Ordering Dimensions with Nested Dropout Normalizing Flows

Citation for published version:

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
Second workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models (ICML 2020)

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.
Ordering Dimensions with Nested Dropout Normalizing Flows

Artur Bekasov 1  Iain Murray 1

Abstract

The latent space of normalizing flows must be of the same dimensionality as their output space. This constraint presents a problem if we want to learn low-dimensional, semantically meaningful representations. Recent work has provided compact representations by fitting flows constrained to manifolds, but hasn’t defined a density off that manifold. In this work we consider flows with full support in data space, but with ordered latent variables. Like in PCA, the leading latent dimensions define a sequence of manifolds that lie close to the data. We note a trade-off between the flow likelihood and the quality of the ordering, depending on the parameterization of the flow.

1. Introduction

Normalizing flows provide a way to parameterize complex distributions in high-dimensional spaces via a sequence of invertible transformations of a simple base distribution. Unlike most other popular flexible generative models (e.g. Kingma & Welling, 2014; Goodfellow et al., 2014; Du & Mordatch, 2019), flows have a tractable likelihood, which simplifies training and model comparison. It is also easier to use tractable representations of distributions in other models and algorithms, such as for variational autoencoder (VAE) priors and posteriors.

However, generative models are not just used for estimating probabilities; we also fit these models to learn representations for downstream tasks (Bengio et al., 2013). The latent spaces learned by normalizing flows can be semantically meaningful (Dinh et al., 2017; Kingma & Dhariwal, 2018) and can be useful for downstream applications, such as classification (Nalisnick et al., 2019).

For standard flows the latent dimensionality must match the input dimensionality, in order to satisfy the invertibility constraint. This constraint presents a limitation for representation learning, where we are often interested in learning lower-dimensional representations (i.e. manifolds) that capture high-level semantic concepts (Bengio et al., 2013).

When data is constrained to a known manifold, it is possible to build a valid tractable flow on that space (Gemici et al., 2016; Rezende et al., 2020). For data without strict constraints, Kumar et al. (2020) and Brehmer & Cranmer (2020) fit an invertible flow defined on a low-dimensional manifold, and bring the manifold close to the data by minimizing square error. Neither of these methods provide a tractable probabilistic model in the data space.

In this work we maintain a full-dimensional flow, but aim to order the latent variables, such that taking the first $K$ of them will reconstruct data with small square error. For a linear flow, our approach implements Principal Components Analysis (PCA). We encode this principle with an additional loss based on nested dropout (ND, Rippel et al., 2014). We show that our models successfully learn low dimensional representations, while obtaining similar likelihoods to standard flows. However, we find that the parameterization of the flow influences the representations, and we must make a trade-off between the compactness of representations and the likelihood of the flow. We hope that these findings will motivate research into flows that more naturally represent low-dimensional structure.

2. Nested dropout

Nested dropout was introduced by Rippel et al. (2014) in the context of autoencoders, with a goal of imposing an ordering on representation dimensions. For a linear autoencoder, nested dropout fits PCA. In a non-linear case, the learned ordering was shown to be useful for efficient retrieval and adaptive compression.

As in dropout (Srivastava et al., 2014): nested dropout randomly “drops” (i.e. zeros out) representation dimensions during training. However, rather than dropping features independently, we sample an index $k \sim p_k(\cdot)$, and drop dimensions $k+1 \ldots K$, i.e. all dimensions above that index, assuming $K$ is the latent dimensionality.

We define $z_{\downarrow k}$ as a representation $z$ where all the dimensions above $k$-th have been dropped, as described above. We then
We combine Eqs. (2), (4) and (5) into a single objective, we train a single flow for both.

To train the autoencoder for a given dataset \( \{x_n\}_{n=1}^N \), we minimize the following objective:

\[
\mathcal{L}(\theta) = \frac{1}{N} \sum_{n=1}^{N} \mathbb{E}_{k \sim p_k} [d(x_n, \tilde{x}_{n,k})],
\]

where \( d \) is the chosen distance metric, typically \( L_2 \) distance. This set-up encourages the auto-encoder to put more information into dimensions that correspond to smaller \( k \) indices.

Rippel et al. used a geometric distribution for \( p_k \), traditionally parameterized as \( p_k(k) = (1-p)^{k-1}p \) for a given Bernoulli probability \( p \). We follow their choice in this work.

