Local Component Analysis

Nicolas Le Roux, Francis Bach

Local Component Analysis

Nicolas Le Roux
nicolas@le-roux.name

Francis Bach
francis.bach@ens.fr

INRIA - SIERRA Project - Team
Laboratoire d'Informatique de l'École Normale Supérieure
Paris, France

Abstract

Kernel density estimation, a.k.a. Parzen windows, is a popular density estimation method, which can be used for outlier detection or clustering. With multivariate data, its performance is heavily reliant on the metric used within the kernel. Most earlier work has focused on learning only the bandwidth of the kernel (i.e., a scalar multiplicative factor). In this paper, we propose to learn a full Euclidean metric through an expectation-minimisation (EM) procedure, which can be seen as an unsupervised counterpart to neighbourhood component analysis (NCA). In order to avoid overfitting with a fully nonparametric density estimator in high dimensions, we also consider a semi-parametric Gaussian-Parzen density model, where some of the variables are modelled through a jointly Gaussian density, while others are modelled through Parzen windows. For these two models, EM leads to simple closed-form updates based on matrix inversions and eigenvalue decompositions. We show empirically that our method leads to density estimators with higher test-likelihoods than natural competing methods, and that the metrics may be used within most unsupervised learning techniques that rely on local distances, such as spectral clustering or manifold learning methods. Finally, we present a stochastic approximation scheme which allows for the use of this method in a large-scale setting.

1 Introduction

Most unsupervised learning methods rely on a metric on the space of observations. The quality of the metric directly impacts the performance of such techniques and a significant amount of work has been dedicated to learning this metric from data when some supervised information is available [27, 16, 2]. However, in a fully unsupervised scenario, most practitioners use the Mahalanobis distance obtained from principal component analysis (PCA). This is an unsatisfactory solution as PCA is essentially a global linear dimension reduction method, while most unsupervised learning techniques, such as spectral clustering or manifold learning, are local.

In this paper, we cast the unsupervised metric learning as a density estimation problem with a Parzen windows estimator based on a Euclidean metric. Using the maximum likelihood framework, we derive in Section 3 an expectation-minimisation (EM) procedure that maximizes the *leave-one-out log-likelihood*, which may be considered as an unsupervised counterpart to neighbourhood component analysis (NCA) [16]. As opposed to PCA, which performs a whitening of the data based on global information, our new algorithm globally performs a whitening of the data using only local information, hence the denomination local component analysis (LCA).

Like all non-parametric density estimators, Parzen windows density estimation is known to overfit in high dimensions [25], and thus LCA should also overfit. In order to keep the modelling flexibility of our density estimator while avoiding overfitting, we propose a semi-parametric Parzen-Gaussian model; following [4], we linearly transform then split our variables in two parts, one which is modelled through a Parzen windows estimator (where we assume the interesting part of the data lies), and one which is modelled as a multivariate Gaussian (where we assume the noise lies). Again, in Section 4, an EM procedure for estimating the linear transform may be naturally derived and leads to simple closed-form updates based on matrix inversions and eigenvalue decompositions. This procedure contains no hyperparameters, all the parameters being learnt from data.

Since the EM formulation of LCA scales quadratically in the number of datapoints, making it impractical for large datasets, we introduce in Section 5 both a stochastic approximation and a subsampling technique

allowing us to achieve a linear cost and thus to scale LCA to much larger datasets.

Finally, in Section 6, we show empirically that our method leads to density estimators with higher test-likelihoods than natural competing methods, and that the metrics may be used within unsupervised learning techniques that rely on such metrics, like spectral clustering.

2 Previous work

Many authors aimed at learning a Mahalanobis distance suited for local learning. While some techniques required the presence of labelled data [16, 27, 2], others proposed ways to learn the metric in a purely unsupervised way, e.g., [28] who used the distance to the k-th nearest neighbour as the local scaling around each datapoint. Most of the other attempts at unsupervised metric learning were developed in the context of kernel density estimation, a.k.a. Parzen windows. The Parzen windows estimator [21] is a nonparametric density estimation model which, given n datapoints $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ in $\mathbb{R}^d$, defines a mixture model of the form $p(\mathbf{x}) = \frac{1}{n}\sum_{j=1}^{n} K(\mathbf{x}, \mathbf{x}_j, \theta)$ where K is a kernel with compact support and parameters θ. We relax the compact support assumption and choose K to be the normal kernel, that is

$$p(\mathbf{x}) = \frac{1}{n}\sum_{j=1}^{n} \mathcal{N}(\mathbf{x}, \mathbf{x}_j, \Sigma) \propto \frac{1}{n\sqrt{|\Sigma|}} \sum_{j=1}^{n} \exp\left[-\frac{1}{2}(\mathbf{x}-\mathbf{x}_j)^\top \Sigma^{-1}(\mathbf{x}-\mathbf{x}_j)\right] ,$$

where Σ is the covariance matrix of each Gaussian. As the performance of the Parzen windows estimator is more reliant on the covariance matrix than on the kernel, there has been a large body of work, originating from the statistics literature, attempting to learn this matrix. However, almost all attempts are focused on the asymptotic optimality of the estimators obtained with little consideration for the practicality in high dimensions. Thus, the vast majority of the work is limited to isotropic matrices, reducing the problem to finding a single scalar h [22, 13, 23, 9, 7, 20, 24], the *bandwidth*, and the few extensions to the non-isotropic cases are numerically expensive [14, 18].

