UNDER REVIEW

RESCIENCE C

Replication

[Re] VAE Approximation Error: ELBO and Exponential Families

Anonymous[1, ID]

[1] Anonymous Institution

Edited by
(Editor)

Reviewed by
(Reviewer 1)
(Reviewer 2)

Received
03 February 2023

Published
–

DOI
–

Reproducibility Summary

Scope of Reproducibility – Variational Autoencoders fail to capture the data distribution when the encoder distribution is inflexible. When an inflexible encoder is trained jointly with a decoder using ELBO objective, the inability of the encoder to represent the posterior propagates into the decoder, causing approximation errors.
We reproduce two experiments in which we separately prepare the ground truth decoder and the encoder. Then, we maximize ELBO with respect to both modules jointly and observe degradation of reconstruction performance.
We also verify how the theoretical insight into the design of a discrete VAE in a Semantic Hashing application influences the choice of input features improving overall performance.

Methodology – We implement and train a toy experiment from scratch on a laptop. We use a mix of authors' code and publicly available code to reproduce a GAN reintepreted as a VAE. We use authors' code to reproduce Semantic Hashing experiment and make our own implementation, improving on the original results. We train models on single GTX 1080 Ti or A100 GPUs. We spend under 100 GPU hours for all experiments.

Results – We observe expected qualitative behavior predicted by the theory on all experiments. We improve the best semantic hashing model's test performance by 5 nats by using a more modern method for gradient estimation of discrete random variables.

What was easy – Following the recipes for experiment setup was easy once the theory details were worked out.

What was difficult – The paper enables verification of exponential family distributions VAE designs of arbitrary complexity, which require probabilistic modeling skills. Figuring out implementation of gradient estimation for discrete random variables is another challenge. We hope to bridge the gap between mathematical notation and implementation details by providing code.

Communication with original authors – We are extremely grateful to the authors for discussing the paper, encouraging us to implement experiments on our own, and suggesting directions for improving results over e-mail.

Copyright © 2023 Anonymous, released under a Creative Commons Attribution 4.0 International license.
Correspondence should be addressed to ()
The authors have declared that no competing interests exists.
Code is available at https://anonymous.4open.science/r/vae-58D2/.

1 Introduction

VAE [1] is a generative model of $p_\theta(x, z)$ where x is observed and z is a latent variable designed to generate novel samples of x given $z \sim p(z)$ using a *decoder* $p_\theta(x|z)$, jointly learned with an *encoder* $q_\phi(z|x)$ using an Evidence Lower Bound (ELBO) objective.
The gap between the true data likelihood and ELBO is the divergence between the approximate encoder and the true posterior:

$$\log p_\theta(x) - \mathcal{L}(\theta, \phi) = \mathbb{E}_{p_d}[D_{KL}(q_\phi(z|x)||p_\theta(z|x))] \tag{1}$$

Impressive results with VAEs involve employing autoregression [2], normalizing flows [3] and hierarchies of latent spaces [4] to make this gap low.

2 Scope of reproducibility

[5] restricts q_ϕ and p_θ to the exponential family that includes continuous Gaussian and discrete Bernoulli distributions. The subset of models that can model the posterior exactly under this restriction is shown to be narrow and is prescribing us to expect approximation errors in complex models. Further, when an encoder is factorized and cannot fully model the posterior, joint optimization forces the decoder to adapt to the posterior at the expense of reconstruction quality.

We support the following claims with our experiments:

- factorization assumptions in the encoder cause approximation errors in the decoder, as shown in Section 3;
- reinterpreting complex generative models as Gaussian VAEs cause approximation errors, as shown in Section 4;
- using term frequency (TF) features for a Bernoulli VAE in a Semantic Hashing task underperforms term counts, as shown in Section 5.

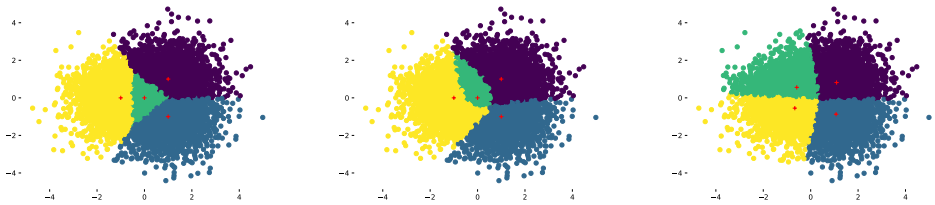
3 Gaussian Mixtures with Factorized Encoder

3.1 Methodology

We study the impact of the encoder design on a toy problem. We implement this experiment from scratch using PyTorch.

Model description — Following [5], we define a ground truth decoder $p^*(x, z) = p^*(z)p^*(x|z)$ to be a mixture of 4 2D Gaussians, where $x \in \mathbb{R}^2, z \in \{1, 2, 3, 4\}$, $p^*(z) = 0.25$, and $p^*(x|z) = \mathcal{N}(x|\mu_\theta(z), \sigma^2 I)$. $\mu(z)$ represents Gaussian means $(-1, 0), (0, 0), (1, 1), (1, -1)$ with an identity covariance scaled by $\sigma = 0.01$.

