Variable Kernel Density Estimation in High-dimensional Feature Spaces

Christiaan M. van der Walt [a,b] and Etienne Barnard [b]
[a] Modelling and Digital Science, CSIR, Pretoria, South Africa
cvdwalt@csir.co.za
[b] Multilingual Speech Technologies Group, North-West University, Vanderbijlpark, South Africa
etienne.barnard@nwu.ac.za

Abstract

Estimating the joint probability density function of a dataset is a central task in many machine learning applications. In this work we address the fundamental problem of kernel bandwidth estimation for variable kernel density estimation in high-dimensional feature spaces. We derive a variable kernel bandwidth estimator by minimizing the leave-one-out entropy objective function and show that this estimator is capable of performing estimation in high-dimensional feature spaces with great success. We compare the performance of this estimator to state-of-the art maximum-likelihood estimators on a number of representative high-dimensional machine learning tasks and show that the newly introduced minimum leave-one-out entropy estimator performs optimally on a number of high-dimensional datasets considered.

Introduction

With the advent of the internet and advances in computing power, the collection of very large high-dimensional datasets has become feasible – understanding and modelling high-dimensional data has thus become a crucial activity, especially in the field of machine learning. Since non-parametric density estimators are data-driven and do not require or impose a pre-defined probability density function on data, they are very powerful tools for probabilistic data modelling and analysis. Conventional non-parametric density estimation methods, however, originated from the field of statistics and were not originally intended to perform density estimation in high-dimensional features spaces - as is often encountered in real-world machine learning tasks. (Scott and Sain 2005) states, for example, that kernel density estimation of a full density function is only feasible up to six dimensions. We therefore define density estimation tasks with dimensionalities of 10 and higher as high-dimensional; and we address the the fundamental problem of non-parametric density estimation in high-dimensional feature spaces in this study.

The first notable attempt to free discriminant analysis from strict distributional assumptions was made in 1951 by (Fix and Hodges 1951; Silverman 1986) with the introduction of the naïve estimator. Since then, many approaches to non-parametric density estimation have been developed, most notably: kernel density estimators (KDEs) (Whittle 1958), k-nearest-neighbour density estimators (Loftsgaarden and Quesenberry 1965), variable KDEs (Breiman et al. 1977), projection pursuit density estimators (Friedman et al. 1984), mixture model density estimators (Dempster et al. 1977; Redner and Walker 1984) and Bayesian networks (Webb 2003; Heckerman 2008).

Recent advances in machine learning have shown that maximum-likelihood (ML) kernel density estimation holds much promise for estimating non-parametric density functions in high-dimensional spaces. Two notable contributions are the Maximum Leave-one-out Likelihood (MLL) kernel bandwidth estimator (Barnard 2010) and the Maximum Likelihood Leave-One-Out (ML-LOO) kernel bandwidth estimator (Leiva-Murillo and Rodríguez 2012). Both estimators were independently derived from the ML objective function with the only differences being the simplifying assumptions made in their derivations to limit the complexity of the bandwidth optimisation problem.

In particular, the ML-LOO estimator constrains the number of free parameters that need to be estimated in the bandwidth optimisation problem by estimating an identical full-covariance or spherical bandwidth matrix for all kernels. The MLL estimator by Barnard assumes that the density function changes slowly throughout feature space, which limits the complexity of the objective function being optimised. Specifically, when the kernel bandwidth $\mathbf{H}_i$ for data point x_i is estimated, it is assumed that the kernel bandwidths of the remaining $N - 1$ kernels are equal to $\mathbf{H}_i$. Therefore, the optimisation of the bandwidth $\mathbf{H}_i$ does not require the estimation of the bandwidths of the remaining *N-1* kernels.

We address the problem of kernel bandwidth estimation for kernel density estimation in this work by deriving a kernel bandwidth estimation technique from the ML framework without the simplifications imposed by the MLL and ML-LOO estimators. This allows us to compare the performance of this more general ML estimator to the MLL and ML-LOO estimators on a number of representative machine learning tasks to gain a better understanding of the practical implications of the simplifying assumptions made in the derivations of the MLL and ML-LOO estimators on real world (RW) density estimation tasks.

In Section 2 we define the classical ML kernel bandwidth estimation problem and derive a novel kernel bandwidth estimator from the ML leave-one-out objective function. We also show how the MLL and ML-LOO estimators can be derived from the same theoretical framework. In Section 3 we describe the experimental design of our comparative simulation studies and in Section 4 we present the results of our simulations. Finally, we make concluding remarks and suggest future work in Section 5.

