Think Global, Adapt Local: Learning Locally Adaptive K-Nearest Neighbor Kernel Density Estimators

Kenny Falkær Olsen Technical University of Denmark
Rasmus Malik Høegh Lindrup University of California, Berkeley
Morten Mørup Technical University of Denmark

Abstract

Kernel density estimation (KDE) is a powerful technique for non-parametric density estimation, yet practical use of KDE-based methods remains limited by insufficient representational flexibility, especially for higher-dimensional data. Contrary to KDE, K-nearest neighbor (KNN) density estimation procedures locally adapt the density based on the K-nearest neighborhood, but unfortunately only provide asymptotically correct density estimates. We present the KNN-KDE method introducing observation-specific kernels for KDE that are locally adapted through priors defined by the covariance of the K-nearest neighborhood, forming a fully Bayesian model with exact density estimates. We further derive a scalable inference procedure that infers parameters through variational inference by optimizing the predictive likelihood exploiting sparsity, batched optimization, and parallel computation for massive inference speedups. We find that KNN-KDE provides valid density estimates superior to conventional KDE and KNN density estimation on both synthetic and real data sets. We further observe that the Bayesian KNN-KDE even outperforms recent neural density estimation procedures on two of the five considered real data sets. The KNN-KDE unifies conventional kernel and KNN density estimation providing a scalable, generic and accurate framework for density estimation. We make our code available on GitHub[1].

[1] `https://github.com/falkaer/lakde`

Proceedings of the 27[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238.

1 INTRODUCTION

A key goal of unsupervised learning is to characterize the data density thereby providing an accurate account of where data occur. This has profound applications for outlier detection (Latecki et al., 2007), in which outliers can be diagnosed as observations occurring in low-density regions of the inferred density, to data augmentation, in which surrogate data can be sampled from the learned density. Importantly, density estimation also has applications within supervised learning. Here the class-posterior probability can be deduced from the learned class specific densities by use of Bayes' theorem. Furthermore, density estimation can be used as a step preceding a supervised procedure quantifying the validity of the supervised approach, i.e. increased validity if the predicted data point is well-supported by the density. Conversely, if the predicted data point is not well-supported by the density, the model has weak support for judging the point. Density estimation is thus at the heart of unsupervised learning with the aim of quantifying the ability to predict properties of test data using the predictive likelihood.

Much research in machine learning has focused on efficient and accurate procedures for density estimation. Ranging from simple linear models for density estimation based on matrix decompositions such as probabilistic principal component analysis (PCA) (Tipping and Bishop, 1999), to non-linear density estimation approaches using parametric modeling methods such as the Gaussian mixture models (GMMs) (Bishop, 2006), non-parametric procedures such as Dirichlet process mixtures (Ferguson, 1973; Rasmussen, 2000) and kernel density estimation (KDE) approaches (Scott, 2015). Recently, deep unsupervised learning approaches have gained much attention due to their flexible characterizations of the data density including variational auto-encoders providing a lower bound of the data likelihood (Kingma and Welling, 2013) to procedures directly optimizing the likelihood using autoregressive models, such as MADE (Germain et al., 2015), or using implicit score-matching as in DDE (Bigdeli et al., 2020) to obtain an

unnormalized log likelihood. Instead of parameterizing a distribution directly, density estimation is also seeing advances through flow procedures that learn to map from a simple, base distribution to a target, complex density trained by relying on invertible neural networks and the change of variable-formula (Dinh et al., 2014, 2016; Papamakarios et al., 2017; Kingma and Dhariwal, 2018; De Cao et al., 2020). Extensions of flows include utilizing ordinary or stochastic differential equations (Chen et al., 2018; Grathwohl et al., 2018; Zhang and Chen, 2021) as a way of defining the mapping between latent variables and data.

KDE is often used for low-dimensional data sets (Scott, 2015). Typically, a Gaussian kernel is used and the bandwidth (i.e., variance) of the Gaussian kernel is inferred by optimizing the predictive likelihood using grid-search within cross-validation (Scott, 2015) or by use of heuristic plug-in estimators for selecting the bandwidth parameter (Sheather and Jones, 1991; Bowman and Azzalini, 1997; Scott, 2015). Unfortunately, cross-validation becomes prohibitively expensive with increasing data dimensionality if the bandwidth parameter is defined separately for each dimension or a full covariance matrix kernel is invoked. While brute force evaluation of the KDE density requires evaluating N^2 pairwise likelihoods, approximate techniques may be used, such as the Fast Gauss Transform (Greengard and Strain, 1991) in which the input space is divided into boxes and only neighboring boxes are used for the evaluation exploiting that the Gaussian kernel rapidly drops to zero and using multi-resolution KD-trees (Gray and Moore, 2003) to bisect the input space. This enables scaling the KDE to $\mathcal{O}(N \log N)$ for large scale analyses, see also Lang et al. (2005) for empirical comparisons of these scaling strategies. However, these scaling abilities rely on the assumption that the kernel is shared throughout the input space or, when using a multi-resolution KD-tree, that the kernel only changes smoothly.

K-nearest neighbor density estimation procedures are inherently locally adaptive using nearest neighbor distances (Loftsgaarden et al., 1965; Devroye and Wagner, 1977) or neighborhood graphs (Von Luxburg and Alamgir, 2013) to form the density, but provide only asymptotically correct density estimates, see also Dasgupta and Kpotufe (2014); Zhao and Lai (2022) and references therein. Varying the scale of the kernel in KDE according to the density was proposed in Abramson (1982) such that high density regions has smaller scale by a factor of dividing by the square-root of the density. This was further investigated in Terrell and Scott (1992) together with nearest neighbor based density estimation and it was discussed that a freely adaptive kernel could be desirable but raises computational difficulties. In Comaniciu et al. (2001); Comaniciu (2003) a locally adaptive KDE was proposed using variable bandwidth mean shifts to locally adapt the kernel bandwidth based on estimates of the gradient of the density. Furthermore, non-stationary adaptive kernels have also been explored in the context of Gaussian Processes, see also Remes et al. (2017).

We presently integrate KNN and KDE forming the KNN-KDE framework, a KDE which locally adapts its density by imposing an observation-specific kernel defined by the covariance of the K-nearest neighborhood. Inspired by prior work (Zhang et al., 2006; Zougab et al., 2014; Bäcklin et al., 2018), KNN-KDE is formulated as a Bayesian learning problem in which the predictive likelihood is explicitly optimized by use of variational inference, yielding the VB-KNN-KDE model which can be further adapted to the local neighborhood through the Bayesian posterior. Whereas the Bayesian inference provides robustness through the prior distribution and uncertainty characterization by the learned posterior distribution, we further obtain high computational efficiency by exploiting parallelism and latent parameter sparsity in the variational formulation, allowing efficient scaling to millions of multivariate data points.

On synthetic and real tabular data sets we demonstrate the utility of the proposed KNN-KDE framework for density estimation when compare to conventional KNN and KDE estimation procedures as well as systematic ablations to simpler kernel structures using our Bayesian formulation. We further compare our approach to recent neural density estimation methods.

2 RELATED WORK

Recently, Bayesian learning of the kernel bandwidth has shown promise to address higher dimensional, efficient inference of kernel bandwidths. In Brewer (2000) and Gangopadhyay and Cheung (2002) Bayesian inference was used to estimate the kernel width in the univariate case and in Zhang et al. (2006) a shared full bandwidth matrix kernel was learned using Bayesian modeling and Markov chain Monte Carlo (MCMC) inference considering the leave-one-out cross-validated likelihood. This approach was demonstrated superior to existing heuristic procedures for setting the kernel parameters, admitting analysis beyond two dimensional data (considering up to five dimensional problems). In Zougab et al. (2014), each observation was endowed its own full bandwidth matrix imposing a global prior defined in terms of the empirical covariance, and the posterior covariance was optimized considering least-squares and cross-entropy losses. The approach was investigated in a context of up to four

dimensional density estimation. This approach was, in Ziane et al. (2015), extended to consider asymmetric kernels and, in Belaid et al. (2018), adapted to count data. In Bäcklin et al. (2018), Bayesian inference was further used to learn a locally adaptive KDE considering the framework of Abramson (1982) using the leave-one-out cross-validated likelihood to tune how kernel bandwidth should be weighted by density. These existing Bayesian procedures for multivariate kernel density estimation have been limited to low dimensional data sets ($D \leq 5$) and, due to poor scaling by $\mathcal{O}(N^2)$, limited numbers of observations ($N \leq 2000$).

Similar to Bayesian approaches to KDEs, non-parametric GMMs shares the merits of being non-parametric and thereby the ability to adapt complexity as N increases (Rasmussen, 2000). Importantly, each cluster has its own covariance matrix thereby—as opposed to the shared-kernel KDE—enabling complex adaptation to the data distribution across the input space, similar to adaptive Bayesian KDEs. Unfortunately, inference of the non-parametric GMM can be computationally challenging as the model can be stuck in suboptimal local minima Relying on advanced inference procedures using birth/death (or, split-merge) moves (Jain and Neal, 2004; Bryant and Sudderth, 2012) and scalable variational inference, which typically invokes a finite representation by a truncated stick breaking construction (Blei and Jordan, 2004) akin to a parametric model.

Deep learning (DL) approaches for density estimation typically requires large data sets in order to adequately learn the complex model representation having a high-dimensional parametric representation. However, when data is plentiful, deep learning approaches can learn to characterize complex high-dimensional densities with favorable performance as measured by predictive likelihood (Dinh et al., 2014; Germain et al., 2015; Dinh et al., 2016; Papamakarios et al., 2017; Kingma and Dhariwal, 2018; Chen et al., 2018; Grathwohl et al., 2018; De Cao et al., 2020; Bigdeli et al., 2020; Zhang and Chen, 2021). The training of such complex densities has been enabled by the ability to handle large data sets efficiently in particular exploiting parallel computations and stochastic optimization procedures.

3 METHODS

3.1 Kernel and K-nearest Neighbor Density Estimation

For data, $\mathbf{X} \in \mathbb{R}^{N \times D}$, with N samples in D-dimensions, we denote the n^{th} sample $\boldsymbol{x}_n$. Both kernel and KNN density estimation build on the idea of estimating density at a point using samples from its immediate neighborhood. In KDE, the shape and size of the neighborhood is implicitly defined by the kernel f that typically is chosen to be a Gaussian kernel (Scott, 2015),

$$p_f(\boldsymbol{x}) = \sum_n^N f(\boldsymbol{x} - \boldsymbol{x}_n), \tag{1}$$

In K-nearest neighbor (KNN) density estimation the size of the neighborhood is defined through the parameter K based on imposing an apriori defined fixed (typically Euclidean) distance metric (Devroye and Wagner, 1977),

$$\widehat{p}_K(\boldsymbol{x}) = \frac{1}{N} \frac{K}{v_D \cdot r_K(\boldsymbol{x})^D}, \tag{2}$$

where v_D is the volume of a D-dimensional Euclidean hypersphere, and $r_K(\boldsymbol{x})$ is the distance from $\boldsymbol{x}$ to its K^{th} nearest neighbor in $\mathbf{X}$.

These approaches come with advantages and disadvantages; because the KDE uses contributions from all points (but windowed by the kernel to form a local neighborhood), it yields a smoother density than KNN-based methods, which rely on order statistics through r_K. On the other hand, the KDE kernel does not vary depending on the location of the neighborhood, and so it cannot adapt to local change in the scale or shape of the density, which KNN density estimation can through changes in r_K. Importantly, the kernel f can be chosen such that $p_f(\mathbf{x})$ forms a valid density, while $\widehat{p}_K(\mathbf{x})$ is not a valid density except in the limit $N \to \infty$ (Dasgupta and Kpotufe, 2014; Zhao and Lai, 2022).

