Variational Mixture of Normalizing Flows

Guilherme Grijó Pen Freitas Pires[1] and Mário A. T. Figueiredo[1,2]

1- Instituto Superior Técnico - Dept. of Electrical and Computer Eng.
Av. Rovisco Pais 1, 1049-001 - Portugal

2- Instituto de Telecomunicações
Av. Rovisco Pais 1, 1049-001 - Portugal

Abstract. In the past few years, deep generative models, such as generative adversarial networks [1], variational autoencoders [2], and their variants, have seen wide adoption for the task of modelling complex data distributions. In spite of the outstanding sample quality achieved by those methods, they model the target distributions *implicitly*, in the sense that the probability density functions approximated by them are not explicitly accessible. This fact renders those methods unfit for tasks that require, for example, scoring new instances of data with the learned distributions. Normalizing flows overcome this limitation by leveraging the change-of-variables formula for probability density functions, and by using transformations designed to have tractable and cheaply computable Jacobians. Although flexible, this framework lacked (until the publication of recent work - [3, 4]) a way to introduce discrete structure (such as the one found in mixtures) in the models it allows to construct, in an unsupervised scenario. The present work overcomes this by using normalizing flows as components in a mixture model, and devising a training procedure for such a model. This procedure is based on variational inference, and uses a variational posterior parameterized by a neural network. As will become clear, this model naturally lends itself to (multimodal) density estimation, semi-supervised learning, and clustering. The proposed model is evaluated on two synthetic datasets, as well as on a real-world dataset.

1 Introduction

Generative models based on neural networks - variational autoencoders (VAEs), generative adversarial networks (GANs), normalizing flows and their variations - have experienced increased interest and progress in their capabilities. Both VAEs and GANs learn *implicit* distributions of the data, in the sense that - if training is successful - one can sample from the learned model, but the likelihood function of the learned distribution is not accessible. Normalizing flows differ from VAEs and GANs in that they allow learning *explicit* distributions of the data[1]. Thus, normalizing flows lend themselves to the task of density estimation.

The goal of this work is to employ normalizing flows in a finite mixture model, so as to better model multimodal datasets. In practice, a neural network classifier is learned jointly with the mixture components. Doing so will naturally produce an approach which lends itself not only to density estimation, but also

[1] In fact, recent work [5] combines the training framework of GANs with the use of normalizing flows, so as to obtain a generator for which it is possible to compute likelihoods.

to clustering - since the classifier can be used to assign points to clusters - and semi-supervised learning, where available labels can be used to refine the classifier and selectively train the mixture components.

Dinh et al. [4], similarly to this work, try to reconcile normalizing flows with a multimodal/discrete structure, by partitioning the latent space into disjoint subsets, using a mixture model where each component has non-zero weight exclusively within its respective subset. Then, using a set identification function and a piecewise invertible function, a variation of the change-of-variable formula is devised. Izmailov et al. [3] also exploit multimodal structure while using normalizing flows for expressiveness. However, while the present work relies on a variational posterior parameterized by a neural network and learns K flows (one for each mixture component), Izmailov et al. [3] resort to a latent mixture of Gaussians as the base distribution for its flow model, and learn a single flow.

2 Normalizing Flows

Given a random variable $\boldsymbol{z} \in \mathbb{R}^D$, with probability density function f_Z, and a bijective and continuous function $g(.) : \mathbb{R}^D \to \mathbb{R}^D$, the probability density function f_X of the random variable $\boldsymbol{x} = g(\boldsymbol{z})$ is given by

$$f_X(\boldsymbol{x}) = f_Z(g^{-1}(\boldsymbol{x}))\left|\det\left(\frac{d}{d\boldsymbol{x}}g^{-1}(\boldsymbol{x})\right)\right|,$$

where $\det\left(\frac{d}{d\boldsymbol{x}}g^{-1}(\boldsymbol{x})\right)$ is the determinant of the Jacobian matrix of $g^{-1}(.)$, computed at $\boldsymbol{x}$. If g is parameterized by some $\boldsymbol{\theta}$, this expression can be optimized, so that it approximates some arbitrary distribution. For that to be feasible, the *base density*, f_Z, has to be computationally "cheap" to evaluate, as well as $\det\left(\frac{d}{d\boldsymbol{x}}g^{-1}(\boldsymbol{x};\boldsymbol{\theta})\right)$, and its gradient w.r.t $\boldsymbol{\theta}$.

