Infinite ShapeOdds: Nonparametric Bayesian Models for Shape Representations

Wei Xing,[1] **Shireen Elhabian,**[1] **Robert M. Kirby,**[1,2] **Ross T. Whitaker,**[1,2] **Shandian Zhe** [2]
[1]Scientific Computing and Imaging Institute, University of Utah
[2]School of Computing, University of Utah
{wxing, shireen, kirby, whitaker}@sci.utah.edu, zhe@cs.utah.edu

Abstract

Learning compact representations for shapes (binary images) is important for many applications. Although neural network models are very powerful, they usually involve many parameters, require substantial tuning efforts and easily overfit small datasets, which are common in shape-related applications. The state-of-the-art approach, ShapeOdds, as a latent Gaussian model, can effectively prevent overfitting and is more robust. Nonetheless, it relies on a linear projection assumption and is incapable of capturing intrinsic nonlinear shape variations, hence may leading to inferior representations and structure discovery. To address these issues, we propose Infinite ShapeOdds (InfShapeOdds), a Bayesian nonparametric shape model, which is flexible enough to capture complex shape variations and discover hidden cluster structures, while still avoiding overfitting. Specifically, we use matrix Gaussian priors, nonlinear feature mappings and the kernel trick to generalize ShapeOdds to a shape-variate Gaussian process model, which can grasp various nonlinear correlations among the pixels within and across (different) shapes. To further discover the hidden structures in data, we place a Dirichlet process mixture (DPM) prior over the representations to jointly infer the cluster number and memberships. Finally, we exploit the Kronecker-product structure in our model to develop an efficient, truncated variational expectation-maximization algorithm for model estimation. On synthetic and real-world data, we show the advantage of our method in both representation learning and latent structure discovery.

Introduction

Many applications, such as in computer vision, biology and medical imaging (Pohl et al. 2007; Liu et al. 2018), involve shapes represented by binary images. In contrast to popular RGB images, shape images lack detailed texture information, and the data often include a limited number of instances (e.g., anatomical and tumor shapes). These present more challenges for learning low-dimensional representations, which are critical for fundamental image processing tasks such as object recognition and segmentation.

Although (deep) neural network based methods, such as variational auto-encoders (VAE) (Kingma and Welling 2013) have achieved a great success in learning image representations (Higgins et al. 2017; Tomczak and Welling 2017), they usually include massive parameters, require much hand-tuning labor, and are prone to overfitting, especially on small datasets, which is often the case for shapes. To overcome this problem, Elhabian and Whitaker (2017) proposed ShapeOdds, a latent Gaussian model that uses sparse projections to prevent overfitting, and Gaussian Markov Random fields (GMRFs) to capture the local/global shape properties. Empirically, ShapeOdds is more robust against missing regions and background clutter and exhibits smaller reconstruction errors, as compared with the shape Boltzman machine (ShapeBM) (Eslami et al. 2014) (a stochastic neural network) and the classical LogOdds based methods (Pohl et al. 2007).

However, a severe limitation of ShapeOdds is that it assumes an oversimplified, linear projection from the latent representations to the shape images. Despite its robustness, ShapeOdds cannot capture the intrinsic nonlinear variations in shapes (Bengio, Courville, and Vincent 2013), and hence may obtain inferior representations that fail to reflect the underlying structures of the data.

To overcome this limitation, we propose infinite ShapeOdds (InfShapeOdds), a Bayesian nonparametric shape representation model that can flexibly capture various nonlinear variations and discover hidden cluster structures. InfShapeOdds includes only a small number of parameters, can effectively avoid overfitting and does not require fine tuning. InfShapeOdds hybridizes latent Gaussian processes (GPs) and Dirichlet processes (DPs) to jointly estimate nonlinear shape representations and discover hidden clusters so as to render the two tightly coupled tasks to benefit each other. Specifically, we first use matrix Gaussian priors, nonlinear feature transformations and the kernel trick to generalize ShapeOdds to a shape-variate GP model that can account for the complex spatial correlations of the pixels within and across shape images so as to capture various nonlinear shape variations. The covariance matrix turns out to be a Kronecker product that enables efficient inverse and log determinant computation. GPs are known to be flexible, robust to noises and resist overfitting (Rasmussen and Williams 2006). We then place a Dirichlet process mixture

(DPM) prior over the latent representations. As a nonparametric mixture prior, DPM allows joint inference of the cluster number and memberships, and hence can discover a variety of cluster structures within the data. Finally, we exploit the Kronecker product properties to efficiently compute the model likelihood and its gradient, based on which we develop a truncated variational expectation-maximization (EM) algorithm for scalable model estimation.

For evaluation, we compared InfShapeOdds with ShapeOdds, GP latent variable model (GPLVM) (Lawrence 2005), and VAEs with different architectures. On real-world benchmark datasets, InfShapeOdds outperforms all competing methods in terms of reconstruction and shape generation/interpolation. For structure discovery, the simulation shows that InfShapeOdds recovers the ground-truth structures much better than ShapeOdds. In real-world datasets, InfShapeOdds often returns more structured and interpretable low-dimensional representations.

Background

Suppose we are given a collection of N shapes, $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_N]^\top$, where each shape is represented by an $I \times J$ binary image, $y_n = [y_{n11}, \ldots, y_{n1J}, \ldots, y_{nI1}, \ldots, y_{nIJ}]^T$ where $1 \leq n \leq N$, $y_{nij} \in \{0, 1\}$ is the pixel (i, j) of the image $\mathbf{y}_n$. We aim to learn a low-dimensional representation $\mathbf{z}_n \in \mathbb{R}^r$ for each image $\mathbf{y}_n$, where $r \ll D = I \times J$. To this end, ShapeOdds uses a latent Gaussian model to generate the image. First, the representation $\mathbf{z}_n$ is sampled from a Gaussian prior, $p(\mathbf{z}_n) = \mathcal{N}(\mathbf{z}_n|\boldsymbol{\mu}, \boldsymbol{\Sigma})$. Then a continuous field $\mathbf{f}_n \in \mathrm{R}^D$ is generated from a linear projection of $\mathbf{z}_n$,

$$\mathbf{f}_n = \mathbf{W}\mathbf{z}_n + \mathbf{w}_0, \tag{1}$$

where $\mathbf{W}$ is the $D \times r$ projection (or loading) matrix, and $\mathbf{w}_0$ the bias. To encode both the local and global spatial properties, $\mathbf{w}_0$ and each column of $\mathbf{W}$ is assigned a Gaussian Markov random field (GMRF) prior,

$$p(\mathbf{w}_j) = \beta_j \mathcal{N}(\mathbf{w}_j|\mathbf{0}, \gamma\mathbf{S}), \tag{2}$$

where $\mathbf{w}_j$ is the j-th column of $\mathbf{W}$ when $j > 0$, $\{\beta_j, \gamma\} \geq 0$, and $\mathbf{S}$ is the stencil of the negative bi-Laplacian operator. To prevent overfitting, a sparse, ARD prior (Neal 2012) is further assigned on β_j to enable the pruning of these columns. Finally, given the latent field $\mathbf{f}_n$, the observed image $\mathbf{y}_n$ is sampled from $p(\mathbf{y}_n|\mathbf{f}_n) = \prod_{i=1}^{I}\prod_{j=1}^{J} p(y_{nij}|f_{nij})$, where $p(y_{nij}|f_{nij}) = 1/\big(1 + \exp(-(2y_{nij} - 1)f_{nij})\big)$.

Model

In spite of its success, ShapeOdds may suffer from the simple linear projection in (1). The linear projection can prevent ShapeOdds from grasping more subtle, nonlinear shape variations (that are ubiquitous in practice), and hence result in low-quality shape representations. To address this issue and to further improve the discovery of the hidden structures, we propose InfShapeOdds, a Bayesian nonparametric shape model, described in the subsequent section.

