Project AI: Discrete Variational Autoencoders with Logit-normals

Petru Neague (11844523) **Mirthe van Diepen** (10327428) **Alessandra van Ree** (10547320)[1]

Abstract

In this short paper we analyze the results of modelling a Variational Autoencoder (VAE) with a Logit-normal distribution as its property of having more weight towards the edges could make it more powerful than Gaussian based VAE's. We compare the experimental results of the Logit-normal with those of the Gaussian and the Concrete distributions to find that the first two are similar in performance while the replication of (Maddison et al., 2016) was not achieved. Thus the results of the Concrete distribution are not comparable. The reconstruction of the latent space using the Logit-normal is similar to that of the Gaussian, proving that the Logit-normal is a suitable candidate for the discrete latent space of a VAE.

1. Introduction

To this day, learning the discrete latent representation in Variational Autoencoders (VAE's; (Kingma & Welling, 2013)) is a challenging task. Originally, in (Kingma & Welling, 2013) the distribution modelling this latent variable was a Gaussian, being a good choice according to Central Limit Theorem. However, the Gaussian is fit for continuous data, but less so for discrete or binary data. At present, the most commonly used application to learn the discrete latent representation is the Gumbel-softmax trick, or the concrete distribution (Jang et al., 2016; Maddison et al., 2016).

In this short paper, we propose to model the discrete latent variable by use of the Logit-normal distribution. Due to the simple reparameterization of this distribution and the analytical properties of the Logit-normal, it makes for a good candidate to model the discrete latent variable in a VAE. As the values of a Logit-normal are between 0 and 1 it can be seen as a relaxation of said variable.

We aim at utilizing the Logit-normal distribution in a VAE and compare its qualitative and quantitative performance in terms of reconstruction error with the Gaussian and the Concrete models. To achieve this, the Logit-normal model has been implemented, and different priors with varying hyper-parameter settings were tested to obtain the best result. Along with the Logit-normal, the Gaussian and Concrete have been implemented as in (Kingma & Welling, 2013) and (Maddison et al., 2016) respectively. While the results of (Maddison et al., 2016) were not replicated, a comparison with the Gaussian model is thoroughly analyzed in the following sections. We show that the Logit-normal model has the potential to outperform the Gaussian model, if properly optimized. In the experiments their performance was very similar with the Gaussian model coming on top on some and the Logit-normal one winning on others; but further hyperparameter optimization of the Logit-normal is necessary to significantly boost its performance.

2. Method

The methodology of this research is described in the following three parts. First, a description of the architecture of the VAE; second, the distributions and their properties for the latent space are explained; and lastly, the composition and assumptions for the loss function for the models is described.

2.1. Architecture

The architecture consists of a VAE which is made up of an encoder, a reparameterization of the main variable to allow the taking of its derivative (and thus use gradient descent) and a decoder. First the input data (see section 4) is encoded by mapping it through two fully connected layers. The activation function between the layers is a $Tanh$ function (standard activation function) and the activation over the latter hidden layer is a $Softplus$ function (standard activation function that turns the output positive) . Thereafter, the layer splits into the different parameters of distribution of the latent space. These parameters are used for the reparameterization trick, i.e. to sample a latent variable. In the decoding part an estimation of the input data will be computed from the sampled latent variable. The decoding part consists of two fully connected linear layers with the same activation functions as in the encoder.

The optimizer of choice was deemed to be the Adaptive Moment algorithm (Adam) as it has been showed to be ”robust and well-suited to a wide range of non-convex optimization problems in the field machine learning” including MNIST data classification (Kingma. & Ba, 2014).

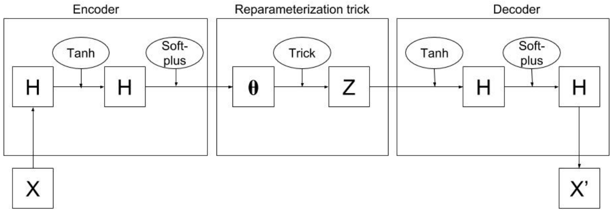

Figure 1. Algorithm architecture: X is an input image, goes through the encoder - 2 fully connected layers (represented here as H) with $Tanh$ and $Softplus$ activation functions, the resulting data is reparameterised to allow Gradient Descent, then another 2 layers compose the decoder part; the end result being a reconstructed image which is compared to the original image to calculate the negative log-Bernoulli loss.

The architecture of the algorithm is presented in Figure 1.

