CENTER FOR
MACHINE PERCEPTION

CZECH TECHNICAL
UNIVERSITY

RESEARCH REPORT

Robust manifold learning

Dick de Ridder[1] and Vojtěch Franc

dick@ph.tn.tudelft.nl, xfrancv@cmp.felk.cvut.cz

[1] Pattern Recognition Group, Dept. of Imaging Science and Technology
Faculty of Applied Sciences, Delft University of Technology
Lorentzweg 1, 2628 CJ Delft, The Netherlands

CTU–CMP–2003-08

May 1, 2003

Available at
ftp://cmp.felk.cvut.cz/pub/cmp/articles/franc/deRidder-TR-2003-08.pdf

This work was sponsored by the European Union under Project
ICA 1-CT-2000-7000 and the Czech Ministry of Education under
Project MSM 212300013

Center for Machine Perception, Department of Cybernetics
Faculty of Electrical Engineering, Czech Technical University
Technická 2, 166 27 Prague 6, Czech Republic
fax +420 2 2435 7385, phone +420 2 2435 7637, www: http://cmp.felk.cvut.cz

Abstract

Probabilistic subspace mixture models, as proposed over the last few years, are interesting methods for learning image manifolds, i.e. nonlinear subspaces of spaces in which images are represented as vectors by their grey-values. Their lack of a global mapping can be remedied by a recently developed method based on locally linear embedding, called locally linear coordination. However, for many practical applications, where outliers are common, this method lacks the necessary robustness. Here, the idea of robust mixture modelling by t-distributions is combined with probabilistic subspace mixture models. The resulting robust subspace mixture model is shown experimentally to give advantages in density estimation and classification of image data sets. It also solves the robustness problems of locally linear coordination, by introducing a weighted reformulation of the embedding step.

Contents

1 Introduction

The last few years have seen an increasing interest in subspace methods, specifically when applied to image data. Many approaches approximate image data, represented by grey-values as vectors in high-dimensional spaces, using linear subspaces. These can be found using a number of well-known subspace techniques, such as principal component analysis (PCA) or factor analysis (FA). The subspaces themselves can be used as models for the projected image data (so-called appearance-based methods), or they can be used as a form of feature extraction to ease further processing.

Work has also proceeded on modelling nonlinear subspaces, i.e. curves or manifolds. Methods to do so broadly fall into three main categories:

- fitting principal curves and surfaces to the data;
- embedding the data;
- modelling the data by (constrained) mixture models.

Methods such as principal curves and surfaces [3, 10] are elegant methods, yet due to their complexity are hard to use to find manifolds of dimensions more than 2 or 3. Manifolds found by neural networks [17] suffer from all neural network problems.

Embeddings, such as multi-dimensional scaling (MDS), locally linear embedding (LLE) [21, 24] and ISOMAP [27], are more flexible in application. They are usually based on some (heuristic) properties to be preserved during the mapping, which makes their operation insightful. However, they do not give a probabilistic model of the data and often require a large amount of computation.

The latter group of methods, mixture models, has received much attention. Most mixture models approximate a manifold using a relatively small number of localised subspaces, which are modelled by Gaussians with restricted covariance matrices [9, 28], possibly with a constraint on the relations between the models [1, 22]. Their popularity is due to the fact that they are usually trained by the well-understood expectation-maximisation (EM) algorithm, for which a large body of literature exists. Methods of optimising the number of models to use, be it by using greedy algorithms or Bayesian learning, are readily available.

However, mixture models still have their drawbacks. First, the EM algorithm can only be guaranteed to converge to a *local* optimum. Second, as the models are density-based, dense data sets are necessary to train them. Third, use of Gaussian densities means these methods are not robust to outliers. Especially when mixture models are used for subsequent classification

(e.g. [6, 8, 11]) this can lead to performance degradation. Although supervised mixture model techniques exist (e.g. [13, 23]), these algorithms are complex, computationally intensive and/or limited to specific data models.

Finally, a major drawback of subspace mixture models for manifold learning is that they do not result in a global mapping from a high-dimensional space to a low-dimensional space. Although some methods have been formulated for achieving a global mapping by coordinating the subspaces in the mixture [22, 29]), these are known to have problems converging [2]. On the other hand, embeddings such as the aforementioned locally linear embedding (LLE, [24]) do achieve a global mapping, but at high computational cost. A recently proposed method reduces this cost by mapping not individual samples, but samples pre-clustered by a mixture of local subspaces. The method is called locally linear coordination (LLC, [26]). A very similar method was, at the same time, proposed by Brand [2].

LLE and LLC will be briefly introduced in Section 2. Though fast, LLC is highly non-robust against outliers, as a single model misfit may cause the global mapping to be incorrect. In Section 3, an alternative robust subspace mixture model based on t-distributions is therefore proposed. It is shown to give good results in density estimation on a toy data set and in a classification problem. In Section 4, the robust subspace mixture model is then applied to in a slight reformulation of the original locally linear coordination. The resulting algorithm, robust LLC (RLLC), is experimentally shown to give better results than ordinary LLC based on mixtures of principal component analysers.

The paper ends with conclusions and an outlook to further applications.

2 Manifold learning

An interesting, recently proposed method for manifold learning is locally linear embedding (LLE, [24]). This method will be discussed in Section 2.1. It suffers from a number of problems, some of which are addressed by a recent variant, which couples LLE with mixtures-of-subspaces. This algorithm, locally linear coordination (LLC) will be discussed in section 2.2.

