flow requires T linear operations to produce a sample from a more flexible posterior with an approximate full-covariance matrix.

4.4.2.2 Variational Posteriors with Sylvester Flows [16]

Motivation

The Householder flows can model only full-covariance Gaussians that is still not necessarily a rich family of distributions. Now, we will look into a generalization of the Householder flows. For this purpose, let us consider the following transformation similar to a single layer MLP with M hidden units and a residual connection:

$$\mathbf{z}^{(t)} = \mathbf{z}^{(t-1)} + \mathbf{A}h(\mathbf{B}\mathbf{z}^{(t-1)} + \mathbf{b}), \tag{4.60}$$

with $\mathbf{A} \in \mathbb{R}^{D\times M}$, $\mathbf{B} \in \mathbb{R}^{M\times D}$, $\mathbf{b} \in \mathbb{R}^{M}$, and $M \leq D$. The Jacobian-determinant of this transformation can be obtained using *Sylvester's determinant identity*, which is a generalization of the matrix determinant lemma.

Theorem 4.3 (Sylvester's Determinant Identity) *For all* $\mathbf{A} \in \mathbb{R}^{D\times M}, \mathbf{B} \in \mathbb{R}^{M\times D}$,

$$\det\left(\mathbf{I}_D + \mathbf{A}\mathbf{B}\right) = \det\left(\mathbf{I}_M + \mathbf{B}\mathbf{A}\right), \tag{4.61}$$

where $\mathbf{I}_M$ *and* $\mathbf{I}_D$ *are* M*- and* D*-dimensional identity matrices, respectively.*

When $M < D$, the computation of the determinant of a $D \times D$ matrix is thus reduced to the computation of the determinant of an $M \times M$ matrix.

Using Sylvester's determinant identity, the Jacobian-determinant of the transformation in Eq. (4.60) is given by:

$$\det\left(\frac{\partial \mathbf{z}^{(t)}}{\partial \mathbf{z}^{(t-1)}}\right) = \det\left(\mathbf{I}_M + \operatorname{diag}\left(h'(\mathbf{B}\mathbf{z}^{(t-1)} + \mathbf{b})\right)\mathbf{B}\mathbf{A}\right). \tag{4.62}$$

Since Sylvester's determinant identity plays a crucial role in the proposed family of normalizing flows, we will refer to them as *Sylvester normalizing flows*.

Parameterization of $\mathbf{A}$ and $\mathbf{B}$

In general, the transformation in (4.60) will not be invertible. Therefore, we propose the following special case of the above transformation:

$$\mathbf{z}^{(t)} = \mathbf{z}^{(t-1)} + \mathbf{Q}\mathbf{R}h(\tilde{\mathbf{R}}\mathbf{Q}^T\mathbf{z}^{(t-1)} + \mathbf{b}), \tag{4.63}$$

where $\mathbf{R}$ and $\tilde{\mathbf{R}}$ are upper triangular $M \times M$ matrices, and

$$\mathbf{Q} = (\mathbf{q}_1 \dots \mathbf{q}_M)$$

with the columns $\mathbf{q}_m \in \mathbb{R}^D$ forming an orthonormal set of vectors. By Theorem 4.3, the determinant of the Jacobian $\mathbf{J}$ of this transformation reduces to:

$$\begin{aligned} \det(\mathbf{J}) &= \det\left(\mathbf{I}_M + \operatorname{diag}\left(h'(\tilde{\mathbf{R}}\mathbf{Q}^T\mathbf{z}^{(t-1)} + \mathbf{b})\right)\tilde{\mathbf{R}}\mathbf{Q}^T\mathbf{Q}\mathbf{R}\right) \\ &= \det\left(\mathbf{I}_M + \operatorname{diag}\left(h'(\tilde{\mathbf{R}}\mathbf{Q}^T\mathbf{z}^{(t-1)} + \mathbf{b})\right)\tilde{\mathbf{R}}\mathbf{R}\right), \end{aligned} \tag{4.64}$$

which can be computed in $O(M)$, since $\tilde{\mathbf{R}}\mathbf{R}$ is also upper triangular. The following theorem gives a sufficient condition for this transformation to be invertible.

Theorem 4.4 *Let* $\mathbf{R}$ *and* $\tilde{\mathbf{R}}$ *be upper triangular matrices. Let* $h : \mathbb{R} \longrightarrow \mathbb{R}$ *be a smooth function with bounded, positive derivative. Then, if the diagonal entries of* $\mathbf{R}$ *and* $\tilde{\mathbf{R}}$ *satisfy* $r_{ii}\tilde{r}_{ii} > -1/\|h'\|_\infty$ *and* $\tilde{\mathbf{R}}$ *is invertible, the transformation given by* (4.63) *is invertible.*

The proof of this theorem could be found in [16].

Preserving Orthogonality of $\mathbf{Q}$

Orthogonality is a convenient property, mathematically, but hard to achieve in practice. In this chapter we consider three different flows based on the theorem above and various ways to preserve the orthogonality of $\mathbf{Q}$. The first two use explicit differentiable constructions of orthogonal matrices, while the third variant assumes a specific fixed permutation matrix as the orthogonal matrix.