Maximum-likelihood Kernel Density Estimation

KDEs estimate the probability density function of a D-dimensional dataset $\mathbf{X}$, consisting of N independent and identically distributed samples $\mathbf{x}_1, ..., \mathbf{x}_N$ with the sum

$$p_{\mathbf{H}}(x_i) = \frac{1}{N}\sum_{j=1}^{N} K_{\mathbf{H}_j}\left(\mathbf{x}_i - \mathbf{x}_j|\mathbf{H}_j\right) \tag{1}$$

where $K_{\mathbf{H}_j}\left(\mathbf{x}_i - \mathbf{x}_j|\mathbf{H}_j\right)$ is the kernel smoothing function fitted over each data point $\mathbf{x}_j$ with bandwidth matrix $\mathbf{H}_j$ that describes the variation of the kernel function.

This formulation shows that the density estimated with the KDE is non-parametric, since no parametric distribution is imposed on the estimate; instead the estimated distribution is defined by the sum of the kernel functions centred on the data points. KDEs thus require the selection of two design parameters, namely the parametric form of the kernel function and the bandwidth matrix. It has been shown that the efficiencies of kernels with respect to the mean squared error between the true and estimated distribution do not differ significantly and that the choice of kernel function should rather be based on the mathematical properties of the kernel function, since the estimated density function inherits the smoothness properties of the kernel function (Silverman, 1986). The Gaussian kernel is therefore often selected in practice for its smoothness properties, such as continuity and differentiability. This thus leaves the estimation of the kernel bandwidth as the only parameter to be estimated.

The ML criterion for kernel density estimation is typically defined as the log-likelihood function and has an inherent shortcoming since the estimated log-likelihood function tends to infinity as the bandwidths of the kernel functions centred on each data point tend to zero. This thus leads to a trivial degenerate solution when kernel bandwidths are estimated by optimising the ML objective function. To address this shortcoming the leave-one-out KDE estimate is used. The leave-one-out estimate removes the effect of sample $\mathbf{x}_i$ from the KDE sum when estimating the likelihood of $\mathbf{x}_i$; optimising the leave-one-out ML objective function with respect to the kernel bandwidth matrix will thus prevent the trivial solution of the ML objective function where $l_{\mathbf{H}}(\mathbf{X}) = \infty$ when $\mathbf{H}_j = 0$. The leave-one-out ML objective function is thus defined as

$$l_{\mathbf{H}}(X) = \sum_{i=1}^{N} \log\left[\frac{1}{N-1}\sum_{j\neq i} K_{\mathbf{H}_j}\left(\mathbf{x}_i - \mathbf{x}_j|\mathbf{H}_j\right)\right] \tag{2}$$

where $\mathbf{H}$ is used to denote the dependency of the log-likelihood on the kernel bandwidths $\mathbf{H}_j$.

Maximum-likelihood Kernel Bandwidth Estimation

KDE bandwidths that optimise the leave-one-out ML objective function can be estimated by finding the partial derivative of the leave-one-out ML objective function in Eq. 2 with respect to each bandwidth $\mathbf{H}_k$

$$\frac{\partial}{\partial \mathbf{H}_k}\left(l_{\mathbf{H}}(X)\right) = \sum_{i=1}^{N} \frac{\frac{\partial}{\partial \mathbf{H}_k}\left[\frac{1}{N-1}\sum_{j\neq i} K_{\mathbf{H}_j}\left(\mathbf{x}_i - \mathbf{x}_j|\mathbf{H}_j\right)\right]}{p_{\mathbf{H}(-i)}(\mathbf{x}_i)} \tag{3}$$

The bandwidth $\mathbf{H}_k$ can thus be estimated by setting this partial derivative to zero and solving for $\mathbf{H}_k$.

MLE kernel bandwidth estimation Based on the formulation of the leave-one-out ML objective function in Eq. 3 we derive a new kernel bandwidth estimator named the minimum leave-one-out entropy (MLE) estimator. (To our knowledge, this is the first attempt where partial derivatives are used to derive variable bandwidths in a closed form solution.)

Since the partial derivative in Eq. 3 will only be non-zero for *j*=*k*, we simplify this equation to give the partial derivative of the MLE objective function

$$\frac{\partial}{\partial \mathbf{H}_k}\left(l_{MLE}(X)\right) = \sum_{i=1}^{N} \frac{\frac{1}{N-1}\frac{\partial}{\partial \mathbf{H}_k}\left[K_{\mathbf{H}_k}\left(\mathbf{x}_i - \mathbf{x}_k|\mathbf{H}_k\right)\right]}{p_{\mathbf{H}(-i)}(\mathbf{x}_i)} \tag{4}$$