Shape-Variate Gaussian Process

We first generalize ShapeOdds to construct a shape-variate Gaussian process, which can flexibly model the complex spatial correlations of the pixels and hence capture all possible nonlinear shape variations. Specifically, to enable nonlinear projections in ShapeOdds, we perform a nonlinear feature mapping on the latent representation $\mathbf{z}_n$, and then substitute the mapped feature vector $\boldsymbol{\phi}(\mathbf{z}_n)$ for $\mathbf{z}_n$ in (1). To be succinct, we rewrite the projection as $\mathbf{f}_n = \mathbf{W}\boldsymbol{\phi}(\mathbf{z}_n)$, because we can always assume the bias $\mathbf{w}_0$ is folded into $\mathbf{W}$ and $\boldsymbol{\phi}(\mathbf{z}_n)$ augmented with a constant feature 1. To account for all possible correlations, we can in general place a matrix Gaussian prior distribution over the loadings $\mathbf{W}$,

$$\begin{aligned} p(\mathbf{W}) &= \mathcal{MN}(\mathbf{0}, \mathbf{K}_D, \mathbf{K}_l) = \mathcal{N}(\mathrm{vec}(\mathbf{W})|\mathbf{0}, \mathbf{K}_l \otimes \mathbf{K}_D) \\ &= \frac{\exp\left(-\frac{1}{2}\mathrm{tr}[\mathbf{K}_D^{-1}\mathbf{W}\mathbf{K}_l^{-1}\mathbf{W}^\top]\right)}{(2\pi)^{Dl/2}|\mathbf{K}_D|^{D/2}|\mathbf{K}_l|^{l/2}} \end{aligned} \tag{3}$$

where l is the dimension of the mapped feature vector, $\mathbf{K}_D \in \mathbb{R}^{D\times D}$ is the row covariance matrix and describes the spatial correlations of the pixels, and $\mathbf{K}_l \in \mathbb{R}^{l\times l}$ is the column covariance matrix and describes the correlations of the mapped features. Note that the prior distribution of $\mathbf{W}$ in ShapeOdds (see (2)) is a special case of (3), where $\mathbf{K}_D = \lambda^{-1}\mathbf{S}$ and $\mathbf{K}_l = \mathrm{diag}([\beta_1, \ldots, \beta_l])$.

Now, let us consider the collection of N fields, $\mathbf{F} = [\mathbf{f}_1, \ldots, \mathbf{f}_N]^\top$. Obviously, we have $\mathbf{F} = \boldsymbol{\Phi}\mathbf{W}$ where $\boldsymbol{\Phi} = [\boldsymbol{\phi}(\mathbf{z}_1), \ldots, \boldsymbol{\phi}(\mathbf{z}_N)]^\top$. Because $\mathbf{F}$ is a linear transformation of $\mathbf{W}$, according to (3), the distribution of $\mathbf{F}$ is matrix Gaussian as well, $p(\mathbf{F}) = \mathcal{MN}_{N\times D}(\mathbf{0}, \boldsymbol{\Phi}\mathbf{K}_l\boldsymbol{\Phi}^\top, \mathbf{K}_D)$. We can further view the row covariance matrix comes from another (nonlinear) feature transformation, $\boldsymbol{\psi}(\mathbf{z}_n) = \mathbf{K}_l^{\frac{1}{2}}\boldsymbol{\phi}(\mathbf{z}_n)$ and $\boldsymbol{\Phi}\mathbf{K}_l\boldsymbol{\Phi}^\top = \boldsymbol{\Psi}\boldsymbol{\Psi}^\top$, where $\boldsymbol{\Psi} = [\boldsymbol{\psi}(\mathbf{z}_1), \ldots, \boldsymbol{\psi}(\mathbf{z}_N)]^\top$. Hence, we have $p(\mathbf{F}|\mathbf{Z}) = \mathcal{MN}_{N\times D}(\mathbf{0}, \boldsymbol{\Psi}\boldsymbol{\Psi}^\top, \mathbf{K}_D)$. Note that each element in $\boldsymbol{\Psi}\boldsymbol{\Psi}^\top$ is an inner product of two mapped feature vectors, $[\boldsymbol{\Psi}\boldsymbol{\Psi}^\top]_{ij} = \langle\boldsymbol{\psi}(\mathbf{z}_i), \boldsymbol{\psi}(\mathbf{z}_j)\rangle$. Explicitly working with the nonlinear feature mapping is costly, especially when the mapping is high dimensional or even infinite dimensional. To reduce the cost, we apply the kernel trick and replace each inner product $\langle\boldsymbol{\psi}(\mathbf{z}_i), \boldsymbol{\psi}(\mathbf{z}_j)\rangle$ by a kernel function $\kappa_z(\mathbf{z}_i, \mathbf{z}_j)$, which is equivalent to performing the feature mapping on the inputs first and then calculating the inner product of the mapped feature vectors. A commonly used kernel function is the SE-ARD kernel, $\kappa_z(\mathbf{z}_i, \mathbf{z}_j) = \theta_0 \exp(-(\mathbf{z}_i - \mathbf{z}_j)^\top \mathrm{diag}(\frac{1}{\boldsymbol{\theta}})(\mathbf{z}_i - \mathbf{z}_j))$ where $\{\theta_0, \boldsymbol{\theta}\}$ are the kernel parameters. Now, the distribution of the fields $\mathbf{F}$ is given by

$$\begin{aligned} p(\mathbf{F}|\mathbf{Z}) &= \mathcal{MN}_{N\times D}(\mathbf{F}|\mathbf{0}, \mathbf{K}_Z, \mathbf{K}_D) \\ &= \mathcal{N}\big(\mathrm{vec}(\mathbf{F})|\mathbf{0}, \mathbf{K}_Z \otimes \mathbf{K}_D\big), \end{aligned} \tag{4}$$

where $\mathbf{K}_Z$ is a kernel matrix on the latent representations $\mathbf{Z}$ and $[\mathbf{K}_Z]_{ij} = \kappa_z(\mathbf{z}_i, \mathbf{z}_j)$.

The distribution in (4) defines a Gaussian process (Rasmussen and Williams 2006) for shapes. Each variate of the process is a continuous field for a shape image. Given a finite set of locations $\mathbf{Z}$, the projected variates of the GP, namely, $\mathbf{F}$, follow a multivariate Gaussian distribution given in (4).

The GP essentially maps the low-dimensional representations to the shape images. We can model any (nonlinear) mapping via choosing an appropriate (nonlinear) covariance (or kernel) function $\kappa_z(\cdot,\cdot)$. We refer to this GP as a shape-variate GP.

