Is Simple Better?: Revisiting Simple Generative Models for Unsupervised Clustering

Jhosimar Arias Figueroa and Adín Ramírez Rivera
Institute of Computing, University of Campinas, Campinas, SP, Brazil
jhosimar.figueroa@students.ic.unicamp.br, adin@ic.unicamp.br

Abstract

In this paper, we proposed a model to learn both clusters and representations of our data in an end-to-end manner. Our model is a modification of the stacked generative model M1+M2 applied to semi-supervised learning, in which, we model our clusters with the Gumbel-Softmax distribution and we consider the use of a deterministic autoencoder to learn latent features, avoiding the problem of hierarchical stochastic variables. Experimental results on three datasets show the effectiveness of our model achieving competitive results with the state-of-the-art. Moreover, we show that our model generates realistic samples.

1 Introduction

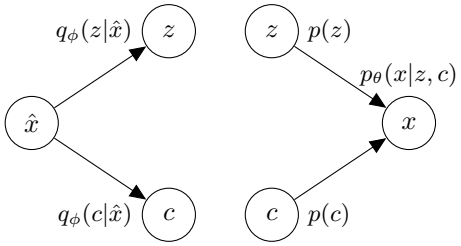
Unsupervised clustering is an open problem in machine learning. Traditional clustering algorithms like k-means and gaussian mixture models (GMM) [1] are commonly used to group data based on hand-crafted features, however they are commonly affected by high-dimensional data. In the last few years, with the help of Deep Neural Networks (DNNs) we learn feature representations in low dimensional space. Recent works take advantage of these representations for clustering, for instance, Xie et al. [2] proposed Deep Embedding Clustering (DEC) where they connect a clustering module to the output layer of a pre-trained DNN and learn jointly both features and clusters, in a similar way, recent works [3, 4] improve DEC by incorporating a reconstruction term. Besides of normal autoencoder-based architectures, other methods take advantage of deep generative models. Jian et al. [5] and Dilokthanakul et al. [6] combined VAE [7] and GMM for clustering. However, the former relies on pre-training and both of them require a complex inference process if the number of clusters increases. Similarly Nalisnick et al. [8] combined VAE and GMM but they use the mixture of Gaussian distribution as the approximate posterior of VAE, improving the capacity of the original VAE. Unlike their work, we use simpler models where instead of modeling the clusters with a GMM, we use an approximation of the categorical distribution.

Motivated by the success of deep generative models in semi-supervised learning [7, 9–11], we propose an unsupervised model for clustering that takes advantage of the generative process to learn powerful latent representations. Experimental results show that a simple model can lead to good results compared with the state-of-the-art without the need of complex models and layer-wise pre-training. Finally we show that our model can generate realistic samples for any specified cluster.

2 Deep Generative Models for Clustering

Our probabilistic model is based on the stacked generative model (M1+M2) proposed by Kingma et al. [7] for semi-supervised learning. Their model uses two stochastic layers that they were not able to train end-to-end, and thus relied on pre-training. In our model we replace the stochastic layer that produces x with a deterministic one, thus avoiding the problem of hierarchical stochastic variables.

Second workshop on Bayesian Deep Learning (NIPS 2017), Long Beach, CA, USA.



(a) Inference Model Q (b) Generative Model P

Figure 1: Probabilistic graphical model of our approach for unsupervised learning. The node $\hat{x}$ in the inference model represent a feature representation obtained from an encoder.

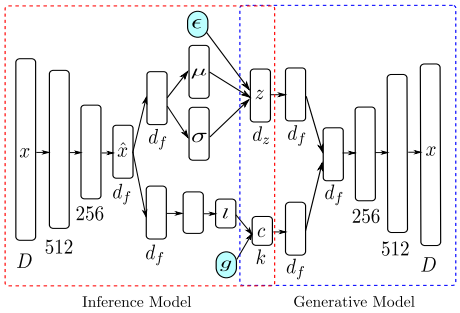


Figure 2: Our proposed architecture. Colored nodes represent stochastic nodes. The variables d_f, d_z, k are the dimensionality of the features, latent Gaussian, and number of clusters, respectively.