A *normalizing flow* - a model proposed in [6], and which has since developed into the basis of multiple state-of-the-art techniques for density estimation [7, 8, 9, 10] - is obtained by applying the change of variables formula to a function g which is the composition of L (parametric) transformations h_ℓ, for $\ell = 0, 1, ..., L-1$, which fulfill the mentioned computational requirements. Let $\boldsymbol{z_0}$ be sampled from f_Z. Applying the change of variables formula to $\boldsymbol{x} = g(\boldsymbol{z_0})$, and taking the logarithm, yields:

$$\log f_X(\boldsymbol{x}) = \log f_Z(g^{-1}(\boldsymbol{x})) - \sum_{\ell=0}^{L-1} \log\left|\det\left(\frac{d}{d\boldsymbol{x_\ell}}h_\ell(\boldsymbol{x_\ell})\right)\right|.$$

The design of transformations that are sufficiently expressive, and whose Jacobians are not computationally heavy, is the main challenge of the framework of normalizing flows. Dinh et al. [8] and Papamakarios et al. [10] present examples of such transformations, which are used in the experiments in this work.

3 Variational Mixture of Normalizing Flows

Consider a mixture of K normalizing flows. Let $p(\boldsymbol{x}|c)$ be the likelihood of data point $\boldsymbol{x}$ under the flow indexed by c, where $c \in \{1, 2, ..., K\}$. Let $q(c|\boldsymbol{x})$ be a neural network with a K-class softmax output. Using $q(c|\boldsymbol{x})$ as a variational posterior over c, the corresponding evidence lower bound (ELBO) is given by:

$$\begin{aligned} \text{ELBO} &= \mathbb{E}_q[\log p(\boldsymbol{x}|c)] + \mathbb{E}_q[\log p(c)] - \mathbb{E}_q[\log q(c|\boldsymbol{x})] \\ &= \sum_{c=1}^{K} q(c|\boldsymbol{x}) \big(\log p(\boldsymbol{x}|c) + \log p(c) - \log q(c|\boldsymbol{x}) \big). \end{aligned} \tag{1}$$

All the terms in (1) are trivial: $q(c|\boldsymbol{x})$ is the forward-pass of a neural network; $\log p(\boldsymbol{x}|c)$ is the log-likelihood of $\boldsymbol{x}$ under the normalizing flow indexed by c; $\log p(c)$ is the log-prior of the component weights, set by the modeller[2]; $-\log q(c|\boldsymbol{x})$ is the negative logarithm of the output of the encoder. We denote this model by *variational mixture of normalizing flows* (VMoNF). In a similar fashion to the variational auto-encoder, proposed by Kingma et al. [2], a VMoNF is fitted by jointly optimizing the parameters of the variational posterior $q(c|\boldsymbol{x})$ and the parameters of the generative process $p(\boldsymbol{x}|c)$, so as to maximize (1). This is done by leveraging modern automatic differentiation frameworks, e.g. [11]. After training, the variational posterior naturally induces a clustering on the data, and can be directly used to assign new data points to the discovered clusters. Moreover, each of the fitted components can be used to generate samples from the cluster it "specialized" in.