If we restrict the MLE estimator to multivariate Gaussian kernels with diagonal covariance matrices, the partial derivative of the kernel function with respect to the diagonal bandwidth H_k in dimension d can be derived with the product rule [1]

$$h_{kd}^{-2} K_{h_{kd}}\left(\frac{x_{id}-x_{kd}}{h_{kd}}\right)\left[h_{kd}^{-2}(x_{id}-x_{kd})^2 - 1\right]\prod_{p\neq d} K_{h_{kp}}\left(\frac{x_{ip}-x_{kp}}{h_{kp}}\right) \tag{5}$$

where $H_{k(d,d)} = h_{kd}^2$ and h_{kd} is the kernel bandwidth in dimension d centred on data point x_k. If we substitute the partial derivative of this kernel function into Eq. 4, set the result to zero and solve for $\mathbf{H}_k$ we obtain the MLE diagonal covariance bandwidth estimate

$$H_{k(d,d)} = \frac{\sum_{i=1}^{N} \frac{K_{\mathbf{H}_k}(\mathbf{x}_i - \mathbf{x}_k|\mathbf{H}_k)(x_{id}-x_{kd})^2}{p_{\mathbf{H}(-i)}(\mathbf{x}_i)}}{\sum_{i=1}^{N} \frac{K_{\mathbf{H}_k}(\mathbf{x}_i - \mathbf{x}_k|\mathbf{H}_k)}{p_{\mathbf{H}(-i)}(\mathbf{x}_i)}} \tag{6}$$

MLL kernel bandwidth estimation A key insight observed by (Gangopadhyay and Cheung, 2002) and by (Barnard, 2010) is that if the density function changes relatively slowly throughout feature space, the optimal kernel bandwidths will also change slowly throughout feature space. If it is assumed that the density function changes slowly throughout feature space, a simplification can be

[1] We provide complete proofs for the MLE diagonal and full covariance, MLL and ML-LOO bandwidth estimators as well as experimental results on convergence of the MLE bandwidhts in (van der Walt 2014).

made by assuming that the bandwidths of kernels in the neighbourhood of a kernel are the same as that of the kernel. Thus, when the bandwidth $\mathbf{H}_i$ is being estimated, it is assumed that all other bandwidths are equal to $\mathbf{H}_i$. Barnard used this assumption to reduce the complexity of the the leave-one-out kernel density estimate and derived a diagonal covariance variable multivariate Gaussian kernel banwidth estimator as

$$H_{i(d,d)} = \frac{\sum_{j\neq i} K_{\mathbf{H}_i}\left(\mathbf{x}_i - \mathbf{x}_j|\mathbf{H}_i\right)\left(x_{id} - x_{jd}\right)^2}{\sum_{j\neq i} K_{\mathbf{H}_i}\left(\mathbf{x}_i - \mathbf{x}_j|\mathbf{H}_i\right)} \quad (7)$$

for each dimension d. This diagonal covariance bandwidth estimator is called the MLL estimator.

ML-LOO bandwidth estimation (Leiva-Murillo and Rodríguez 2012) also simplified the bandwidth optimisation problem for ML kernel bandwidth estimation by estimating an identical spherical or full-covariance bandwidth matrix for all kernels – thus limiting the number parameters to be optimised to a single bandwidth or single covariance matrix. If the definition of the leave-one-out KDE for an identical spherical Guassian bandwidth is substituted into Eq. 2, the ML-LOO spherical bandwidth, h, can be solved as [2]

$$h^2 = \frac{1}{D}\frac{\sum_{i=1}^{N}\frac{1}{p_{\mathbf{H}(-i)}(\mathbf{x}_i)}\sum_{j\neq i} K\left(\mathbf{x}_i - \mathbf{x}_j|h\right)\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2}{\sum_{i=1}^{N}\frac{1}{p_{\mathbf{H}(-i)}(\mathbf{x}_i)}\sum_{j\neq i} K\left(\mathbf{x}_i - \mathbf{x}_j|h\right)} \quad (8)$$

Similarly, if the definition of the leave-one-out KDE for an identical full covariance Gaussian bandwidth is substituted into Eq. 2, the ML-LOO full covariance bandwidth matrix, H, can be solved as

$$\mathbf{H} = \frac{\sum_{i=1}^{N}\frac{1}{p_{\mathbf{H}(-i)}(\mathbf{x}_i)}\sum_{j\neq i}(\mathbf{x}_i-\mathbf{x}_j)(\mathbf{x}_i-\mathbf{x}_j)^T K_H(\mathbf{x}_i-\mathbf{x}_j|H)}{\sum_{i=1}^{N}\frac{1}{p_{\mathbf{H}(-i)}(\mathbf{x}_i)}\sum_{j\neq i} K_H(\mathbf{x}_i-\mathbf{x}_j|H)} \quad (9)$$

Global MLE bandwidth estimation Motivated by the bias-variance trade-off, we derive a new estimator (named the global MLE estimator) by defining the leave-one-out kernel density estimate for a diagonal covariance bandwidth matrix, $\mathbf{H}_g$, that is identical for all kernels. If we substitute this definition of the leave-one-out KDE into Eq. 2, $\mathbf{H}_g$ can be solved as

$$H_{g(d,d)} = \frac{\sum_{i=1}^{N}\frac{\sum_{j\neq i} K_{\mathbf{H}_g}(x_{id}-x_{jd}|\mathbf{H}_g)(x_{id}-x_{jd})^2}{p_{\mathbf{H}_g(-i)}(\mathbf{x}_i)}}{\sum_{i=1}^{N}\frac{\sum_{j\neq i} K_{\mathbf{H}_g}(x_{id}-x_{jd}|\mathbf{H}_g)}{p_{\mathbf{H}_g(-i)}(\mathbf{x}_i)}} \quad (10)$$

for each dimension d.