We learn these representations with the use of a nonlinear encoder, i.e.,

$$\hat{x} = g(x). \tag{1}$$

We consider this layer as our feature representations of the data.

2.1 Generative Unsupervised model

We use two latent variables to model the data, the continuous variable z and the categorical variable c, i.e., we have $p(x, z, c)$. The generative model P is defined as $p(c)p(z)p_\theta(x|z, c)$, cf. Fig. 1(b),

$$p(c) = Cat(c|\pi), \tag{2}$$
$$p(z) = \mathcal{N}(z|0, I), \tag{3}$$
$$p_\theta(x|z, c) = f(x; z, c, \theta), \tag{4}$$

where $Cat(c|\pi)$ is a multinomial distribution, $f(x; z, c, \theta)$ is a likelihood function represented by a non-linear combination of the latent variables c and z, we use deep neural networks to represent this non-linearity.

2.2 Variational Lower Bound

Computing the posterior $p_\theta(z, c|x)$ is intractable, thus we approximate this posterior with a tractable one, $q_\phi(z, c|\hat{x})$, by using variational inference [12] and a deterministic representation (1) of the data. Therefore, our inference model Q is defined as $q_\phi(c|\hat{x})q_\phi(z|\hat{x})$, cf. Fig. 1(a),

$$q_\phi(c|\hat{x}) = Cat(c|\pi_\phi(\hat{x})), \tag{5}$$
$$q_\phi(z|\hat{x}) = \mathcal{N}\left(z|\mu_\phi\left(\hat{x}\right), \text{diag}\left(\sigma^2_\phi\left(\hat{x}\right)\right)\right), \tag{6}$$

where $\pi_\phi(\hat{x})$ is a probability vector, $\sigma_\phi(\hat{x})$ is vector of standard deviations and $\mu_\phi(\hat{x})$ is a vector of means, all the functions that define these vectors are represented as neural networks.

Our categorical variable c is a discrete node that represents a categorical distribution that we use to associate each cluster with each category. Nevertheless, we cannot backpropagate through discrete nodes. Thus, we need to reformulate it into a deterministic path within the model. We use the Gumbel-Softmax distribution [13, 14] to approximate our categorical distribution with a continuous one.

As we are dealing with an unsupervised method, we need to infer both latent variables, i.e., c and z. Thus, we optimize the model by maximizing the lower bound:

$$\log p_\phi(x) \geq \mathbb{E}_{c,z\sim q_\phi(c,z|\hat{x})}\left[\log p_\theta(x|c, z)\right] - KL\left(q_\phi\left(c|\hat{x}\right)||p\left(c\right)\right) - KL\left(q_\phi\left(z|\hat{x}\right)||p\left(z\right)\right), \tag{7}$$

where the first term can be approximated with the reconstruction loss, in our experiments we represent this loss with binary cross entropy (binary data) and mean square error (continuous data). We specify the prior of the categories as a standard uniform distribution $p(c) \sim U(0, 1)$.

Table 1: Clustering performance, ACC (%) and NMI (%), on all datasets. The standard deviation (%) is shown in parenthesis when available.

Method	MNIST		USPS		REUTERS-10K	
	ACC	NMI	ACC	NMI	ACC	NMI
k-means [3]	53.24	-	66.82	-	51.62	-
GMM [5]	53.73	-	-	-	54.72	-
AE+k-means [3]	81.82	74.73	69.31	66.20	70.52	39.79
AE+GMM [5]	82.18	-	-	-	70.13	-
GMVAE [6]	82.31 ($\pm$ 4)	-	-	-	-	-
DCN [4]	83.00	81.00	-	-	-	-
DEC [2]	86.55	83.72	74.08	75.29	73.68	49.76
IDEC [3]	88.06	86.72	76.05	78.46	75.64	49.81
VaDE [5]	94.46	-	-	-	79.83	-
Proposed	85.75 ($\pm$ 8)	82.13 ($\pm$ 5)	72.58 ($\pm$ 3)	65.48 ($\pm$ 1)	76.74 ($\pm$ 6)	52.42 ($\pm$ 5)

3 Experiments

We evaluate our proposed method on two image datasets and one text dataset: MNIST [15] consists of 70000 images of size 28 $\times$ 28, and 10 classes, USPS consists of 9298 images of size 16 $\times$ 16 and 10 classes, REUTERS-10K [16] is a subset of 10000 examples taken from the original REUTERS dataset, features and number of categories were specified following DEC [2]. For our problem, we consider the number of classes as the number of clusters. We use the pre-defined train/test splits of each data set, from the training set we consider 80% for training and 20% for validation.