Although the GP endows us with nonlinear shape modeling power, learning this GP is problematic because we need to estimate the $D \times D$ correlation matrix $\mathbf{K}_D$. Since D is the number of pixels, $\mathbf{K}_D$ can be tremendous, even with small-sized shape images, e.g., 32×32. To reduce the parameters while keeping the flexibility of capturing complex spatial correlations between the pixels, we for each dimension of the images introduce a set of latent spatial variables, $\mathbf{S} = \{\mathbf{s}_1, \ldots, \mathbf{s}_I\}$ and $\mathbf{T} = \{\mathbf{t}_1, \ldots, \mathbf{t}_J\}$ (the image size is $I \times J$). These variables characterize the spatial properties of the shapes at different positions or coordinates. We then define $\mathbf{K}_D$ as a kernel matrix on the spatial variables to capture the spatial correlations, and the kernel function has a composite form, $[\mathbf{K}_D]_{dd'} = \kappa_s(\mathbf{s}_{i_d}, \mathbf{s}_{i_{d'}})\kappa_t(\mathbf{t}_{j_d}, \mathbf{t}_{j_{d'}})$, where $\kappa_s(\cdot,\cdot)$ and $\kappa_t(\cdot,\cdot)$ are arbitrary kernels that measure the similarity of the spatial variables in each dimension (and can be different), and (i_d, j_d) and $(i_{d'}, j_{d'})$ are the coordinates for the d-th and d'-th pixel, respectively. It is easy to see that $\mathbf{K}_D = \mathbf{K}_S \otimes \mathbf{K}_T$ where $\mathbf{K}_S$ and $\mathbf{K}_T$ are the kernel matrices on $\mathbf{S}$ and $\mathbf{T}$, respectively. The Kronecker structure reduces the number of parameters from $\mathcal{O}(D^2)$ to $\mathcal{O}(I + J)$. Meanwhile, the complex spatial correlations can be fully grasped by the spatial variables and rich (nonlinear) kernel functions.

Now, with the spatial latent variables $\mathbf{S}$ and $\mathbf{T}$, the finite probability of our shape-variate GP is given by

$$p(\mathbf{F}|\mathbf{Z}, \mathbf{S}, \mathbf{T}) = \mathcal{N}(\mathrm{vec}(\mathbf{F})|\mathbf{0}, \mathbf{K}_Z \otimes \mathbf{K}_S \otimes \mathbf{K}_T). \quad (5)$$

The covariance between any two outputs (i.e., pixels) f_{nij} and $f_{n'i'j'}$ is

$$\mathrm{cov}(f_{nij}, f_{n'i'j'}) = \kappa_z(\mathbf{z}_n, \mathbf{z}_{n'})\kappa_s(\mathbf{s}_i, \mathbf{s}_{i'})\kappa_t(\mathbf{t}_j, \mathbf{t}_{j'}).$$

The covariance integrates the (nonlinear) spatial correlations within shapes and the cross-correlations between different shapes (i.e., $\kappa_z(\mathbf{z}_n, \mathbf{z}_{n'})$). In this way, our model is flexible enough to capture a variety of nonlinear shape variations to obtain high-quality compact representations.

Dirichlet Process Mixture Prior

Next, to uncover the hidden structures within the shapes, we place a Dirichlet process mixture (DPM) prior (Antoniak 1974) over each latent representation $\mathbf{z}_n$. As a nonparametric mixture prior, DPM can sample an unbounded number of mixture components (i.e., cluster centers), through which we can identify both the cluster number and memberships from posterior inference. Specifically, to sample $\mathbf{z}_n$, we first sample an infinite collection of random variables $\mathbf{v} = \{v_1, v_2, \cdots\}$ and cluster centers $\boldsymbol{\eta} = \{\eta_1, \eta_2, \cdots\}$ from $p(\mathbf{v}|\alpha) = \prod_{m=1}^{\infty} \mathrm{Beta}(v_m|1, \alpha)$, and $p(\boldsymbol{\eta}) = \prod_{m=1}^{\infty} \mathcal{N}(\boldsymbol{\eta}_m|\mathbf{0}, \mathbf{I})$ where $\alpha > 0$ is the concentration hyper-parameter. Then we sample a cluster membership u_n, and then sample $\mathbf{z}_n$ according to the assigner cluster center,

$$\begin{aligned} p(\mathbf{z}_n, u_n|\mathbf{v}, \boldsymbol{\eta}) &= p(\mathbf{z}_n|u_n, \boldsymbol{\eta})p(u_n|\mathbf{v}) \\ &= \prod_{m=1}^{\infty} \pi_m(\mathbf{v})^{\delta(u_n=m)} \mathcal{N}(\mathbf{z}_n|\boldsymbol{\eta}_{u_n}, \lambda\mathbf{I}), \end{aligned} \quad (6)$$

where $\pi_m(\mathbf{v}) = v_m \prod_{m'=1}^{m-1}(1 - v_{m'})$, λ is the variance parameter that controls the distance from $\mathbf{z}_n$ to the cluster center, and $\delta(\cdot)$ is the indicator function.

Finally, given the continuous field $\mathbf{f}_n$, we sample the observed binary shape image $\mathbf{y}_n$ from a probit regression model, $p(\mathbf{y}_n|\mathbf{f}_n) = \prod_{n,i,j} \Phi(f_{nij})^{y_{nij}}(1 - \Phi(f_{nij}))^{1-y_{nij}}$, where $\Phi(\cdot)$ is the standard Gaussian CDF function. Given (5) and (6), we can obtain the joint probability of our model by $p(\mathbf{Z}, \mathbf{S}, \mathbf{T}, \mathbf{F}, \mathbf{Y}, \mathbf{u}, \mathbf{v}, \boldsymbol{\eta}) = \prod_{n=1}^{N} p(\mathbf{z}_n|u_n, \boldsymbol{\eta})p(u_n|\mathbf{v})p(\mathbf{v}|\alpha)p(\mathbf{F}|\mathbf{Z}, \mathbf{S}, \mathbf{T})p(\mathbf{S})p(\mathbf{T})$ $p(\boldsymbol{\eta}) \prod_{n=1}^{N} p(\mathbf{y}_n|\mathbf{f}_n)$, where $\mathbf{u} = \{u_1, \ldots, u_N\}$ are the cluster memberships, and $p(\mathbf{S})$ and $p(\mathbf{T})$ are standard Gaussian prior distributions for the spatial latent variables $\mathbf{S}$ and $\mathbf{T}$.

Model Estimation

We now present the model estimation algorithm. In a Bayesian framework, the estimation amounts to computing the posterior distributions of the latent variables. However, exact posterior computation is infeasible since we cannot calculate the normalization constant. Hence, we resort to a variational expectation-maximization (EM) approach: in the E step, we approximate the posterior distributions for a set of latent variables, such as the fields $\mathbf{F}$ and the cluster memberships $\mathbf{u}$; in the M step, based on these posteriors, we optimize the representations $\mathbf{Z}$ and other parameters.

Specifically, for a closed-form update, we first introduce an augmented variable h_{nij} for each pixel y_{nij} and decompose each likelihood $p(y_{nij}|f_{nij})$ into $p(h_{nij}|f_{nij}) = \mathcal{N}(h_{nij}|f_{nij}, 1)$, where $p(y_{nij}|h_{nij}) = \delta(y_{nij} = 1)\delta(h_{nij} > 0) + \delta(y_{nij} = 0)\delta(h_{nij} \leq 0)$. This is equivalent to the original probit model (Albert and Chib 1993). Denote all the augmented variables $\{h_{nij}\}$ by $\mathbf{H}$. The joint probability now is

$$\begin{aligned} p(\mathbf{Z}, \mathbf{S}, \mathbf{T}, \mathbf{F}, \mathbf{H}, \mathbf{Y}, \mathbf{u}, \mathbf{v}, \boldsymbol{\eta}) = \prod_{n=1}^{N} p(\mathbf{z}_n|u_n, \boldsymbol{\eta})p(u_n|\mathbf{v})p(\mathbf{S}) \\ p(\mathbf{T})p(\boldsymbol{\eta})p(\mathbf{v}|\alpha)p(\mathbf{F}|\mathbf{Z}, \mathbf{S}, \mathbf{T}) \prod_{n,i,j} p(h_{nij}|f_{nij})p(y_{nij}|h_{nij}). \end{aligned}$$