3.2 Bayesian KNN-KDE

We presently formulate a KDE that shares the beneficial properties of both KNN and conventional kernel density estimation by defining observation-specific kernels in terms of the covariance of the K-nearest neighborhood, and refer to it as KNN-KDE. The simplest instantiation of this premise is to let the kernel be a multivariate Gaussian parameterized directly by the empirical covariance of the K-nearest neighborhood $\widehat{\boldsymbol{\Sigma}}_n^{(K)}$,

$$p(\boldsymbol{x}) = \frac{1}{N} \sum_n^N \mathcal{N}(\boldsymbol{x} \,|\, \boldsymbol{x}_n, \boldsymbol{\Sigma}_n^{(K)}), \tag{3}$$

$$\widehat{\boldsymbol{\Sigma}}_n^{(K)} = \frac{1}{K} \sum_{k \in N_K(\boldsymbol{x}_n)} (\boldsymbol{x}_k - \boldsymbol{x}_n)(\boldsymbol{x}_k - \boldsymbol{x}_n)^\top, \tag{4}$$

where $N_K(\boldsymbol{x}_n)$ is the set of Euclidean K-nearest neighbors of $\boldsymbol{x}_n$. This model only requires specifying the

	$p(\boldsymbol{x})$	Kernel	Distance measure
KDE	Valid density	Fixed (typically Gaussian)	Fixed (Mahalanobis for Gaussian)
KNN	Rely on asymptotics	Locally adapt to neighborhood	Fixed (typically Euclidean)
B-KNN-KDE	Valid density	Locally adapt to neighborhood	Learned local Mahalanobis

Table 1: Properties of conventional KDE and KNN density estimation procedures in comparison to B-KNN-KDE. For Bayesian KNN-KDEs we further devise an efficient variational Bayesian inference procedure exploiting sparsity for improved scaling.

size of the neighborhood through the K parameter, and combines properties of both KNN and kernel density estimation by locally adapting to the K-nearest neighborhood while admitting a valid density, but still implicitly relies on a fixed, global metric to define the K-nearest neighborhood.

Surprisingly, to the best of our knowledge eq. (3) has not been explored previously. Notably, Breiman et al. (1977) discussed choosing the length scale of the kernel $\widehat{\boldsymbol{\Sigma}}_n^{(K)} = r_K(\boldsymbol{x}_n)^2\mathbf{I}$ in KDE to be the distance to the K^{th} nearest neighbor, and in Terrell and Scott (1992) nearest neighbor distance was suggested defined in terms of a fixed Mahalanobis distance. It was further discussed that a distance metric varying freely would be desirable but would raise computational difficulties and therefore not pursued. Equation (3) resembles the local likelihood density estimation procedure of Gao et al. (2017) in which the mean and covariance of the local likelihood function used for density estimation is defined by weighting samples according to a Gaussian kernel with length scale defined by $r_K(\boldsymbol{x}_n)$.

To further enhance the expressiveness of the proposed KNN-KDE model, we extend it to a fully Bayesian procedure which we refer to as VB-KNN-KDE generalizing eq. (3) by parametrizing each observation-specific kernel in terms of a Wishart-distributed precision matrix $\boldsymbol{\Lambda}_n$ with prior distribution

$$p(\boldsymbol{\Lambda}_n) = \mathcal{W}\big(\big(\nu\tau_n\widehat{\boldsymbol{\Sigma}}_n^{(K)}\big)^{-1}, \nu\big), \tag{5}$$

$$p(\tau_n) = \text{Gam}(a, b), \tag{6}$$

such that $\mathbb{E}_p\left[\boldsymbol{\Lambda}_n^{-1}\right] = \tau_n\widehat{\boldsymbol{\Sigma}}_n^{(K)}$. The introduction of the τ_n parameters allows the model to further vary the scale of the prior covariance $\widehat{\boldsymbol{\Sigma}}_n^{(K)}$. This enables flexible scaling of the K-nearest neighbor covariance kernels also potentially enabling local adaptation of scale according to density as suggested for KDEs in Abramson (1982) and Bäcklin et al. (2018).

Directly optimizing the KNN-KDE likelihood (3) based on $\mathbf{X}$ (i.e., $p(\mathbf{X}) = \prod_m^N p(\boldsymbol{x}_m)$) would result in τ_n, or equivalently the norm of $\boldsymbol{\Lambda}_n^{-1}$, collapsing to zero. Thus, in order to perform generalizable inference over the per-observation parameters $\{\boldsymbol{\Lambda}_n, \tau_n\}$ we consider the leave-one-out predictive likelihood also considered in Zhang et al. (2006); Zougab et al. (2014); Bäcklin et al. (2018),

$$p(\mathbf{X} \,|\, \boldsymbol{\Lambda}) = \prod_m^N \left[\frac{1}{N-1}\sum_{n\neq m}^N \mathcal{N}(\boldsymbol{x}_m \,|\, \boldsymbol{x}_n, \boldsymbol{\Lambda}_n^{-1})\right]. \tag{7}$$

We can expand the likelihood (7) to that of a latent variable model by associating every data point $\boldsymbol{x}_n$ with N binary latent variables $\boldsymbol{z}_n = \{z_{nm}\}_m^N$ where $\sum_n^N z_{nm} = 1$ and $z_{nn} = 0$, such that each data point is drawn from one-of-N-minus-one Gaussians,

$$p(\mathbf{X} \,|\, \mathbf{Z}, \boldsymbol{\Lambda}) = \prod_n^N\prod_m^N \mathcal{N}(\boldsymbol{x}_m \,|\, \boldsymbol{x}_n, \boldsymbol{\Lambda}_n^{-1})^{z_{nm}}, \tag{8}$$

$$p(\mathbf{Z}) = \prod_n^N\prod_m^N \left(\frac{1}{N-1}\right)^{z_{nm}}. \tag{9}$$

From a generative perspective these latent variables model which Gaussian was responsible for generating the observed data point $\boldsymbol{x}_m$, and constraining $z_{nn} = 0$ prevents a data point from being responsible for generating itself (as defined by the leave-one-out predictive likelihood). Thus, marginalizing $p(\mathbf{X}, \mathbf{Z} \,|\, \boldsymbol{\Lambda})$ over $\mathbf{Z}$ recovers eq. (7).

To perform inference over the distributions of the latent parameters $\mathbf{Z}$ and non-latent parameters $\boldsymbol{\Lambda} = \{\boldsymbol{\Lambda}_1, \boldsymbol{\Lambda}_2, \ldots, \boldsymbol{\Lambda}_N\}$, $\boldsymbol{\tau} = \{\tau_1, \tau_2, \ldots, \tau_N\}$, we turn to *variational Bayesian* (VB) inference (Blei et al., 2017; Bishop, 2006). VB is a well-established method used to obtain an analytical approximation of the true posterior $p(\mathbf{Z}, \boldsymbol{\Lambda}, \boldsymbol{\tau} \,|\, \mathbf{X})$ by proposing a family of approximate distributions $q(\mathbf{Z}, \boldsymbol{\Lambda}, \boldsymbol{\tau})$ and analytically optimizing the difference between the true and approximate posteriors, with the measure of dissimilarity typically being the Kullback-Leibler (KL) divergence. The optimal approximate posterior $q^*(\mathbf{Z}, \boldsymbol{\Lambda}, \boldsymbol{\tau})$ is then

$$q^*(\mathbf{Z}, \boldsymbol{\Lambda}, \boldsymbol{\tau}) = \arg\min_q D_{\text{KL}}\left(q(\mathbf{Z}, \boldsymbol{\Lambda}, \boldsymbol{\tau}) \,\|\, p(\mathbf{Z}, \boldsymbol{\Lambda}, \boldsymbol{\tau} \,|\, \mathbf{X})\right). \tag{10}$$

Utilizing a family of variational posteriors which is fully factorized in the parameters, i.e. with

$q(\mathbf{Z}, \mathbf{\Lambda}, \boldsymbol{\tau}) = q(\mathbf{Z})\, q(\mathbf{\Lambda})\, q(\boldsymbol{\tau})$, and applying coordinate ascent optimization rules to derive variational distributions optimizing the KL divergence yields the following variational distribution over $\mathbf{Z}$,

$$q^*(\mathbf{Z}) = \prod_n^N \prod_m^N r_{nm}^{z_{nm}}, \tag{11}$$

$$r_{nm} \triangleq \mathbb{E}_q\left[z_{nm}\right] = \begin{cases} 0 & \text{for } n = m \\ \frac{\exp \rho_{nm}}{\sum_{i \neq n}^N \exp \rho_{im}} & \text{otherwise} \end{cases}, \tag{12}$$

$$\begin{aligned} \rho_{nm} &\triangleq \mathbb{E}_q\left[\ln \mathcal{N}(\boldsymbol{x}_m \,|\, \boldsymbol{x}_n, \mathbf{\Lambda}_n^{-1})\right] \qquad (13) \\ &\propto \frac{1}{2}\left[\mathbb{E}_q\left[\ln |\mathbf{\Lambda}_n|\right] - (\boldsymbol{x}_m - \boldsymbol{x}_n)^\top \mathbb{E}_q\left[\mathbf{\Lambda}_n\right](\boldsymbol{x}_m - \boldsymbol{x}_n)\right]. \end{aligned}$$

The variational posterior $q^*(\mathbf{\Lambda})$ takes the same form as its prior due to conjugacy, with new variational parameters (marked by tildes),

$$q^*(\mathbf{\Lambda}) = \prod_n^N \mathcal{W}(\mathbf{\Lambda}_n \,|\, \widetilde{\mathbf{W}}_n, \widetilde{\nu}_n), \tag{14}$$

$$\widetilde{\mathbf{W}}_n^{-1} = \nu\, \mathbb{E}_q\left[\tau_n\right] \widehat{\mathbf{\Sigma}}_n^{(K)} + \mathbf{R}_n, \tag{15}$$

$$\mathbf{R}_n = \sum_m^N r_{nm}\,(\boldsymbol{x}_m - \boldsymbol{x}_n)\,(\boldsymbol{x}_m - \boldsymbol{x}_n)^\top, \tag{16}$$

$$\widetilde{\nu}_n = \nu + \sum_m^N r_{nm}, \tag{17}$$

and so does the variational posterior $q^*(\boldsymbol{\tau})$,

$$q^*(\boldsymbol{\tau}) = \prod_n^N \mathrm{Gam}(\tau_n \,|\, \widetilde{a}, \widetilde{b}_n), \tag{18}$$

$$\widetilde{a} = a + \frac{\nu D}{2}, \quad \widetilde{b}_n = b + \frac{\nu}{2} \operatorname{Tr}\left[\widehat{\mathbf{\Sigma}}_n^{(K)}\, \mathbb{E}_q\left[\mathbf{\Lambda}_n\right]\right]. \tag{19}$$

These update rules provide interlocked sets of variables which are continually re-estimated until convergence. Convergence may be monitored by the *evidence lower bound* (ELBO) (Blei et al., 2017) of the marginalized predictive likelihood (8), i.e. $\int p(\mathbf{X} \,|\, \mathbf{Z}, \mathbf{\Lambda}, \boldsymbol{\tau})\, p(\mathbf{Z})\, p(\mathbf{\Lambda})\, p(\boldsymbol{\tau})\, \mathrm{d}\mathbf{Z}\, \mathrm{d}\mathbf{\Lambda}\, \mathrm{d}\boldsymbol{\tau} \geq \text{ELBO}$, which can be evaluated directly from expectations of the variational factors (see supplementary material for further details).

The variational distributions optimizing eq. (10) effectively construct posterior KNN covariances through eq. (15) with each contribution in the sum (16) being weighted by a learned, implicit distance metric through eq. (13) (being proportional to an exponentiated Mahalanobis distance in which $\mathbb{E}_q\left[\mathbf{\Lambda}_n\right]$ defines the inverse covariance). Importantly, this enables the posterior to adapt the kernel to be defined in terms of a learned neighborhood given by the learned normalized log-likelihoods (12) as opposed to being strictly formed by the K-nearest neighbors as defined from the prior using the Euclidean neighborhood, see also table 1. This supports a freely varying Mahalanobis distance as discussed but not pursued in Terrell and Scott (1992), providing a computationally feasible framework for realizing such a property.

3.3 Batched Variational Updates

As the kernels within the KNN-KDE likelihood are observation-specific and do not necessarily change smoothly across the input space existing procedures to scale KDE (i.e., using multi-resolution KD-trees and the Fast Gauss Transform) cannot be applied. However, the structure of the variational update rules afford several opportunities for efficient implementation.