An exception is the approach proposed in [26], which is very similar to our method, as the authors learn the covariance matrix of the Parzen windows estimator using local neighbourhoods. However, their algorithm does not minimize a well-defined cost function, making it unsuitable for kernels other than the Gaussian one, and the locality used to compute the covariance matrix depends on parameters which must be hand-tuned or cross-validated. Also, the modelling of all the dimensions using the Parzen windows estimator makes the algorithm unsuitable when the data lie on a high-dimensional manifold. In an extension to [26], [3] uses a neural network to compute the leading eigenvectors of the local covariance matrix at every point in the space, then uses these matrices to do density estimation and classification. Despite the algorithm's impressive performance, it does not correspond to a linear reparametrisation of the space and thus cannot be used as a preprocessing step.

3 Local Component Analysis

Seeing the density as a mixture of Gaussians, one can easily optimize the covariances using the EM algorithm [12]. However, maximizing the standard log-likelihood of the data would trivially lead to the degenerate solution where Σ goes to 0 to yield a sum of Dirac distributions. One solution to that problem is to penalize some norm of the precision matrix to prevent it from going to infinity. Another, more compelling, way is to optimize the leave-one-out log-likelihood, where the probability of each datapoint $\mathbf{x}_i$ is computed under the distribution obtained when $\mathbf{x}_i$ has been removed from the training set. This technique is not new and has already been explored both in the supervised [16, 15] and in the unsupervised setting [13]. However, in the latter case, the cross-validation was then done by hand, which explains why only one bandwidth parameter could be optimized[1]. We will thus use the following criterion:

$$\mathcal{L}(\Sigma) = -\sum_{i=1}^{n} \log\left[\frac{1}{n-1}\sum_{j\neq i} \mathcal{N}(x_i, x_j, \Sigma)\right] \quad (1)$$

$$\leq \text{cst} - \sum_{i=1}^{n}\sum_{j\neq i} \lambda_{ij} \log \mathcal{N}(x_i, x_j, \Sigma) + \sum_{i=1}^{n}\sum_{j\neq i} \lambda_{ij}\log\lambda_{ij} , \quad (2)$$

with the constraints $\forall i$, $\sum_{j\neq i}\lambda_{ij} = 1$. This variational bound is obtained using Jensen's inequality.

The EM algorithm optimizes the right-hand side of Eq. (2) by alternating between the optimisations of λ and Σ in turn. The algorithm is guaranteed to converge, and does so to a stationary point of the true

[1] Most of the literature on estimating the covariance matrix discards the log-likelihood cost because of its sensitivity to outliers and prefers AMISE (see, e.g., [14]). However, in all our experiments, the number of datapoints was large enough so that LCA did not suffer from the presence of outliers.

function over Σ defined in Eq. (1). At each step, the optimal solutions are:

$$\lambda_{ij}^* = \frac{\mathcal{N}(x_i, x_j, \Sigma)}{\sum_{k \neq i} \mathcal{N}(x_i, x_k, \Sigma)} \text{ if } j \neq i \quad (3)$$

$$\lambda_{ii}^* = 0 \quad (4)$$

$$\Sigma^* = \frac{\sum_{ij} \lambda_{ij} (\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T}{n} . \quad (5)$$

The "responsibilities" λ_{ij}^* define the relative proximity of $\mathbf{x}_j$ to $\mathbf{x}_i$ (compared to the proximity of all the $\mathbf{x}_k$'s to $\mathbf{x}_i$) and Σ^* is the average of all the local covariance matrices.

This algorithm, which we coin LCA, for local component analysis, transforms the data to make it locally isotropic, as opposed to PCA which makes it globally isotropic. Fig. (1) shows a comparison of PCA and LCA on the word sequence "To be or not to be". Whereas PCA is highly sensitive to the length of the text, LCA is only affected by the local shapes, thus providing a much less distorted result.[2]

First, one may note at this point that Manifold Parzen Windows [26] is equivalent to LCA with only one step of EM. This makes Manifold Parzen Windows more sensitive to the choice of the original covariance matrix whose parameters must be carefully chosen. As we shall see later in the experiments, running EM to convergence is important to get good accuracy when using spectral clustering on the transformed data. Second, it is also worth noting that, similarly to Manifold Parzen Windows, LCA can straightforwardly be extended to the cases where each datapoint uses its own local covariance matrix (possibly with a smoothing term), or where the covariance Σ^* is the sum of a low-rank matrix and some scalar multiplied by the identity matrix.

Not only may LCA be used to learn a linear transformation of the space, but it also defines a density model. However, there are two potential negative aspects associated with this method. First, in high dimensions, Parzen windows is prone to overfitting and must be regularized [25]. Second, if there are some directions containing a small Gaussian noise, the local isotropy will blow them up, swamping the data with clutter. This is common to all the techniques which renormalise the data by the inverse of some variance. A solution to both of these issues is to consider a product of two densities: one is a low-dimensional Parzen windows estimator, which will model the interesting signal, and the other is a Gaussian, which will model the noise.

[2]Since both methods are insensitive to any linear reparametrisation of the data, we do not include the original data in the figure.