Experimental Design

In this section we describe the experimental design to compare and investigate the performance of the MLE, global MLE, MLL, spherical ML-LOO and full covariance ML-LOO estimators on a number of representative RW machine learning datasets.

[2]Leiva-Murillo and Rodríguez further simplify this expression by substituting $\sum_{j\neq i} K\left(\mathbf{x}_i - \mathbf{x}_j|h\right) = (N-1)p_{\mathbf{H}(-i)}(\mathbf{x}_i)$

Table 1: RW dataset summary

Dataset	**C**	**D**	**Ntr**	**Nte**
Letter	26	16	16000	4000
Segmentation	7	18	2100	210
Landsat	6	36	4435	2000
Optdigits	10	64	3823	1797
Isolet	26	617	6238	1559

Datasets

We select five RW datasets (with independent train and test sets) from the UCI Machine Learning Repository (Lichman 2013) for the purpose of simulation studies. These data sets are selected to have relatively high dimensionalities, since high-dimensions are typical of many real-world machine learning tasks. The selected datasets are the Letter, Segmentation, Landsat, Optdigits and Isolet datasets. We summarise these datasets in Table 1 and denote the number of classes with "C", the dimensionality with "D", the number of training samples with "Ntr" and the number of test samples with "Nte". We refer the reader to (Lichman 2013) for a more detailed description of these datasets.

Data pre-processing

Principal Component Analysis (PCA) is performed on all RW datasets as a pre-processing step prior to density estimation. PCA is used to reduce the dimensionality of datasets by performing a linear transformation that re-aligns the feature axes to the directions of most variation, thus minimizing the variance orthogonal to the projection. Features with smallest eigenvalues (or variance in the transformed feature space) may thus be disregarded for the purpose of dimensionality reduction. PCA also ensures that the features of the transformed feature space are orthogonal, thus ensuring the features are decorrelated. We retain only the transformed features with eigenvalues larger than 1% of the eigenvalue of the principal component. [3] We perform the PCA transformation on each class individually, since it has been shown (Barnard, 2010) that this approach is more effective in compressing features than when all classes are transformed simultaneously. We denote this pre-processing step for dimensionality reduction as "PCA1".

A more subtle implication of PCA is that it serves as a form of bandwidth regularisation for ML kernel bandwidth estimators, which is a crucial task for kernel density estimation in high-dimensional spaces. Because of the formulation of the ML objective function, an infinite likelihood score is obtained when two data points have corresponding values in any dimension. When, for example, a Gaussian kernel is fitted over a training data point, and a test point has the same value in any of the dimensions, the bandwidth that will maximise the likelihood score in the dimension with correspond-

[3]The more conventional approach is to keep the features with largest eigenvalues such that their eigenvalues sum to 95% of the total of all eigenvalues. This approach attempts to capture 95% of the variance, but fails when, for example, all the features have approximately the same eigenvalues.

ing values will be 0. This phenomenon becomes more problematic as dimensionality increases, since the probability of having two observations with the same value in any dimension increases significantly, thus leading to more degenerate kernel bandwidths. PCA reduces the probability of identical values in a dimension since the linear transformation of a new feature is the weighted sum of the values of all dimensions in the original feature space. Two data points thus need to satisfy the same linear equation to have the same value in a transformed dimension.

Kernel bandwidth initialisation

It was shown in (van der Walt and Barnard 2013) that the Silverman rule-of-thumb bandwidth estimator performed reliably on a number of machine learning tasks. Based on this empirical evidence and the intuitive theoretical motivation that the Silverman estimator optimises the asymptotic mean integrated squared error (assuming a Gaussian reference distribution), we make use of the Silverman bandwidth estimator for bandwidth initialisation.

Since the Silverman estimator estimates a unique bandwidth per dimension, we initialise the diagonal MLE and MLL bandwidths of each kernel with the Silverman bandwidths for each dimension. Similarly, the full covariance ML-LOO bandwidth matrix and global MLE bandwidth matrix is initialised with a diagonal bandwidth matrix consisting of the Silverman bandwidths. The spherical ML-LOO bandwidth is initialised with the average of the Silverman bandwidths.