Variational Approximation

We then, in the E step, use variational inference to approximate the posterior distributions of the latent fields $\mathbf{F}$, the augmented variables $\mathbf{H}$, the cluster memberships $\mathbf{u}$ and the DPM-related variables $\mathbf{v}$ and $\boldsymbol{\eta}$. Specifically, we use a factorized distribution, $q(\mathbf{F})q(\mathbf{H})q(\mathbf{u})q(\mathbf{v})q(\boldsymbol{\eta})$ to approximate the true posterior $p(\mathbf{F}, \mathbf{H}, \mathbf{u}, \mathbf{v}, \boldsymbol{\eta}|\mathbf{Y})$. To obtain the optimal approximation, we minimize the Kullback-Leibler (KL) divergence between the approximate and exact posterior. Each time, we update one posterior, say, $q(\mathbf{v})$, while fixing the others, and cyclically refine each posterior until convergence. To handle the infinite supports in $\mathbf{v}$, $\boldsymbol{\eta}$ and the cluster memberships $\mathbf{u}$, we use a truncated variational posterior (Blei, Jordan, and others 2006). Specifically, we set

a truncation level M and enforce $q(v_M = 1) = 1$ so that $q(u_n > M) = 0$ $(1 \leq n \leq N)$. The truncated variational posteriors for $\mathbf{v}$, $\mathbf{u}$ and $\boldsymbol{\eta}$ are updated by

$$
\begin{aligned}
q(u_n) &= \text{Multinomial}(u_n|\zeta_{n1}, \cdots, \zeta_{nM}),\\
q(v_m) &= \text{Beta}(v_m|\gamma_{m1}, \gamma_{m2}),\\
q(\boldsymbol{\eta}_m) &= \mathcal{N}(\boldsymbol{\eta}_m|\boldsymbol{\mu}_m, s_m\mathbf{I}),
\end{aligned}
$$

where

$$
\begin{aligned}
\zeta_{nm} \propto \exp(&\mathbb{E}_q\left[\log(v_m)\right] + \sum_{m'=1}^{m-1}\mathbb{E}_q\left[\log(1-v_{m'})\right]\\
&- \frac{1}{2\lambda}\mathbb{E}_q\left[||\boldsymbol{\eta}_m||^2\right] + \frac{1}{\lambda}\mathbf{z}_n{}^T\mathbb{E}_q\left[\boldsymbol{\eta}_m\right]),\\
\gamma_{m1} = 1 + \sum_{n=1}^{N}\zeta_{nm}, &\quad \gamma_{m2} = \alpha + \sum_{n=1}^{N}\sum_{m'=m+1}^{M}\zeta_{nm'},\\
s_m = \frac{1}{1+\lambda^{-1}\sum_{a=1}^{N}\zeta_{nm}}, &\quad \boldsymbol{\mu}_m = \frac{\sum_{n=1}^{N}\zeta_{nm}\mathbf{z}_n}{\lambda + \sum_{n=1}^{N}\zeta_{nm}}.
\end{aligned}
$$

The moments are calculated by $\mathbb{E}_q[\log(1-v_m)] = \psi(\gamma_{m2}) - \psi\left(\gamma_{m1}+\gamma_{m2}\right)$, $\mathbb{E}_q[\log v_m] = \psi(\gamma_{m1}) - \psi\left(\gamma_{m1}+\gamma_{m2}\right)$, $\mathbb{E}_q[\boldsymbol{\eta}_m] = \boldsymbol{\mu}_m$, and $\mathbb{E}_q[\|\boldsymbol{\eta}_m\|^2] = \|\boldsymbol{\mu}_m\|^2 + Ls_m$ where $\psi(\cdot) = \frac{d}{dx}\ln\Gamma(\cdot)$. The variational posteriors $q(\mathbf{F})$ and $q(\mathbf{H})$ are updated by

$$
\begin{aligned}
q(\mathbf{F}) &= \mathcal{N}(\text{vec}(\mathbf{F})|\text{vec}(\mathbb{E}_q[\mathbf{H}]), \boldsymbol{\Sigma}_f),\\
q(\mathbf{H}) &\propto \mathcal{N}\left(\text{vec}(\mathbf{H})|\text{vec}(\mathbb{E}_q[\mathbf{F}]), \mathbf{I}\right) \odot \delta(\text{vec}(\mathbf{H}) \geq \mathbf{1}),
\end{aligned}
$$

where $\Sigma_f = \boldsymbol{\Sigma}(\mathbf{I}+\boldsymbol{\Sigma})^{-1}$, $\odot$ is the Hadamard (or element-wise) product, and

$$
\text{vec}(\mathbb{E}_q[\mathbf{H}]) = \text{vec}(\mathbb{E}_q[\mathbf{F}]) + \frac{\bar{\mathbf{Y}} \odot \mathcal{N}(\text{vec}(\mathbb{E}_q[\mathbf{F}])|\mathbf{0}, \mathbf{I})}{\Phi(\bar{\mathbf{Y}} \odot \text{vec}(\mathbb{E}_q[\mathbf{F}]))}.
$$

Here $\bar{\mathbf{Y}} = 2\text{vec}(\mathbf{Y}) - \mathbf{1}$ and $\boldsymbol{\Sigma} = \mathbf{K}_Z \otimes \mathbf{K}_S \otimes \mathbf{K}_T$ is the covariance marix of the shape-variate GP.

Estimating Shape Representations

In the M step, we optimize the low-dimensional representations $\mathbf{Z}$, the spatial variables $\mathbf{S}$ and $\mathbf{T}$ and the kernel parameters based on the variational posteriors updated in the E step. Specifically, we maximize the expected log joint probability $\mathcal{L}$ under these variational posterior distributions,

$$
\begin{aligned}
\mathcal{L} &= \mathbb{E}_q\left[\log p\left(\mathbf{Z}, \mathbf{S}, \mathbf{T}, \mathbf{F}, \mathbf{H}, \mathbf{Y}, \mathbf{u}, \mathbf{v}, \boldsymbol{\eta}\right)\right]\\
&= \log|\boldsymbol{\Sigma}| + \text{tr}(\boldsymbol{\Sigma}^{-1}\boldsymbol{\Sigma}_f) + \text{vec}(\mathbb{E}_q[\mathbf{F}])^T\boldsymbol{\Sigma}^{-1}\text{vec}(\mathbb{E}_q[\mathbf{F}])\\
&- \sum_{n=1}^{N}\frac{\lambda}{2}\mathbb{E}_q[\|\mathbf{z}_n - \sum_{m=1}^{M}\delta(z_n = m)\boldsymbol{\eta}_m\|^2] + \text{const.} \qquad (7)
\end{aligned}
$$