4 LCA with a multiplicative Gaussian component

We now assume that there are irrelevant dimensions in our data which can be modelled by a Gaussian. In other words, we consider an invertible linear transformation $(B_G, B_L)^\top \mathbf{x}$ of the data, modelling $B_G^\top \mathbf{x}$ as a multivariate Gaussian and $B_L^\top \mathbf{x}$ through kernel density estimation, the two parts being independent, leading to $p(\mathbf{x}) \propto p(B_G^\top \mathbf{x}, B_L^\top \mathbf{x}) = p_G(B_G^\top \mathbf{x}) p_L(B_L^\top \mathbf{x})$, where p_G is a Gaussian and p_L is the Parzen windows estimator, i.e.,

$$p(\mathbf{x}_i) \propto \frac{\left|B_G B_G^\top + B_L B_L^\top\right|^{\frac{1}{2}}}{n-1} \times \exp\left[-\frac{1}{2}(\mathbf{x}_i - \mu)^\top B_G B_G^\top (\mathbf{x}_i - \mu)\right] \times \left(\sum_{j \neq i} \exp\left[-\frac{1}{2}(\mathbf{x}_i - \mathbf{x}_j)^\top B_L B_L^\top (\mathbf{x}_i - \mathbf{x}_j)\right]\right) ,$$

with (B_G, B_L) a full-rank square matrix. Using EM, we can upper-bound the negative log-likelihood:

$$-2 \sum_i \log p(\mathbf{x}_i) \leq \operatorname{tr}(B_G^\top C_G B_G) + \operatorname{tr}(B_L^\top C_L B_L) - \log \left|B_G B_G^\top + B_L B_L^\top\right| , \quad (6)$$

with

$$C_G = \frac{1}{n} \sum_i (\mathbf{x}_i - \mu)(\mathbf{x}_i - \mu)^\top ,$$

$$C_L = \frac{1}{n} \sum_{ij} \lambda_{ij} (\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^\top .$$

The matrices B_G and B_L minimizing the right-hand side of Eq. (6) may be found using the following proposition (see proof in the appendix):

Proposition 1 *Let $B_G \in \mathbb{R}^{d \times d_1}$ and $B_L \in \mathbb{R}^{d \times d_2}$, with $d = d_1 + d_2$ and $B = (B_G, B_L) \in \mathbb{R}^{d \times d}$ invertible. Consider two symmetric positive matrices M_1 and M_2 in $\mathbb{R}^{d \times d}$. The problem*

$$\min_{B_G, B_L} \operatorname{tr} B_G^\top M_1 B_G + \operatorname{tr} B_L^\top M_2 B_L - \log\det(B_G B_G^\top + B_L B_L^\top) \quad (7)$$

has a finite solution only if M_1 and M_2 are invertible and, if these conditions are met, reaches its optimum at

$$B_G = M_1^{-1/2} U_+ , \quad B_L = M_1^{-1/2} U_- D_-^{-1/2} ,$$

where U_+ are the eigenvectors of $M_1^{-1/2} M_2 M_1^{-1/2}$ associated with eigenvalues greater than or equal to 1, U_- are the eigenvectors of $M_1^{-1/2} M_2 M_1^{-1/2}$ associated

Figure 1: Results obtained when transforming "To be or not to be" using PCA (left) and LCA (right). **Left**: To make the data globally isotropic, PCA awkwardly compresses the letters horizontally. **Right**: Since LCA is insensitive to the spacing between letters and to the length of the text, there is no horizontal compression.

with eigenvalues smaller than 1 and D_- is the diagonal matrix containing the eigenvalues of $M_1^{-1/2} M_2 M_1^{-1/2}$ smaller than 1.

The resulting procedure is described in Algorithm 1, where all dimensions are initially modelled by the Parzen windows estimator, which empirically yielded the best results.

Algorithm 1 LCA - Gauss

Input: X (dataset), iterMax (maximum number of iterations), ν (regularisation)

Output: B_G (Gaussian part transformation), B_L (Parzen windows transformation)

$C_G \leftarrow \text{cov}(X) + \nu I_d$ {Initialize C to the global covariance}

$I_G \leftarrow C_G^{-\frac{1}{2}}$

$B_G = 0, B_L = \text{chol}(C_G^{-1})$ {Assign all dimensions to the Parzen windows estimator}

for iter = 1:iterMax **do**

$M_{ij} \leftarrow \exp\left[-\frac{(x_i - x_j)^\top B_L^\top B_L (x_i - x_j)}{2}\right], \quad M_{ii} \leftarrow 0$

$\lambda_{ij} \leftarrow \frac{M_{ij}}{\sum_k M_{ik}}$

$C_L \leftarrow \frac{\sum_{ij} \lambda_{ij} (x_i - x_j)(x_i - x_j)^\top}{n} + \nu I_d$

$[V, D] \leftarrow eig(I_G C_L I_G)$ {Eigendecomposition of $I_G C_L I_G$}

$t_1 = \max_z D(z, z) \leq 1$ {Cut-off between eigenvalues smaller and larger than 1}

$t_+ = \{t | t_1 \leq t \leq d\}$, $t_- = \{t | 1 \leq t < t_1\}$

$V_+ \leftarrow V(:, t_+), \quad V_- \leftarrow V(:, t_-), \quad D_- = D(t_-, t_-)$

$B_L = I_G V_- D_-^{-1/2}, \quad B_G = I_G V_+$

end for

Relationship with ICA. Independent component analysis (ICA) can be seen as a density model where $\mathbf{x} = A\mathbf{s}$ and $\mathbf{s}$ has independent components (see, e.g., [17]). In the Parzen windows framework, this corresponds to modelling the density of $\mathbf{s}$ by a product of univariate kernel density estimators [6]. This however causes two problems: first, while this assumption is appropriate in settings such as source separation, it is violated in most settings, and having a multivariate kernel density estimation is preferable. Second, most algorithms are dedicated to finding independent components which are *non-Gaussian*. In the presence of more than one Gaussian dimension, most ICA frameworks become unidentifiable, while our explicit modelling of such Gaussian components allows us to tackle this situation (a detailed analysis of the identifiability of our Parzen/Gaussian model is out of the scope of this paper).