Kernel bandwidth regularisation

In practice the optimal ML kernel bandwidth, h_{id}, centred on data point x_i in dimension d, is degenerate under certain circumstances and tends to 0. Pre-processing with PCA reduces the number of such occurrences significantly as explained earlier. As a further measure of regularisation we make use of the theoretical lower bound of the optimal bandwidth for the ML criterion as derived in (Leiva-Murillo and Rodríguez 2012). The lower bound states that the minimum optimal bandwidth, h_{id}^2, is equal to the squared Euclidean distance to the nearest-neighbour of x_i. In practice it often happens that two samples have the same values in a certain dimension, thus making the nearest-neighbour distance 0. We set the lower bound of a bandwidth in each dimension to the nearest data point with a non-zero distance. For all estimators we validate that all bandwidths are above their respective lower bounds, after each iteration of the bandwidth optimisation procedure. The bandwidths that are below their respective lower bounds after an iteration are thus replaced by the corresponding lower bound value.

Performance evaluation

The RW datasets in Table 1 all have independent test sets. We therefore perform 10-fold cross-validation on each class specific training set to find the optimal number of training iterations for each class conditional density function. We then calculate the likelihood scores of the test samples for each class to estimate the sample entropy per class.

The optimal kernel bandwidth is obtained with a direct approach where the right-hand side of each bandwidth estimation equation is initialised with the Silverman bandwidth, the left hand side is updated, and the updated bandwidth is then substituted into the right-hand side again. This process is repeated for 10 iterations on each training set.

Experimental setup

We compare the performance of the MLE, global MLE, MLL, spherical ML-LOO and full covariance ML-LOO estimators on the Letter, Segmentation, Landsat, Optdigits and Isolet datasets summarised in Table 1. PCA1 class-specific transformations are performed on each class for all datasets and the entropy score of each estimator is calculated per class.

Results

In this section we present the results of the comparative simulation study described in the experimental design section. We denote the class number with “C”, the number of principal components with “K”, the number of samples in the class with “NC”, the MLE estimator with “MLE”, the global MLE estimator with “MLE(g)”, the MLL estimator with “MLL”, the spherical covariance ML-LOO estimator with “ML-LOO(s)” and the full covariance ML-LOO estimator with “ML-LOO(f)”. We make use of colour scales to visually represent the relative performance of the estimators, where green indicates the lowest entropy for a specific class and red the highest entropy for a specific class.

The *Letter* dataset results in Table 2 show that the ML-LOO(f) estimator performs optimally on all classes except class 9, while the MLL and ML-LOO(s) estimator underperform on most classes.

The *Segmentation* dataset results in Table 3 show that the MLE estimator performs optimally on all classes except classes 2 and 7, and the MLL estimator performs competitively on the same classes. The ML-LOO(f) estimator performs optimally on classes 2 and 7, underperforms on classes 3 and 4 and performs moderately on the remaining classes. The MLE(g) and ML-LOO(s) estimators underperform on four classes, and perform moderately on the remaining classes.

The *Landsat* dataset results in Table 4 show that the ML-LOO(f) estimator performs optimally and near optimally on all classes except class 2 and the global MLE estimator performs competitively on most classes except class 2. The MLE estimator performs optimally on class 2 and moderately on the remaining classes, while the MLL estimator performs competitively on class 2 and moderately on the remaining classes. The ML-LOO(s) clearly underperforms on all classes.

The *Optdigits* dataset results in Table 5 show that the MLE(g) estimator performs optimally on most classes. The MLE and MLL estimators perform optimally on class 7 and perform moderately on the remaining classes. The ML-LOO(s) estimator performs near optimally and optimally on classes 3 and 5, moderately on classes 2 and 8 and underperforms on the remaining classes. The ML-LOO(f) estimator

Table 2: Letter test entropy results (class-specific PCA1)

C	K	NC	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
1	15	789	9.2873	10.2996	10.5922	7.9763	10.1448
2	15	766	11.8521	12.4504	13.2300	10.6214	12.5788
3	15	736	10.9114	10.9152	11.4798	10.2541	11.8211
4	14	805	10.6806	10.8480	11.2551	9.8986	11.1240
5	15	768	9.0948	10.0307	10.6545	7.4805	9.8392
6	13	775	9.3735	9.8056	10.0067	8.2964	10.0040
7	15	773	11.6837	12.4585	12.7124	10.6560	12.3793
8	14	734	10.6712	10.6509	10.6427	8.9762	11.4651
9	13	755	8.7703	7.4315	7.4505	7.6537	9.1916
10	13	747	8.0048	8.2247	8.5730	6.9398	8.5478
11	15	739	10.1285	10.7803	10.6981	9.1266	11.0117
12	12	761	6.9545	7.0843	7.2469	5.9676	7.4122
13	14	792	8.3489	8.4041	8.7171	6.7962	9.0203
14	14	783	9.3522	8.7992	9.7257	8.4742	10.1868
15	16	753	12.7908	13.0909	13.9332	11.5703	13.7332
16	15	803	9.9907	11.1816	11.5641	9.4843	10.8242
17	15	783	10.8606	11.8567	12.0359	9.6828	11.8571
18	15	758	10.9078	12.1141	12.4231	9.9914	11.3758
19	15	748	9.0280	9.8751	10.0427	7.5570	9.7640
20	13	796	7.8998	8.1263	8.4534	6.6746	8.7879
21	13	813	8.8809	9.0425	9.2017	7.9219	9.2352
22	14	764	10.0613	10.3505	10.6758	8.7185	10.8547
23	15	752	10.5571	10.9732	11.4986	9.3987	11.8726
24	15	787	11.0415	11.1224	11.6708	9.7728	12.2909
25	14	786	8.7009	9.3734	9.8168	7.4867	9.2290
26	15	734	10.2744	10.0557	10.8011	9.1422	11.2895