Firstly, the variational parameters are interlocked in such a way that it is sufficient to store either $\{\widetilde{\mathbf{W}}_n\}_n^N$ or $\{r_{nm}\}_{n,m}^N$ in memory at any point, and as each r_{nm} is proportional to $\exp \rho_{nm}$, i.e. a Gaussian likelihood, we can expect high sparsity in the latent variables r_{nm} as the likelihood of faraway points under each Gaussian kernel rapidly drops to zero. Storing only the sparsified latent variables leads to significant compute and memory savings.

Secondly, updates to the variational parameters may be computed in a batched fashion and interleaved such that low memory requirements can be maintained during training. Given a batch of indices $\mathcal{B}$ the parameters of $\mathbf{\Lambda}_{n \in \mathcal{B}}$ can be estimated by eqs. (15) and (17) using a $B \times N$ "slice" of the latent variables, $\{r_{nm}\}_{n \in \mathcal{B}}$, where $B = |\mathcal{B}|$. These parameters can in turn be used to update the variational parameters of $\mathbf{\Lambda}_{n \in \mathcal{B}}$, and to produce new unnormalized log-likelihoods $\{\rho'_{nm}\}_{n \in \mathcal{B}}$, which may be normalized using eq. (12).

To correctly normalize the new $\{\rho'_{nm}\}_{n \in \mathcal{B}}$ and renormalize the now invalidated $\{r_{nm}\}_{n \notin \mathcal{B}}$ we maintain the normalization constants needed in eq. (12) defined as $c_m = \sum_n^N \exp \rho_{nm}$, such that after producing $\{\rho'_{nm}\}_{n \in \mathcal{B}}$ the updated normalization constants can be obtained by adding the new contributions and subtracting $\{\rho_{nm}\}_{n \in \mathcal{B}}$, which are recovered by unnormalizing $\{r_{nm}\}_{n \in \mathcal{B}}$, $c'_m = c_m + \sum_{n \in \mathcal{B}} (\exp \rho'_{nm} - c_m r_{nm})$. The correctly normalized r'_{nm} is thereby obtained as,

$$r'_{nm} = \begin{cases} \frac{1}{c'_m} \exp \rho'_{nm} & \text{for } n \in \mathcal{B} \\ \frac{c_m}{c'_m} r_{nm} & \text{for } n \notin \mathcal{B} \end{cases}. \tag{20}$$

The procedure is shown in fig. 1.

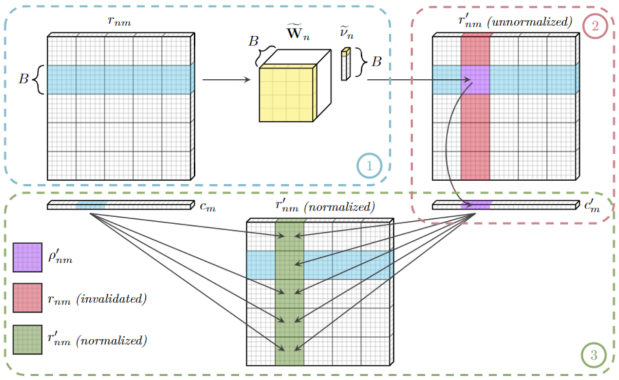

Figure 1: Diagram of the batched update rules. **Step 1:** A row of r_{nm} blocks is used to produce a block of $\mathbf{\Lambda}_n$ parameters. **Step 2:** The $\mathbf{\Lambda}_n$ parameters are used to produce an updated non-sparse ρ'_{nm} block and normalization constants c'_m. **Step 3:** The updated block and the invalidated column are renormalized using c_m and c'_m from the previous and current iterations.

3.4 Predictive Density

In order to perform inference on previously unobserved data points $\widehat{\boldsymbol{x}}$ and latent variables $\widehat{\boldsymbol{z}}$ that accurately reflects the uncertainty in the parameters, we consider the posterior predictive density, that is, the distribution of unobserved data points conditioned on the observed data,

$$
\begin{aligned}
p(\widehat{\boldsymbol{x}} \mid \mathbf{X}) &= \sum_{\widehat{\boldsymbol{z}}} \int p(\widehat{\boldsymbol{x}} \mid \widehat{\boldsymbol{z}}, \mathbf{X}, \mathbf{\Lambda})\, p(\widehat{\boldsymbol{z}})\, p(\mathbf{\Lambda} \mid \mathbf{X})\, \mathrm{d}\mathbf{\Lambda} \\
&\approx \sum_{\widehat{\boldsymbol{z}}} \int p(\widehat{\boldsymbol{x}} \mid \widehat{\boldsymbol{z}}, \mathbf{X}, \mathbf{\Lambda})\, p(\widehat{\boldsymbol{z}})\, q^*(\mathbf{\Lambda} \mid \mathbf{X})\, \mathrm{d}\mathbf{\Lambda} \\
&= \frac{1}{N} \sum_{n}^{N} \mathrm{St}(\widehat{\boldsymbol{x}} \mid \boldsymbol{x}_n, (\widetilde{\nu}_n + 1 - D)\widetilde{\mathbf{W}}_n, \widetilde{\nu}_n + 1 - D),
\end{aligned}
$$

where we use the learned tractable posterior $q^*(\mathbf{\Lambda} \mid \mathbf{X})$, and $\mathrm{St}(\cdot)$ is the multivariate Student's t-distribution.

4 RESULTS AND DISCUSSION

We perform experiments comparing our proposed VB-KNN-KDE methods with existing kernel and KNN density estimators, as well as mixture-models and neural density estimators on both real and synthetic data sets. Throughout our experiments we set the hyperparameters of the scale parameter $\tau_n \sim \mathrm{Gam}(a, b)$ to be non-informative ($a = b = 10^{-8}$) and set other hyperparameters such as ν and K using Bayesian black-box optimization (BO) where not otherwise noted, see supplementary for details. We initialize the latent variables r_{nm} in VB-KNN-KDE to $\frac{1}{K}$ for each point's Euclidean K-nearest neighbors.

4.1 Synthetic Data

Comparing our methods to KNN density estimation is complicated by the inexact density estimates provided by the KNN estimator. While KNN density estimation converges to a valid density in the limit of large data $N \to \infty$, it is otherwise biased, see Zhao and Lai (2022) for a thorough analysis. A less biased estimator is also proposed in Loftsgaarden et al. (1965); Zhao and Lai (2022), multiplying by $K - 1$ in place of K in eq. (2).

To facilitate comparison and quantize the bias of KNN density estimates we consider synthetic 2-dimensional data sets and numerically integrate the inexact KNN density estimates to obtain the normalization constant required to form a valid density. Specifically, we consider 3 data sets "pinwheel", "2spirals", and "checkerboard", some of which are also considered in Grathwohl et al. (2018)[2].

In order to compare to conventional shared-kernel KDEs we also implement variants parametrized by a single scalar bandwidth, a diagonal precision matrix, and a full precision matrix using VB and denote them VB-Scalar-KDE, VB-Diag-KDE, and VB-Full-KDE, respectively. The shared-kernel KDEs behave identically on the synthetic 2-dimensional data sets so we omit VB-Scalar-KDE and VB-Diag-KDE results for these, but we report their performance on tabular data.

In fig. 2 (left panel) we plot the log-likelihood estimates for VB-KNN-KDE, VB-Full-KDE and (numerically normalized) KNN density estimators on the three synthetic data sets as a function of data set size, N, having grid-searched the optimal value of K separately for each estimator and data set size. VB-KNN-KDE consistently outperforms the numerically normalized KNN density.

In fig. 2 (right panel) we plot the densities for $N = 1\,000$ training observations. The estimated densities are used to highlight the qualitative difference between the proposed VB-KNN-KDE and conventional KNN density estimation and shared-covariance kernel KDE (VB-Full-KDE). Notably, it can here be observed that the conventional KNN density estimation approach provides overly fine-grained densities resulting in low density estimates inside the pinwheels, spirals and checkerboards which explains the inferior predictive performance observed in the left panel when comparing the correctly normalized KNN estimator to VB-KNN-KDE and VB-Full-KDE. In contrast, the

[2]We use the same data generation scripts as in the FFJORD repository: `https://github.com/rtqichen/ffjord`.

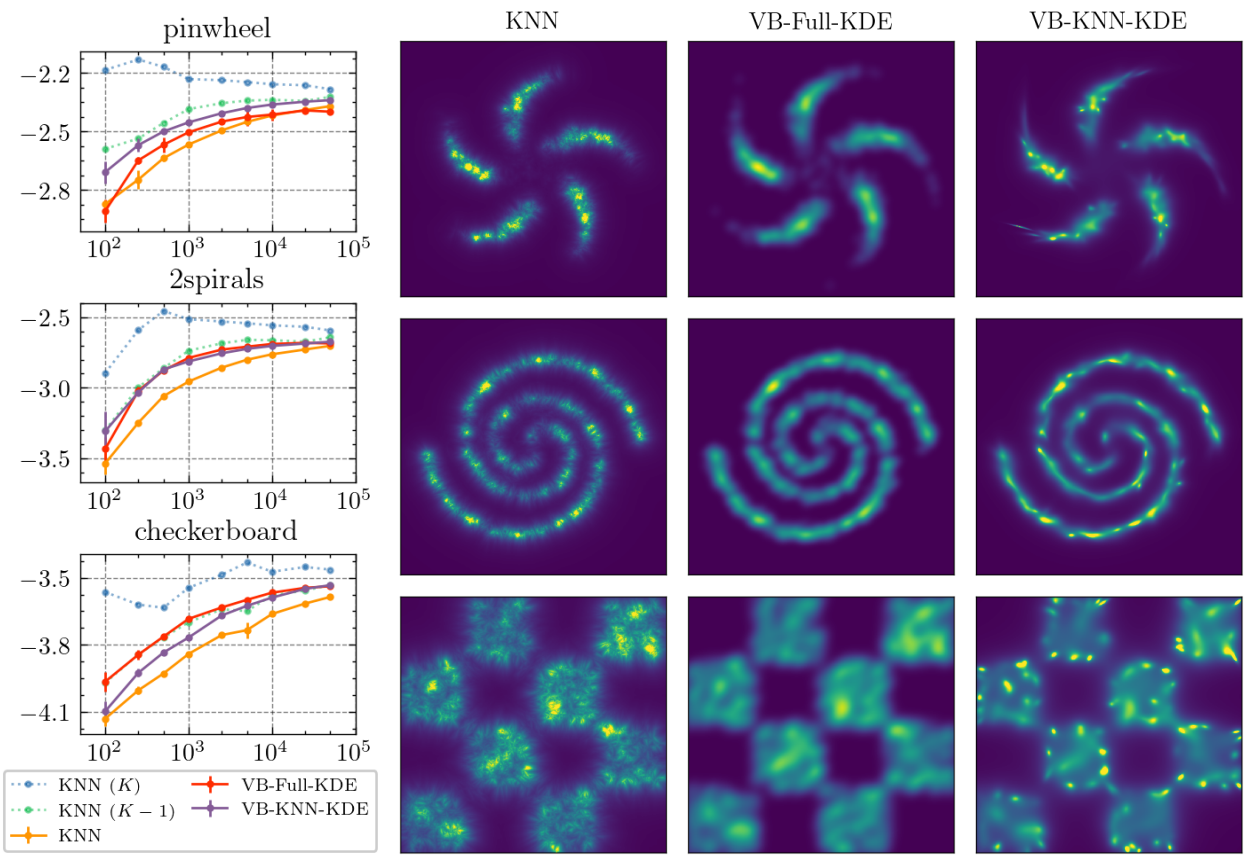

Figure 2: Left panel: Average estimated test log-likelihood on synthetic data sets (higher is better) over 5 runs with varying number of observations N for VB-KNN-KDE, VB-Full-KDE and KNN density estimator. The KNN density estimator given is numerically normalized to be correct and we observe that the true normalized density is significantly overestimated with both K and $K-1$ normalizations for the finite samples considered, and consistently outperformed by VB-KNN-KDE. Right panel: Examples of estimated density for $N = 1\,000$ training observations on synthetic data sets (from the top: "pinwheel", "2spirals", and "checkerboard").

VB-Full-KDE is overly smooth resulting in blurred characterizations of the three densities. VB-KNN-KDE strikes a balance between the two other methods providing a smoother density than conventional KNN density estimation, without over-smoothing as seen with the conventional shared-kernel VB-Full-KDE.

