\phi(z|x)}[\log(p_\theta(x|z)]\right] \tag{1}$$

When $\beta = 1$, β-VAE recovers the regular VAE. $\beta > 1$ values put more weight on the gradients of the KL divergence term $D_{KL}(q_\phi(z|x)||p(z))$ compared to the log-likelihood term and the $\beta < 1$ values do the opposite. We will see the effects of such manipulations when we discuss the results later in this subsection.

2 About the code

We minimize the cost function given by (1). Typical β-VAE framework works with images as inputs, we have a special case here where the input set is 10 dimensional jointly Gaussian samples. Our low dimensional random variable z is of dimension 2.

Since GitHub does not allow me to upload larger files, I have uploaded a small dataset of only 100 data points for your reference. You can generate and use your own data to implement the code.

3 How the code works

the 'beta-vae-main' file has all the hyper-parameters with default values that you can change as required. 'beta-vae-layers' file has the structure of the neural networks i.e. the encoder and the decoder laid down. 'beta-vae-models' file defines the model input output relationships, calculates the cost function and calls in the encoder and decoder structures form 'vae-layers' to do so.

The 'beta-vae-main' file has the 'training step' defined in it that requires the the calculation of cost function defined in 'beta-vae-models'. 'beta-vae-main' has the training process ordered in terms of the number of epochs and iterations. 'beta-vae-main' calls in the 'beta-vae-models' to calculate and update the gradients of the cost function with respect to the model parameters in every iteration. The program stops after it runs for the default number of 'epochs' set at the parameter list in the 'beta-vae-main' file. The file 'data-processing' is called in by the main file 'beta-vae-main' to get the dataset shuffled and batched according to the batch size before every epoch.

The 'beta-vae-after-training' file is mainly used as a generative network after the actual training is done. Using the trained parameters of the decoder network, with latent Gaussian samples as input to the decoder we can produce the VAE generated data. Another type of data that can be generated using 'beta-vae-after-training' is encoder-decoder data. For that we provide the original samples at the encoder, and the encoder induced samples in the latent space are fed to the decoder and the generated data is collected at the decoder output.

References

[1] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "beta-vae: Learning basic visual concepts with a constrained variational framework," 2016.

[2] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.