Fig. 2 shows the architecture used in our model, we consider fully connected multilayer perceptron (MLP) to represent our networks. All the layers, except the input, output and embeddings, are followed by batch normalization [17] and rectified linear unit (ReLU) as non-linearity. The architecture of the encoder used for learning our representations $\hat{x}$ is $D - 512 - 256 - d_f$, where D is the input dimensionality and d_f is the feature dimensionality. Our inference and generative models are parameterized by three neural networks: latent inference model $q_\phi(z|\hat{x})$, categorical model $q_\phi(c|\hat{x})$ and generative model $p_\theta(x|z,c)$. For the latent inference model we approximate the stochastic variables by using two linear layers for μ and σ respectively, we then apply reparameterization trick using these outputs. The architecture for the categorical model is $d_f - d_f - d_f/2 - k$ where k is the number of clusters, this last layer represents the logits l used in Gumbel-Softmax [13, 14] to approximate a categorical distribution. Finally the generative model is given by the combination of the latent variables z and c, we use linear layers to obtain vectors of dimensionality d_f, then we add these two vectors and pass them through a decoder network $d_f - 256 - 512 - D$.

For training we used Adam [18] optimizer with a learning rate of 0.001. We iterate for 300 epochs, and in every epoch we consider a different random permutation of our data. We use a batch size of 300, feature size (d_f) of 100 and latent size (d_z) of 150 for MNIST and 50 for USPS and REUTERS-10K. We smooth the temperature used in Gumbel-Softmax from 1 to 0.5 in the first 100 epochs. While training we found that our model did not converge without assigning a weight to the gaussian regularizer, w_G, we believe this problem happens because the loss functions are not in the same range. We consider this weight w_G as a hyperparameter in the model. In Fig. 3 we show the accuracy of our best run on MNIST given different values of w_G, as we can see, without specifying this weight the model does not converge, we obtained the best results by using a weight of 5.

3.1 Quantitative Evaluation

We evaluated our clustering results with two metrics commonly used in the literature, clustering accuracy (ACC) and normalized mutual information (NMI) [19]. As we can see in Table 1, our model achieves comparable results with the state-of-the-art, something important to consider is that most of the related works rely on layer-wise pretraining and are based on complex models where clustering algorithms [2, 4, 3] or complex priors [6, 5] are required.

Existing unsupervised methods [8, 12] that learn data representations are an excellent benchmark for our model. However, since their objective is not clustering, we will use the k-Nearest Neighbors classifier (kNN) on the features learned on MNIST to evaluate them. Specifically, we compared

Table 2: MNIST test error-rate (%) for kNN on latent space.

Method	k		
	3	5	10
VAE [12]	18.43	15.69	14.19
DLGMM [8]	9.14	8.38	8.42
VaDE [5]	2.20	2.14	2.22
Proposed	3.46	3.30	3.44

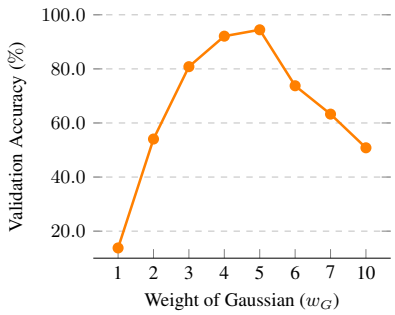


Figure 3: Accuracy (%) of the model while varying the Gaussian-weight-regularizer hyperparameter (w_G) on the MNIST dataset.

our model against DLGMM [8] and VAE [12]. In Table 2, we can see that our model outperforms DLGMM and VAE significantly, and shows competitive results against VaDE [5]. Note that none of these models required labels during training.