Orthogonal Sylvester Flows First, we consider a Sylvester flow using matrices with M orthogonal columns (O-SNF). In this flow we can choose $M < D$ and thus introduce a flexible bottleneck. Similar to [56], we ensure orthogonality of $\mathbf{Q}$ by applying the following differentiable iterative procedure proposed by Björck and Bowie [57] and Kovarik [58]:

$$\mathbf{Q}^{(k+1)} = \mathbf{Q}^{(k)}\left(\mathbf{I} + \frac{1}{2}\left(\mathbf{I} - \mathbf{Q}^{(k)\top}\mathbf{Q}^{(k)}\right)\right), \tag{4.65}$$

with a sufficient condition for convergence given by $\|\mathbf{Q}^{(0)\top}\mathbf{Q}^{(0)} - \mathbf{I}\|_2 < 1$. Here, the 2-norm of a matrix $\mathbf{X}$ refers to $\|\mathbf{X}\|_2 = \lambda_{\max}(\mathbf{X})$, with $\lambda_{\max}(\mathbf{X})$ representing the largest singular value of $\mathbf{X}$. In our experimental evaluations we ran the iterative procedure until $\|\mathbf{Q}^{(k)\top}\mathbf{Q}^{(k)} - \mathbf{I}\|_F \leq \epsilon$, with $\|\mathbf{X}\|_F$ the Frobenius norm, and ϵ a small convergence threshold. We observed that running this procedure up to 30 steps was sufficient to ensure convergence with respect to this threshold. To minimize the computational overhead introduced by orthogonalization, we perform this orthogonalization in parallel for all flows.

Since this orthogonalization procedure is differentiable, it allows for the calculation of gradients with respect to $\mathbf{Q}^{(0)}$ by backpropagation, allowing for any standard optimization scheme such as stochastic gradient descent to be used for updating the flow parameters.

Householder Sylvester Flows Second, we study Householder Sylvester flows (H-SNF) where the orthogonal matrices are constructed by products of Householder reflections. Householder transformations are reflections about hyperplanes. Let $\mathbf{v} \in \mathbb{R}^D$, then the reflection about the hyperplane orthogonal to $\mathbf{v}$ is given by Eq. (4.58).

It is worth noting that performing a single Householder transformation is very cheap to compute, as it only requires D parameters. Chaining together several Householder transformations results in more general orthogonal matrices, and Theorem 4.2 shows that any $M \times M$ orthogonal matrix can be written as the product of $M - 1$ Householder transformations. In our Householder Sylvester flow, the number of Householder transformations H is a hyperparameter that trades off the number of parameters and the generality of the orthogonal transformation. Note that the use of Householder transformations forces us to use $M = D$, since Householder transformations result in square matrices.

Triangular Sylvester Flows Third, we consider a triangular Sylvester flow (T-SNF), in which all orthogonal matrices $\mathbf{Q}$ alternate per transformation between the identity matrix and the permutation matrix corresponding to reversing the order of $\mathbf{z}$. This is equivalent to alternating between lower and upper triangular $\tilde{\mathbf{R}}$ and $\mathbf{R}$ for each flow.

Amortizing Flow Parameters

When using normalizing flows in an amortized inference setting, the parameters of the base distribution as well as the flow parameters can be functions of the datapoint $\mathbf{x}$ [19]. Figure 4.15 (left) shows a diagram of one SNF step and the amortization procedure. The inference network takes datapoints $\mathbf{x}$ as input and provides as an output the mean and variance of $\mathbf{z}^{(0)}$ such that $\mathbf{z}^{(0)} \sim \mathcal{N}(\mathbf{z}|\mu^0, \sigma^0)$. Several SNF transformations are then applied to $\mathbf{z}^{(0)} \rightarrow \mathbf{z}^{(1)} \rightarrow \ldots \mathbf{z}^{(T)}$, producing a flexible posterior distribution for $\mathbf{z}^{(T)}$. All of the flow parameters ($\mathbf{R}$, $\tilde{\mathbf{R}}$, and $\mathbf{Q}$ for each transformation) are produced as an output by the inference network and are thus fully amortized.