4.2 Tabular Data

We also consider the tabular data sets studied in Papamakarios et al. (2017), available in their preprocessed format from Papamakarios (2018) and compare VB-KNN-KDE to standard KDEs with shared-covariance kernels (VB-Scalar-KDE, VB-Diag-KDE, and VB-Full-KDE), a non-parametric GMM denoted DP-GMM of Hughes and Sudderth (2013)[3], as well as recent neural density estimators on tabular data sets. For completeness we also included the conventional Parzen KDE as implemented in scikit-learn (Pedregosa et al., 2011), where the bandwidth is decided using grid-search on validation data. Results are shown in Table 2, with standard deviation estimated over 3 runs due to shuffling data prior to using batched optimization steps.

To validate our KNN-informed prior we also implement a version of VB-KNN-KDE where the prior of each observation-specific kernel is based on the global empirical covariance, as also considered in Zougab et al. (2014) (referred to as VB-LA-KDE), and also consider simplifications of VB-KNN-KDE such as not adapting the scale of the K-nearest covariance (VB-KNN-KDE$_{\tau_n=1}$), not adapting the posterior distributions of $\mathbf{\Lambda}_n$ (VB-KNN-KDE$_{\nu\to\infty}$), and not allowing any posterior adaptation (VB-KNN-KDE$^{\tau_n=1}_{\nu\to\infty}$), which corresponds directly to eq. (3).

We also compare to prominent neural density estimation works, including FFJORD, MAF, MADE, DDE, B-NAF, and DiffFlow. The results are also shown

[3] Available from `https://github.com/bnpy/bnpy`.

	POWER	GAS	HEPMASS	MINIBOONE	BSDS300
#dimensions	*6*	*8*	*21*	*43*	*63*
#train/val./test	*1660k/184k/205k*	*852k/95k/105k*	*315k/65k/175k*	*30k/3k/4k*	*1000k/50k/250k*
PARZEN	-0.59	6.08	-25.574	-35.16	104.71
DP-GMM	-0.07 ± 0.01	8.11 ± 0.37	-21.45 ± 0.14	-16.86 ± 0.04	161.50 ± 0.87
VB-SCALAR-KDE	-0.59 ± 0.001	4.23 ± 1.47	-25.63 ± 0.01	-37.04 ± 0.04	97.93 ± 0.01
VB-DIAG-KDE	-0.14 ± 0.001	10.10 ± 0.38	-24.17 ± 0.02	-34.56 ± 0.01	97.95 ± 0.26
VB-FULL-KDE	-0.10 ± 0.001	10.85 ± 0.12	-23.28 ± 0.02	-28.16 ± 0.01	112.87 ± 0.29
VB-LA-KDE	0.06 ± 0.001	12.61 ± 0.05	-23.96 ± 0.02	-25.73 ± 0.01	156.30 ± 0.21
VB-KNN-KDE$^{\tau_n=1}_{\nu\to\infty}$	0.12	12.49	-22.70	-15.63	151.21
VB-KNN-KDE$_{\nu\to\infty}$	0.19 ± 0.001	12.71 ± 0.01	-23.31 ± 0.01	-20.60 ± 0.01	154.35 ± 0.08
VB-KNN-KDE$_{\tau_n=1}$	0.17 ± 0.001	12.74 ± 0.01	-23.23 ± 0.01	-19.06 ± 0.01	165.07 ± 0.02
VB-KNN-KDE	0.22 ± 0.01	12.86 ± 0.02	-23.17 ± 0.01	-18.52 ± 0.01	165.29 ± 0.08
FFJORD†	0.46 ± 0.01	8.59 ± 0.12	-14.92 ± 0.08	-10.43 ± 0.04	157.40 ± 0.19
MAF†	0.24 ± 0.01	10.08 ± 0.02	-17.70 ± 0.02	-11.75 ± 0.44	155.69 ± 0.28
MADE†	-3.08 ± 0.03	3.56 ± 0.04	-20.98 ± 0.02	-15.59 ± 0.50	148.85 ± 0.28
MADE MOG†	0.40	8.47	-15.15	-12.27	153.71
DDE	0.97 ± 0.18	9.73 ± 1.14	-11.30 ± 0.16	-6.94 ± 1.81	–
B-NAF	0.61	12.06	-14.71	-8.95	157.36
DIFFFLOW	0.62	11.96	-15.09	-8.86	157.73

Table 2: Results of density estimation reported as log-likelihoods (in nats, higher is better) on validation sets for tabular data. †Results for FFJORD (Grathwohl et al., 2018), MAF (Papamakarios et al., 2017) and MADE (Germain et al., 2015) are from Grathwohl et al. (2018). Data set highest log-likelihood in green and lowest in red.

in Table 2. We note that DDE obtains the highest log-likelihoods on the test data for 3 of the 5 considered data sets (POWER, HEPMASS and MINIBOONE), whereas VB-KNN-KDE performs best on GAS and BSDS300. Neither the dimensionality nor data set size indicate whether neural density estimators or KNN-KDE approaches will perform best. We further observe that VB-KNN-KDE generally outperforms VB-KNN-KDE$_{\nu\to\infty}$ and VB-KNN-KDE$_{\tau_n=1}$ and that it outperforms VB-LA-KDE. Notably, the VB-KNN-KDE$^{\tau_n=1}_{\nu\to\infty}$ that defines the KNN covariance kernel with no posterior adaptation is despite its simplicity also performing well. For all data sets we observe inferior performances when using shared-kernel KDEs.

When scaling to larger data sets maintaining sufficient sparsity in the latent variables r_{nm} becomes crucial. For all considered data sets we found that setting a sparsity cutoff of 10^{-7} was sufficient to perform tractable inference, and did not impact the final log-likelihood for all $N \leq 50\,000$. We find that the resulting amount of non-zero r_{nm} becomes log-linear in the size of data, N, see fig. 3.

While the developed VB-KNN-KDE procedure favorably scales in memory in terms of the sparsity of the responsibilities (substantially reducing the $\mathcal{O}(N^2)$), the procedure does, however, require estimating the observation-specific parameter $\widetilde{\mathbf{W}}_n$ from its inverse which scales by $\mathcal{O}\left(D^3\right)$ in computation using Cholesky decomposition. We presently exploited parallel computations on GPUs but for large dimensional analyses the scaling in D may become prohibitive. In the very large D setting, we note that PCA can be used as a preprocessing step to reduce dimensionality exploiting that the density estimation can still be established in the high-dimensional space using that the change of variable to the representation induced by PCA decomposes into a product of the density of the signal space (preserved PCA components) and residual space (removed components) as described in Moghaddam and Pentland (1997).

A benefit of the present framework is that the approach can be used without careful tuning of network architecture as in neural density estimation. The only hyperparameters to tune in the VB-KNN-KDE are the number of neighbors K and degrees of freedom ν. These parameter can either be set as low as possible to the dimensionality of the data or optimized (i.e., using Bayesian optimization) in terms of the value providing best predictions.

We presently used VB inference to account for uncertainty while still providing efficient and scalable inference. However, other inference procedures could be readily invoked such as sampling from the posterior distributions as opposed to taking expectations or maximizing the posterior distribution (MAP). Importantly, as the procedure is based on an objective optimizing the predictive likelihood we can expect overfitting to be less of an issue and therefore even MAP inference as opposed to VB or MCMC sampling procedures to work well.

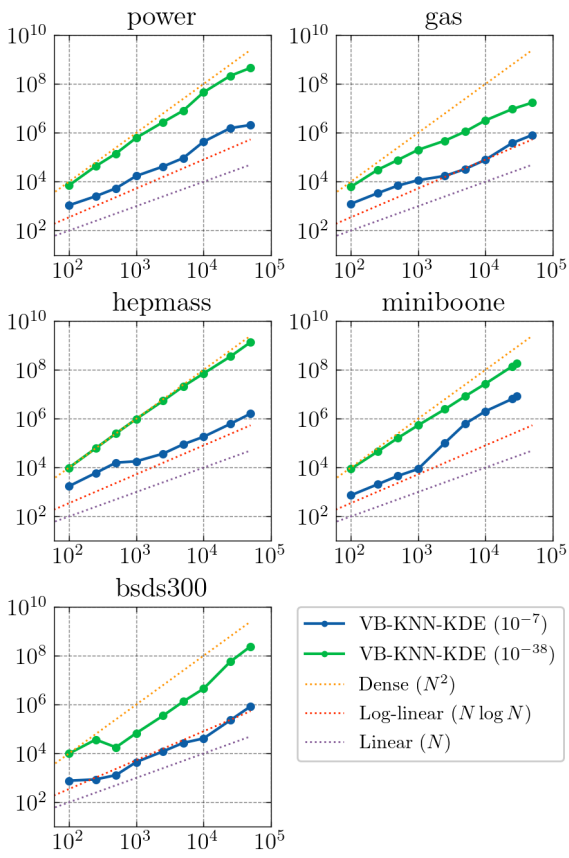

Figure 3: Amount of non-zero latent variables r_{nm} as a function of data set size N. We observe log-linear ($\approx \mathcal{O}(N \log N)$) scaling in memory use when pruning near-zero latent variables.

We presently evaluated our density estimators in terms of (predictive) log-likelihood. However, as discussed in Theis et al. (2016) there are other relevant evaluation metrics, including performance in downstream tasks and evaluating the quality of generated samples. Interestingly, for generative modeling approaches that do not warrant log-likelihood estimates directly, a strategy is to generate samples and use Parzen window KDE, typically relying on a Gaussian kernel which is known to perform poorly facing high-dimensional data (Theis et al., 2016). We also observed poor performance of Parzen-based KDE when exploring our scalable variational inference procedure (i.e., VB-Scalar-KDE) and the extensions to diagonal and full covariance kernels (i.e., VB-Diag-KDE and VB-Full-KDE). Thus, the proposed VB-KNN-KDE framework forms a promising alternative providing more accurate characterizations of the data density.

Additionally, while we considered only the leave-one-out data likelihood our approach could be readily extended to using the leave-K-out data likelihood by forcing appropriate latent variables z_{nm} to zero in eq. (12). Whereas leave-one-out likelihood is the least biased estimator we note that variance introduced by changing training data can be better accounted for using five or ten fold cross-validation (Hastie et al., 2009), implying that leaving more samples out of the data likelihood may improve generalization as well.

While we used the KNN neighborhood directly to form $\mathbf{\Sigma}_n^{(K)}$ and then use Bayesian modeling for further adaptation through the posterior, other approaches could be used to obtain observation-specific covariance matrices based on the KNN neighborhood, such as averaging over the KNN covariance matrices for several values of K, or directly modeling the distribution of KNN covariances using Wishart processes (Wilson and Ghahramani, 2010).

Compared to the VB-LA-KDE specifications considering a fixed prior for the kernel (Zougab et al., 2014), we found the proposed VB-KNN-KDE procedure to perform the best. We attribute this to the importance of adapting the prior to the input space and thereby enhancing the flexibility of the VB-KNN-KDE procedure. Notably, we systematically contrasted the VB-KNN-KDE to the corresponding procedure fixing the kernel structure to the neighborhood (VB-KNN-KDE$_{\nu\to\infty}$) as well as fixing the scale of the KNN covariance prior (VB-KNN-KDE$_{\tau_n=1}$ and VB-KNN-KDE$_{\tau_n=1}^{\nu\to\infty}$). Whereas these procedures also performed well they were in general inferior to the VB-KNN-KDE. This points to the importance of the posterior adaptation of the KNN covariance prior both in terms of scale and structure of the covariance.

5 CONCLUSION

We proposed the KNN-KDE framework parametrizing KDE in terms of the K-nearest neighbor covariance, and extend it to the VB-KNN-KDE procedure, which further locally adapts observation-specific kernels to a learned Mahalanobis distance of the local neighborhood. We further devised a scalable $\mathcal{O}(N \log N)$ inference procedure exploring sparsity of the learned latent variables. We contrasted the procedure to conventional KNN and KDE procedures, as well as other kernel specifications formulated using VB inference, and found that VB-KNN-KDE exhibited favorable performance. We further contrasted the approach to recent neural density estimation procedures finding that the VB-KNN performed best on two of the considered real data sets. The developed VB-KNN-KDE serves as an easy to use, flexible KDE approach that admits considerably more flexible density estimation than conventional KNN- and kernel-based density estimation while retaining favorable scaling properties.