A single Linear layer $\theta : \mathbb{R}^4 \to \mathbb{R}^2$ encodes means as matrix columns. We apply θ to one hot encoded choice of z or its soft combination and obtain a mean parameter necessary to compute the log probability of a point under Gaussian distribution.



(a) (b) (c)

Figure 1. Factorized Encoder (a) ground truth (b) fixed decoder (b) joint training

We now refer to encoder definition: $q_\phi(z|x) \propto \exp\langle g_\phi(x), z_b\rangle$, where $\langle\cdot,\cdot\rangle$ is an inner product. g_ϕ is a network with randomly initialized three Linear layers with input dimension 2, hidden dimensions of 64, and output dimension of 2 with ReLU activations. We choose $z_b \in \{0,1\}^2$ to restrict the encoder to factor over two bits. To get $q_\phi(z|x)$, we evaluate $g_\phi(x)$ once and compute the inner products of its activations with every possible value of $z_b = [0,0],[0,1],[1,0],[1,1]$ and apply the softmax function computes exp and normalizes, making the output proper probability distribution.

Dataset — We make a training set x by sampling 2000 points from the ground truth decoder after setting the random seed to 0.

Experimental setup — We train the encoder by minimizing negative ELBO keeping the decoder frozen for 10000 steps. We use Adam with a learning rate of 1e-3. Finally, we unfreeze the decoder and train both modules jointly for 10000 steps. We use a batch size of 2000.

To calculate the reconstruction loss part of ELBO, we evaluate expectation

$$\mathbb{E}_{q_\phi} \log p_\theta(x|z) = \sum_i q_\phi(z_i|x) \log p_\theta(x|z = z_i)$$

exactly, running the decoder forward once for every possible z.

It takes under 10 minutes to run all experiments on a laptop CPU.

3.2 Results

After training the encoder alone, we achieved ELBO of -3.37 and observed a slight improvement during joint training to -3.31. The magnitude of the change looks insignificant; however completely changes the clusters' geometric structure as shown in Figure 1. ELBO values do not match the original paper due to differences in cluster center choices.

Increasing encoder dimensionality — We verify that getting rid of the factorization helps the encoder exactly model the posterior. We replace the encoder network with a single Linear layer with input dimension 2 and output dimension 4 and use the softmax activation function.

4 DCGAN turned Gaussian VAE on CelebA

4.1 Methodology

In this experiment, we train a ground truth decoder using a GAN objective, pretrain the encoder using conditional likelihood and then train both models jointly as a VAE.
We analyze the convergence of ELBO and how ELBO influences image generation. We use the code of [6] for implementing DCGAN [7]. We use the authors' code for conditional likelihood training of the encoder and joint fine-tuning.

Model description — We designate a trained DCGAN $g_\theta : \mathbb{R}^{100} \to \mathbb{R}^{64x64x3}$ as a ground truth decoder, making it probabilistic by adding Gaussian noise. We use a copy of DCGAN's generator as an encoder.

Dataset — We use an aligned and cropped version of the CelebA dataset. It contains 202,599 face images of 10,177 celebrity identities [8].

Experimental setup — To train DCGAN, we follow hyperparameters from [6], training for up to 100 epochs. We stop one epoch short of mode collapse. Next, we update the encoder using Adam optimizer with a learning rate of 2e-4 for 1e6 steps using conditional likelihood objective. Next, we use Adam optimizer with a learning rate of 2e-4 to jointly update both modules for 1000000 iterations.

Computational requirements — Pretraining of DCGAN takes about 5 hours. Training the encoder takes about 1 hour. Joint training takes about 3 hours.

4.2 Results

We observe ELBO improve after pretraining the encoder while FID increases, signaling a decrease in image quality.

Table 1. Train ELBO (higher is better)

Experiment	ELBO	FID
optimize encoder (conditional likelihood)	-19072.746	1.3108
optimize encoder and decoder (ELBO, fixed σ_d)	5137.171	71.852

5 Semantic Hashing on 20 Newsgroups

5.1 Methodology

We investigate a Bernoulli VAE for the Semantic Hashing task. In this task, we aim to learn an encoder that maps our document to a binary code we can index using, for example, Hamming distance. We compare the performance of term frequencies as features in [9] to term counts suggested in [5].
We first become acquainted with the experiment by running the code provided by the author. One experiment for one set of hyperparameters runs for about 30 minutes on one GPU. The whole set of experiments takes about 16 GPU hours in total. Next, we study the author's code by reimplementing it and improving results as described in Section 5.2.1.