### 2.1. Normalizing flows with nested dropout

We start with the standard training objective of a normalizing flow. For a given parametric invertible function \( f_\theta \) and a base density \( \pi \), the density of a given datapoint \( x \) is:

\[
p_\theta(x) = \pi(f_\theta(x)) \left| \text{det} \left( \frac{\partial f_\theta}{\partial x} \right) \right|.
\]

Given a dataset \( \{x_n\}_{n=1}^N \), we fit the flow parameters \( \theta \) by minimizing the negative log likelihood objective:

\[
\mathcal{L}(\theta) = -\frac{1}{N} \sum_{n=1}^{N} \log p_\theta(x_n).
\]

We can redefine a lower-dimensional reconstruction, Eq. (1), for a normalizing flow by treating the invertible function \( f_\theta \) as the “encoder”, and its inverse \( f_\theta^{-1} \) as the “decoder”:

\[
\tilde{x}_{i,k} = f_\theta^{-1}(f_\theta(x)_{i,k}).
\]

We combine Eqs. (2), (4) and (5) into a single objective, which specifies that the flow should have high likelihood, while also giving good reconstructions from low-dimensional parts of its representation:

\[
\mathcal{L}(\theta) = \frac{1}{N} \sum_{n=1}^{N} \left( -\log p_\theta(x_n) + \lambda \mathbb{E}_{k \sim p_k} [d(x_n, \tilde{x}_{n,k})] \right),
\]

where \( \lambda \) is a hyper-parameter that balances the two losses. In line with Rippel et al. we estimate the expectation with a single Monte-Carlo sample.

The loss is similar to the one used by Brehmer & Cranmer (2020, Eq. 21), with two distinctions. First, \( k \) is sampled from \( p_k \) for each datapoint, rather than pre-set according to the desired manifold dimensionality. Second, while Brehmer & Cranmer use two separate flows, one for each part of the objective, we train a single flow for both.

<table>
<thead>
<tr>
<th></th>
<th>LL</th>
<th>MSE(2)</th>
<th>MSE(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCA</td>
<td>$-$</td>
<td>0.003</td>
<td>0.037</td>
</tr>
<tr>
<td>QR</td>
<td>$-0.804 \pm 0.000$</td>
<td>0.240 ± 0.053</td>
<td>0.321 ± 0.025</td>
</tr>
<tr>
<td>QR*</td>
<td>$-0.804 \pm 0.000$</td>
<td>0.003 ± 0.000</td>
<td>0.037 ± 0.000</td>
</tr>
<tr>
<td>LU</td>
<td>$-0.805 \pm 0.000$</td>
<td>0.051 ± 0.042</td>
<td>0.257 ± 0.130</td>
</tr>
<tr>
<td>LU*</td>
<td>$-0.805 \pm 0.001$</td>
<td>0.008 ± 0.007</td>
<td>0.037 ± 0.001</td>
</tr>
</tbody>
</table>

Figure 1. 2D projections \( z \) of the 3D synthetic dataset, colored by the value of the first principal component of PCA. Superscript * for models trained with nested dropout.

### 3. Experiments

#### 3.1. Synthetic dataset

As a simple test case, we sample data from a centered 3-dimensional normal distribution that is scaled along the axis and then rotated. The eigenvalues of the covariance are 1, 0.1, and 0.01, so the target distribution looks like a fuzzy elliptical disc embedded into 3 dimensions. We sample \( 10^4 \) points for training, and another \( 10^4 \) points for evaluation.

We use a simple flow with a standard normal base distribution and a single invertible linear transformation. Such a flow can express the target distribution by learning to scale/rotate the base distribution as necessary. However, due to the spherical symmetry of the base distribution, the likelihood is invariant to permuting the dimensions (or any orthogonal transformation) of the base distribution.

The invertible linear transformation is parameterized by either an LU decomposition with a random, fixed permutation matrix \( P \) (Kingma & Dhariwal, 2018), or a QR decomposition (Hoogeboom et al., 2019) with the orthogonal matrix parameterized by 3 Householder transformations (Tomczak et al., 2020).
We use a RQ-NSF (C) image flow (Durkan et al., 2019) as a baseline for reconstruction results, where we project onto 1 or 2 principal components. Likelihood and reconstruction results are summarized in Table 1. We visualise some of the projections in Fig. 1.

For both parameterizations the additional loss improves reconstructions from 1 to 2 dimensions. For the QR parameterization, matching PCA’s optimal representation isn’t a restriction, and the test likelihood is maintained. The results for the LU parameterization have higher variance with nested dropout. The permutation matrix $P$ is not learned, and its initialization matters.

### 3.2. Images

To evaluate the method on high-dimensional data, we fit a normalizing flow with nested dropout to Fashion-MNIST images (Xiao et al., 2017). To simplify the model architecture, we pad the images by 2 pixels on each side, giving $32 \times 32$ or 1024-dimensional images. We use the provided test set, but split the provided training set into 50,000 training images and 10,000 validation images. We dequantize the images by adding uniform noise $\mathbf{U} \in [0, 1)^{32 \times 32}$.

We use a RQ-NSF (C) image flow (Durkan et al., 2019) containing 3 multi-scale levels with 4 transformation steps per level, where each step consists of an activation normalization layer, a 1x1 convolution, and a rational-quadratic coupling transform. Residual convolutional networks with 3 blocks and 128 hidden channels parameterize the 4-bin rational-quadratic splines in the coupling transforms. We train all models for $10^4$ iterations, with a batch size of 500. We set the reconstruction coefficient to $\lambda = 20$ for nested dropout runs, with $p_k$ set to a geometric distribution with $p = 0.33$.