Acknowledgements

This work is partly funded by WS Audiology and the Innovation Fund Denmark (IFD) under File No. 3129-00075B.

References

Abramson, I. S. (1982). On bandwidth variation in kernel estimates-a square root law. *The annals of Statistics*, pages 1217–1223.

Bäcklin, C. L., Andersson, C., and Gustafsson, M. G. (2018). Self-tuning density estimation based on bayesian averaging of adaptive kernel density estimations yields state-of-the-art performance. *Pattern Recognition*, 78:133–143.

Bakshy, E., Dworkin, L., Karrer, B., Kashin, K., Letham, B., Murthy, A., and Singh, S. (2018). Ae: A domain-agnostic platform for adaptive experimentation. In *Conference on Neural Information Processing Systems*, pages 1–8.

Belaid, N., Adjabi, S., Kokonendji, C. C., and Zougab, N. (2018). Bayesian adaptive bandwidth selector for multivariate discrete kernel estimator. *Communications in Statistics-Theory and Methods*, 47(12):2988–3001.

Bigdeli, S. A., Lin, G., Portenier, T., Dunbar, L. A., and Zwicker, M. (2020). Learning generative models using denoising density estimators. *arXiv preprint arXiv:2001.02728*.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer-Verlag, Berlin, Heidelberg.

Blei, D. M. and Jordan, M. I. (2004). Variational methods for the dirichlet process. In *Proceedings of the twenty-first international conference on Machine learning*, page 12.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877.

Bowman, A. W. and Azzalini, A. (1997). *Applied smoothing techniques for data analysis: the kernel approach with S-Plus illustrations*, volume 18. OUP Oxford.

Breiman, L., Meisel, W., and Purcell, E. (1977). Variable kernel estimates of multivariate densities. *Technometrics*, 19(2):135–144.

Brewer, M. J. (2000). A bayesian model for local smoothing in kernel density estimation. *Statistics and Computing*, 10(4):299–309.

Bryant, M. and Sudderth, E. B. (2012). Truly nonparametric online variational inference for hierarchical dirichlet processes. In *Advances in Neural Information Processing Systems*, pages 2699–2707.

Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. (2018). Neural ordinary differential equations. In *Advances in neural information processing systems*, pages 6571–6583.

Comaniciu, D. (2003). An algorithm for data-driven bandwidth selection. *IEEE Transactions on pattern analysis and machine intelligence*, 25(2):281–288.

Comaniciu, D., Ramesh, V., and Meer, P. (2001). The variable bandwidth mean shift and data-driven scale selection. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 1, pages 438–445. IEEE.

Dasgupta, S. and Kpotufe, S. (2014). Optimal rates for k-nn density and mode estimation. In *NIPS*, volume 27, pages 2555–2563.

De Cao, N., Aziz, W., and Titov, I. (2020). Block neural autoregressive flow. In *Uncertainty in artificial intelligence*, pages 1263–1273. PMLR.

Devroye, L. P. and Wagner, T. J. (1977). The strong uniform consistency of nearest neighbor density estimates. *The Annals of Statistics*, pages 536–540.

Dinh, L., Krueger, D., and Bengio, Y. (2014). Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. (2016). Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*.

Ferguson, T. S. (1973). A bayesian analysis of some nonparametric problems. *The annals of statistics*, pages 209–230.

Gangopadhyay, A. and Cheung, K. (2002). Bayesian approach to the choice of smoothing parameter in kernel density estimation. *Journal of nonparametric statistics*, 14(6):655–664.

Gao, W., Oh, S., and Viswanath, P. (2017). Density functional estimators with k-nearest neighbor bandwidths. In *2017 IEEE International Symposium on Information Theory (ISIT)*, pages 1351–1355. IEEE.

Genz, A. and Bretz, F. (2009). *Computation of multivariate normal and t probabilities*, volume 195. Springer Science & Business Media.

Germain, M., Gregor, K., Murray, I., and Larochelle, H. (2015). Made: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning*, pages 881–889.

Grathwohl, W., Chen, R. T., Bettencourt, J., Sutskever, I., and Duvenaud, D. (2018). Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*.

Gray, A. G. and Moore, A. W. (2003). Nonparametric density estimation: Toward computational tractability. In *Proceedings of the 2003 SIAM International Conference on Data Mining*, pages 203–211. SIAM.

Greengard, L. and Strain, J. (1991). The fast gauss transform. *SIAM Journal on Scientific and Statistical Computing*, 12(1):79–94.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media.

Hughes, M. C. and Sudderth, E. B. (2013). Memoized online variational inference for Dirichlet process mixture models. In *Neural Information Processing Systems (NIPS)*.

Jain, S. and Neal, R. M. (2004). A split-merge markov chain monte carlo procedure for the dirichlet process mixture model. *Journal of computational and Graphical Statistics*, 13(1):158–182.

Kingma, D. P. and Dhariwal, P. (2018). Glow: Generative flow with invertible 1x1 convolutions. In *Advances in neural information processing systems*, pages 10215–10224.

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

Lang, D., Klaas, M., and de Freitas, N. (2005). Empirical testing of fast kernel density estimation algorithms.

Latecki, L. J., Lazarevic, A., and Pokrajac, D. (2007). Outlier detection with kernel density functions. In *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, pages 61–75. Springer.

Loftsgaarden, D. O., Quesenberry, C. P., et al. (1965). A nonparametric estimate of a multivariate density function. *The Annals of Mathematical Statistics*, 36(3):1049–1051.

Moghaddam, B. and Pentland, A. (1997). Probabilistic visual learning for object representation. *IEEE Transactions on pattern analysis and machine intelligence*, 19(7):696–710.

Papamakarios, G. (2018). Preprocessed datasets for maf experiments.

Papamakarios, G., Pavlakou, T., and Murray, I. (2017). Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pages 2338–2347.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Rasmussen, C. E. (2000). The infinite gaussian mixture model. In *Advances in neural information processing systems*, pages 554–560.

Remes, S., Heinonen, M., and Kaski, S. (2017). Nonstationary spectral kernels. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30, pages 4642–4651. Curran Associates, Inc.

Scott, D. W. (2015). *Multivariate density estimation: theory, practice, and visualization*. John Wiley & Sons.

Sheather, S. J. and Jones, M. C. (1991). A reliable data-based bandwidth selection method for kernel density estimation. *Journal of the Royal Statistical Society: Series B (Methodological)*, 53(3):683–690.

Terrell, G. R. and Scott, D. W. (1992). Variable kernel density estimation. *The Annals of Statistics*, pages 1236–1265.

Theis, L., van den Oord, A., and Bethge, M. (2016). A note on the evaluation of generative models. In *International Conference on Learning Representations*.

Tipping, M. E. and Bishop, C. M. (1999). Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622.

Von Luxburg, U. and Alamgir, M. (2013). Density estimation from unweighted k-nearest neighbor graphs: a roadmap. In *Advances in Neural Information Processing Systems*, pages 225–233.

Wang, Z. and Jegelka, S. (2017). Max-value entropy search for efficient bayesian optimization. In *International Conference on Machine Learning*, pages 3627–3635. PMLR.

Wilson, A. G. and Ghahramani, Z. (2010). Generalised wishart processes. *arXiv preprint arXiv:1101.0240*.

Zhang, Q. and Chen, Y. (2021). Diffusion normalizing flow. *Advances in Neural Information Processing Systems*, 34:16280–16291.

Zhang, X., King, M. L., and Hyndman, R. J. (2006). A bayesian approach to bandwidth selection for multivariate kernel density estimation. *Computational Statistics & Data Analysis*, 50(11):3009–3031.

Zhao, P. and Lai, L. (2022). Analysis of knn density estimation. *IEEE Transactions on Information Theory*, 68(12):7971–7995.

Ziane, Y., Adjabi, S., and Zougab, N. (2015). Adaptive bayesian bandwidth selection in asymmetric kernel density estimation for nonnegative heavy-tailed data. *Journal of Applied Statistics*, 42(8):1645–1658.

Zougab, N., Adjabi, S., and Kokonendji, C. C. (2014). Bayesian estimation of adaptive bandwidth matrices in multivariate kernel density estimation. *Computational Statistics & Data Analysis*, 75:28–38.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Not Applicable]
   (b) Complete proofs of all theoretical results. [Not Applicable]
   (c) Clear explanations of any assumptions. [Not Applicable]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Yes]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Supplementary Materials

6 EXPERIMENTAL DETAILS

All our experiments are implemented in PyTorch and run on a single NVIDIA A100 40GB GPU. When training our models we monitor the predictive log-likelihood on validation data sets, and stop training when the validation log-likelihood decreases or plateaus. We use a batch size of $B = 50\,000$ on all data sets except for BSDS300 where we use a smaller batch size of $B = 10\,000$. While the batch size does influence the optimization procedure, we did not observe the final converged solution to be significantly affected by the choice of batch size.

6.1 Data Availability

The five considered tabular data sets are the same pre-processed data sets as considered in Grathwohl et al. (2018) and Papamakarios et al. (2017), which are available from `https://zenodo.org/records/1161203#.Wmtf_XVl8eN` and licensed under CC BY 4.0. The synthetic data sets are also from Grathwohl et al. (2018), and the code to generate these is MIT licensed.

6.2 Efficient Hyperparameter Tuning by Bayesian Optimization

When scaling to very large data sets, it can become prohibitively expensive to exhaustively search for hyperparameters using grid search. To avoid this, we use Bayesian optimization (BO) with the Ax[4] (Bakshy et al., 2018) framework to search for hyperparameters on subsets of the larger data sets that are then used as hyperparameters on the full-size data sets. We find this approach to be robust to the noise introduced by subsampling, and that optimal hyperparameters do not change drastically as the size of subsets increases. The BO loop uses the max-value entropy search (MES) (Wang and Jegelka, 2017) acquisition function to propose trial hyperparameters, which are used to fit and evaluate model on random subsets of the data.

We use BO to select hyperparameters for all considered models (KNN-KDE, VB-KNN-KDE, VB-LA-KDE, etc.) which include the size of the KNN neighborhood K, and the degrees-of-freedom ν for the Bayesian models.

In fig. 4, all three runs sample subsets with $N = 50\,000$ observations and shows that BO consistently finds the same optima even when trial evaluations are noisy due to initialization of the BO procedure and sampling of subsets. In fig. 5, BO is used with subsets of $N = \{10\,000, 50\,000, 100\,000\}$ observations (increasing from top to bottom), and shows that the optimal hyperparameters do not change drastically as the training set size increases.