2.1 Locally linear embedding (LLE)

As input, LLE takes a set of n d-dimensional vectors (each of which may represent an image, for example) assembled in a matrix $\mathbf{X}$ of size $d \times n$. Its output is a set of n m-dimensional vectors ($m \ll d$) assembled in a matrix $\mathbf{Y}$ of size $m \times n$, where the ith column vector of $\mathbf{Y}$ corresponds to the ith column

vector of $\mathbf{X}$. First, the $n \times n$ squared distance matrix $\mathbf{\Delta}$ between all samples is constructed. For each sample $\mathbf{x}_i$, $i = 1, \ldots, n$, its q nearest neighbours are then sought; their indices are stored in an $n \times q$ matrix $\mathbf{\Omega}$, such that Ω_{ij} is the index of the j-nearest neighbour of sample $\mathbf{x}_i$.

In the first step, each sample $\mathbf{x}_i$ is approximated by a weighted linear combination of its K nearest neighbours, making use of the assumption that neighbouring samples will lie on a locally linear patch of the nonlinear manifold. To find the reconstruction weight matrix $\mathbf{W}$, where $\mathbf{W}_{i\Omega_{ij}}$ contains the weight of neighbour j in the reconstruction of sample $\mathbf{x}_i$, the following expression has to be minimised w.r.t. $\mathbf{W}$ [24]:

$$\varepsilon_I(\mathbf{W}) = \sum_{i=1}^{n} \parallel \mathbf{x}_i - \sum_{j=1}^{q} W_{i\Omega_{ij}} \mathbf{x}_{\Omega_{ij}} \parallel^2, \tag{1}$$

subject to the constraint $\sum_{j=1}^{q} W_{i\Omega_{ij}} = 1$. It is easy to show that each weight can be calculated individually [24]. For each sample $\mathbf{x}_i$, construct a matrix $\mathbf{Q}$ with $Q_{jm} = \frac{1}{2}(\Delta_{i\Omega_{ij}} + \Delta_{i\Omega_{im}} - \Delta_{\Omega_{ij}\Omega_{im}})$. Let $\mathbf{R} = (\mathbf{Q} + r\mathbf{I})^{-1}$, where r is a suitably chosen regularisation constant (see [7]). Then $W_{i\Omega_{ij}} = (\sum_{l=1}^{q} R_{jl})/(\sum_{l,p=1}^{q} R_{lp})$.

In the second and final step, the weights stored in $\mathbf{W}$ are kept fixed and an embedding in $\mathbb{R}^m$ is found by minimising w.r.t. $\mathbf{Y}$:

$$\varepsilon_{II}(\mathbf{Y}) = \sum_{i=1}^{n} \parallel \mathbf{y}_i - \sum_{j=1}^{q} W_{i\Omega_{ij}} \mathbf{y}_{\Omega_{ij}} \parallel^2 . \tag{2}$$

This minimisation problem can be solved by introducing the constraint that the embedded data should have unit covariance, i.e. $\frac{1}{n}\mathbf{Y}\mathbf{Y}^T = \mathbf{I}$ (otherwise, $\mathbf{Y} = \mathbf{0}$ would minimise (2)). As a result, (2) is minimised by carrying out an eigen-decomposition of the matrix $\mathbf{M} = (\mathbf{I} - \mathbf{W})^T(\mathbf{I} - \mathbf{W})$ [24]. The eigenvectors corresponding to the 2nd to $(m+1)^{st}$ smallest eigenvalues then form the final embedding $\mathbf{Y}$; the eigenvector corresponding to the smallest eigenvalue corresponds to the mean of the embedded data, and can be discarded to obtain an embedding centered at the origin.

After embedding, a new sample can be mapped quickly by calculating the weights for reconstructing it by its q nearest neighbours in the training set, as in the first step of LLE. Its embedding is then found by taking a weighted combination of the embeddings of these neighbours [7, 24].

LLE has been shown to be useful for analysis of high-dimensional data sets [7, 15, 16, 21]. A typical example is visualisation of a sequence of images, e.g. showing a person's face slowly rotating from left to right. For such data sets, LLE finds embeddings in which the individual axes correspond (roughly)

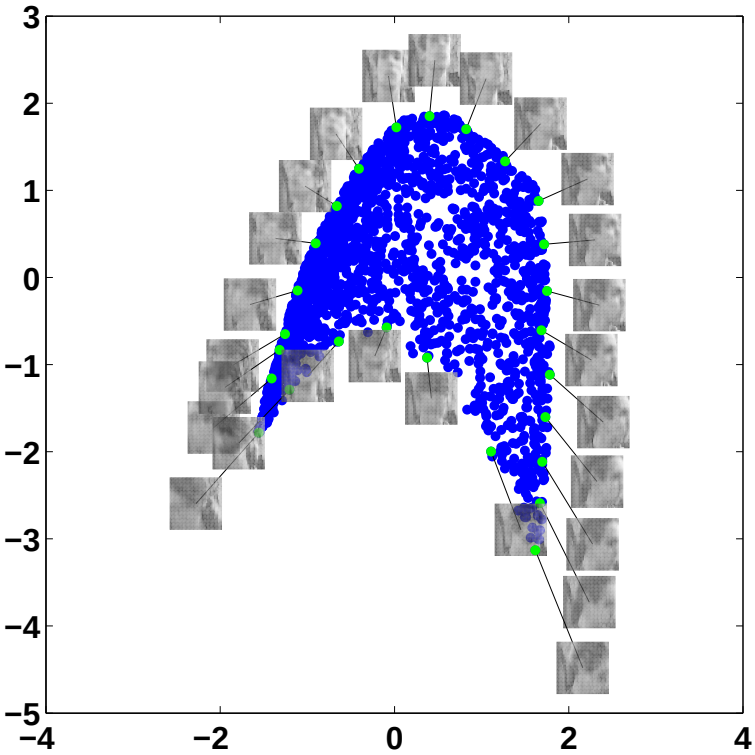


Figure 1: A set of 2000 face images, mapped down to 2D using LLE. Before applying LLE, a global PCA mapping was used to reduce the number of dimensions to 100. Other parameters: $q = 25$, $m = 2$, automatic regularisation [7].

to the small number of degrees of freedom present in the data. Figure 1 illustrates this.