Relationship with NGCA. NGCA [4] makes an assumption similar to ours (they rather assume an additive Gaussian noise on top of a low-dimensional non-Gaussian signal) but uses a projection pursuit algorithm to iteratively find the directions of non-Gaussianity. Unlike in FastICA, the contrast functions used to find the interesting directions can be different for each direction. However, like all projection pursuit algorithms, the identification of interesting directions gets much harder in higher dimensions, as most of them will be almost Gaussian. Our use of a non-parametric density estimator with a log-likelihood cost allows us to globally optimize all directions simultaneously and does not rely on the model being correct. Finally, LCA estimates all its parameters from data as opposed to NGCA which requires the number of non-Gaussian directions to be set.

Escaping local optima. Though our model allows for the modification of the number of dimensions modelled by the Gaussian through the analysis of the spectrum of $C_G^{-1/2} C_L C_G^{-1/2}$, it is sensitive to local optima. It is for instance rare that a dimension modelled by a Gaussian is switched to the Parzen windows estimator. Even though the algorithm will more easily switch from the Parzen windows estimator to the Gaussian model, it will typically stop too early, that is model many dimensions using the Parzen windows estimator rather than the better Gaussian. To solve these issues, we propose an alternate algorithm, *LCA-Gauss-Red*, which explores the space of dimensions modelled by a Gaussian more aggressively using a search algorithm, namely:

1. We run the algorithm LCA - Gauss for a few iterations (40 in our experiments);

2. We then "transfer" some columns from B_L (the Parzen windows model) to B_G (the Gaussian

model), and rerun LCA - Gauss using these new matrices as initialisations;

3. We iterate step 2 using a dichotomic search of the optimal number of dimensions modelled by the Gaussian, until a local optimum is found;

4. Once we have a locally optimum number of dimensions modelled by the Gaussian model, we run LCA - Gauss to convergence.

5 Speeding up LCA

Computing the local covariance matrix of the points using Eq. (3), (4) and (5) has a complexity in $O(dn^2 + d^2n + d^3)$, with d the dimensionality of the data and n the number of training points. Since this is impractical for large datasets, we can resort to sampling to keep the cost linear in the number of datapoints. We may further use low-rank or diagonal approximation to achieve a complexity which grows quadratically with d instead of cubically.

5.1 Averaging a subset of the local covariance matrices

Instead of averaging the local covariances over all datapoints, we may only average them over a subset of datapoints. This estimator is unbiased and, if the local covariance matrices are not too dissimilar, which is the assumption underlying LCA, then its variance should remain small. This is equivalent to using a minibatch procedure: every time we have a new minibatch of size B, we compute its local covariance $\widehat{C_L}$, which is then averaged with the previously computed C_L using

$$C_L \leftarrow \gamma^{\frac{B}{n}} C_L + (1 - \gamma^{\frac{B}{n}})\widehat{C_L} \tag{8}$$

to yield the updated C_L. The exponent B/n is so that γ, the discount factor, determines the weight of the old covariance matrix after an entire pass through the data, which makes it insensitive to the particular choice of batch size. As opposed to many such algorithms where the choice of γ is critical as it helps retaining the information of previous batches, the locality of the EM estimate makes it less so. However, if the number of datapoints used to estimate C_L is not much larger than the dimension of the data, we need to set a higher γ to avoid degenerate covariance matrices. In simulations, we found that using a value of $\gamma = .6$ worked well. Similarly, the size of the minibatch influences only marginally the final result and we found a value of 100 to be large enough.

5.2 Computing the local covariance matrices using a subset of the datapoints

Rather than using only a subset of local covariance matrices, one may also wonder if using the entire dataset to compute these matrices is necessary. Also, as the number of datapoints grows, the chances of overfitting increase. Thus, one may choose to use only a subset of the datapoints to compute these matrices. This will increase the local covariances, yielding a biased estimate of the final result, but may also act as a regulariser. In practice, for very large datasets, one will want the largest neighbourhood size while keeping the computational cost tractable.

Denoting n_i the number of locations at which we estimate the local covariance and n_j the number of neighbours used to estimate this covariance, the cost per update is now $O(d^2[n_i + n_j] + dn_in_j + d^3)$. Since only n_j should grow with n, this is linear in the total number of datapoints.

Though they may appear similar, these are not "landmark" techniques (see, e.g., [11]) as there is still one Gaussian component per datapoint, and the n_i datapoints around which we compute the local covariances are randomly sampled at every iteration.

6 Experiments

LCA has three main properties: first, it transforms the data to make it locally isotropic, thus being well-suited for preprocessing the data before using a clustering algorithm like spectral clustering; second, it extracts relevant, non-Gaussian components in the data; third, it provides us with a good density model through the use of the Parzen windows estimator.