[4] `https://ax.dev`

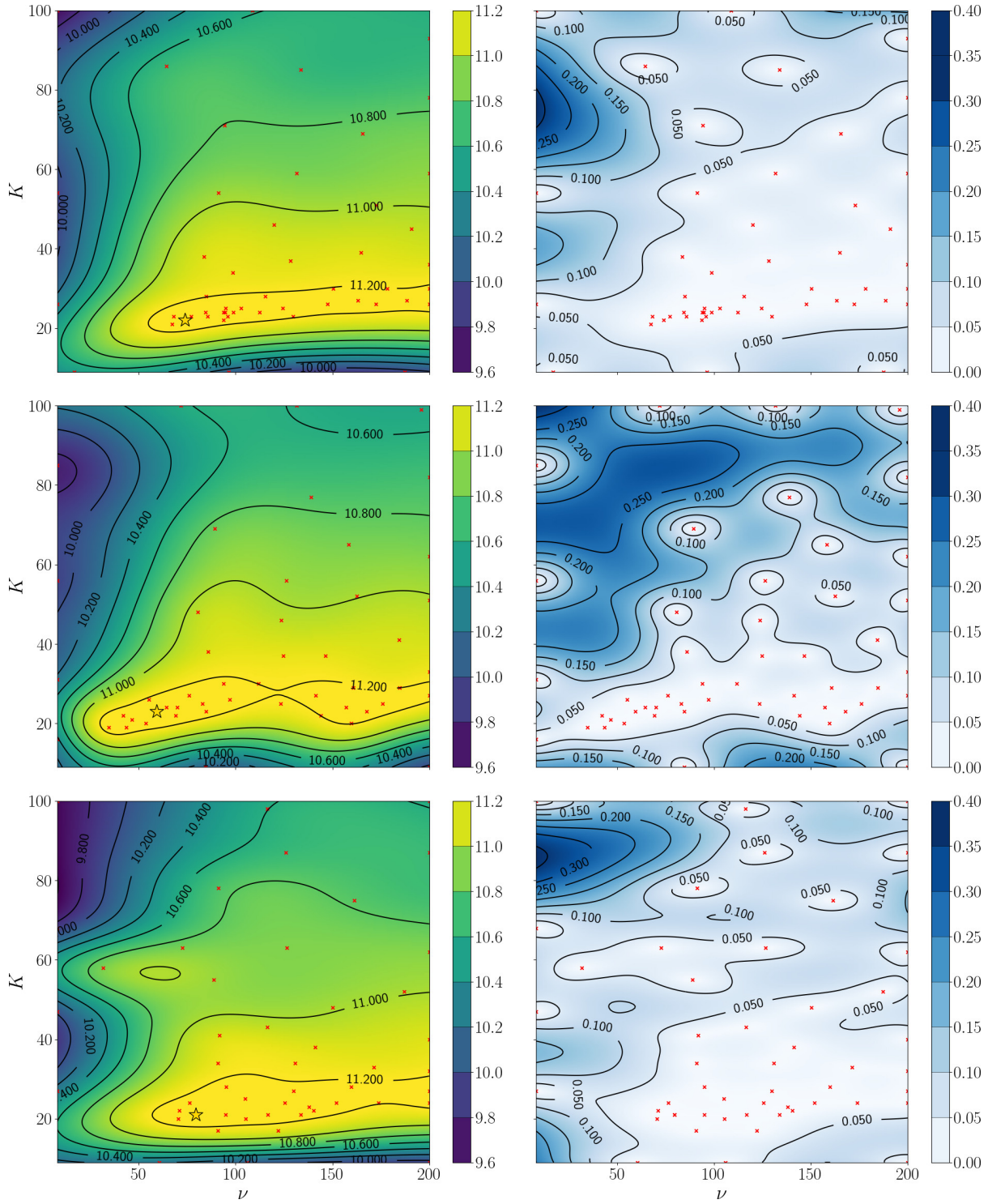

Figure 4: Estimated hyperparameter landscape of GAS based on 3 separate runs of Bayesian optimization using the MES acquisition function. In each trial a VB-KNN-KDE model is fitted to a new $N = 50\ 000$ subset — differences between runs are due to data subsampling and sampling of the acquisition function. Left: The estimated final average log-likelihood of a VB-KNN-KDE model given hyperparameters K and ν. Right: The standard deviation of the estimate.

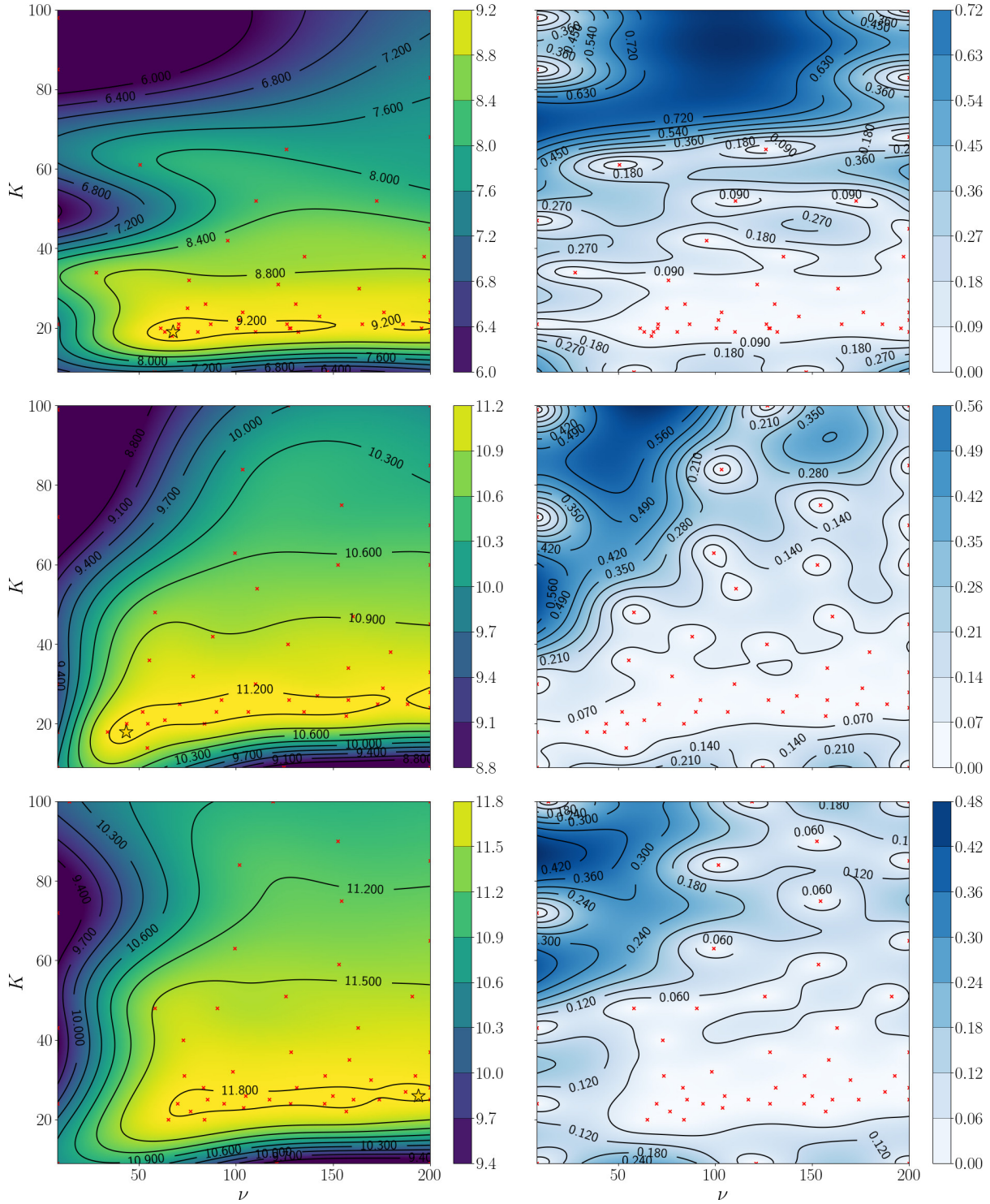

Figure 5: Estimated hyperparameter landscape of GAS as in fig. 4, but for increasing (top to bottom) sizes of data subsets $N = \{10\,000, 50\,000, 100\,000\}$. While the estimates of the hyperparameter landscapes vary away from the optima, the locations of the optima remain largely stationary as the size of the sampled subsets increases.

7 ADDITIONAL DERIVATIONS

The variational update rules for the shared covariance models (VB-Full-KDE, VB-Diag-KDE, and VB-Scalar-KDE) are given in Section 7.1, and the predictive likelihoods are given in Section 7.2. Detailed descriptions of the computation of the associated evidence lower bounds (ELBO) are given in Section 7.3.

7.1 Update Rules for Alternative KDEs Specifications

The model proposed in Zougab et al. (2014) specifies the same data likelihood as VB-KNN-KDE, but parameterizes each $\boldsymbol{\Lambda}_n$ in terms of the empirical covariance,

$$p(\boldsymbol{\Lambda}_n) = \mathcal{W}\big(\big(\nu\widehat{\boldsymbol{\Sigma}}\big)^{-1}, \nu\big) \tag{21}$$

where $\widehat{\boldsymbol{\Sigma}}$ is the empirical covariance matrix. Since the purpose of comparing to this model is to validate the effect of the KNN-based prior, we augment this model by imposing observation-specific scale parameters τ_n as in VB-KNN-KDE, and impose a learned shared covariance matrix instead of estimating it empirically.

These modifications bring the representational capacity of the model closer to that of VB-KNN-KDE, such that the most significant difference between the two becomes whether the prior covariance structure is locally adaptive or not. We refer to the resulting model as VB-LA-KDE, which is specified as

$$p(\overline{\boldsymbol{\Sigma}}) = \mathcal{W}\big(\bar{\nu}^{-1}\widehat{\boldsymbol{\Sigma}}, \bar{\nu}\big), \quad p(\tau_n) = \mathrm{Gam}(a, b), \tag{22}$$

$$p(\boldsymbol{\Lambda}_n) = \mathcal{W}\big(\big(\nu\tau_n\overline{\boldsymbol{\Sigma}}\big)^{-1}, \nu\big). \tag{23}$$

The variational posterior distributions become

$$q^*(\mathbf{Z}) = \prod_n^N \prod_m^N r_{nm}^{z_{nm}}, \quad r_{nm} \triangleq \mathbb{E}_q\left[z_{nm}\right] = \frac{\exp\rho_{nm}}{\sum_i^N \exp\rho_{im}}, \tag{24}$$

$$\rho_{nm} \propto \mathbb{E}_q\left[\ln|\boldsymbol{\Lambda}_n|\right] - \left(\boldsymbol{x}_m - \boldsymbol{x}_n\right)^\top \mathbb{E}_q\left[\boldsymbol{\Lambda}_n\right]\left(\boldsymbol{x}_m - \boldsymbol{x}_n\right), \tag{25}$$

$$q^*(\overline{\boldsymbol{\Sigma}}) = \mathcal{W}(\overline{\boldsymbol{\Sigma}} \,|\, \widetilde{\overline{\mathbf{W}}}, \widetilde{\bar{\nu}}), \tag{26}$$

$$q^*(\boldsymbol{\Lambda}) = \prod_n^N \mathcal{W}(\boldsymbol{\Lambda}_n \,|\, \widetilde{\mathbf{W}}_n, \widetilde{\nu}_n), \tag{27}$$

$$q^*(\boldsymbol{\tau}) = \prod_n^N \mathrm{Gam}(\tau_n \,|\, \widetilde{a}, \widetilde{b}_n), \tag{28}$$

$$\tag{29}$$

with corresponding update rules,

$$\widetilde{\mathbf{W}}_n^{-1} = \nu\,\mathbb{E}_q\left[\tau_n\right]\mathbb{E}_q\left[\overline{\boldsymbol{\Sigma}}\right] + \mathbf{R}_n, \quad \widetilde{\nu}_n = \nu + \sum_m^N r_{nm}, \tag{30}$$

$$\widetilde{\overline{\mathbf{W}}}^{-1} = \bar{\nu}\widehat{\boldsymbol{\Sigma}}^{-1} + \nu\sum_n^N \mathbb{E}_q\left[\tau_n\right]\mathbb{E}_q\left[\boldsymbol{\Lambda}_n\right], \quad \widetilde{\bar{\nu}} = \bar{\nu} + N\nu, \tag{31}$$

$$\widetilde{a} = a + \frac{\nu D}{2}, \quad \widetilde{b}_n = b + \frac{\nu}{2}\,\mathrm{Tr}\left[\mathbb{E}_q\left[\overline{\boldsymbol{\Sigma}}\right]\mathbb{E}_q\left[\boldsymbol{\Lambda}_n\right]\right]. \tag{32}$$