2.2. Reparameterization trick

For the first model the approximation of the distribution of the latent variable (i.e. the posterior) is defined as a Gaussian distribution. In this case the parameters $\boldsymbol{\mu}$ (mean) and $\boldsymbol{C}$ (covariance matrix) are trained. The choice of the covariance matrix may be important for the final results. The covariance matrix is defined as a diagonal matrix where the diagonal entries are squares of the entries of a vector $\boldsymbol{\sigma} \in \mathbb{R}^K$ where K is the dimension of the latent space. Sampling from this model can be done by $Z = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \epsilon$ where $\epsilon \sim \mathcal{N}(\mathbf{0}_K, I_K)$ is a sample of the standard normal-distribution. It is assumed that the prior of the latent variable has a standard-normal distribution.

The second model has a Concrete distribution as the posterior. Sampling from this distribution is the continuous relaxation of the Gumbel-Max trick. A sample from the Concrete distribution $Z \sim \text{Concrete}(\boldsymbol{\alpha}, \lambda)$ is of the form

$$Z_k = \frac{\exp((\log \alpha_k + G_k)/\lambda)}{\sum_{i=1}^{K} \exp((\log \alpha_i + G_i)/\lambda)}$$

where K is the dimension of the latent space, $\boldsymbol{\alpha} \in \mathbb{R}^K_{>0}$ are the locations, $\lambda \in \mathbb{R}_{>0}$ is the temperature and $G_k \sim \text{Gumbel}(0, 1)$. The density over $\boldsymbol{z}$ is given by

$$P(z) = \Gamma(K)\lambda^{K-1} \prod_{k=1}^{K} \frac{\alpha_k z_k^{-(\lambda+1)}}{\sum_{i=1}^{K} \alpha_i z_i^{-\lambda}}.$$

According to (Maddison et al., 2016) this method can be affected by underflow. A solution is sampling from the ExpConcrete distribution. This distribution preserves the density of the variable, see appendix C of (Maddison et al., 2016) for more details. In the implementation the posterior and prior are both defined as an ExpConcrete distribution, where the posterior has trained locations and temperature 0.5, and the prior has uniform locations and temperature 1.

For the latter model the posterior is defined as a Logit-normal distribution. The density of the multivariate Logit-normal distribution over $\boldsymbol{z}$ is

$$P(z) = \frac{1}{|2\pi \boldsymbol{C}|^{\frac{1}{2}}} \cdot \frac{1}{\prod_{i=1}^{K} z_i(1 - z_i)} e^{-\frac{1}{2}(\boldsymbol{y}-\boldsymbol{\mu})^T \boldsymbol{C}^{-1}(\boldsymbol{y}-\boldsymbol{\mu})}$$

where

$$\boldsymbol{y} = \ln\left(\frac{\boldsymbol{z}}{1 - \boldsymbol{z}}\right), \tag{1}$$

K is the dimension of the latent space, $\boldsymbol{C}$ is the covariance matrix and $\boldsymbol{\mu}$ is the mean. The covariance matrix is again a diagonal matrix with squares on the diagonal. In order to sample $Z \sim P(\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{C}))$, first sample $Y \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{C})$ and take the inverse of equation (1). It is assumed that the prior $p(\boldsymbol{z})$ has Logit-normal distribution with zero mean and a covariance matrix of the form $\alpha \cdot I_K$ with $\alpha \in \mathbb{R}_{>0}$.

2.3. Lower bound

According to (Kingma & Welling, 2013) the lower bound of a VAE is defined by

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}^{(i)}) = &- D_{\text{KL}}(q_{\boldsymbol{\phi}}(\boldsymbol{z} \mid \boldsymbol{x}^{(i)}) \| p_{\boldsymbol{\theta}}(\boldsymbol{z})) \\ &+ \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}^{(i)})}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)} \mid \boldsymbol{z})\right]\end{aligned} \tag{2}$$

where $p_{\boldsymbol{\theta}}$ is the prior and $q_{\boldsymbol{\phi}}$ is the approximation of the posterior. In order to maximize the lower bound, it is necessary to minimize the KL-divergence. Using Monte Carlo's rule the KL-divergence can be approximated by

$$\frac{1}{L}\sum_{l=1}^{L} -(\log p_{\boldsymbol{\theta}}(\boldsymbol{z}^{(l)}) - \log q_{\boldsymbol{\phi}}(\boldsymbol{z}^{(l)} \mid \boldsymbol{x}^{(i)})).$$

Besides minimizing the KL-divergence, the negative log likelihood must also be optimized. We will assume that the input data has a Bernoulli distribution which is an accurate assumption given the binary nature of the data. Hence, the negative log Bernoulli loss of $\boldsymbol{x}^{(i)}$ with probabilities $\hat{\boldsymbol{x}}$ must be minimized. The ELBO of a specific data point is given by the sum of the approximation of the KL-divergence with $L = 1$ and the negative log Bernoulli loss, see Equation (2). For each batch we take the mean over all values of ELBO to obtain the loss.