The major computational challenge in (7) is the $ND \times ND$ covariance matrix $\boldsymbol{\Sigma}$. Since D is the number of pixels, even a small collection (e.g., 100) of images (e.g., 32×32) can lead to a huge $\boldsymbol{\Sigma}$ (over $100K \times 100K$) and prohibitive cost to compute its inverse, log determinant and the gradient. To overcome this problem, we utilize the properties of the Kronecker product in $\boldsymbol{\Sigma}$. Specifically, since $\boldsymbol{\Sigma} = \mathbf{K}_Z \otimes \mathbf{K}_S \otimes \mathbf{K}_T$, we have $|\boldsymbol{\Sigma}| = |\mathbf{K}_Z|^{IJ}|\mathbf{K}_S|^{NJ}|\mathbf{K}_T|^{NI}$ where the exponent of each determinant is the product of the ranks of the other two kernel matrices. Then, we have $\log|\boldsymbol{\Sigma}| = IJ\log|\mathbf{K}_Z| + NJ\log|\mathbf{K}_S| + NI\log|\mathbf{K}_T|$. To compute the gradient of $\log|\boldsymbol{\Sigma}|$, we need only to take the gradient of the log determinant of each kernel matrix, which is much cheaper. Next, to compute $\boldsymbol{\Sigma}^{-1}$ related terms, say, $\text{tr}(\boldsymbol{\Sigma}^{-1}\boldsymbol{\Sigma}_f)$, we first perform eigendecomposition for each kernel matrix: $\boldsymbol{\Sigma}_Z = \mathbf{U}_Z\boldsymbol{\Lambda}_Z\mathbf{U}_Z^\top$, $\boldsymbol{\Sigma}_S = \mathbf{U}_S\boldsymbol{\Lambda}_S\mathbf{U}_S^\top$ and $\boldsymbol{\Sigma}_T = \mathbf{U}_T\boldsymbol{\Lambda}_T\mathbf{U}_T^\top$, where $\{\mathbf{U}_Z, \mathbf{U}_S, \mathbf{U}_T\}$ are egienvectors, and $\{\boldsymbol{\Lambda}_Z, \boldsymbol{\Lambda}_S, \boldsymbol{\Lambda}_T\}$ diagonal matrices with eigenvalues in the diagonal. Hence, we have $\boldsymbol{\Sigma} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\top$ where $\mathbf{U} = \mathbf{U}_Z \otimes \mathbf{U}_S \otimes \mathbf{U}_T$ and $\boldsymbol{\Lambda} = \boldsymbol{\Lambda}_Z \otimes \boldsymbol{\Lambda}_S \otimes \boldsymbol{\Lambda}_T$. Since $\boldsymbol{\Sigma}_f = \boldsymbol{\Sigma}(\mathbf{I}+\boldsymbol{\Sigma})^{-1}$, we can derive that $\text{tr}(\boldsymbol{\Sigma}^{-1}\boldsymbol{\Sigma}_f) = \text{tr}\big((\mathbf{I}+\boldsymbol{\Sigma})^{-1}\big) = \text{tr}\big(\mathbf{U}(\mathbf{I}+\boldsymbol{\Lambda})^{-1}\mathbf{U}^\top\big) = \text{tr}\big((\mathbf{I}+\boldsymbol{\Lambda})^{-1}\big)$. Since $\mathbf{I}+\boldsymbol{\Lambda}$ is a diagonal matrix, the inverse and trace computation is trivial. For the gradient of $\text{tr}(\boldsymbol{\Sigma}^{-1}\boldsymbol{\Sigma}_f)$, let us take $\mathbf{K}_Z$ as an example. We observe that $\nabla\mathbf{K}_Z = \mathbf{U}_Z(\mathbf{U}_Z^\top\nabla\mathbf{K}_Z\mathbf{U}_Z)\mathbf{U}_Z^\top$. Therefore, $\nabla\boldsymbol{\Sigma} = \nabla\mathbf{K}_Z \otimes \mathbf{K}_S \otimes \mathbf{K}_T = \mathbf{U}_Z(\mathbf{U}_Z^\top\nabla\mathbf{K}_Z\mathbf{U}_Z)\mathbf{U}_Z^\top \otimes \mathbf{U}_S\boldsymbol{\Lambda}_S\mathbf{U}_S^\top \otimes \mathbf{U}_T\boldsymbol{\Lambda}_T\mathbf{U}_T^\top = \mathbf{U}(\mathbf{U}_Z^\top\nabla\mathbf{K}_Z\mathbf{U}_Z \otimes \boldsymbol{\Lambda}_S \otimes \boldsymbol{\Lambda}_T)\mathbf{U}^\top$. With this in hand, we can derive that $\nabla\text{tr}(\boldsymbol{\Sigma}^{-1}\boldsymbol{\Sigma}_f) = \text{tr}\big(\nabla(\mathbf{I}+\boldsymbol{\Sigma})^{-1}\big) = -\text{tr}\big((\mathbf{I}+\boldsymbol{\Sigma})^{-1}\nabla\boldsymbol{\Sigma}(\mathbf{I}+\boldsymbol{\Sigma})^{-1}\big) = -\text{tr}\big(\boldsymbol{\Lambda}^{-1}\boldsymbol{\Lambda}^{-1}\big(\text{diag}(\mathbf{U}_Z^\top\nabla\mathbf{K}_Z\mathbf{U}_Z) \otimes \boldsymbol{\Lambda}_S \otimes \boldsymbol{\Lambda}_T\big)\big)$. Now, since the computation is taken place on diagonal matrices again, it becomes much easier and cheaper. We then can use the chain rule to calculate the gradient w.r.t the representations $\mathbf{Z}$ and the kernel parameters. Computing the gradient regarding $\mathbf{K}_S$ and $\mathbf{K}_T$ is accomplished the same way. Finally, we use L-BFGS to maximize the expected log joint probability $\mathcal{L}$ due to its excellent performance.

Backward and Forward Mapping

Suppose we have conducted the variational EM algorithm to estimate our model on a collection of shape images $\mathbf{Y}$. Now, if we see a new image $\mathbf{y}_*$, how can we estimate its compact representation $\mathbf{z}_*$? This estimation is called backward mapping. To utilize the previous learning results, we can run the same variational EM algorithm, but fixing the variational posteriors for the DPM related variables $\mathbf{v}$ and $\boldsymbol{\eta}$, the spatial latent variables $\mathbf{S}$ and $\mathbf{T}$ and the kernel parameters that have already been estimated. Then, we obtain the representation $\mathbf{z}_*$ and the cluster membership u_* that align with the existing shape representations.

Conversely, given an arbitrary compact representation $\mathbf{z}_*$, how can we generate the shape image $\mathbf{y}_*$? This is called forward mapping. First, we can obtain a conditional distribution of the continuous field $\mathbf{f}_*$ by

$$
p(\mathbf{f}_*|\mathbf{z}_*, \mathbf{Z}, \mathbf{S}, \mathbf{T}, \mathbf{Y}) = \mathcal{N}(\text{vec}(\mathbf{f}_*)|\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*),
$$

where $\boldsymbol{\mu}_* = \mathbf{K}_*^T(\boldsymbol{\Sigma}+\mathbf{I})^{-1}\text{vec}(\mathbf{H})$, $\boldsymbol{\Sigma}_* = \mathbf{K}_{**} - \mathbf{K}_*^T(\boldsymbol{\Sigma}+\mathbf{I})^{-1}\mathbf{K}_*$, $\mathbf{K}_* = \mathbf{k}_{n*} \otimes \mathbf{K}_S \otimes \mathbf{K}_T$, $\mathbf{k}_{n*} = [\kappa_z(\mathbf{z}_*, \mathbf{z}_1), \ldots, \kappa_z(\mathbf{z}_*, \mathbf{z}_N)]^T$, and $\mathbf{K}_{**} = \kappa_z(\mathbf{z}_*, \mathbf{z}_*)(\mathbf{K}_S \otimes \mathbf{K}_T)$. Then, we compute the predictive distribution of $\mathbf{y}_*$ by

$$
\begin{aligned}
p(\mathbf{y}_*|\mathbf{z}_*) &= \int p(\mathbf{y}_*|\mathbf{h}_*)p(\mathbf{h}_*|\mathbf{f}_*)p(\mathbf{f}_*|\mathbf{z}_*, \mathbf{Z}, \mathbf{S}, \mathbf{T}, \mathbf{Y})d\mathbf{h}_*d\mathbf{f}_*\\
&= \Phi\big((2\mathbf{y}_* - 1) \circ \frac{\boldsymbol{\mu}_*}{\sqrt{\mathbf{1} + \text{diag}(\boldsymbol{\Sigma}_*)}}\big).
\end{aligned}
$$

Algorithm Complexity

The naive implement of our variational EM approach will incur a bottleneck in the calculation related to $\mathbf{\Sigma}$ (see (7)), i.e., the covaraince matrix of the shape-variate GP, and leads to $\mathcal{O}\big((NIJ)^3\big)$ time and $\mathcal{O}\big((NIJ)^2\big)$ space complexity. However, using the Kronecker-product properties, we can reduce the time complexity to $\mathcal{O}(N^3+I^3+J^3)$ and the space complexity to $\mathcal{O}(N^2+I^2+J^2)$, and we need to compute and maintain the kernel matrices only for the representations and spatial variables, which are much smaller.