For VB-Full-KDE a single covariance matrix parameterizes the KDE,

$$p(\boldsymbol{\Lambda}) = \mathcal{W}(\Big(\nu\widehat{\boldsymbol{\Sigma}}\Big)^{-1}, \nu), \tag{33}$$

$$p(\mathbf{X} \,|\, \mathbf{Z}, \boldsymbol{\Lambda}) = \prod_n^N \prod_m^N \mathcal{N}(\boldsymbol{x}_m \,|\, \boldsymbol{x}_n, \boldsymbol{\Lambda}^{-1})^{z_{nm}} \tag{34}$$

and we obtain the following variational distributions and update rules

$$q^*(\mathbf{Z}) = \prod_n^N \prod_m^N r_{nm}^{z_{nm}}, \quad r_{nm} \triangleq \mathbb{E}\left[z_{nm}\right] = \frac{\exp \rho_{nm}}{\sum_i^N \exp \rho_{im}}, \tag{35}$$

$$\rho_{nm} \propto \mathbb{E}_q\left[\ln |\mathbf{\Lambda}|\right] - \left(\boldsymbol{x}_m - \boldsymbol{x}_n\right)^\top \mathbb{E}_q\left[\mathbf{\Lambda}\right]\left(\boldsymbol{x}_m - \boldsymbol{x}_n\right), \tag{36}$$

$$q^*(\mathbf{\Lambda}) = \mathcal{W}(\mathbf{\Lambda} \,|\, \widetilde{\mathbf{W}}, \widetilde{\nu}), \tag{37}$$

$$\widetilde{\mathbf{W}}^{-1} = \nu \widehat{\mathbf{\Sigma}} + \sum_n^N \sum_m^N r_{nm} (\boldsymbol{x}_m - \boldsymbol{x}_n)(\boldsymbol{x}_m - \boldsymbol{x}_n)^\top, \quad \widetilde{\nu} = \nu + N. \tag{38}$$

For VB-Diag-KDE the covariance is restricted to a diagonal matrix with each diagonal entry λ_i^2 modelled as a gamma distribution with prior variance σ_i^2,

$$\mathbf{\Lambda} = \operatorname{diag}(\boldsymbol{\lambda}^2), \quad p(\lambda_i^2) = \operatorname{Gam}(\nu, \nu\sigma_i^2) \tag{39}$$

$$p(\mathbf{X} \,|\, \mathbf{Z}, \mathbf{\Lambda}) = \prod_n^N \prod_m^N \mathcal{N}(\boldsymbol{x}_m \,|\, \boldsymbol{x}_n, \mathbf{\Lambda}^{-1})^{z_{nm}} = \prod_n^N \prod_m^N \prod_i^D \mathcal{N}(x_{mi} \,|\, x_{ni}, \lambda_i^{-2})^{z_{nm}}. \tag{40}$$

The variational distributions and update rules are

$$q^*(\mathbf{Z}) = \prod_n^N \prod_m^N r_{nm}^{z_{nm}}, \quad r_{nm} \triangleq \mathbb{E}\left[z_{nm}\right] = \frac{\exp \rho_{nm}}{\sum_i^N \exp \rho_{im}}, \tag{41}$$

$$\rho_{nm} \propto \sum_i^D \mathbb{E}_q\left[\ln \lambda_i^2\right] - \left(x_{mi} - x_{ni}\right)^2 \mathbb{E}_q\left[\lambda_i^2\right], \tag{42}$$

$$q^*(\lambda_i^2) = \operatorname{Gam}(\lambda_i^2 \,|\, \widetilde{\nu}, \widetilde{w}_i), \tag{43}$$

$$\widetilde{\nu} = \nu + \frac{N}{2}, \quad \widetilde{w}_i = \nu\sigma_i^2 + \frac{1}{2}\sum_n^N \sum_m^N r_{nm}\left(x_{mi} - x_{ni}\right)^2. \tag{44}$$

VB-Scalar-KDE restricts VB-Diag-KDE to a shared distribution along the diagonal,

$$\mathbf{\Lambda} = \operatorname{diag}(\lambda^2), \quad p(\lambda^2) = \operatorname{Gam}(\nu, \nu\sigma^2), \tag{45}$$

$$p(\mathbf{X} \,|\, \mathbf{Z}, \mathbf{\Lambda}) = \prod_n^N \prod_m^N \mathcal{N}(\boldsymbol{x}_m \,|\, \boldsymbol{x}_n, \mathbf{\Lambda}^{-1})^{z_{nm}} = \prod_n^N \prod_m^N \prod_i^D \mathcal{N}(x_{mi} \,|\, x_{ni}, \lambda^{-2})^{z_{nm}}. \tag{46}$$

The variational distributions and update rules are

$$q^*(\mathbf{Z}) = \prod_n^N \prod_m^N r_{nm}^{z_{nm}}, \quad r_{nm} \triangleq \mathbb{E}\left[z_{nm}\right] = \frac{\exp \rho_{nm}}{\sum_i^N \exp \rho_{im}}, \tag{47}$$

$$\rho_{nm} \propto D\,\mathbb{E}_q\left[\ln \lambda^2\right] - \mathbb{E}_q\left[\lambda^2\right] \sum_i^D \left(x_{mi} - x_{ni}\right)^2, \tag{48}$$

$$q^*(\lambda^2) = \operatorname{Gam}(\lambda^2 \,|\, \widetilde{\nu}, \widetilde{w}), \tag{49}$$

$$\widetilde{\nu} = \nu + \frac{ND}{2}, \quad \widetilde{w} = \nu\sigma^2 + \frac{1}{2}\sum_i^D \sum_n^N \sum_m^N r_{nm}\left(x_{mi} - x_{ni}\right)^2. \tag{50}$$

7.2 Predictive Likelihoods for Alternative KDEs Specifications

The predictive likelihood for VB-LA-KDE is identical to that of VB-KNN-KDE, as they share the same posterior distributions.

For VB-Full-KDE the predictive likelihood is a mixture of multivariate Student's t-distributions with a shared covariance,

$$p(\widehat{\boldsymbol{x}} \,|\, \mathbf{X}) = \frac{1}{N}\sum_n^N \operatorname{St}(\widehat{\boldsymbol{x}} \,|\, \boldsymbol{x}_n, (\widetilde{\nu} + 1 - D)\widetilde{\mathbf{W}}, \widetilde{\nu} + 1 - D). \tag{51}$$

VB-Diag-KDE becomes a mixture of products of independent univariate Student's t-distributions,

$$p(\widehat{\boldsymbol{x}} \,|\, \mathbf{X}) = \frac{1}{N} \sum_{n}^{N} \prod_{i}^{D} \mathrm{St}(\widehat{x}_i \,|\, x_{ni}, \frac{\widetilde{\nu}_i}{\widetilde{w}_i}, 2\widetilde{\nu}). \tag{52}$$

VB-Scalar-KDE becomes a mixture of multivariate Student's t-distributions with a diagonal covariance matrix of identical elements,

$$p(\widehat{\boldsymbol{x}} \,|\, \mathbf{X}) = \frac{1}{N} \sum_{n}^{N} \mathrm{St}(\widehat{\boldsymbol{x}} \,|\, \boldsymbol{x}_n, \mathrm{diag}(\sqrt{\frac{\widetilde{\nu}}{\widetilde{w}}}), 2\widetilde{\nu}). \tag{53}$$

7.3 Lower Bound on Marginal Likelihood

A lower bound on the marginal likelihood (evidence), the ELBO, can be evaluated directly from expectations with respect to the variational factors and used to monitor for convergence during model fitting. These expectations are given below for the three considered KDE models.

Let $\Gamma_D(x) = \pi^{\frac{D(D-1)}{4}} \prod_{i=1}^{D} \Gamma\left(x + \frac{1-i}{2}\right)$ be the multivariate gamma function and $\mathrm{H}[\cdot]$ the entropy operator. Any unevaluated expectations are standard results for normal, Wishart and gamma distributions explicitly given in Appendix B of Bishop (2006).

7.3.1 ELBO for VB-KNN-KDE

$$\begin{aligned}\mathcal{L} = {} & \mathbb{E}_q[\ln p(\mathbf{X} \,|\, \mathbf{Z}, \boldsymbol{\Lambda})] + \mathbb{E}_q[\ln p(\mathbf{Z})] + \mathbb{E}_q[\ln p(\boldsymbol{\Lambda} \,|\, \boldsymbol{\tau})] + \mathbb{E}_q[\ln p(\boldsymbol{\tau})] \\ & - \mathbb{E}_q[\ln q^*(\mathbf{Z})] - \mathbb{E}_q[\ln q^*(\boldsymbol{\Lambda})] - \mathbb{E}_q[\ln q^*(\boldsymbol{\tau})]\end{aligned} \tag{54}$$

$$\mathbb{E}_q[\ln p(\mathbf{X} \,|\, \mathbf{Z}, \boldsymbol{\Lambda})] = -\frac{ND}{2}\ln(2\pi) + \sum_{n}^{N}\sum_{m}^{N} r_{nm} \ln \rho_{nm} \tag{55}$$

$$= -\frac{ND}{2}\ln(2\pi) + \sum_{m}^{N} \ln c_m + \sum_{n}^{N} r_{nm} \ln r_{nm} \tag{56}$$

$$\mathbb{E}_q[\ln p(\mathbf{Z})] = -N \ln(N-1) \tag{57}$$

$$\begin{aligned}\mathbb{E}_q[\ln p(\boldsymbol{\Lambda} \,|\, \boldsymbol{\tau})] = {} & \sum_{n}^{N} \mathbb{E}_q\left[\ln B\left(\left(\tau_n \nu \widehat{\boldsymbol{\Sigma}}_n^{(K)}\right)^{-1}, \nu\right)\right] + \frac{\nu - D - 1}{2}\mathbb{E}_q[\ln|\boldsymbol{\Lambda}_n|] \\ & - \frac{\nu}{2}\mathbb{E}_q[\tau_n]\,\mathrm{Tr}\left[\widehat{\boldsymbol{\Sigma}}_n^{(K)}\mathbb{E}_q[\boldsymbol{\Lambda}_n]\right]\end{aligned} \tag{58}$$

$$\begin{aligned}\mathbb{E}_q\left[\ln B\left(\left(\tau_n \nu \widehat{\boldsymbol{\Sigma}}_n^{(K)}\right)^{-1}, \nu\right)\right] = {} & \frac{\nu}{2}\left(\ln\left|\widehat{\boldsymbol{\Sigma}}_n^{(K)}\right| + D\,\mathbb{E}_q[\ln\tau_n] + D\ln\nu - D\ln 2\right) \\ & - \ln\Gamma_D\left(\frac{\nu}{2}\right)\end{aligned} \tag{59}$$

$$\mathbb{E}_q[\ln p(\boldsymbol{\tau})] = \sum_{n}^{N} -\ln\Gamma(a_n) + a_n \ln b_n + (a_n - 1)\,\mathbb{E}_q[\ln\tau_n] - b_n\,\mathbb{E}_q[\tau_n] \tag{60}$$

$$\mathbb{E}_q[\ln q^*(\mathbf{Z})] = \sum_{n}^{N}\sum_{m}^{N} r_{nm}\ln r_{nm} \tag{61}$$

$$\mathbb{E}_q[\ln q^*(\boldsymbol{\Lambda})] = \sum_{n}^{N} -\mathrm{H}_q[\boldsymbol{\Lambda}_n] \tag{62}$$

$$\mathbb{E}_q[\ln q^*(\boldsymbol{\tau})] = \sum_{n}^{N} -\mathrm{H}_q[\tau_n] \tag{63}$$