4.4.2.3 Hyperspherical Latent Space

Motivation

In the VAE framework, choosing Gaussian priors and Gaussian posteriors from the mathematical convenience leads to Euclidean latent space. However, such choice could be limiting for the following reasons:

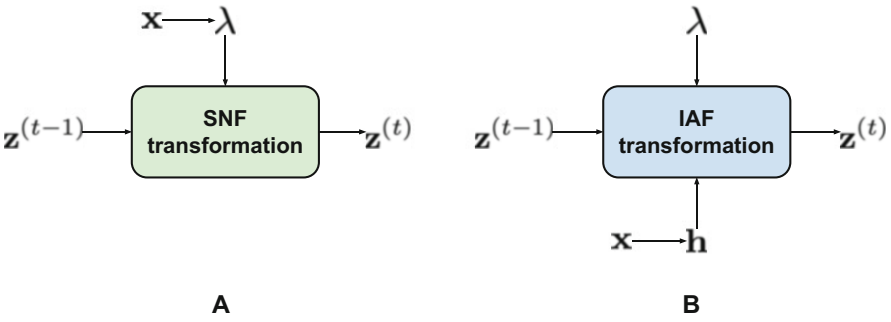

Fig. 4.15 Different amortization strategies for Sylvester normalizing flows and Inverse Autoregressive Flows. (**a**) Our inference network produces amortized flow parameters. This strategy is also employed by planar flows. (**b**) Inverse Autoregressive Flow [18] introduces a measure of **x** dependence through a context variable $\mathbf{h}(\mathbf{x})$. This context acts as an additional input for each transformation. The flow parameters themselves are independent of **x**

- In low dimensions, the standard Gaussian probability presents a concentrated probability mass around the mean, encouraging points to cluster in the center. However, this is particularly problematic when the data is divided into multiple clusters. Then, a better suited prior would be *uniform*. Such a uniform prior, however, is not well-defined on the hyperplane.
- It is a well-known phenomenon that the standard Gaussian distribution in high dimensions tends to resemble a uniform distribution on the surface of a hypersphere, with the vast majority of its mass concentrated on the hyperspherical shell (the so-called soap bubble effect). A natural question is whether it would be better to use a distribution defined on the hypersphere.

A distribution that would allow solving both problems at once is the von Mises–Fisher distribution. It was advocated in [33] to use this distribution in the context of VAEs.

von Mises–Fisher Distribution

The *von Mises–Fisher* (vMF) distribution is often described as the Normal Gaussian distribution on a hypersphere. Analogous to a Gaussian, it is parameterized by $\mu \in \mathbb{R}^m$ indicating the mean direction, and $\kappa \in \mathbb{R}_{\geq 0}$ the concentration around μ. For the special case of $\kappa = 0$, the vMF represents a Uniform distribution. The probability density function of the vMF distribution for a random unit vector $\mathbf{z} \in \mathbb{R}^m$ (or $\mathbf{z} \in \mathcal{S}^{m-1}$) is then defined as

$$q(\mathbf{z}|\boldsymbol{\mu}, \kappa) = C_m(\kappa) \exp(\kappa \boldsymbol{\mu}^T \mathbf{z}) \tag{4.66}$$

$$C_m(\kappa) = \frac{\kappa^{m/2-1}}{(2\pi)^{m/2} \mathcal{I}_{m/2-1}(\kappa)}, \tag{4.67}$$

where $||\boldsymbol{\mu}||^2 = 1$, $C_m(\kappa)$ is the normalizing constant, and $\mathcal{I}_v$ denotes the modified Bessel function of the first kind at order v.

Interestingly, since we define a distribution over a hypersphere, it is possible to formulate a uniform prior over the hypersphere. Then it turns out that if we take the vMF distribution as the variational posterior, it is possible to calculate the Kullback–Leibler divergence between the vMF distribution and the uniform defined over $\mathcal{S}^{m-1}$ analytically [33]:

$$KL[\text{vMF}(\mu, \kappa)||\text{Unif}(\mathcal{S}^{m-1})] = \kappa + \log C_m(\kappa) - \log \left(\frac{2(\pi^{m/2})}{\Gamma(m/2)} \right)^{-1}. \tag{4.68}$$

To sample from the vMF, one can follow the procedure of [59]. Importantly, the reparameterization cannot be easily formulated for the vMF distribution. Fortunately, [60] allows extending the reparameterization trick to the wide class of distributions that can be simulated using rejection sampling. [33] presents how to formulate the acceptance–rejection sampling reparameterization procedure. Being equipped with the sampling procedure and the reparameterization trick, and having an analytical form of the Kullback–Leibler divergence, we have everything to be able to build a hyperspherical VAE. However, please note the all these procedures are less trivial than the ones for Gaussians. Therefore, a curious reader is referred to [33] for further details.

4.5 Hierarchical Latent Variable Models

4.5.1 Introduction

The main goal of AI is to formulate and implement systems that can interact with an environment, process, store, and transmit information. In other words, we wish an AI system *understands* the world around it by identifying and disentangling hidden factors in the observed low-sensory data [61]. If we think about the problem of building such a system, we can formulate it as learning a probabilistic model, i.e., a joint distribution over observed data, $\mathbf{x}$, and hidden factors, $\mathbf{z}$, namely, $p(\mathbf{x}, \mathbf{z})$. Then learning a *useful representation* is equivalent to finding a posterior distribution over the hidden factors, $p(\mathbf{z}|\mathbf{x})$. However, it is rather unclear what we really mean by *useful* in this context. In a beautiful blog post [62], Ferenc Huszar outlines why learning a latent variable model by maximizing the likelihood function is not necessarily useful from the representation learning perspective. Here, we will use it