3.2 Qualitative Evaluation

Besides clustering, we also evaluated the generative part of our model. In Fig. 4(a) we show how our model has learned to separate style and category information. Additionally, we demonstrate the impact of choosing different number of clusters K for MNIST. To this end, we chose different number of clusters and generate images from these clusters. In Fig. 4(b) we can see that, if K is smaller than the number of true clusters, similar digits are clustered together, such as 4 with 9, and 3 with 8. On the other hand, if K is larger than the number of true clusters, some clusters are repeated but we can consider them as sub-clusters where each sub-cluster contains similar style information, like digit 6 where one cluster generates fatter 6's, while the other thinner 6's, cf. Fig. 4(d). And if the true number of cluster is known, our model yields an accurate representation, cf. Fig. 4(c).

Fig. 5 shows the feature representations of MNIST at different epochs using t-SNE [20], we reduced the dimensionality of the feature representations to 2D and plot the MNIST test set. Different colors indicate ground-truth classes. We can see that after a number of epochs, we can learn more discriminative representations improving the clustering process.

4 Conclusions

In this paper, we proposed a simple model for unsupervised clustering where we avoid the problem of hierarchical stochastic layers by using deterministic layers to learn representations in an end-to-end manner. Experimental results show that our approach generates clusters that accurately represent their given class, and that produces realistic samples when conditioned on cluster information. Our future work focuses on the use of clustering algorithms over the feature representations to improve the learning process.

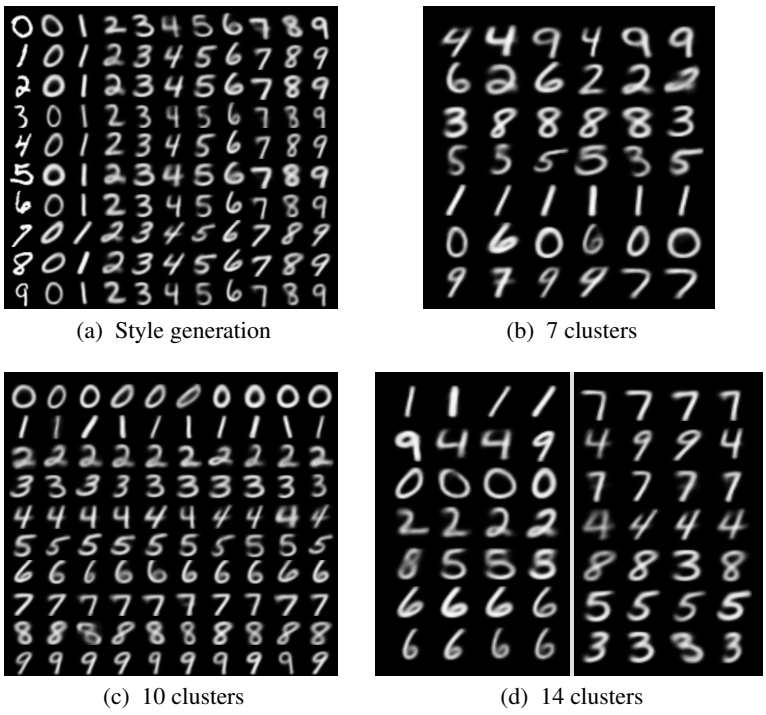
Acknowledgments

This work was supported in part by CNPq-Brazil through Grant 132848/2015-5, and in part by the São Paulo Research Foundation (FAPESP) under Grant 2016/19947-6. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan X Pascal GPU used for this research.

References

[1] C. M. Bishop. Pattern recognition and machine learning. 2006.