LLE as described here suffers from a number of problems:

- the choice of parameters influences the end result to a large extent: the number of neighbours to use, q; the number of dimensions to map down to, m; and the amount of regularisation to apply the local Gram matrix R.
- the second step entails an eigendecomposition of an $n \times n$ matrix, which for large n (say, $n > 5000$) becomes infeasible in terms of time and space complexity.

To addres the first problem, a number of improvements were proposed in [7]. The number of neighbours q still has to be set, but regularisation parameter r and the intrinsic dimensionality m can be estimated if a desired percentage-of-variance retained $0 < v \leq 1$ is specified.

The second problem is also addressed in [7], by giving a measure of "necessity" for each sample, based on a local linearity estimate. The data set can then be pruned by removing unnecessary samples. This approach suffers from the obvious problems of dependence on the local linearity estimate, and the question of how to decide how many samples one needs to retain to still obtain a good mapping. A more elegant solution is to use mixtures-of-subspaces to obtain a reduction in the computation needed. This will be discussed next.

2.2 Locally linear coordination (LLC)

Teh and Roweis [26] proposed to use LLE in conjunction with mixtures-of-subspaces. Their algorithm works as follows.

First, the first stage of LLE is applied find the reconstruction weight matrix $\mathbf{W}$, using q neighbours in $\mathbb{R}^d$. Regularise with r_{LLE}, or use automatic regularisation. Next, a mixture-of-subspaces is trained, such as a mixture of principal component analysers (MPPCA, [28]) or a mixture of factor analysers (MFA, [9]); these models are discussed in more detail in Section 3.1. Such a subspace mixture model have k models of m dimensions each. It is regularised with r_{PCA}, or the automatic regularisation found by LLE is used (applicable when $n/k \approx q$) is used.

Each vector $\mathbf{x}_i$ to be mapped can now be expressed by its position *inside* each of the subspaces, i.e. its "internal coordinates". For the model in (8), this is:

$$\mathbf{t}_{ij} = \mathbf{A}_j^T(\mathbf{x}_i - \boldsymbol{\mu}_j) \tag{3}$$

where $\mathbf{A}_j$ contains the basis vectors of subspace j as its columns, and $\boldsymbol{\mu}_j$ is the origin of subspace j.

The key of LLC is that the vectors $\mathbf{y}_i$ looked for can be seen as weighted, rotated and shifted versions of the $\mathbf{t}_i$. LLE can find the rotations and shifts, for the entire subspace at a time. The weights are given by the responsibilities:

$$\mathbf{y}_i = \sum_{j=1}^{k} r_{ij}(\mathbf{L}_j\mathbf{t}_{ij} + \mathbf{o}_j) \tag{4}$$

where r_{ij} is the responsibility of model j for vector $\mathbf{x}_i$ (18), and $\mathbf{t}_{ij}$ is the vector of internal coordinates of vector $\mathbf{x}_i$ in subspace j. $\mathbf{L}_j$ is the $m \times m$ rotation matrix for subspace j, and $\mathbf{o}_j$ is its offset.

By rewriting, the previous expression can be split into a known part (r_{ij}, $\mathbf{t}_{ij}$) and an unknown part ($\mathbf{L}_j$, $\mathbf{o}_j$):

$$\mathbf{y}_i = \sum_{j=1}^{k} \left(\left[r_{ij} \mathbf{t}_{ij}^T | 1 \right] \right) \mathbf{L}'_j \tag{5}$$

where the 1 is a scalar bias term, and $\mathbf{L}'_j$ is an $(m+1) \times m$ matrix containing the $m \times m$ rotation matrix and the offset.

Now the entire $n \times m$ matrix $\mathbf{Y}$ can be expressed as $\mathbf{Y} = \mathbf{UP}$, where $\mathbf{U}$ contains the first term in the previous equation and $\mathbf{P}$ the unknown parameter matrices $\mathbf{L}'_j$, repacked to a $j(m+1) \times m$ matrix:

$$\mathbf{Y} = \left[R_1 \mathbf{t}_1^T 1 | R_2 \mathbf{t}_2^T 1 | \ldots | R_k \mathbf{t}_k^T 1 \right] \mathbf{P} = \mathbf{UP} \tag{6}$$

Given this, the second stage of LLE (2) can be written as minimising

$$\begin{aligned} \operatorname{tr}\left(\mathbf{Y}^T (\mathbf{I} - \mathbf{W})^T (\mathbf{I} - \mathbf{W}) \mathbf{Y} \right) &= \operatorname{tr}\left(\mathbf{P}^T \left[\mathbf{U}^T (\mathbf{I} - \mathbf{W})^T (\mathbf{I} - \mathbf{W}) \mathbf{U} \right] \mathbf{P} \right) \\ &= \operatorname{tr}\left(\mathbf{P}^T \mathbf{G} \mathbf{P} \right) \end{aligned} \tag{7}$$

under constraints $\frac{1}{n} \mathbf{1}^T \mathbf{UP} = 0$ (which can again be done by dropping the first smallest eigenvector) and $\frac{1}{n} \mathbf{P}^T \mathbf{U}^T \mathbf{UP} = \mathbf{P}^T \mathbf{HP} = \mathbf{I}$. This leads to a generalised eigenproblem: solving $\mathbf{Gv} = \lambda \mathbf{Hv}$.