4 Experiments

In this section, the proposed model is applied to two benchmark synthetic datasets (Pinwheel and Two-circles) and one real-world dataset (MNIST). On one of the synthetic datasets, one shortcoming of the model is brought to attention, but is overcome in a semi-supervised setting. On the real-world dataset, the model's clustering capabilities are evaluted, as well as its capacity to model complex distributions. A technique inspired in [12] was employed to improve training speed and quality of results. This consists in dividing the inputs of the softmax layer in the variational posterior by a "temperature" value, T, which follows an exponential decay schedule during training. This makes the variational posterior "more certain" as training proceeds, while allowing all components to be exposed to the whole data, during the initial epochs. Moreover, it discourages components from being "subtrained" during the initial epochs and, subsequently, from being prematurely discarded.

[2]When there is no *a priori* knowledge about the true component weights, the best assumption is that they correspond to a uniform distribution

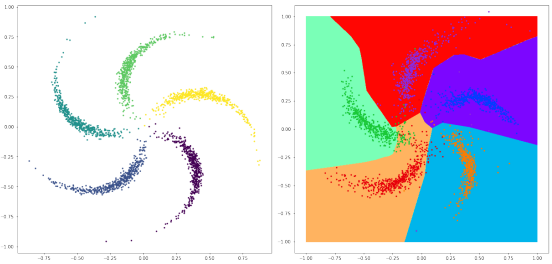

Figure 1: (a) Original dataset (b) Samples from the learned model. Each dot is colored according to the component it was sampled from. The background colors denote the regions where each component has maximum probability assigned by the variational posterior. (Note that the background colors were chosen so as to not match the dot colors, otherwise it dots wouldn't be visible)

4.1 Toy datasets

4.1.1 Pinwheel dataset

This dataset is constituted by five non-linear "wings". See Figure 1 for the results of running the model on this dataset. As expected, the variational posterior has learned to partition the space so as to attribute each "wing" to a component of the mixture.

4.1.2 Two-circles dataset

This dataset consists of two concentric circles. The experiment on this dataset, makes evident one shortcoming of the proposed model: the way in which the variational posterior partitions the space is not necessarily guided by the intrisic structure in the data (see Figure 2-b). In the case of the two-circles dataset, it was found that the most common partitioning arrived at consisted simply of a split into two half-spaces. However, in a semi-supervised setting, this behaviour can be corrected and the model successfully learns to separate the two circles, as shown in Figure 2-d). In this setting, the model was pretrained on the labeled instances and then trained with the normal procedure. There were 1024 labeled instances, and 64 unlabeled instances. In this case, the model has the chance to selectively refine both the variational posterior and each of the components. As is clearly visible in Figure 2, the model struggles with learning full, closed, circles; this is because it is unable to "pierce a hole" in the base distribution, due to the nature of the transformations that are applicable. Thus, to model a circle, the model has to learn to stretch the blob formed by the base distribution, and "bend it over itself". This difficulty is also what keeps the model from learning a structurally interesting solution in the fully unsupervised case: it is easier for each component to learn to distort space so as to model half of the two circles. Moreover, the points in diametrically opposed regions of the same circle are more dissimilar (in the geometrical sense) than points in the same region of the two circles. Therefore, when completely uninformed by labels, the variational posterior's layers will tend to have similar activations for points in the latter

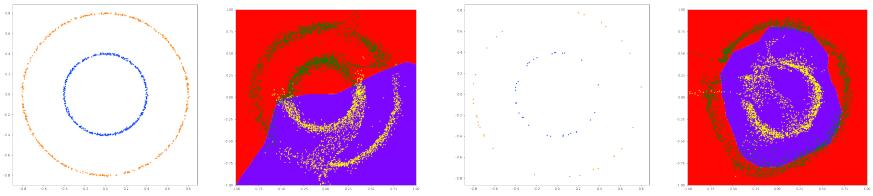

Figure 2: a) Original dataset. b) Samples from the learned model, without any labels. Coloring logic is the same as in Figure 1. c) Labeled points used in semi-supervised scenario. d) Samples from the model trained in the semi-supervised scenario.