Related Work

InfShapeOdds hybridizes a shape-variate Gaussian process (GP) and Dirichlet process (Antoniak 1974) to fulfill the nonlinear shape modeling and latent structure discovery. A closely related approach is the classical Gaussian process latent variable model (GPLVM) (Lawrence 2005) that uses vanilla GPs and assumes all the outputs (i.e., pixels) are independent given the representations. Hence, GPLVM ignores the spatial coherence within the shape images, and might be less capable of finding good representations. We can reduce InfShapeOdds to GPLVM by setting the spatial covariance matrices (see (5)) $\mathbf{K}_S = \mathbf{I}$ and $\mathbf{K}_T = \mathbf{I}$. The computational benefit of the Kronecker product has been realized in the GP community and exploited in quite a few models (Saatcci 2012; Luttinen and Ilin 2012; Rakitsch et al. 2013; Flaxman et al. 2015; Yu, Li, and Liu 2018; Zhe, Xing, and Kirby 2019). Although often being utilized to enable exact inference, the Kronecker product recently has also been used for efficient approximate inference with factorized kernels (e.g., RBF) (Wilson and Nickisch 2015; Izmailov, Novikov, and Kropotov 2018). These excellent works lay inducing points/variational posteriors on the grids over input dimensions to construct Kronecker products and combine with interpolations to accelerate the computation.

Experiments

Evaluating Compact Shape Representations

We first evaluated the quality of the learned representations by our method in the tasks of reconstruction and shape interpolation.

Datasets. For a fair comparison, we used the same benchmark datasets as in the ShapeOdds paper (Elhabian and Whitaker 2017): the *Weizmann horse* dataset (Borenstein and Ullman 2008), including 328 silhouettes of horses facing to the left with different poses, and the *Caltech-101 motorcycle* dataset (Fei-Fei, Fergus, and Perona 2007), containing 798 silhouettes of different motorcycles facing to the right. Following (Eslami et al. 2014), we cropped and normalized *Weizmann horse* and *Caltech-101 motocycle* to 32×32 and 64×64 images, respectively.

Competing methods. We compared InfShapeOdds with the original ShapeOdds (Elhabian and Whitaker 2017), Gaussian process latent variable model (GPLVM) (Lawrence 2005) (modified and equipped with a probit link function) and VAE (Kingma and Welling 2013). Note that we did not compete against other methods, such as Shape Boltzmann Machines (ShapeBM) (Eslami et al. 2014), because they have been shown inferior to ShapeOddes (Elhabian and Whitaker 2017) for these datasets. For our method and GPLVM, we used the SE-ARD kernel and ran a maximum of 300 iterations for model estimation. Both methods were based on the L-BFGS optimization algorithm. We compete with two VAEs: VAE-1, which has two fully connected hidden layers in the encoder, each layer with half of the number of neurons in the previous layer, and the decoder with the same structure in the reverse direction; VAE-2, which is similar to VAE-1 but includes 3 fully connected hidden layers. We applied a ReLU activation function for all the hidden layers, except for the last layer of the decoder, for which we used a Sigmoid activation function to match the binary outputs (i.e., pixels). The Adam (Kingma and Ba 2014) algorithm was used for training. To ensure the best performance for VAEs, we held 20% of the training data for validation. At each epoch, we examined the validation error and stored the model. After running 1000 epochs, the model with the lowest validation error was chosen for testing.

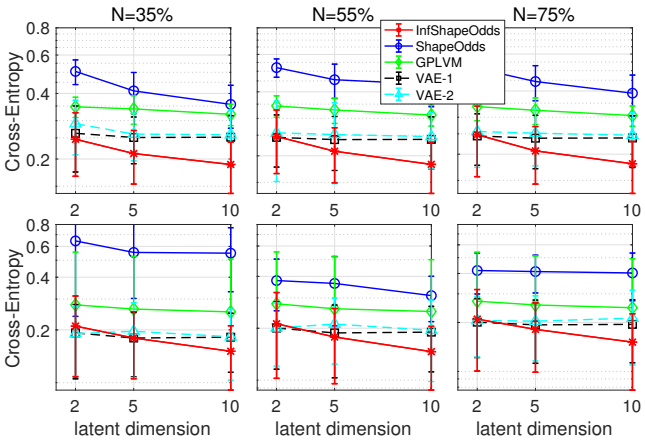


Figure 1: Reconstruction error: *horses* (top row) and *motorcycles* (bottom row). Training sample ratios are $N = 35\%, 55\%, 75\%$.

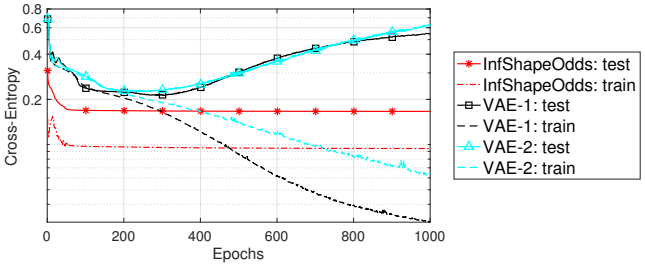


Figure 2: Running test and training errors on the *horse* dataset with $N = 75\%$ and $r = 10$.

Reconstruction. To evaluate the quality of the compact representations, we first examined how well they can reconstruct the original shape images. To this end, for each test shape image $\mathbf{y}_*$, we estimated the representation $\mathbf{z}_*$ (from backward mapping), and then used $\mathbf{z}_*$ to generate an image $\hat{\mathbf{y}}_*$ (via forward mapping). We then computed the cross-

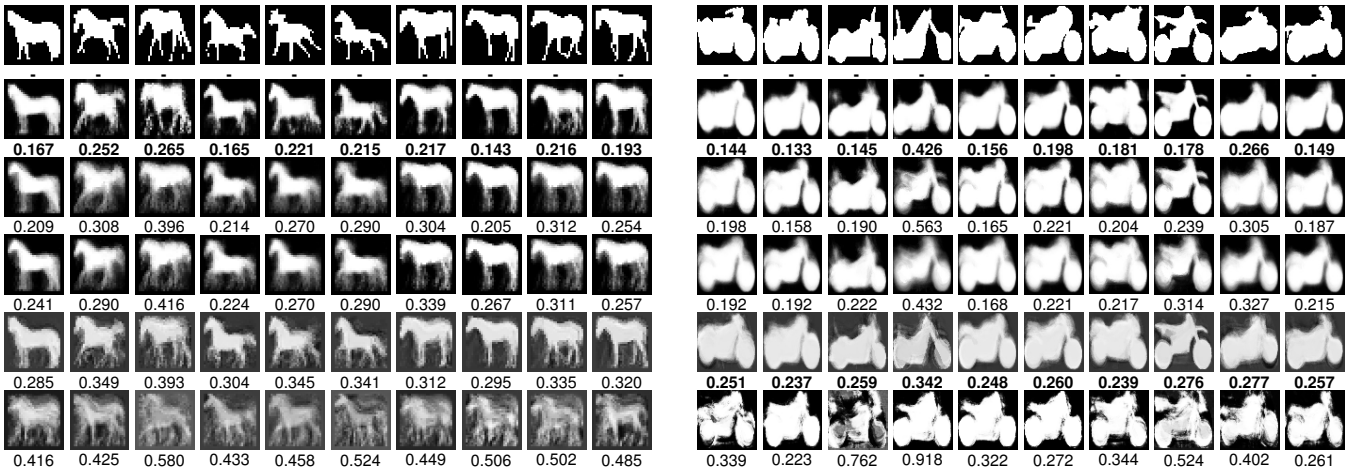
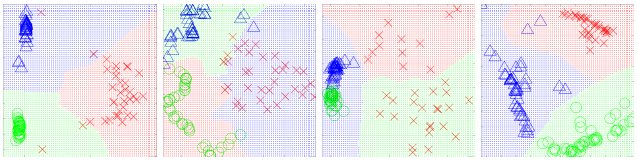


Figure 3: Reconstruction details for the *horse* and *motorcycle* datasets ($N = 75\%$ and $r = 10$). The rows from top to bottom: ground truth, InfShapeOdds, VAE-1, VAE-2, GPLVM, and ShapeOdds. Numbers are cross-entropies.