We note that the reconstruction curve is drastically improved, a sudden drop in error after including $\approx 250$ dimensions from the reversed order. In our implementation of the multi-scale transform, it is this last quarter of the variables that are transformed at all 3 multi-scale levels. We use the reversed order in all further experiments, both during training with nested dropout, and during evaluation.

#### 3.2.2. Training with Nested Dropout

We now try to understand whether we can use nested dropout to improve upon the baseline RQ-NSF (C) model results in terms of reconstruction accuracy. We set the reconstruction coefficient $\lambda = 10^{-3}$, and we set $p = 10^{-3}$ for the geometric distribution $p_k$. Reconstruction and likelihood results are shown in Fig. 3.

We note that the reconstruction curve is drastically improved, in particular for $< 250$ dimension projections. However, we also note that the test log-likelihood of the model suffers, being reduced by $\approx 4\%$ as a result of applying nested dropout. We hypothesize that this trade-off is controlled by the values of $\lambda$ and $p$, which we explore further in Section 3.2.3.

We show samples in the ambient space and reconstructions for a baseline flow and a nested dropout flow in Fig. 4.
Ordering Dimensions with Nested Dropout Normalizing Flows

Figure 4. Top: samples from the flow. Bottom: reconstructions for an input image in the top left, decaying the latent dimension from 510 to 1 (left-to-right, top-to-bottom). Left: RQ-NSF (C) baseline. Right: RQ-NSF (C) with nested dropout.

Figure 5. Samples from RQ-NSF (C) trained with nested dropout. Each row shows samples with a different number of latent dimensions retained: 2, 4, 8, 16, 32, 64 for each row in order.

Even though the likelihoods of the two models differ, their samples are of comparable perceptual quality. It has been noted that the relationship between the perceptual quality of samples and likelihood is complex (Theis et al., 2016). The reconstructions demonstrate that nested dropout indeed guides the flow towards ordering the representations.

For the nested dropout flow, Fig. 5 shows samples from the flow varying the number of dropped dimensions. Using 2 dimensions gives a generic, blurry item, which becomes sharper and more detailed given more latent features.

3.2.3. Hyper-parameters

We perform a limited grid-search for the $\lambda$ and $p$ hyper-parameters. We perturb each hyper-parameter independently, starting with the baseline values used in the previous section. Results are shown in Fig. 6.

The value of $p$ has a limited effect on the reconstruction curve. A lower value (which causes more dimensions to be dropped on average during the run) marginally improves the MSE numbers for $< 200$ latent dimensions.

$\lambda = 10^{-3}, p = 1 \times 10^{-3}, \text{ bpd: 2.347}$
$\lambda = 10^{-2}, p = 1 \times 10^{-3}, \text{ bpd: 2.288}$
$\lambda = 10^{-4}, p = 1 \times 10^{-3}, \text{ bpd: 2.263}$
$\lambda = 10^{-3}, p = 1 \times 10^{-2}, \text{ bpd: 2.290}$
$\lambda = 10^{-3}, p = 5 \times 10^{-4}, \text{ bpd: 2.256}$

Figure 6. Mean squared error of Fashion-MNIST reconstructions for RQ-NSF (C) flow trained with nested dropout against the number of retained dimensions, varying hyper-parameters $\lambda$ and $p$. The dotted line for the baseline nested dropout model. bpd is the negative test log-likelihood in bits per dimension (lower is better).

The effect of changing $\lambda$ is more pronounced. Larger values cause a noticeable improvement in reconstruction results, while lower values have the opposite effect.

All 4 models improve upon the baseline in terms of test log-likelihood, which is surprising. There could be an interaction between $\lambda$ and $p$ hyper-parameters, or large variation across runs. Unfortunately, all but one result, including the one with the best reconstruction results, have slightly lower likelihoods than the standard flow.

4. Discussion

Nested dropout is a simple way to encourage a flow to represent data as closely as possible in the top, principal elements of its latent representation. Given the large redundancy in the way flows parameterize distributions, we can choose flows with similar likelihoods that provide much better low dimensional representations.

The small reduction in likelihood suggests that existing flows have some inductive bias against interpretable latent spaces. We also can’t always apply nested dropout, because the flow must be evaluated in both directions on every iteration. For architectures with one-pass sampling, training is typically slower by a factor of two. Flows without a cheap inverse, e.g. auto-regressive flows, would be impractical.

If data really lies on a low-dimensional manifold, densities in data space are not well defined, and other work is more appropriate (Kumar et al., 2020; Brehmer & Cranmer, 2020). However, PCA is often applied when data aren’t really restricted to a subspace of fixed dimensionality. We hope that nested dropout flows will be useful in similar circumstances.
Acknowledgements
This work was supported in part by the EPSRC Centre for Doctoral Training in Data Science, funded by the UK Engineering and Physical Sciences Research Council (grant EP/L016427/1) and the University of Edinburgh.

References