Note that $\mathbf{G}$ and $\mathbf{H}$ are now $j(m+1) \times j(m+1)$ matrices, which in general will be much smaller than the original $n \times n$ matrix $\mathbf{W}$. However, $\mathbf{G}$ will no longer be sparse as it was in LLE.

LLC has a few more parameters than the original LLE: the number of models k to use, their dimension m, and the regularisation r_{PCA} needed for training the mixture model. Usually, one will pick m to be equal to the number of dimensions we want to map down to (which can be found automatically by specifying the percentage of variance to retain, see [7]) – although in some of the experiments in [26], this is not the case. Finally, k might be chosen such that $n/k \approx q$, i.e. samples are "summarised" by subspaces at the same scale that LLE operates. However, as mixtures-of-subspaces are density-based, they might need more than q samples (on average) to be estimated well.

LLC couples in a very interesting way two ideas, those of mixtures-of-subspaces and embedding. The mixture models ease the application of LLE, whereas LLE provides a way of post-coordinating the mixture models (that is, aligning them to provide a global mapping).

2.3 Experiments

2.3.1 3D toy data

LLC was first applied to a toy data set, samples distributed on a 3D sine-wave [7]. The subspace mixture model used was a mixture of probabilistic principal component analysers (MPPCA). Figure 2 shows the results. For low noise levels (Figures 2(a)-(c)), the LLC mapping found is even better than that found by LLE: there is less skew.

However, LLC is extremely sensitive to noise. When the local noise level is increased, mappings quickly become very poor (Figures 2(d)-(f)). The same holds when just a few outliers (10, i.e. just 2%) are added (Figures 2(g)-(i)). Note that LLC is far more sensitive to outliers than LLE, for which results are shown in the same figure. This non-robustness of LLC may be alleviated by employing more robust methods of fitting mixtures of subspaces. This will be addressed in Section 3.

2.3.2 Rotating face images

In a larger experiment, the face data set used in Figure 1 was embedded using LLC, both randomly initialised and initialised by k-means. However, initialisation by k-means never led to convergence, so these results are not reported. Results for various values of k are shown in Figure 3. The subspace mixture models were regularised using the value found by LLE; all other settings were kept the same as before.

Note that LLC at first glance gives good embeddings, in that the data is "spread out" over the 2D plane. However, careful inspection shows that it not unfolded the manifold completely (in contrast to LLE). For example, if the face is traced around the edge the manifold, it is nowhere facing upwards; this indicates that these images must have been mapped somewhere inside the convex hull of the overall mapped data set. It seems that even for this (non-noisy) data set, LLC is harder to apply than LLE.

3 A robust subspace mixture model

In this section, the problem of robustness to outliers will be addressed. First, an overview of probabilistic subspace models will be given in Section 3.1 and the algorithm for fitting a mixture of PCA subspaces will be discussed in Section 3.2. Inspired by recent developments in statistics [20], a mixture of subspace t-distributions will be proposed in Section 3.4. This method can be shown to be much more robust against outliers than methods based on