(a) Purity=0.96 InfShapeOdds (b) Purity=0.91 GPLVM (c) Purity=0.55 ShapeOdds (d) Purity=0.98 VAE-2

Figure 4: Estimated latent clusters

entropy between $\mathbf{y}_*$ and $\hat{\mathbf{y}}_*$ to measure the reconstruction error. We varied the portion of the training images from $\{35\%, 55\%, 75\%\}$ and reconstructed the remaining images. The average cross-entropies and their standard deviations are reported in Fig. 1. As we can see, InfShapeOdds nearly always obtains the smallest reconstruction errors and in particular significantly outperforms ShapeOdds. GPLVM and VAEs outperform ShapeOdds as well, confirming the advantages of nonlinear shape modeling. Although we used early stopping to prevent VAEs from overfitting, their performance in most cases are still inferior to our method. This demonstrates the advantage of our nonparametric models on small datasets. We also tried the dropout regularization (Srivastava et al. 2014) in training VAEs but obtained similar comparison results. Furthermore, we examined how the reconstruction error varies along with the epochs/iterations for InfShapeOdds and VAEs. From Fig. 2, we can see that InfShapeOdds never overfits the data — the test and training errors vary almost in the same trend and converge quickly in 100 iterations. In contrast, for VAEs, the training errors keep decreasing while at some stage the test error starts to increase, which exhibits a typical overfitting behaviour.

To examine the reconstruction details, we randomly show a set of reconstructed images from all the methods (training portion 75%, $r = 10$) and the ground-truth in Fig. 3. It can be seen that InfShapeOdds recovered the images better than all competitors, especially on the edges of abrupt changes. The images reconstructed by ShapeOdds are blurred, especially in the *horse* dataset (see the last row of Fig. 3), implying that ShapeOdds failed to capture the nonlinear variations of the shapes. The reason for the inferior performance of GPLVM might be that it ignores the spatial correlations inside the images, despite its nonlinear modeling capability. Visually, the VAE results are close to our method, but the images are more blurred on edges, e.g., horse legs and motorcycle shells.

Shape interpolation. Good representation learning approaches should find a latent space in which most valid shapes can reside. To evaluate this point, we examined all the methods through the shape interpolation. Specifically, from the training set, we randomly selected a pair of shape images that are significantly different (in terms of Hamming distance). After training, we obtained the compact representations of the two shapes, based on which we produced 18 intermediate representations from a linear interpolation. we then generated the shape images for those intermediate representations. Again, we set $r = 10$ and chose 75% of the examples in the *horse* dataset to train each method. The interpolated shapes are shown in Figure 5. As we can see, the horse shapes interpolated from InfShapeOdds are clear, valid and smooth (see the top row), which is consistent with our previous results. The horse shapes interpolated by ShapeOdds, however, are very fuzzy and overlapping, indicating that ShapeOdds failed to find a good latent space to accommodate valid shapes. The results of GPLVM are in-between, but still contain many artifacts around the edges (see the fourth row). VAEs produced quite a few sharp images on the left. However, the interpolated images from the middle to the right are more blurred than our approach, especially on the horse legs. We also tested all the methods on the *motorcycle* dataset and observed similar results. Due to the space limit, we include the results in the supplementary material.

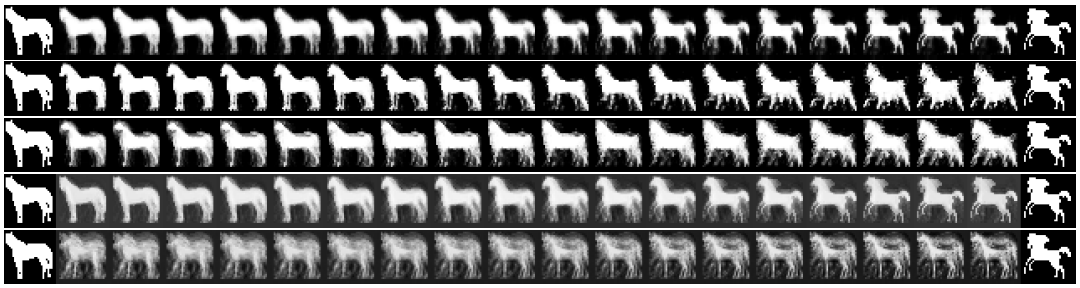


Figure 5: Shape interpolation between the left and right shapes. From top to bottom: InfShapeOdds, VAE-1, VAE-2, GPLVM and ShapeOdds.

Latent Structure Discovery

Shapes in many applications exhibit strong multi-modality (Trinh and Kimia 2011), implying rich underlying structures. Therefore, we also evaluated InfShapeOdds in uncovering the hidden structures. To this end, we first generated a synthetic dataset of ellipse shapes. Each ellipse is 128×128. We first sampled shape representations in a two-dimensional space. Each representation was sampled from a Gaussian mixture model (GMM), with three components of equal weights (i.e., $\frac{1}{3}$) and centers $(64, 0), (0, 64), (64, 64)$. The covariance matrix of each component is $64\mathbf{I}$. Given a representation $\mathbf{z} = [z_1, z_2]$, each pixel (i, j) of the corresponding ellipse $\mathbf{y}$ was generated by $y_{ij} = \delta(\exp(-(i-64)^2/z_1^2 - (j-64)^2/z_2^2) > 0.5)$.

We then ran InfShapeOdds, ShapeOdds, GPLVM and VAE-2 to estimate the representations. For InfShapeOdds, we set the truncation level for the DPM prior to 10, and it automatically returned the correct number of clusters and the cluster membership of each representation. For GPLVM, ShapeOdds and VAE-2, we ran k-means to cluster their learned representations. We set the number of clusters to 3. Note, however, that in practice we usually do not know the ground-truth cluster number. We computed the purity of the estimated clusters. The purity is defined as $P(T, C) = \frac{1}{N}\sum_j \max|t_k \cap c_j|$ where $T = \{t_1, \ldots, t_K\}$ are the ground-truth classes and $C = \{c_1, \ldots, c_J\}$ the clusters found by the algorithm. Higher purity indicates better performance. The results are shown in Fig. 4. The cluster regions are filled with different background colors, and the markers indicate the ground-truth class. As we can see, both InfShapeOdds and VAE-2 recovered well the ground-truth structure and achieved the highest purity (Fig. 4c and d). Although VAE-2 obtains a slightly better purity, its clusters are more expanded. Furthermore, InfShapeOdds can jointly infer the number of clusters and memberships but VAE-2 cannot. In Fig. 4b, although the clusters of the representations learned by GPLVM also align with the ground truth, those clusters are much more expanded, as compared with InfShapeOdds, and tends to spread out to the boundaries. In Fig. 4c, the representations from ShapeOdds severely deviate from the ground-truth structure and has a much worse purity of 0.55. It implies that ShapeOdds failed to capture the nonlinear variations in the simulated ellipses.