In the experiments, we will assess the performance of the following algorithms: *LCA*, the original algorithm; *LCA-Gauss*, using a multiplicative Gaussian component, as described in Section 4; *LCA-Gauss-Red*, the variant of *LCA-Gauss* using the more aggressive search to find a better number of dimensions to be modelled by the Gaussian component. The MATLAB code for *LCA*, *LCA-Gauss* and *LCA-Gauss-Red* is available at `http://nicolas.le-roux.name/code.html`.

6.1 Improving clustering methods

We first try to solve three clustering problems: one for which the clusters are convex and the direction of interest does not have a Gaussian marginal (Fig. (2), left), one for which the clusters are not convex (Fig. (2), middle), and one for which the directions

of interest have almost Gaussian marginals (Fig. (2), right). Following [1, 2], the data is progressively corrupted by adding dimensions of white Gaussian noise, then whitened. We compare here the clustering accuracy, which is defined as $\frac{100}{n} \min_P \mathrm{tr}(EP)$ where E is the confusion matrix and P is the set of permutations over cluster labels, obtained with the following five techniques:

1. Spectral clustering (SC) [19] on the whitened data (using the code of [8]);
2. SC on the projection on the first two components found by FastICA using the best contrast function and the correct number of components;
3. SC on the data transformed using the metric learnt with LCA;
4. SC on the data transformed using the metric learnt with the product of LCA and a Gaussian;
5. SC on the projection of the data found using NGCA [4] with the correct number of components.

Our choice of spectral clustering stems from its higher clustering performance compared to K-means. Results are reported in Fig. (3). Because of the whitening, the Gaussian components in the first dataset are shrunk along the direction containing information. As a result, even with little noise added, the information gets swamped and spectral clustering fails completely. On the other hand, LCA and its variants are much more robust to the presence of irrelevant dimensions. Though NGCA works very well on the first dataset, where there is only one relevant component, its performance drops quickly when there are two relevant components (note that, for all datasets, we provided the true number of relevant dimensions as input to NGCA). This is possibly due to the deflation procedure which is not adapted when no single component can be clearly identified in isolation. This is in contrast with LCA and its variants which circumvent this issue, thanks to their global optimisation procedure. Note also that *LCA-Gauss* allows us to perform *unsupervised* dimensionality reduction with the same performance as previously proposed supervised algorithms (e.g., [2]).

Figure (4) shows the clustering accuracy on the three datasets for various numbers of EM iterations, one iteration corresponding to Manifold Parzen Windows [26] with a Gaussian kernel whose covariance matrix is the data covariance kernel. As one can see, running the EM algorithm to convergence yields a significant improvement in clustering accuracy. The performance of Manifold Parzen Windows could likely have been improved with a careful initialisation of the original kernel, but this would have been at the expense of the simplicity of the algorithm.

6.2 LCA as a density model

We now assess the quality of LCA as a density model. We build a density model of the USPS digits dataset, a 256-dimensional dataset of handwritten digits. We compared several algorithms:

- An isotropic Parzen windows estimator with the bandwidth estimated using LCA (replacing Σ^* of Eq. (5) by λI so that the two matrices have the same trace);
- A Parzen windows estimator with diagonal metric (equal to the diagonal of Σ^* in Eq. (5);
- A Parzen windows estimator with the full metric as obtained using LCA;
- A single Gaussian model;
- A product of a Gaussian and a Parzen windows estimator (as described in Section 4).

The models were trained on a set of 2000 datapoints and regularized by penalizing the trace of Σ^{-1} (in the case of the last model, both covariance matrices, local and global, were penalized). The regularisation parameter was optimized on a validation set of 1000 datapoints. For the last model, the regularisation parameter of the global covariance was set to the one yielding the best performance for the full Gaussian model on the validation set. Thus, we only had to optimize the regularisation parameter for the local covariance.

The final performance was then evaluated on a set of 3000 datapoints which had not been used for training nor validation. We ran the experiment 20 times, randomly selecting the training, validation and test set each time.

Fig. (5) shows the mean and the standard error of the negative log-likelihood on the test set. As one can see, modelling all dimensions using the Parzen windows estimator leads to poor performance in high dimensions, despite the regulariser and the leave-one-out criterion. On the other hand, LCA-Gauss and LCA-Gauss-Red clearly outperform all the other models, justifying our choice of modelling some dimensions using a Gaussian. Also, as opposed to the previous experiments, there is no performance gain induced by the use of LCA-Gauss-Red as opposed to LCA-Gauss, which we believe stems from the fact that the switch from one model to the other is easier to make when there are plenty of dimensions to choose from. The poor performance of

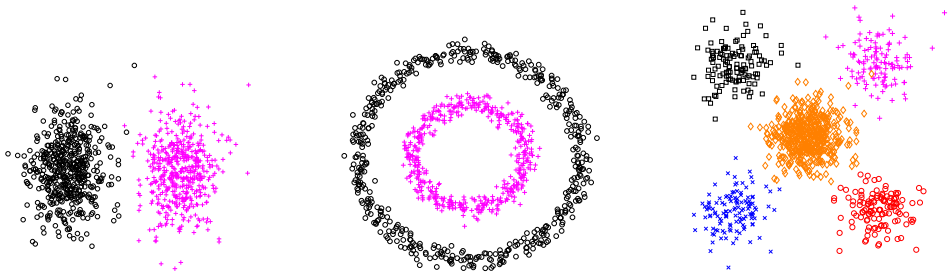

Figure 2: Noise-free data used to assess the robustness of K-means to noise. Left: mixture of two isotropic Gaussians of unit variance and means $[-3, 0]^\top$ and $[3, 0]^\top$. Centre: two concentric circles with radii 1 and 2, with added isotropic Gaussian noise of standard deviation .1. Right: mixture of five Gaussians. The centre cluster contains four times as many datapoints as the other ones.