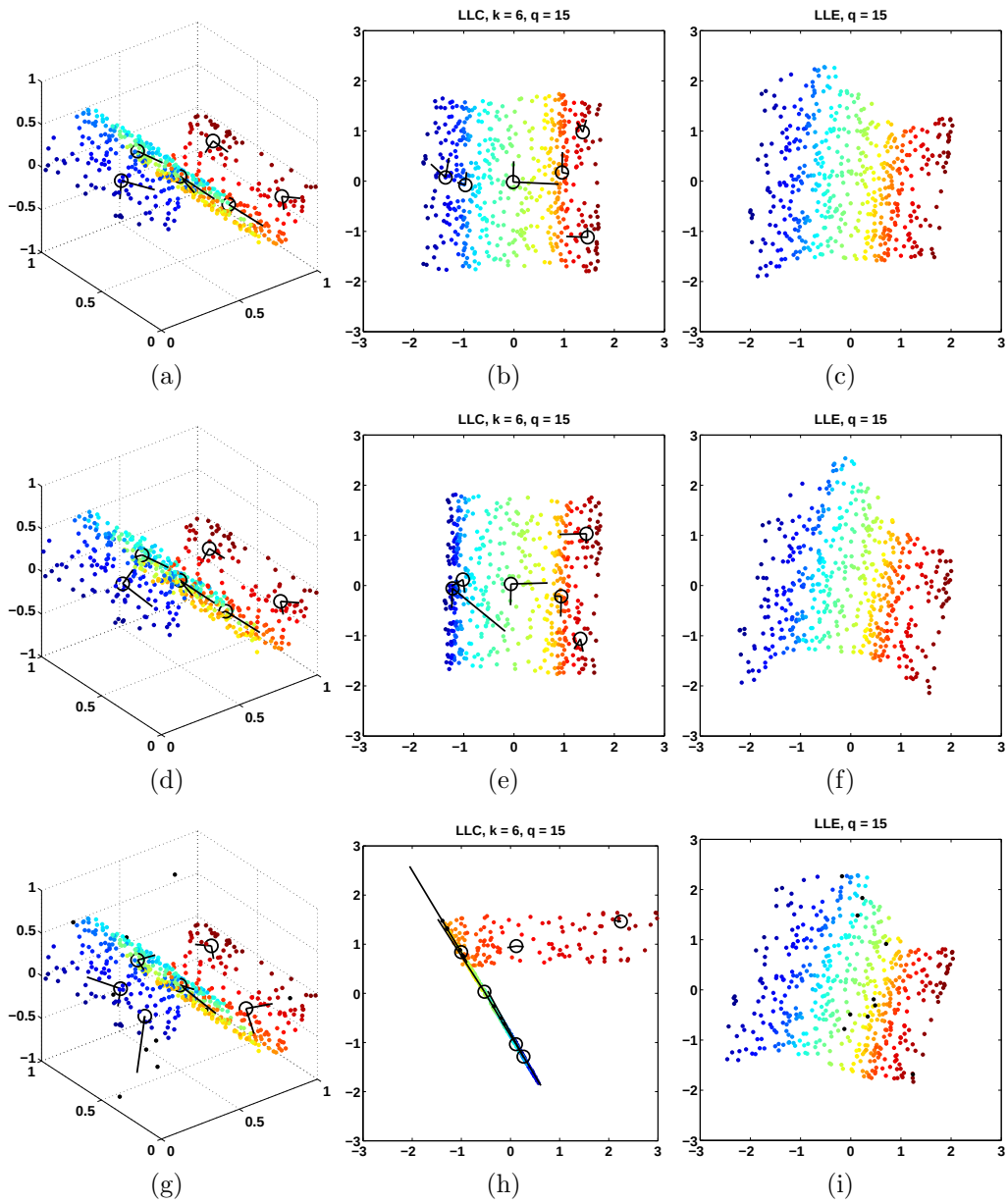


Figure 2: 500 samples on a 3D sine wave (left column) mapped to 2D using LLC (middle column) and LLE (right column). (a)-(c) with added Gaussian noise, $\sigma = 0.01$. (d)-(f) with added Gaussian noise, $\sigma = 0.025$. (g)-(i) with added Gaussian noise, $\sigma = 0.01$ and 10 uniformly distributed outliers. Other parameters: $k = 6$, $q = 15$, $m = 2$, r_{LLE} automatic, $r_{PCA} = 0$.

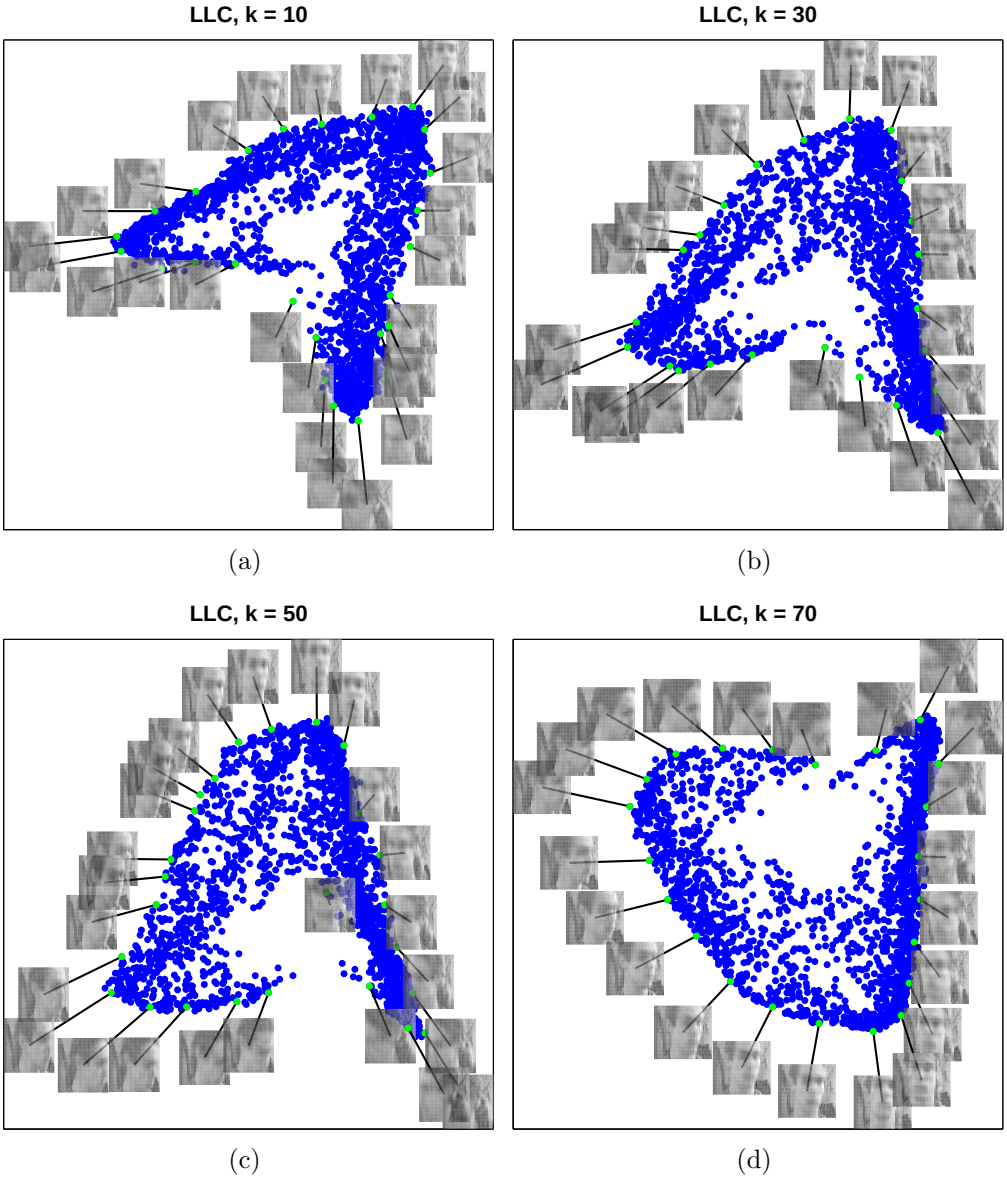


Figure 3: A set of 2000 face images, mapped down to 2D using LLC with randomly initialiased MPPCA, for various values of k. Before applying LLC, a global PCA mapping was used to reduce the number of dimensions to 100. Other parameters: $q = 25$, $m = 2$, automatic regularisation.