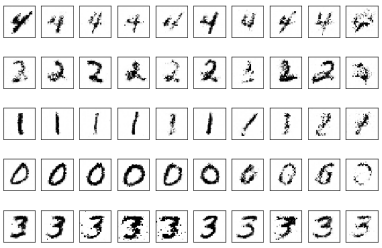

Figure 3: Samples from the fitted mixture components. Each row is sampled from the same component

	0	1	2	3	4
0	0.00	0.01	0.00	0.98	0.00
1	0.00	0.02	0.98	0.00	0.00
2	0.00	0.95	0.01	0.01	0.03
3	0.00	0.48	0.30	0.00	0.21
4	0.65	0.35	0.01	0.00	0.00

Table 1: Normalized contingency table for the clustering induced by the model. Rounded to two decimal places. Rows: true label. Columns: cluster index.

case, and thus tend to have similar outputs for them.

4.2 Real-world dataset

In this subsection, the proposed model is evaluated on the well-known MNIST dataset [13]. For this experiment, only the images corresponding to the digits from 0 to 4 were considered. In Figure 3, samples from the components obtained after training can be seen. Moreover, a normalized contingency table is presented, where the performance of the variational posterior as a clustering function can be assessed. Note that the cluster indices induced by the model have no semantic meaning. From Table 1 and Figure 3 it is possible to see that although there is some confusion, the model successfully clusters the MNIST digits.

5 Discussion

The main shortcoming of the proposed model is that the variational posterior does not always partition the space in the intuitively correct manner. Potentially, this could be improved by using a consistency loss regularization term. In fact, this idea has been pursued in [3]. During the experimentation phase, it was found that a balance between the complexity of the variational posterior and that of the components of the mixture, is crucial for the convergence to interesting

solutions. This is intuitive: if the components are too complex, the variational posterior tends to ignore most of them and assigns most points to a single or few components. In principle, the model should be able to ignore unneeded components, if the complexities of the posterior and the flows are well calibrated. This needs to be evaluated empirically. The effect of using different architectures for the neural networks used was also not evaluated.

References

[1] I. Goodfellow et al. Generative Adversarial Nets. In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani et al. Curran Associates, Inc., 2014, 2672–2680.

[2] D. P. Kingma et al. Auto-encoding variational Bayes. In: *International Conference on Learning Representations (ICLR)*. 2014.

[3] P. Izmailov et al. Semi-Supervised Learning with Normalizing Flows. In: *International Conference on Machine Learning. Workshop on Invertible Neural Networks and Normalizing Flows*. 2019.

[4] L. Dinh et al. *A RAD approach to deep mixture models*. 2019. eprint: `arXiv:1903.07714`.

[5] A. Grover et al. Flow-GAN: Combining Maximum Likelihood and Adversarial Learning in Generative Models. In: *AAAI Conference on Artificial Intelligence*. 2018.

[6] D. Rezende et al. Variational Inference with Normalizing Flows. In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by F. Bach et al. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, July 2015, 1530–1538.

[7] D. P. Kingma et al. Glow: Generative Flow with Invertible 1x1 Convolutions. In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio et al. Curran Associates, Inc., 2018, 10215–10224.

[8] L. Dinh et al. Density estimation using Real NVP. In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. 2017.

[9] N. De Cao et al. Block Neural Autoregressive Flow. *35th Conference on Uncertainty in Artificial Intelligence (UAI19)*, 2019.

[10] G. Papamakarios et al. Masked Autoregressive Flow for Density Estimation. In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., 2017, 2338–2347.

[11] A. Paszke et al. Automatic Differentiation in PyTorch. In: *NIPS Autodiff Workshop*. 2017.

[12] D. Zhang et al. *Deep Unsupervised Clustering Using Mixture of Autoencoders*. 2017. eprint: `arXiv:1712.07788`.

[13] Y. LeCun et al. MNIST handwritten digit database, 2010.