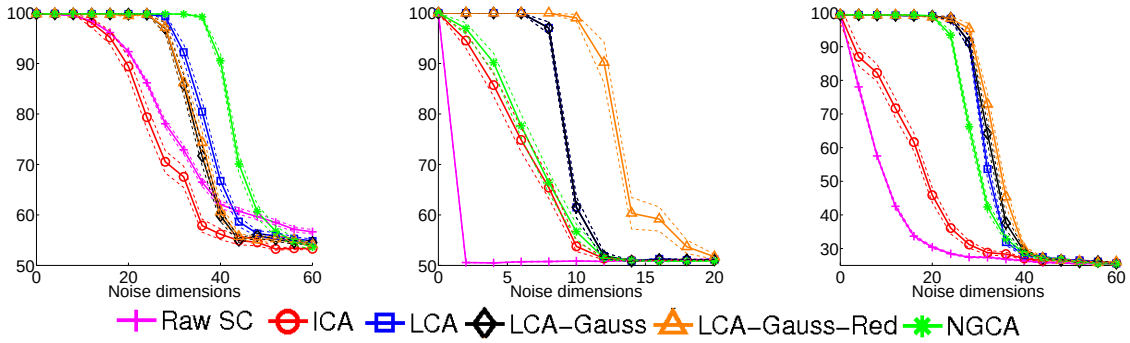

Figure 3: Average clustering accuracy (100% = perfect clustering, chance is 50% for the first two datasets and 20% for the last one) on 100 runs for varying number of dimensions of noise added. The error bars represent one standard error. Left: mixture of isotropic Gaussians presented in Fig. (2) (left). Centre: two concentric circles presented in Fig. (2) (centre). Right: mixture of five Gaussians presented in Fig. (2) (right).

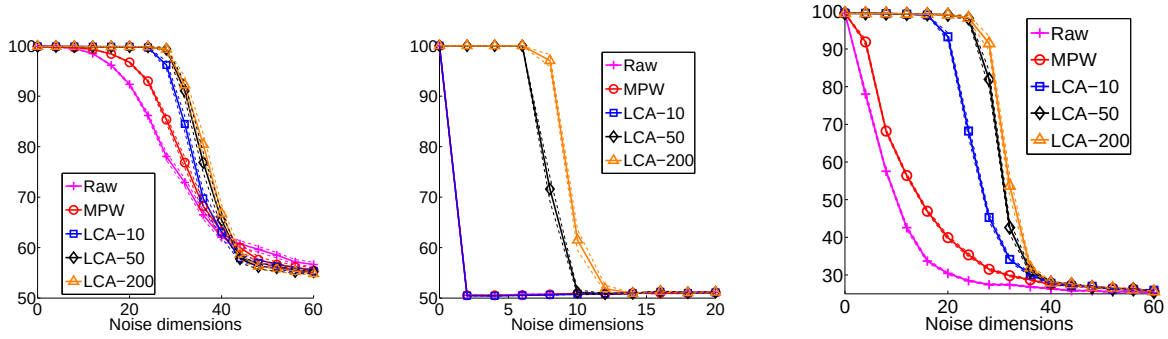

Figure 4: Average clustering accuracy (100% = perfect clustering, chance is 50% for the first two datasets and 20% for the last one) on 100 runs for varying number of dimensions of noise added and varying number of EM iterations in the LCA algorithm (MPW = one iteration). The error bars represent one standard error. Left: mixture of isotropic Gaussians presented in Fig. (2) (left). Centre: two concentric circles presented in Fig. (2) (centre). Right: mixture of five Gaussians presented in Fig. (2) (right).

LCA-Full is a clear indication of the problems suffered by Parzen windows in high dimensions.

6.3 Subsampling

We now evaluate the loss in performance incurred by the use the subsampling procedure described in Section 5, both on the train and test negative log-

	N = 1000	N = 3000	N = 6000	N = 1000	N = 3000	N = 6000
B = 1000	6.43 ± 0.10	2.70 ± 0.06	-0.10 ± 0.03	6.43 ± 0.10	2.80 ± 0.06	-0.17 ± 0.02
B = 3000	6.58 ± 0.07	2.73 ± 0.05	-0.06 ± 0.03	6.54 ± 0.07	2.80 ± 0.06	-0.11 ± 0.02
B = 6000	6.22 ± 0.08	2.21 ± 0.03	0.00 ± 0.01	6.18 ± 0.07	1.98 ± 0.03	0.02 ± 0.02

	N = 1000	N = 3000	N = 6000	N = 1000	N = 3000	N = 6000
B = 1000	2.12 ± 0.16	0.20 ± 0.08	0.65 ± 0.05	2.01 ± 0.17	0.07 ± 0.09	0.15 ± 0.03
B = 3000	2.11 ± 0.16	0.30 ± 0.07	0.46 ± 0.04	2.01 ± 0.17	0.16 ± 0.09	0.09 ± 0.03
B = 6000	1.54 ± 0.16	-0.75 ± 0.07	-0.01 ± 0.01	1.43 ± 0.15	-1.07 ± 0.08	0.01 ± 0.02

Figure 6: Train (top) and test (bottom) negative log-likelihood differences induced by the use of smaller batch and neighbourhood sizes compared to the original model ($\gamma = 0$, $B = 6000$, $N = 6000$) for $\gamma = 0.3$ (left) and $\gamma = 0.6$ (right). A negative value means better performance.