Table 3: Segmentation test entropy results (class-specific PCA1)

C	K	NC	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
1	13	330	14.7612	14.3396	14.5128	15.2068	16.6483
2	11	330	13.3975	13.5591	15.3803	12.9305	13.8543
3	12	330	10.8880	2.7401	3.5781	10.5969	9.6920
4	11	330	15.4156	7.7225	11.9012	17.5817	13.2768
5	11	330	15.4297	6.7261	7.2938	9.9464	17.4564
6	11	330	11.2927	10.7688	11.0891	11.3063	11.7564
7	10	330	7.1770	6.8176	6.8371	3.1038	7.3919

Table 4: Landsat test entropy results (class-specific PCA1)

C	K	NC	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
1	8	1533	10.6246	10.6571	10.6958	10.5527	11.0285
2	10	703	11.4557	11.2188	11.2724	11.3094	12.0536
3	14	1358	16.6442	16.7766	17.1863	16.4641	17.8911
4	13	626	14.6303	14.6583	14.7188	14.6150	15.8079
5	11	707	14.1853	14.6757	14.6641	14.2129	14.8374
6	11	1508	14.2989	14.2268	14.5458	14.2334	15.0980

performs optimally on class 10, performs moderately on four classes and underperforms on the remaining classes.

The *Isolet* dataset results in Table 6 show that the MLE(g) estimator performs optimally on 20 of the 26 classes, while the MLE estimator performs optimally on five classes and performed optimally tied with the MLE(g) estimator on class 14. The MLL estimator performs optimally on class 9 and performs optimally with the MLE estimator on classes 3, 20 and 26. The ML-LOO(f) estimator performs optimally with the MLL estimator on class 9, while the ML-LOO(s) estimator consistently underperforms on all classes.

In general, we have observed that the ML-LOO(f) estimator performed optimally on most classes of the Letter and Landsat datasets; the MLE estimator performed optimally on most classes of the Segmentation dataset and the MLE(g) estimator performed optimally on most classes of the Optdigits and Isolet datasets.

We also observed that the MLE and MLL estimators exhibited very similar relative performance behaviour and that the MLE generally performed better. If we compare the MLE bandwidth estimator in Eq. 6 and the MLL bandwidth estimator in Eq. 7 we find that these estimators are identical except that the numerator and denominator of the MLE estimation equation is normalised with the leave-one-out KDE likelihood score of each sample x_i. This explains the similar performance behaviour of these estimators and shows that this normalisation factor leads to an improvement in performance. This term may be regarded as a regularisation term since the effect on the estimated bandwidth of data points that fall within dense regions is reduced while the effect of data points that lie in lower density regions are increased. To illustrate the regularisation effect, consider two univariate data points, x_1 and x_2, that lie very close to each other. Both x_1 and x_2 will have relatively high density val-

Table 5: Optdigits test entropy results (class-specific PCA1)

C	K	NC	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
1	46	554	52.7315	54.9642	54.9642	55.0121	57.2316
2	36	571	37.8930	39.9837	39.9837	40.0183	39.8571
3	41	557	55.1442	60.8972	60.8972	61.1132	55.9001
4	50	572	63.7679	67.9808	67.9808	69.7068	69.8641
5	39	568	45.1030	45.1030	45.1030	45.1332	44.2904
6	46	558	55.7444	58.8748	58.8748	58.9782	59.6404
7	37	558	46.8292	46.7069	46.7069	46.9650	48.8804
8	41	566	52.9752	55.9436	55.9436	55.9758	55.6049
9	50	554	61.2910	63.9965	63.9965	64.2079	65.3601
10	43	562	57.9241	56.2229	56.2229	51.2123	59.1375

Table 6: Isolet test entropy results (class-specific PCA1)