Gaussian densities. Its usefulness will be demonstrated on a toy example and a real-world problem in Section 3.5.

3.1 Probabilistic subspace models

The two main models for mixtures of subspaces are mixtures of probabilistic principal component analysers (PPCA, [28]) and mixtures of factor analysers (FA, [9]). Both are based on a subspace model in which an observed variable $\mathbf{x} \in \mathbb{R}^d$ is generated by a some low-dimensional variable $\mathbf{s} \in \mathbb{R}^m$, where typically $m \ll d$:

$$\mathbf{x} = \mathbf{A}\mathbf{s} + \boldsymbol{\mu} + \boldsymbol{\varepsilon} \tag{8}$$

The low-dimensional variable is thus shifted w.r.t. to the center $\boldsymbol{\mu}$ of the subspace, mapped into the high-dimensional space by a projection operator $\mathbf{A}$, and i.i.d. Gaussian noise $\boldsymbol{\varepsilon}$ is added. The distributions of the variables are:

$$p(\mathbf{s}) = N(\mathbf{s}; \mathbf{0}, \mathbf{I}) \tag{9}$$

$$p(\boldsymbol{\varepsilon}) = N(\boldsymbol{\varepsilon}; \mathbf{0}, \boldsymbol{\Psi}) \tag{10}$$

$$p(\mathbf{x}|\mathbf{s}) = N(\mathbf{x}; \mathbf{A}\mathbf{s} + \boldsymbol{\mu}, \boldsymbol{\Psi}) \tag{11}$$

As both $p(\mathbf{s})$ and $p(\mathbf{x}|\mathbf{s})$ are Gaussian, the marginal distribution of $\mathbf{x}$ will be so as well:

$$p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{s})p(\mathbf{s})d\mathbf{s} \tag{12}$$

whose mean and covariance matrix can easily be found from (8):

$$E(\mathbf{x}) = \boldsymbol{\mu} \quad \text{and} \quad E(\mathbf{x}\mathbf{x}^T) = \mathbf{A}E(\mathbf{s}\mathbf{s}^T)\mathbf{A}^T + \boldsymbol{\Psi} = \mathbf{A}\mathbf{A}^T + \boldsymbol{\Psi} \tag{13}$$

i.e., $\mathbf{x}$ is distributed as:

$$p(\mathbf{x}) = N(\mathbf{x}; \boldsymbol{\mu}, \mathbf{A}\mathbf{A}^T + \boldsymbol{\Psi}) \tag{14}$$

The difference between principal component analysis and factor analysis lies in the assumed noise model. FA assumes $\boldsymbol{\Psi}$ to be a diagonal matrix, whereas PPCA assumes $\boldsymbol{\Psi}$ to be a multiple of the identity matrix, $\sigma^2\mathbf{I}$. FA thus models the individual noise level in each of the dimensions (i.e. pixels, for image data), PPCA assumes all dimensions have an equal noise level. In this paper, only PPCA will be discussed; however, all observations are equally applicable to FA.

3.2 Mixtures of PPCAs (MPPCA)

Tipping and Bishop [28] derived an EM algorithm for maximum likelihood learning of a mixture of k PPCA's. It will be summarised here in enough detail to allow the reader to understand the effects of using a different density model later on; for a more detailed derivation, see the original paper. The probability of observing a sample $\mathbf{x}$ under a mixture model with k models is: is given by:

$$p(\mathbf{x}) = \sum_{j=1}^{k} \pi_j p(\mathbf{x}|j) \tag{15}$$

where the π_j are mixing parameters, for which $\sum_{j=1}^{k} \pi_j = 1$, and $p(\mathbf{x}|j)$ is the probability of $\mathbf{x}$ under model j given by (14). The log-likelihood of observing a data set $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ is:

$$\mathcal{L}(\mathbf{X}) = \log \prod_{i=1}^{n} p(\mathbf{x}_i) = \sum_{i=1}^{n} \log \sum_{j=1}^{k} \pi_j p(\mathbf{x}_i|j) \tag{16}$$

To maximise (16) w.r.t. the parameters $\{\pi_j, \boldsymbol{\mu}_j, \mathbf{A}_j, \sigma_j^2\}$, $j = 1, \ldots, k$, a hidden indicator variable z_{ij} is introduced, which is 1 when sample $\mathbf{x}_i$ is generated by model j, and 0 otherwise; the mixing parameters π_j are then equal to $p(z_{ij} = 1)$. The complete-data distribution can now be written as:

$$p(\mathbf{x}, \mathbf{s}, \mathbf{z}) = \prod_{i=1}^{n} p(\mathbf{x}_i|\mathbf{s}_i, \mathbf{z}_i) p(\mathbf{s}_i|\mathbf{z}_i) p(\mathbf{z}_i) = \prod_{i=1}^{n} \prod_{j=1}^{k} \left[\pi_j p(\mathbf{x}_i|\mathbf{s}_i, j) p(\mathbf{s}_i|j)\right]^{z_{ij}} \tag{17}$$

where n is the number of samples in the dataset, $p(\mathbf{x}_i|\mathbf{s}_i, j)$ and $p(\mathbf{s}_i|j)$ are given by (11) and (9), respectively, and $p(\cdot|j)$ is short for $p(\cdot|z_{ij} = 1)$.