Table 2. Train Negative ELBO (lower is better)

	d=0			d=1			d=2		
b	e1	e2	e3	e1	e2	e3	e1	e2	e3
8	**406**	419	432	**325**	382	407	**325**	363	415
16	**372**	383	418	**191**	298	397	**169**	287	406
32	**332**	343	409	**153**	181	391	**118**	170	405
64	**288**	304	405	**151**	174	391	**122**	165	401

Table 3. Test Negative ELBO (lower is better)

	d=0				d=1			d=2		
b	e1	ARM	e2	e3	e1	e2	e3	e1	e2	e3
8	**423**	**423**	444	439	**413**	453	427	**419**	431	431
16	**409**	**409**	451	430	**403**	466	431	**408**	448	441
32	**397**	**396**	439	433	**396**	454	455	**402**	466	473
64	**387**	392	460	467	**393**	486	481	**401**	484	478

Dataset — We use a preprocessed 20newsgroups dataset [10]. The vocabulary is restricted to 10000 most common words. Each document is represented as a vector of smoothed word counts. The training set contains 11269 documents, and a test set contains 7505 documents. We do not use text categories.

Model description — The encoder takes a vector of 10000 word counts as input and passes them through one linear layer with output dimension $b = 64$, representing logits of b Bernoulli variables. The decoder takes b binary digits as input and passes them through $d = 0$ hidden layers with hidden dimension 512 and output dimension matching vocabulary size. The final activation of the decoder is log of softmax.

Experimental Setup — We use a seed of 1, Adam with a learning rate of 1e-3 and a batch size of 256. We train the model for 1000 epochs estimating ELBO using an antithetic gradient estimator. We're additionally computing ELBO without augmentations to estimate training and test losses. Next, we train the model for 500 more epochs, averaging the loss over 10 backward passes for further gradient variance reduction.

We sweep over the following hyperparameters: varying latent dimension $b = 8, 16, 32, 64$, varying decoder hidden layer count $d = 0, 1, 2$, converting word counts to word frequencies together with an extra hidden layer of dimension 512 in the encoder (configuration `e2` in Section 5.2 below) and using word frequencies without extra hidden layers (configuration `e3`).

5.2 Results

We observe proposed word count features along with a simple encoder consistently achieve lower negative ELBO, as shown in Tables 2 and 3.

Gradient Estimation for Bernoulli Variables — Evaluation of ELBO involves estimating the decoder likelihood over the approximate posterior distribution. In discrete settings, implementations resort to Monte Carlo sampling. A commonly used method for gradient estimation of discrete variables is REINFORCE algorithm [11]. This estimator is unbiased and has high variance. Reducing variance involves taking many samples, trading variance for bias [12], and sample augmentation.

The ARM [13] estimator aims to reduce the variance by augmenting the random variable space re-parametrizing Bernoulli variables using a continuous Logistic distribution, and evaluating each random variable twice at "opposite" points in the space. However, authors of DisARM [14] observe that the Logistic reparametrization comes at the cost of introducing variance on its own and marginalizing out continuous variables by computing an expectation over them.
In our implementation, we use DisARM to estimate gradients and see the best performance improve by 5 nats, as shown in Table 3.

6 Discussion

We focused on the most fundamental examples of the exponential family models: continuous Gaussian and a discrete Bernoulli. The original paper provides a general framework to analyze more complex distributions that fall into exponential family, like Markov Random Fields, which we would like to explore in the future.

6.1 Communication with original authors

We would like to thank the authors for discussing the paper, encouraging us to implement experiments on our own, and suggesting directions for improving results over e-mail.

References

1. D. P. Kingma and M. Welling. **Auto-Encoding Variational Bayes**. 2013.
2. D. P. Kingma, T. Salimans, and M. Welling. "Improved Variational Inference with Inverse Autoregressive Flow." In: **ArXiv** abs/1606.04934 (2016).
3. D. J. Rezende and S. Mohamed. "Variational Inference with Normalizing Flows." In: **International Conference on Machine Learning**. 2015.
4. A. Vahdat and J. Kautz. "NVAE: A Deep Hierarchical Variational Autoencoder." In: **ArXiv** abs/2007.03898 (2020).
5. A. Shekhovtsov, D. Schlesinger, and B. Flach. "VAE Approximation Error: ELBO and Exponential Families." In: **International Conference on Learning Representations**. 2022.
6. N. Inkawhich. **DCGAN Tutorial**. https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html. Accessed: 2022-12-18.
7. A. Radford, L. Metz, and S. Chintala. **Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks**. 2015.
8. Z. Liu, P. Luo, X. Wang, and X. Tang. "Deep Learning Face Attributes in the Wild." In: **Proceedings of International Conference on Computer Vision (ICCV)**. Dec. 2015.
9. S. Chaidaroon and Y. Fang. "Variational Deep Semantic Hashing for Text Documents." In: **Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval** (2017).
10. J. Rennie and K. Lang. "20 Newsgroups." In: **Proceedings of the Twelfth International Conference on Machine Learning**. 1995, pp. 331–339.
11. R. J. Williams. "Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning." In: **Mach. Learn.** 8.3–4 (May 1992), pp. 229–256.
12. A. Shekhovtsov. "Bias-Variance Tradeoffs in Single-Sample Binary Gradient Estimators." In: **ArXiv** abs/2110.03549 (2021).
13. M. Yin and M. Zhou. **ARM: Augment-REINFORCE-Merge Gradient for Stochastic Binary Networks**. 2018.
14. Z. Dong, A. Mnih, and G. Tucker. "DisARM: An Antithetic Gradient Estimator for Binary Latent Variables." In: (2020).