LCA - Isotropic	269.78 ± 0.18
LCA - Diagonal	109.59 ± 0.56
LCA - Full	32.98 ± 0.35
Gaussian	32.27 ± 0.36
LCA - Gauss	19.09 ± 0.39
LCA - Gauss - Red	19.09 ± 0.39

Figure 5: Test negative log-likelihood on the USPS digits dataset, averaged over 20 runs.

likelihoods. For that purpose, we used the USPS digit recognition dataset, which contains 8298 datapoints in dimension 256, which we randomly split into a training set of $n = 6000$ datapoints, using the rest as the test set. We tested the following hyperparameters:

- Discount factor $\gamma = 0.3, 0.6, 0.9$,
- Batch size $B = 1000, 3000, 6000$,
- Neighbourhood size $N = 1000, 3000, 6000$.

Fig. (6) show the log-likelihood differences induced by the use of smaller batch sizes and neighbourhood sizes. For each set of hyperparameters, 20 experiments were run using different training and test sets, and the means and standard errors are reported. The results for $\gamma = 0.9$ were very similar and are not included due to space constraints.

Three observations may be made. First, reducing the batchsize has little effect, except when γ is small. Second, reducing the neighbourhood size has a regularizing effect at first but drastically hurts the performance if reduced too much. Third, the value of γ, the discount factor, has little influence, but larger values proved to yield more consistent test performance, at the expense of slower convergence. The consistency of these results shows that it is safe to use subsampling (with values of $\gamma = 0.6$, $B = 100$ and $N = 3000$, for instance) especially if the training set is very large.

7 Conclusion

Despite its importance, the learning of local or global metrics is usually an overseen step in many practical algorithms. We have proposed an extension of the general bandwidth selection problem to the multidimensional case, with a generalisation to the case where several components are Gaussian. Additionally, we proposed an approximate scheme suited to large datasets which allows to find a local optimum in linear time. We believe LCA can be an important preprocessing tool for algorithms relying on local distances, such as manifold learning methods or many semi-supervised algorithms. Another use would be to cast LCA within the mean-shift algorithm, which finds the modes of the Parzen windows estimator, in the context of image segmentation [10]. In the future, we would like to extend this model to the case where the metric is allowed to vary with the position in space, to account for more complex geometries in the dataset.

Acknowledgements

Nicolas Le Roux and Francis Bach are supported in part by the European Research Council (SIERRA-ERC-239993). We would also like to thank Warith Harchaoui for its valuable input.

References

[1] F. Bach and Z. Harchaoui. Diffrac: a discriminative and flexible framework for clustering. In *Advances in Neural Information Processing Systems 20*, 2007.

[2] F. Bach and M. I. Jordan. Learning spectral clustering. In *Advances in Neural Information Processing Systems 16*, 2004.

[3] Y. Bengio, H. Larochelle, and P. Vincent. Non-local manifold Parzen windows. In *Advances in Neural Information Processing Systems 18*, pages 115–122. MIT Press, 2006.

[4] G. Blanchard, M. Kawanabe, M. Sugiyama, V. Spokoiny, and K.-R. Müller. In search of non-Gaussian components of a high-dimensional distribution. *J. Mach. Learn. Res.*, 7:247–282, December 2006.

[5] J.M. Borwein and A.S. Lewis. Convex analysis and nonlinear optimization, theory and examples, 2000.

[6] R. Boscolo, H. Pan, and V.P. Roychowdhury. Independent component analysis based on nonparametric density estimation. *Neural Networks, IEEE Transactions on*, 15(1):55–65, 2004.

[7] A. Bowman. An alternative method of cross-validation for the smoothing of density estimates. *Biometrika*, 71(2):pp. 353–360, 1984.

[8] W.-Y. Chen, Y. Song, H. Bai, C.-J. Lin, and E. Y. Chang. Parallel spectral clustering in distributed systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(3):568–586, 2011.

[9] Y.-S. Chow, S. Geman, and L.-D. Wu. Consistent cross-validated density estimation. *The Annals of Statistics*, 11(1):pp. 25–38, 1983.

[10] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:603–619, 2002.

[11] V. De Silva and J.B. Tenenbaum. Sparse multidimensional scaling using landmark points. *Technology*, pages 1–41, 2004.

[12] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):pp. 1–38, 1977.

[13] R.P.W. Duin. On the choice of smoothing parameters for Parzen estimators of probability density functions. *Computers, IEEE Transactions on*, C-25(11):1175 –1179, 1976.

[14] T. Duong and M. Hazelton. Cross-validation bandwidth matrices for multivariate kernel density estimation. *Scandinavian Journal of Statistics*, 32(3):pp. 485–506, 2005.

[15] A. Globerson and S. Roweis. Metric learning by collapsing classes. In *Advances in Neural Information Processing Systems*, volume 18, pages 451–458. MIT Press, 2006.