The EM algorithm uses the fact that the maximum of the log of (17) can easily be found analytically, to maximise (16) w.r.t. the parameters [19]. The M step of the EM algorithm for mixtures of PPCAs [28] actually consists of two steps: in the first, only the mixing parameters π_j and the means $\boldsymbol{\mu}_j$ are updated; in the second, new values for the $\mathbf{A}_j$ and σ_j^2 are found. Combined, this gives:

- **E step**: for all i and j, calculate the posterior responsibility of each model for each sample, i.e. the expectation of z_{ij}:

$$r_{ij} = E(z_{ij}) = \frac{\pi_j p(\mathbf{x}_i|j)}{p(\mathbf{x}_i)} = \frac{\pi_j p(\mathbf{x}_i|j)}{\sum_{l=1}^{k} \pi_l p(\mathbf{x}_i|l)} \tag{18}$$

- **M step**: for all j, update the estimates of the parameters (a prime indicating new parameters):

$$\pi_j' = \frac{1}{n}\sum_{i=1}^{n} r_{ij} \quad \text{and} \quad \boldsymbol{\mu}_j{}' = \frac{\sum_{i=1}^{n} r_{ij}\mathbf{x}_i}{\sum_{i=1}^{n} r_{ij}} \tag{19}$$

  The per-model weighted sample covariance matrix $\mathbf{S}_j'$,

$$\mathbf{S}_j{}' = \frac{\sum_{i=1}^{n} r_{ij}(\mathbf{x}_i - \boldsymbol{\mu}_j{}')(\mathbf{x}_i - \boldsymbol{\mu}_j{}')^T}{\sum_{i=1}^{n} r_{ij}} \tag{20}$$

  can then be used to find $\mathbf{A}_j'$ and $\sigma_j^{2\prime}$, using normal eigendecomposition:

$$\mathbf{A}_j' = \mathbf{E}_j(\boldsymbol{\Lambda}_j - \sigma_j^{2\prime}\mathbf{I})^{\frac{1}{2}} \quad \text{and} \quad \sigma_j^{2\prime} = \frac{1}{d-m}\sum_{l=m+1}^{d} \lambda_l \tag{21}$$

  where $\boldsymbol{\Lambda}_j$ contains the m leading eigenvalues of $\mathbf{S}_j{}'$ on its diagonal, $\mathbf{E}_j$ the corresponding eigenvectors, and the λ_l^j are the trailing eigenvalues.

Tipping and Bishop actually describe a faster version of the EM algorithm as well, updating $\mathbf{A}_j$ and σ_j^2 iteratively. However, this still uses just $\mathbf{S}_j{}'$.

3.3 Lack of robustness

The mixture model described above is clearly a mixture of Gaussians, in which the covariance matrices are restricted to a specific form, controlled by the parameter m. Although the Gaussian distribution is mathematically elegant and allows one to derive the EM algorithm above, it is not necessarily optimal for manifold learning. The problem is that it may assign high probability to large empty volumes in space, in the presence of a few outliers. This is illustrated in Figure 5(a)-(c) for a simple 2D problem, where 200 samples are distributed along 2 1D curves, with some added Gaussian noise. When just a few uniformly distributed outliers are present, one or more of the Gaussians are used to model these. The resulting model no longer describes the manifolds well.

What is needed for cases such as these is a more robust mixture model. There has recently been interest in the vision and statistics communities in modelling single subspaces more robustly (e.g. [4, 18]), as manifold learning is applied to increasingly more problems[1]. However, many of the proposed

[1]Note that not all robust approaches are only concerned with ignoring outliers; some also aim at being robust against outlying pixel values [5, 25]. However, we believe factor analysis (FA) can be used quite effectively to model noise in individual pixels, also in the framework proposed in this paper.

techniques are cumbersome, iterative, take a large amount of computation or are not applicable in a mixture model.

We propose that for manifold learning, finding an exact local PCA solution is not necessary, *as long as the main axes of the densities are aligned with the manifold*. Then, more robust densities can be used in the mixture, resulting in models that will mainly assign high probability to samples on the manifold. From the statistics literature, there is a wide range of possible robust approaches (see e.g. [12]). A simple solution in mixture modelling consists of adding a uniform density into the mixture to capture the outliers. The problem with this approach is that the range over which this uniform distribution is defined will have to be set, which is not trivial in high-dimensional spaces. Another interesting approach is to use two Gaussian components per model in the mixture:

$$p(\mathbf{x}) = (1-c)N(\mathbf{x}; \boldsymbol{\mu}, \mathbf{C}) + cN(\mathbf{x}; \boldsymbol{\mu}, \alpha\mathbf{C}) \tag{22}$$

Here, the second Gaussian is used to model outliers. Obviously, this leads to the question how to set or learn α. Below, a generalised and more elegant implementation of this idea, using the t-distribution, will be discussed.