3. Algorithm reliability

The algorithm presented above has constantly throughout the project been subjected to reliability evaluations. In order to check whether it was working correctly a number of methods (of varying degrees of easiness of implementation) have been devised:

- Is the loss presented at the end of each epoch a real positive number? - required answer: “yes” Is it decreasing in subsequent epochs in subsequent epochs? - required answer: “yes”;

- Run the model to find the latent representation of the test data set; then check if it is clustered in the required clusters (e.g. for MNIST: the ‘ones’ belong in a different cluster than the ‘twos’ and so on) - required answer: “yes”;

- If the model was run for a 2-dimensional latent space, then plot the decoded version of a random subset belonging to the latent variables and check if they look like the training data set itself;

- Run a few data points through the algorithm (encoding, reparameterizing and decoding) and then check if the output is similar to the input.

The hyperparameters used in the model were taken from the mathematical recommendations (for example the Gumbel temperature was known to be optimally located between zero and ten). If the mathematical model of the algorithm does not actually suggest hyperparameter values, then information could be sought from similar models used by other research papers; and if that is not available either, targeted trials of believed-to-be-good hyperparameters could be implemented and their results compared; such that after much testing, the best set of hyperparameters would be kept.

The data set (MNIST) is composed of discrete classes (digits from zero to nine) so the Bernoulli classification mechanism is well chosen; if the data would have been continuous, this distribution would have had to be different;

4. Data

The data used for the experiments is the MNIST (LeCun et al., 2010) digit dataset, containing 60.000 training images and 10.000 test images. Due to the nature of the experiments and the models, the data needed to be binarized based on the grayscale values in the MNIST dataset. As in (Salakhutdinov & Murray, 2008), for every epoch the data was stochastically binarized based on the intensity of every pixel. By use of this intensity, for every pixel a sample was drawn from a Bernoulli distribution.

5. Experiments

The training has been done on 60.000 MNIST images with a training batch size of 50. All models were trained for 200 epochs, for latent dimension sizes $\in \{2, 4, 8, 20, 40\}$. The Logit-normal was subjected to a range of prior values of its covariance $C = \sigma I_k$ with $\sigma \in \{0.32, 0.56, 1., 1.78\}$. The Logit-normal with the best performing σ will be compared with the Gaussian and Concrete.

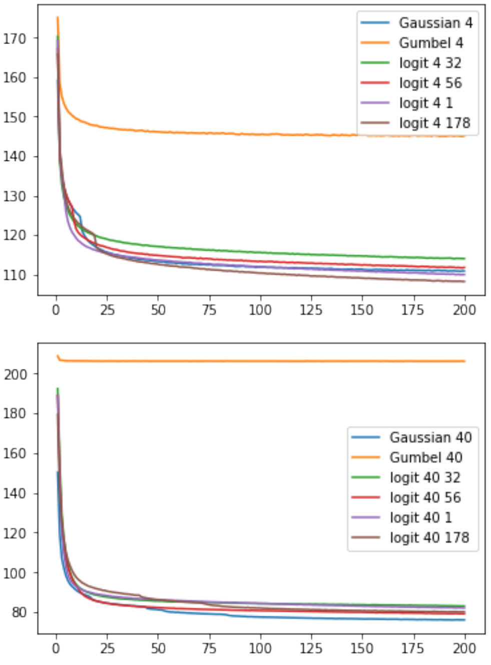

Figure 2. Evolution of Negative Log-Bernoulli loss for all distributions with latent dimension of 4 (top graph) and 40 (bottom graph).

These experiments will be evaluated in threefold.