[16] J. Goldberger, S. Roweis, G.E. Hinton, and R. Salakhutdinov. Neighbourhood components analysis. In *Advances in Neural Information Processing Systems 17*, 2004.

[17] A. Hyvärinen, J. Karhunen, and E. Oja. *Independent component analysis*. Wiley, New York, 2001.

[18] M. C. Jones and D. A. Henderson. Maximum likelihood kernel density estimation: On the potential of convolution sieves. *Comput. Stat. Data Anal.*, 53:3726–3733, August 2009.

[19] A. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14*, 2001.

[20] B. Park and J. S. Marron. Comparison of data-driven bandwidth selectors. *Journal of the American Statistical Association*, 85(409):pp. 66–72, 1990.

[21] E. Parzen. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 33(3):pp. 1065–1076, 1962.

[22] M. Rosenblatt. Remarks on some nonparametric estimates of a density function. *The Annals of Mathematical Statistics*, 27(3):pp. 832–837, 1956.

[23] M. Rudemo. Empirical choice of histograms and kernel density estimators. *Scandinavian Journal of Statistics*, 9(2):pp. 65–78, 1982.

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

[25] B. W. Silverman. Density estimation for statistics and data analysis. 1986.

[26] P. Vincent and Y. Bengio. Manifold Parzen windows. In *Advances in Neural Information Processing Systems 15*, pages 825–832. MIT Press, 2002.

[27] E. Xing, A. Ng, M. I. Jordan, and S. Russell. Distance metric learning with application to clustering with side-information. In *Advances in Neural Information Processing Systems 14*, 2002.

[28] L. Zelnik-Manor and P. Perona. Self-tuning spectral clustering. In *Advances in Neural Information Processing Systems 17*, 2004.

Appendix

We prove here Proposition 1.

Proof If M_1 is singular, then the minimum value is $-\infty$, because we can have $B_G^\top M_1 B_G$ bounded while $B_G B_G^\top$ tends to $+\infty$ (for example, if $d_1 = 1$, and u_1 is such that $M_1 u_1 = 0$, select $B_G = \lambda u_1$ with $\lambda \to +\infty$). The reasoning is similar for M_2.

We thus assume that M_1 and M_2 are invertible. We consider the eigendecomposition of $M_1^{-1/2} M_2 M_1^{-1/2} = U \operatorname{Diag}(e) U^\top$, which corresponds to the generalized eigendecomposition of the pair (M_1, M_2).

Denoting $A_2 = U^\top M_1^{1/2} B_L$ and $A_1 = U^\top M_1^{1/2} B_G$, we have:

$$\begin{aligned}
& \operatorname{tr} B_G^\top M_1 B_G + \operatorname{tr} B_L^\top M_2 B_L - \log\det(B_G B_G^\top + B_L B_L^\top) \\
= \; & \operatorname{tr} A_1^\top A_1 + \operatorname{tr} A_2^\top \operatorname{Diag}(e) A_2 \\
& - \log\det(A_1 A_1^\top + A_2 A_2^\top) + \log\det M_1 \\
= \; & \operatorname{tr} A_1^\top A_1 + \operatorname{tr} A_2^\top \operatorname{Diag}(e) A_2 - \log\det(A_2^\top A_2) \\
& - \log\det(A_1^\top (I - A_2 (A_2^\top A_2)^{-1} A_2^\top) A_1) + \log\det M_1 \; .
\end{aligned}$$

By taking derivatives with respect to A_1, we get

$$A_1 = (I - \Pi_2) A_1 (A_1^\top (I - \Pi_2) A_1)^{-1} \; , \qquad (9)$$

with $\Pi_2 = A_2 (A_2^\top A_2)^{-1} A_2^\top$. By left-multiplying both sides of Eq. (9) by $A_2^\top$, we obtain

$$A_2^\top A_1 = 0 \; .$$

By left-multiplying by $A_1^\top$, we get

$$A_1^\top A_1 = I \; .$$

Thus, we now need to minimize with respect to A_2 the following cost function

$$d_1 + \operatorname{tr} A_2^\top \operatorname{Diag}(e) A_2 - \log\det(A_2^\top A_2) + \log\det M_1$$

Let s be the vector of singular values of A_2, ordered in *decreasing* order and let the e_i be ordered in *increasing* order. We have:

$$\operatorname{tr} \operatorname{Diag}(e) A_2 A_2^\top = -\operatorname{tr}(-\operatorname{Diag}(e) A_2 A_2^\top) \geqslant \sum_i e_i s_i^2 \; ,$$

with equality if and only if the eigenvectors of $A_2 A_2^\top$ are aligned with the ones of $\operatorname{Diag}(e)$ (the $-e_i$ being also in decreasing order) (Theorem 1.2.1, [5]).

Thus, we have $A_2 A_2^\top = \operatorname{diag}(s)^2$ with only d_2 nonzero elements in s. Let J_2 be the index of non zero-elements. We thus need to minimize

$$d_1 + \log\det M_1 + \sum_{j \in J_2} (e_j s_j^2 - \log s_j^2) \; ,$$

with optimum $s_j^2 = e_j^{-1}$ and value:

$$d_1 + d_2 + \log\det M_1 + \sum_{j \in J_2} \log e_j \; .$$

Thus, we need to take J_2 corresponding to the smallest eigenvalues e_j. If we also optimize with respect to d_2, then J_2 must only contain the elements smaller than 1. ■