C	K	NC	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
1	92	300	189.0759	191.2731	190.8883	190.8883	205.5452
2	107	300	211.6703	210.8217	212.5131	212.5131	227.0795
3	98	300	203.3099	202.9658	202.9658	202.9675	218.5153
4	101	300	205.9877	209.6424	207.7975	207.8165	220.4039
5	103	300	213.3857	215.8229	218.8905	218.9028	230.3118
6	100	300	199.3268	199.7004	200.6295	200.6515	219.3737
7	90	300	186.8359	189.5903	188.2592	188.2732	198.8510
8	88	300	180.7441	182.4256	181.7845	182.0967	195.6949
9	108	300	230.2376	236.3707	226.6773	226.6732	243.0001
10	85	300	182.6589	184.5528	184.5528	184.6028	193.5564
11	89	300	189.1564	190.9196	190.9196	190.9604	199.9672
12	118	300	241.4164	252.6089	249.7367	250.0967	267.3455
13	123	300	244.3422	248.1532	249.8309	249.8319	269.9968
14	119	300	239.3669	239.3669	261.6026	241.8353	264.3237
15	103	300	209.7407	215.6002	213.0124	213.1697	232.1384
16	104	300	206.8094	208.4278	207.3090	207.3090	221.1164
17	108	300	222.6612	227.7650	227.7650	227.7496	244.1215
18	106	300	217.0238	224.1280	221.0936	221.2660	235.8874
19	96	300	199.3421	201.4060	203.1794	203.7217	219.1281
20	100	300	203.0767	202.6541	202.6541	202.6541	217.6525
21	121	300	239.3333	241.4055	244.0612	244.1695	260.1166
22	113	300	228.0789	226.9123	230.1929	230.1947	248.3391
23	104	300	207.1714	208.6037	209.0199	209.0204	220.9262
24	92	300	194.4274	199.8320	197.9024	197.9059	209.3233
25	109	300	218.3860	219.9983	220.8091	221.0781	239.4411
26	100	300	206.8940	206.6910	206.6910	206.7172	224.4596

ues for $p_{h(-1)}(x_1)$ and$p_{h(-2)}(x_2)$, since they are close in feature space and will increase each other's density. If the bandwidth,h_1, is estimated for the kernel centred on x_1, the distance $(x_1 - x_2)^2$ will be very small, but since the value of $p_{H(-1)}(x_2)$ will be large, the contribution of $(x_1 - x_2)^2$ to h_1 will be reduced, thus preventing h_1 from becoming too small.

The ML-LOO(s) estimator estimates the single bandwidth for all dimensions and all kernels, and the general under-performance of this estimator compared to the global MLE estimator shows that local scale variations should differ between dimensions.

The MLE(g) estimator estimates an identical covariance bandwidth matrix for all kernels, and the superior performance of this estimator on the Optdigits and Isolet datasets show that these datasets require scale variations between features but no not require drastic local scale variations within dimensions. We derive this conclusion from the fact that the MLE and MLL estimators are capable of modelling drastic changes in local scale variation within features (since the bandwidth is adapted for each kernel in each dimension), and since the MLE(g) estimator generally outperforms these estimators on these two datasets, it shows an interesting case of the bias-variance trade-off: having fewer parameters, the MLE(g) estimator is less flexible than the MLE and MLL estimators; however, those parameters can be estimated with greater reliability, leading to the best performance in many cases. It is important to note that the MLE(g) estimator can model local scale variations to some extent within features, since a kernel function is placed on each data point, and if the locations of data points within a dimension vary the estimated density function for the dimension will also vary according to the locations of the data points, thus capturing local scale variations. However, the bandwidths placed on each data point are identical within a dimension and therefore drastic scale variations cannot be modelled accurately. Similarly, the MLE(g) estimator can also model correlation between features to some extent since a kernel is placed on each data point, and if data points between features vary in the same direction the correlation between these features will be captured by the density of the kernels placed on these data points. However, since the MLE(g) estimator has a diagonal covariance bandwidth matrix, local variations are modelled parallel to the feature axes. Thus if local variations are not parallel to the feature axes, the ML-LOO(f) estimator will perform better since the ML-LOO(f) estimator can model local density that is not parallel to the feature axes, by making use of correlation coefficients in the full covariance bandwidth matrix.

The ML-LOO(f) estimator estimates an identical full covariance matrix for all kernels, and the superior performance of this estimator on the Letter and Landsat datasets show that these datasets require a density estimate that can model local variations that are not necessarily parallel to the feature axes. This conclusion is derived from the fact that the ML-LOO(f) generally outperforms the MLE(g) estimator on these two datasets, and since both estimators estimate an identical bandwidth matrix for each kernel, the only difference is that the ML-LOO(f) has a full covariance bandwidth matrix that models local correlation in the density estimate by making use of correlation coefficients in the kernel bandwidth matrix.