7.3.2 ELBO for VB-LA-KDE

$$\mathcal{L} = \mathbb{E}_q\left[\ln p(\mathbf{X} \mid \mathbf{Z}, \mathbf{\Lambda})\right] + \mathbb{E}_q\left[\ln p(\mathbf{Z})\right] + \mathbb{E}_q\left[\ln p(\mathbf{\Lambda} \mid \overline{\mathbf{\Sigma}}, \boldsymbol{\tau})\right] + \mathbb{E}_q\left[\ln p(\overline{\mathbf{\Sigma}})\right] + \mathbb{E}_q\left[\ln p(\boldsymbol{\tau})\right] \tag{64}$$
$$- \mathbb{E}_q\left[\ln q^*(\mathbf{Z})\right] - \mathbb{E}_q\left[\ln q^*(\mathbf{\Lambda})\right] - \mathbb{E}_q\left[\ln q^*(\overline{\mathbf{\Sigma}})\right] - \mathbb{E}_q\left[\ln q^*(\boldsymbol{\tau})\right]$$

$$\mathbb{E}_q\left[\ln p(\mathbf{X} \mid \mathbf{Z}, \mathbf{\Lambda})\right] = -\frac{ND}{2}\ln(2\pi) + \sum_n^N \sum_m^N r_{nm} \ln \rho_{nm} \tag{65}$$

$$= -\frac{ND}{2}\ln(2\pi) + \sum_m^N \ln c_m + \sum_n^N r_{nm} \ln r_{nm} \tag{66}$$

$$\mathbb{E}_q\left[\ln p(\mathbf{Z})\right] = -N\ln(N-1) \tag{67}$$

$$\mathbb{E}_q\left[\ln p(\mathbf{\Lambda} \mid \overline{\mathbf{\Sigma}}, \boldsymbol{\tau})\right] = \sum_n^N \mathbb{E}_q\left[\ln B\left(\left(\tau_n \nu \overline{\mathbf{\Sigma}}\right)^{-1}, \nu\right)\right] + \frac{\nu - D - 1}{2}\mathbb{E}_q\left[\ln|\mathbf{\Lambda}_n|\right] \tag{68}$$
$$- \frac{\nu}{2}\mathbb{E}_q\left[\tau_n\right]\mathrm{Tr}\left[\mathbb{E}_q\left[\overline{\mathbf{\Sigma}}\right]\mathbb{E}_q\left[\mathbf{\Lambda}_n\right]\right]$$

$$\mathbb{E}_q\left[\ln B\left(\left(\tau_n \nu \overline{\mathbf{\Sigma}}\right)^{-1}, \nu\right)\right] = -\frac{\nu}{2}\left(D\ln 2 - \mathbb{E}_q\left[\ln\left|\overline{\mathbf{\Sigma}}\right|\right] - D\,\mathbb{E}_q\left[\ln\tau_n\right] - D\ln\nu\right) \tag{69}$$
$$- \ln\Gamma_D\left(\frac{\nu}{2}\right)$$

$$\mathbb{E}_q\left[\ln p(\mathbf{\Sigma})\right] = \mathbb{E}_q\left[\ln B\left(\bar{\nu}^{-1}\widehat{\mathbf{\Sigma}}, \bar{\nu}\right)\right] + \frac{\bar{\nu} - D - 1}{2}\mathbb{E}_q\left[\ln\left|\overline{\mathbf{\Sigma}}\right|\right] \tag{70}$$
$$- \frac{\bar{\nu}}{2}\mathrm{Tr}\left[\widehat{\mathbf{\Sigma}}^{-1}\mathbb{E}_q\left[\overline{\mathbf{\Sigma}}\right]\right]$$

$$\mathbb{E}_q\left[\ln B\left(\bar{\nu}^{-1}\widehat{\mathbf{\Sigma}}, \bar{\nu}\right)\right] = -\frac{\bar{\nu}}{2}\left(D\ln 2 + \ln\left|\widehat{\mathbf{\Sigma}}\right| - D\ln\bar{\nu}\right) - \ln\Gamma_D\left(\frac{\bar{\nu}}{2}\right) \tag{71}$$

$$\mathbb{E}_q\left[\ln p(\boldsymbol{\tau})\right] = \sum_n^N -\ln\Gamma\left(a_n\right) + a_n\ln b_n + \left(a_n - 1\right)\mathbb{E}_q\left[\ln\tau_n\right] - b_n\,\mathbb{E}_q\left[\tau_n\right] \tag{72}$$

$$\mathbb{E}_q\left[\ln q^*(\mathbf{Z})\right] = \sum_n^N \sum_m^N r_{nm}\ln r_{nm} \tag{73}$$

$$\mathbb{E}_q\left[\ln q^*(\mathbf{\Lambda})\right] = \sum_n^N -\mathrm{H}_q\left[\mathbf{\Lambda}_n\right] \tag{74}$$

$$\mathbb{E}_q\left[\ln q^*(\overline{\mathbf{\Sigma}})\right] = -\mathrm{H}_q\left[\overline{\mathbf{\Sigma}}\right] \tag{75}$$

$$\mathbb{E}_q\left[\ln q^*(\boldsymbol{\tau})\right] = \sum_n^N -\mathrm{H}_q\left[\tau_n\right] \tag{76}$$

7.3.3 ELBO for VB-Full-KDE

$$\begin{aligned}\mathcal{L} = \mathbb{E}_q\left[\ln p(\mathbf{X} \mid \mathbf{Z}, \mathbf{\Lambda})\right] + \mathbb{E}_q\left[\ln p(\mathbf{Z})\right] + \mathbb{E}_q\left[\ln p(\mathbf{\Lambda})\right] \\ - \mathbb{E}_q\left[\ln q^*(\mathbf{Z})\right] - \mathbb{E}_q\left[\ln q^*(\mathbf{\Lambda})\right]\end{aligned} \tag{77}$$

$$\mathbb{E}_q\left[\ln p(\mathbf{X} \mid \mathbf{Z}, \mathbf{\Lambda})\right] = -\frac{ND}{2}\ln(2\pi) + \sum_n^N \sum_m^N r_{nm} \ln \rho_{nm} \tag{78}$$

$$= -\frac{ND}{2}\ln(2\pi) + \sum_m^N \ln c_m + \sum_n^N r_{nm} \ln r_{nm} \tag{79}$$

$$\mathbb{E}_q\left[\ln p(\mathbf{Z})\right] = -N\ln(N-1) \tag{80}$$

$$\mathbb{E}_q\left[\ln p(\mathbf{\Lambda})\right] = \mathbb{E}_q\left[\ln B\left(\left(\nu\widehat{\mathbf{\Sigma}}\right)^{-1}, \nu\right)\right] + \frac{\nu - D - 1}{2}\mathbb{E}_q\left[\ln|\mathbf{\Lambda}|\right] \tag{81}$$

$$- \frac{\nu}{2}\operatorname{Tr}\left[\widehat{\mathbf{\Sigma}}\,\mathbb{E}_q\left[\mathbf{\Lambda}\right]\right]$$

$$\mathbb{E}_q\left[\ln B\left(\left(\nu\widehat{\mathbf{\Sigma}}\right)^{-1}, \nu\right)\right] = -\frac{\nu}{2}\left(D\ln 2 - \ln\left|\widehat{\mathbf{\Sigma}}\right| - D\ln\nu\right) - \ln\Gamma_D\left(\frac{\nu}{2}\right) \tag{82}$$

$$\mathbb{E}_q\left[\ln q^*(\mathbf{Z})\right] = \sum_n^N \sum_m^N r_{nm}\ln r_{nm} \tag{83}$$

$$\mathbb{E}_q\left[\ln q^*(\mathbf{\Lambda})\right] = -\mathrm{H}_q\left[\mathbf{\Lambda}\right] \tag{84}$$

7.3.4 ELBO for VB-Diag-KDE

$$\begin{aligned}\mathcal{L} = \mathbb{E}_q\left[\ln p(\mathbf{X} \mid \mathbf{Z}, \mathbf{\Lambda})\right] + \mathbb{E}_q\left[\ln p(\mathbf{Z})\right] + \mathbb{E}_q\left[\ln p(\mathbf{\Lambda})\right] \\ - \mathbb{E}_q\left[\ln q^*(\mathbf{Z})\right] - \mathbb{E}_q\left[\ln q^*(\mathbf{\Lambda})\right]\end{aligned} \tag{85}$$

$$\mathbb{E}_q\left[\ln p(\mathbf{X} \mid \mathbf{Z}, \mathbf{\Lambda})\right] = -\frac{ND}{2}\ln(2\pi) + \sum_n^N \sum_m^N r_{nm} \ln \rho_{nm} \tag{86}$$

$$= -\frac{ND}{2}\ln(2\pi) + \sum_m^N \ln c_m + \sum_n^N r_{nm} \ln r_{nm} \tag{87}$$

$$\mathbb{E}_q\left[\ln p(\mathbf{Z})\right] = -N\ln(N-1) \tag{88}$$

$$\mathbb{E}_q\left[\ln p(\mathbf{\Lambda})\right] = \sum_i^D -\ln\Gamma(\nu) + \nu\left(\ln\nu + \ln\sigma_i^2\right) + (\nu - 1)\,\mathbb{E}_q\left[\ln\lambda_i^2\right] \tag{89}$$

$$- \nu\sigma_i^2\,\mathbb{E}_q\left[\lambda_i^2\right]$$

$$\mathbb{E}_q\left[\ln q^*(\mathbf{Z})\right] = \sum_n^N \sum_m^N r_{nm}\ln r_{nm} \tag{90}$$

$$\mathbb{E}_q\left[\ln q^*(\mathbf{\Lambda})\right] = -\mathrm{H}_q\left[\mathbf{\Lambda}\right] \tag{91}$$

7.3.5 ELBO for VB-Scalar-KDE

$$\begin{aligned}\mathcal{L} = \mathbb{E}_q\left[\ln p(\mathbf{X} \mid \mathbf{Z}, \mathbf{\Lambda})\right] + \mathbb{E}_q\left[\ln p(\mathbf{Z})\right] + \mathbb{E}_q\left[\ln p(\mathbf{\Lambda})\right] \\ - \mathbb{E}_q\left[\ln q^*(\mathbf{Z})\right] - \mathbb{E}_q\left[\ln q^*(\mathbf{\Lambda})\right]\end{aligned} \tag{92}$$

$$\mathbb{E}_q\left[\ln p(\mathbf{X} \mid \mathbf{Z}, \mathbf{\Lambda})\right] = -\frac{ND}{2}\ln(2\pi) + \sum_n^N \sum_m^N r_{nm} \ln \rho_{nm} \tag{93}$$

$$= -\frac{ND}{2}\ln(2\pi) + \sum_m^N \ln c_m + \sum_n^N r_{nm} \ln r_{nm} \tag{94}$$

$$\mathbb{E}_q\left[\ln p(\mathbf{Z})\right] = -N\ln(N-1) \tag{95}$$

$$\begin{aligned}\mathbb{E}_q\left[\ln p(\mathbf{\Lambda})\right] = &-\ln\Gamma(\nu) + \nu\left(\ln\nu + \ln\sigma^2\right) + (\nu-1)\,\mathbb{E}_q\left[\ln\lambda^2\right] \\ &- \nu\sigma^2\,\mathbb{E}_q\left[\lambda^2\right]\end{aligned} \tag{96}$$

$$\mathbb{E}_q\left[\ln q^*(\mathbf{Z})\right] = \sum_n^N \sum_m^N r_{nm} \ln r_{nm} \tag{97}$$

$$\mathbb{E}_q\left[\ln q^*(\mathbf{\Lambda})\right] = -\mathrm{H}_q\left[\mathbf{\Lambda}\right] \tag{98}$$

7.4 Sampling the Posterior Predictive Likelihood

The posterior predictive likelihood for the VB-KNN-KDE model takes the form of a mixture of multivariate Student's t-distributions, which may be sampled by first picking a component in the mixture uniformly at random, and then sampling that component through computing (Genz and Bretz, 2009)

$$\widehat{\boldsymbol{x}} \leftarrow \boldsymbol{x}_n + \sqrt{\frac{u}{\widetilde{\nu}_n + 1 - D}}\boldsymbol{y}, \tag{99}$$

where

$$\boldsymbol{y} \sim \mathcal{N}(\mathbf{0}, (\widetilde{\nu}_n + 1 - D)\,\widetilde{\mathbf{W}}_n), \quad u \sim \chi^2(\widetilde{\nu}_n + 1 - D). \tag{100}$$

8 ASYMPTOTIC COMPLEXITY

Time-complexity: For a data set of N samples in D dimensions the inversion of each observation-specific kernel relies on the Cholesky factorization with $\mathcal{O}(D^3)$ time complexity, while the evaluation of a latent variable r_{nm} between the n'th and m'th data points involves evaluating a quadratic form taking $\mathcal{O}(D^2)$ time. Thus the evaluation of all pairs of latent variables and Cholesky factorization of all kernels takes $\mathcal{O}(N^2D^2 + ND^3)$ time corresponding to one complete pass over the training data. However, these computations are embarrassingly parallel across N and the Cholesky factorization and quadratic form computations are implemented using parallel GPU algorithms, yielding further parallellization.

Space-complexity: Naïve implementation requires a space complexity of either $\mathcal{O}(N^2)$ by storing all responsibilities or alternatively $\mathcal{O}(ND^2)$ by storing all observation-specific precision matrices. By exploiting sparsity in the responsibilities we in our experiments empirically observe log-linear scaling ($\approx \mathcal{O}(N \log N)$) in memory for all considered data sets.