1. Quantitative evaluation by inspecting the negative evidence lower bound (ELBO) and its components;

2. Quantitative evaluation of 2D latent space by inspecting a scatter plot and quantitative evaluation of high-dimensional latent space by inspecting a scatter plot of t-SNE;

3. Qualitative evaluation of histogram of latent variable’s values;

4. A reconstruction of the latent sapce for the Logit-normal.

6. Results

6.1. Loss

The negative log-Bernoulli loss is the one that can be compared between models (since it represents the reconstruction error); its evolution for the Logit-normal (with all its priors), the Gaussian and the Concrete distribution are shown in Figure 2 for a latent dimension of size 4 and 40. In Tables 1 and Table 2 the values ELBO and KL-divergence of the

	0.32	**0.56**	**1.00**	**1.78**
2	145.42	142.67	141.33	**140.21**
4	124.04	121.73	120.10	**118.63**
8	105.02	109.43	**101.61**	105.57
20	99.98	98.45	97.62	**96.97**
40	104.13	**100.25**	100.45	104.02

Table 1. The ELBO of Logit-normal posterior and prior. The columns are different values for $\alpha = 0.32, 0.56, 1, 1.78$ which refer to the prior with covariance matrix αI_K. And the rows correspond to the different dimensions of the latent space.

	0.32	**0.56**	**1.00**	**1.78**
2	7.00	6.79	6.53	6.37
4	10.03	9.97	10.14	10.43
8	16.02	14.92	16.71	15.47
20	21.02	21.37	20.13	22.17
40	21.02	21.01	18.31	23.84

Table 2. The KL-divergence of Logit-normal posterior and prior. The columns are different values for $\alpha = 0.32, 0.56, 1, 1.78$ which refer to the prior with covariance matrix αI_K. And the rows correspond to the different dimensions of the latent space.

Logit-normal models are displayed, respectively. Table 3 compares the ELBO methods of the Logit-normal with prior 1.78, Gaussian and the Gumbel models.

	Logit-normal	**Gaussian**	**Concrete**
2	140.21	**138.59**	185.88
4	**118.63**	121.87	167.86
8	105.57	**100.54**	158.62
20	96.97	**96.52**	169.05
40	104.02	**96.21**	227.97

Table 3. Comparison between the ELBO of the best Logit-normal model (the one having an $\alpha = 1.78$) with the ELBO values of the Gaussian model and Concrete model. The rows correspond to the different dimensions of the latent space.

The most visible feature of those graphs is that the performance of the Gumbel distribution seems to not be on par with the other models. This is in contrast with the results from (Jang et al., 2016) and is testament to the fact that we were not able to replicate their performance. It is not clear why that is but more research would be needed in this direction. Another feature is that we can see all of the rest models converging towards more or less the same region of performance, i.e. around 110 for 4 dimensions and around 80 for 40 dimensions. It can also be seen that the Gaussian performs slightly better at 40 dimensions while the Logit-normal is more efficient at 4 dimensions meaning that depending on the assignment and architecture one might be better than the other but the results will not be across the board. The different priors seem to be performing quite similarly on based on the ELBO and its components as can be seen in Table 1 and Table 2. However, the prior with $\sigma = 1.78$ has the best general performance on different dimensions in both the loss and the KL-divergence.

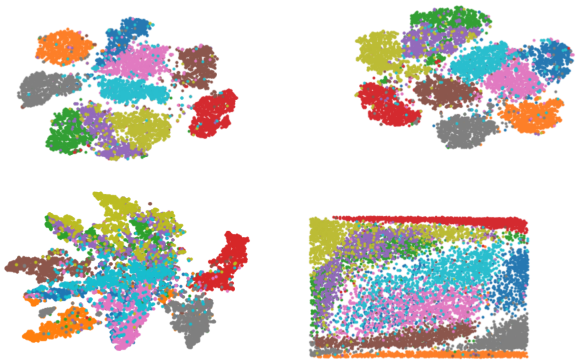

Figure 3. Encoding of 10.000 test-set images using Logit-normal with prior 0.32 (top left), Gaussian (top right) and concrete (bottom left) distributions for 8 dimensions - reduced to two dimensions with t-SNE algorithm; and the encoding of Logit-normal with prior 0.32 for 2 dimensions - without t-SNE (bottom right)

6.2. Latent clusters

In Figure 3 the performance of the encoding algorithm is shown. The different colors represent images of different digits (i.e. red dots represent images of the digit 'one' etc.). The plots have been reduced from 8 to two dimensions with the help of the T-distributed stochastic neighbor embedding algorithm (t-SNE, (Maaten & Hinton, 2008)). The figure shows a clear differentiation between all digits for the case of the Logit-normal and Gaussian distributions, while the concrete distribution lags behind them (due to the experiment replication issue discussed above). The 2-dimensional reprezentation does indeed cluster the numbers fairly well, although understandably inferior to higher dimensional processes.