Conclusion and future work

We have shown that none of the ML estimators investigated performed optimally on all tasks, and based on theoretical motivations confirmed with empirical results it is clear that the optimal estimator depends on the degree of scale variation between features and the degree of changes in scale variation with features.

The results show that the full covariance ML-LOO and global MLE estimators (which estimate an identical full and diagonal covariance matrix respectively for each kernel) performed optimally on four of the five datasets investigated, while the MLE estimator (which estimates a unique bandwidth for each kernel) performed optimally on only one dataset. This is an interesting case of the bias-variance trade-off: having fewer parameters, the full covariance ML-LOO and global MLE estimators are less flexible than the MLE and MLL estimators; however, those parameters can be estimated with greater reliability, leading to the best performance in many cases.

From the theoretical and empirical results in this work it is clear that the optimal estimator should somehow function like the full covariance ML-LOO estimator in regions with low spatial variability, and must function like the MLE estimator and be able to adapt bandwidths in regions with high spatial variability, especially for outliers.

We therefore believe that it would be interesting to investigate a hybrid kernel bandwidth estimator by first detecting and removing outliers. Clustering can then be performed on the remaining data and the full covariance ML-LOO bandwidth estimator can be used to estimate a unique full covariance kernel bandwidth for each cluster - each kernel will thus make use of the full covariance bandwidth matrix of the cluster to which it is assigned. The MLE estimator can then be used to estimate a unique bandwidth for each outlier; since the MLE estimator can model scale variations, this will ensure that outliers have sufficiently wide bandwidths. This proposed hybrid approach will thus generally function like the full covariance ML-LOO estimator in the clustered regions and has the added ability to change the direction of local scale variations between clusters. This estimator will also be capable of making the bandwidths of kernel function centred on outliers sufficiently wide. We therefore propose to implement this hybrid ML kernel bandwidth estimator in future work and perform a comparative study between this approach, the MLE, full covariance ML-LOO and the first hybrid approach proposed above.

In summary, the results of this investigation show that the contribution of the global MLE and MLE estimators are extremely valuable since they provide two alternative kernel bandwidth estimators to employ in high-dimensional feature spaces.

References

Barnard, E., 2010. Maximum leave-one-out likelihood for kernel density estimation, in: *Proceedings of the twenty-first annual symposium of the Pattern Recognition Association of South Africa (PRASA),* Stellenbosch, South Africa.

Breiman, L., Meisel, W., Purcell, E., 1977. Variable kernel estimates of multi-variate densities. *Technometrics* 19, 135144.

Dempster, A.P., Laird, N.M., Rubin, D.B., 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*. Series B (Methodological) , 138.

Fix, E., Hodges, J.L., 1951. Discriminatory analysis, nonparametric estimation: consistency properties. Technical Report, Report No. 4, Project No. 21-49-004. USAF School of Aviation Medicine. Randolph Filed, Texas.

Friedman, J.H., Stuetzle, W., Schroeder, A., 1984. Projection pursuit density estimation. *Journal of the American Statistical Association* 79, 599608.

Gangopadhyay, A., Cheung, K., 2002. Bayesian approach to the choice of smoothing parameter in kernel density estimation. *Journal of Nonparametric Statistics* 14, 655664.

Heckerman, D., 2008. A tutorial on learning with Bayesian networks. Springer.

Leiva-Murillo, J.M., Rodríguez, A., 2012. Algorithms for maximum-likelihood bandwidth selection in kernel density estimators. *Pattern Recognition Letters* 33, 17171724.

Lichman, M. ”UCI machine learning repository, 2013.” URL http://archive.ics.uci.edu/ml (2016).

Loftsgaarden, D.O., Quesenberry, C.P., 1965. A nonparametric estimate of a multivariate density function. *The Annals of Mathematical Statistics*, 10491051.

Redner, R.A., Walker, H.F., 1984. Mixture densities, maximum likelihood and the em algorithm. *SIAM* review 26, 195239.

Scott, D.W., Sain, S.R., 2005. Multi-dimensional density estimation. *Handbook of Statistics* 24, 229261.

Silverman, B.W., 1986. *Density estimation for statistics and data analysis*. Volume 26. CRC press.

van der Walt, C.M., Barnard, E., 2013. Kernel bandwidth estimation for non- parametric density estimation: a comparative study, in: *Proceedings of the twenty-fourth annual symposium of the Pattern Recognition Association of South Africa (PRASA),* Johannesburg, South Africa.

van der Walt, C. M. 2014. Maximum-likelihood kernel density estimation in high-dimensional feature-spaces. Ph.D. diss., North-West University, South Africa.

Webb, A.R., 2003. *Statistical pattern recognition*. Wiley.

Whittle, P., 1958. On the smoothing of probability density functions. *Journal of the Royal Statistical Society*. Series B (Methodological) , 334343.