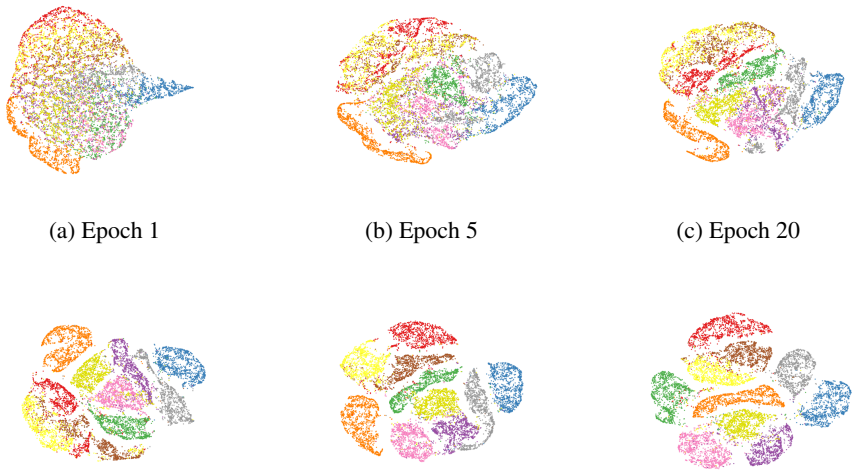
[2] J Xie, R. Girshick, and A. Farhadi. Unsupervised deep embedding for clustering analysis. In *International Conference on Machine Learning (ICML)*, 2016.



(a) Style generation (b) 7 clusters

(c) 10 clusters (d) 14 clusters

Figure 4: Generated images of our proposed model trained on MNIST. (a) The latent space has learned styles and can transfer it, we show this behavior by inputting a test image x (first column) through $q_\phi(z|\hat{x})$ and generate samples $p_\theta(x|z,c)$ for each category c. The model also learned different digits grouping by varying the number of clusters, we show the results of (b) seven, (c) ten, and (d) fourteen clusters (each row shows the output of one cluster). Notice that diverging from the true cluster number forces the model to group similar representations together.



(a) Epoch 1 (b) Epoch 5 (c) Epoch 20

(d) Epoch 50 (e) Epoch 150 (f) Epoch 300

Figure 5: Visualization of the feature representations on MNIST test set at different epochs.

[3] X. Liu J. Yin X. Guo, L. Gao. Improved deep embedded clustering with local structure preservation. In *International Joint Conference on Artificial Intelligence*, 2017.

[4] B. Yang, X. Fu, N. D. Sidiropoulos, and M. Hong. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *International Conference on Machine Learning (ICML)*, 2017.

[5] Z. Jiang, Y. Zheng, H. Tan, B. Tang, and H. Zhou. Variational deep embedding: A generative approach to clustering. In *International Joint Conference on Artificial Intelligence*, 2017.

[6] N. Dilokthanakul, P. A.M. Mediano, M. Garnelo, M. C.H. Lee, H. Salimbeni, K. Arulkumaran, and M. Shanahan. Deep unsupervised clustering with gaussian mixture variational cvprs. In *arXiv preprint arXiv:1611.02648*, 2016.

[7] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[8] E. Nalisnick, L. Hertel, , and P. Smyth. Approximate inference for deep latent gaussian mixtures. In *NIPS Workshop on Bayesian Deep Learning*, 2016.

[9] L. Maaløe, C. K. Sønderby, S. K. Sønderby, and O. Winther. Auxiliary deep generative models. In *International Conference on Machine Learning (ICML)*, 2016.

[10] L. Maaløe, M. Fraccaro, and O. Winther. Semi-supervised generation with cluster-aware generative models. In *arXiv preprint arXiv:1704.00637*, 2017.

[11] J. Arias and A. Ramírez. Learning to cluster with auxiliary tasks: A semi-supervised approach. In *Conference on Graphics, Patterns and Images (SIBGRAPI)*, 2017.

[12] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

[13] E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations (ICLR)*, 2016.

[14] C. J. Maddison, A. Mnih, and Y. W. Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations (ICLR)*, 2016.

[15] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

[16] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. Rcv1: A new benchmark collection for text categorization research. In *The Journal of Machine Learning Research*, 2004.

[17] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Learning Representations (ICLR)*, 2015.

[18] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[19] W. Xu, X. Liu, and Y. Gong. Document clustering based on non-negative matrix factorization. In *International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, pages 267–273, 2003.

[20] L. v. d. Maaten and G. Hinton. Visualizing high-dimensional data using t-sne. In *The Journal of Machine Learning Research*, pages 2579–2605, 2008.