The reconstruction of the latent space (from 0 to 1) for a two dimensional Logit-normal with a covariance prior $\alpha = 1.78$ is shown Figure 5. It can be seen that all MNIST classes are represented (i.e. all digits from zero to nine). This proves that the decoding part of the algorithm is indeed capable of reconstructing the necessary features to be able to clearly differentiate between classes.

6.3. Histogram z-values

A slice of one of the dimensions of the Gaussian and logit is shown in Figure 4. Comparing the distributions with the slice it is possible to see clearly the resemblance between

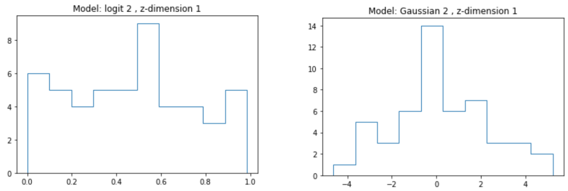

Figure 4. Standard slice of 40-Dimension Gaussian and Logit-normal with $\alpha = 1.78$ histogram of z-values

them. This is generally how slices have looked like for all the 40 dimensional models, meaning that all dimensions are well used and further increases in dimensionality might yield better results.

6.4. Reconstruction of latent space

In Figure 5 the reconstruction of the 2 dimensional latent space of the Logit-normal is displayed. This shows that all of the numbers from 0 to 9 can be reconstructed by the latent space based on the Logit-normal.

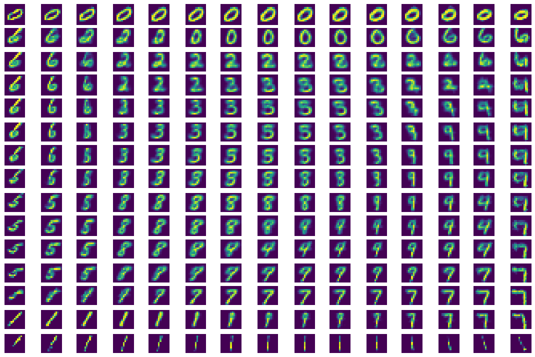

Figure 5. Latent space reconstruction for the Logit-normal model.

7. Conclusion

Despite the failure of replicating the results of using the concrete distribution in (Jang et al., 2016) a reliable comparison can still be made with the Gaussian distribution of (Kingma & Welling, 2013).
The performance of the Gaussian comes very close to that of the Logit-normal distribution, regardless of what prior is assumed; the priors for the Logit-normal do have an influence on the overall performance and it is possible that higher value priors would be more effective as in these cases the Logit-normal would be more concentrated towards the edges (see Figure 6).

On that note, further research should concentrate on trying larger priors and changing the datasets with more varied and complex features to test the Logit-normal distribution's viability in that area as well. An increase in dimensionality would also be a recommendation as all dimensions for the largest-dimensional experiment are used, meaning that it is still possible that the Logit-normal might make better use of additional dimensions than the Gaussian.

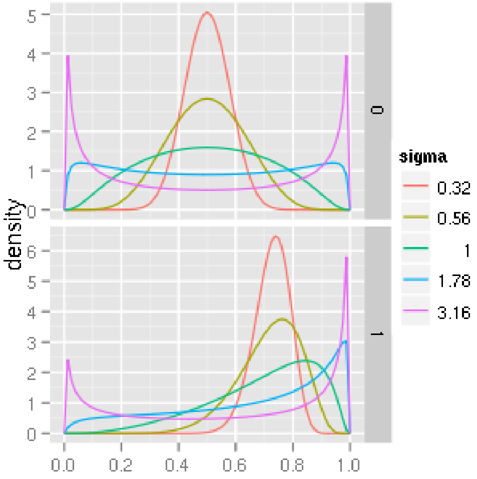

Figure 6. Logit-normal distribution with different priors (represented by the colored lines) for a mean $\mu = 0$ (top) and $\mu = 1$ (bottom)

References

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Kingma, D. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kingma., D. P. and Ba, J. L. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

LeCun, Y., Cortes, C., and Burges, C. Mnist handwritten digit database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2, 2010.

Maaten, L. v. d. and Hinton, G. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov): 2579–2605, 2008.

Maddison, C., Mnih, A., and Teh, Y. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

Salakhutdinov, R. and Murray, I. On the quantitative analysis of deep belief networks. In *Proceedings of the 25th international conference on Machine learning*, pp. 872–879. ACM, 2008.