Next, we examined our method on a real-world dataset, *fashion MNIST* (https://github.com/zalandoresearch/fashion-mnist). This dataset comprises 28×28 gray-scale images for 10 categories of fashion products, including shoes, skirts, shirts, etc. We first converted these images into binary shapes with a threshold of 0.05. We then generate six datasets, each of which contains $1,024$ random images, based on the number of categories being used. For each training set, we ran all the methods to estimate compact representations (with dimension 2). To determine the number of clusters for GPLVM and ShapeOdds, we ran the Dirichlet Process k-means (Kulis and Jordan 2011) algorithm to cluster their representations. The results are listed in Table 1. InfShapeOdds achieves the highest purity in most cases, indicating the best clustering performance, which confirms the advantage of our method in recovering the hidden structure from real data.

#	InfShapeOdds	GPLVM	ShapeOdds	VAE-2
3	**0.6436**	0.6377	0.5762	0.4648
4	0.4854	0.5059	**0.5273**	0.2607
5	0.4258	0.3896	0.3779	**0.4863**
6	**0.4434**	0.1777	0.3281	0.3525
7	**0.5088**	0.3486	0.2627	0.2900
8	**0.4434**	0.1387	0.2646	0.2559
9	**0.4277**	0.2324	0.2324	0.3965

Table 1: The purity of the estimated clusters from *fashion MNIST*.

Finally, as a case study, we show in Fig. 6 the cluster structure found by InfShapeOdds from the *horse* dataset. These clusters of the latent representations are meaningful — they actually reflect different horse poses, e.g., running, standing and grazing. Visually, the distribution of the representations is clearly multi-modal, implying a structure in the data. We also found meaningful clusters from the *motorcycle* dataset. The results are provided in the supplementary material.

Conclusion

We have proposed a nonparametric Bayesian shape representation model based on Gaussian process and Dirichlet process. Our model can flexibly capture nonlinear shape variations and discover latent structures. The performance of our method is better than or comparable to neural-network-based approaches. However, our method does not suffer

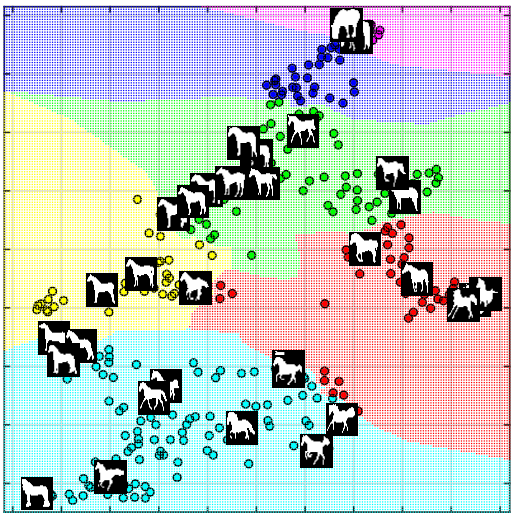


Figure 6: Latent clusters found by InfShapeOdds on the *horse* dataset.

from overfitting, requires fewer parameters, and is easier to train.

Acknowledgment

This work has been supported by DARPA TRADES Award HR0011-17-2-0016

References

Albert, J. H., and Chib, S. 1993. Bayesian Analysis of Binary and Polychotomous Response Data. *Journal of the American Statistical Association* 88(422):669.

Antoniak, C. E. 1974. Mixtures of dirichlet processes with applications to bayesian nonparametric problems. *The annals of statistics* 1152–1174.

Bengio, Y.; Courville, A.; and Vincent, P. 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence* 35(8):1798–1828.

Blei, D. M.; Jordan, M. I.; et al. 2006. Variational inference for dirichlet process mixtures. *Bayesian analysis* 1(1):121–143.

Borenstein, E., and Ullman, S. 2008. Combined top-down/bottom-up segmentation. *IEEE Transactions on pattern analysis and machine intelligence* 30(12):2109–2125.

Elhabian, S. Y., and Whitaker, R. T. 2017. Shapeodds: Variational bayesian learning of generative shape models. In *CVPR*, 2185–2196.

Eslami, S. A.; Heess, N.; Williams, C. K.; and Winn, J. 2014. The shape boltzmann machine: a strong model of object shape. *International Journal of Computer Vision* 107(2):155–176.

Fei-Fei, L.; Fergus, R.; and Perona, P. 2007. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer vision and Image understanding* 106(1):59–70.

Flaxman, S.; Wilson, A.; Neill, D.; Nickisch, H.; and Smola, A. 2015. Fast kronecker inference in gaussian processes with non-gaussian likelihoods. In *International Conference on Machine Learning*, 607–616.

Higgins, I.; Matthey, L.; Pal, A.; Burgess, C.; Glorot, X.; Botvinick, M.; Mohamed, S.; and Lerchner, A. 2017. beta-vae: Learning basic visual concepts with a constrained variational framework. *ICLR* 2(5):6.

Izmailov, P.; Novikov, A.; and Kropotov, D. 2018. Scalable gaussian processes with billions of inducing inputs via tensor train decomposition. In *International Conference on Artificial Intelligence and Statistics*, 726–735.

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

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

Kulis, B., and Jordan, M. I. 2011. Revisiting k-means: New algorithms via bayesian nonparametrics. *arXiv preprint arXiv:1111.0352*.

Lawrence, N. 2005. Probabilistic non-linear principal component analysis with gaussian process latent variable models. *Journal of machine learning research* 6(Nov):1783–1816.

Liu, F.; Xie, L.; Xia, Y.; Fishman, E. K.; and Yuille, A. L. 2018. Joint shape representation and classification for detecting pdac. *arXiv preprint arXiv:1804.10684*.

Luttinen, J., and Ilin, A. 2012. Efficient gaussian process inference for short-scale spatio-temporal modeling. In *Artificial Intelligence and Statistics*, 741–750.

Neal, R. M. 2012. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media.

Pohl, K. M.; Fisher, J.; Bouix, S.; Shenton, M.; McCarley, R. W.; Grimson, W. E. L.; Kikinis, R.; and Wells, W. M. 2007. Using the logarithm of odds to define a vector space on probabilistic atlases. *Medical Image Analysis* 11(5):465–477.

Rakitsch, B.; Lippert, C.; Borgwardt, K.; and Stegle, O. 2013. It is all in the noise: Efficient multi-task gaussian process inference with structured residuals. In *Advances in neural information processing systems*, 1466–1474.

Rasmussen, C. E., and Williams, C. K. 2006. *Gaussian process for machine learning*. MIT press.

Saatcci, Y. 2012. *Scalable inference for structured Gaussian process models*. Ph.D. Dissertation, Citeseer.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15(1):1929–1958.

Tomczak, J. M., and Welling, M. 2017. Vae with a vampprior. *arXiv preprint arXiv:1705.07120*.

Trinh, N. H., and Kimia, B. B. 2011. Skeleton search: Category-specific object recognition and segmentation using a skeletal shape model. *International Journal of Computer Vision* 94(2):215–240.

Wilson, A., and Nickisch, H. 2015. Kernel interpolation for scalable structured gaussian processes (kiss-gp). In *International Conference on Machine Learning*, 1775–1784.

Yu, R.; Li, G.; and Liu, Y. 2018. Tensor regression meets gaussian processes. In *International Conference on Artificial Intelligence and Statistics*, 482–490.

Zhe, S.; Xing, W.; and Kirby, R. M. 2019. Scalable high-order gaussian process regression. In *The 22nd International Conference on Artificial Intelligence and Statistics*, 2611–2620.