3.4 Mixtures of t-distributed subspaces (MTS)

A recent development in the statistics community is that of a mixture of t-distributions [20]. The motivation is as follows: a generalisation of the two-component normal mixture model (22) is the normal scale model, where

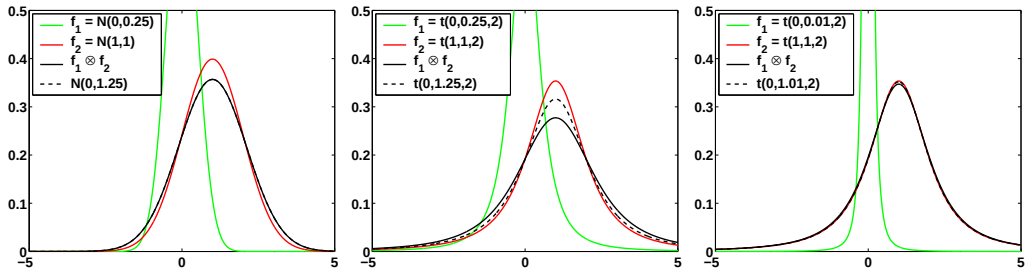
$$p(\mathbf{x}) = \int N(\mathbf{x}; \boldsymbol{\mu}, u^{-1}\mathbf{C}) dH(u) \tag{23}$$

This is identical to (22) when H is a pdf with $H(1) = 1-c$, $H(\frac{1}{c}) = c$ and 0 elsewhere. If, however, H is replaced by a χ^2 pdf with ν degrees of freedom, with a gamma prior on u of $p(u) = \gamma(u; \nu/2, 2/\nu)$, the resulting distribution is a t-distribution [14, 19, 20]. Here,

$$\gamma(u; \alpha, \theta) = u^{\alpha-1} \exp(-u\theta^{-1}) \Gamma(\alpha)^{-1} \theta^{-\alpha} \tag{24}$$

The t-distribution is a heavier-tailed alternative to the Gaussian, with an additional parameter ν, the degrees of freedom. The pdf of a random variable $\mathbf{x}$ with a multivariate t-distribution is given by:

$$t(\mathbf{x}; \boldsymbol{\mu}, \mathbf{C}, \nu) = (\pi\nu)^{-\frac{d}{2}} \Gamma\left(\frac{\nu+d}{2}\right) \Gamma\left(\frac{\nu}{2}\right)^{-1} \left((\mathbf{x}-\boldsymbol{\mu})^T \mathbf{C}^{-1} (\mathbf{x}-\boldsymbol{\mu}) + 1\right)^{-\frac{\nu+d}{2}} \tag{25}$$



(a) (b) (c)

Figure 4: (a) The convolution of two Gaussians $N(x; \mu, \sigma^2)$ is itself a Gaussian; (b) in general, the convolution of two t-pdfs $t(x; \mu, \sigma^2, \nu)$ is not a t-pdf, however (c) for small σ^2 it approximates it well.

where Γ is the Gamma function, $\boldsymbol{\mu}$ is the mean and $\mathbf{C}$ the covariance matrix. For $\nu \to \infty$, the t-distribution becomes a Gaussian distribution.

The key element in deriving the EM algorithm for mixtures of PPCA was that the convolution of two Gaussians is itself a Gaussian; this was used to derive (14). For t-distributions, the convolution of

$$p(\mathbf{x}|\mathbf{s}) = t(\mathbf{x}; \mathbf{A}\mathbf{s} + \boldsymbol{\mu}, \sigma^2\mathbf{I}) \quad \text{and} \quad p(\mathbf{s}) = t(\mathbf{s}; \mathbf{0}, \mathbf{I}) \tag{26}$$

is not necessarily a t-distribution. However, it is to good approximation for large ν (as the t-distribution becomes like a Gaussian) *or for small* σ^2 in (26). The latter is a consequence of the fact that for small σ^2, convolution with a Gaussian will approximate convolution with a delta function, as is illustrated in Figure 4 for the univariate case.

In manifold learning, one will typically observe small values of σ^2 (provided the manifold dimensionality is chosen correctly). Approximately, then:

$$p(\mathbf{x}_i|j) \approx t(\mathbf{x}_i; \boldsymbol{\mu}_j, \mathbf{A}_j{\mathbf{A}_j}^T + \sigma_j^2\mathbf{I}, \nu_j) \tag{27}$$

This allows us to use the EM algorithm derived for mixtures of constrained Gaussians above, and apply it to a mixture of constrained t-distributions. Again, a full derivation is not given; please see [20] for more information.

In using the EM algorithm to maximise (16) when a mixture of t-distributions is used, a new hidden variable u_{ij} is introduced. If $\mathbf{x}_i$ belongs to model j (i.e. $z_{ij} = 1$), then:

$$p(\mathbf{x}_i|\mathbf{s}_i, u_{ij}, j) = N(\mathbf{x}_i; \mathbf{A}\mathbf{s}_i + \boldsymbol{\mu}_j, u_{ij}^{-1}\boldsymbol{\Psi}_j) \tag{28}$$

Intuitively, u_{ij} is the weight assigned by model j to sample $\mathbf{x}_i$: outliers will be given low weight, and hence be described by a Gaussian with high covariance

matrix elements. As above, given model j, the u_{ij} are assumed independently distributed according to a gamma distribution,

$$p(u_{ij}|j) = \gamma(u_{ij}; \nu_j/2, 2/\nu_j) \tag{29}$$

The complete-data distribution becomes:

$$\begin{aligned} p(\mathbf{x}, \mathbf{s}, \mathbf{z}, \mathbf{u}) &= \prod_{i=1}^{n} p(\mathbf{x}_i|\mathbf{s}_i, \mathbf{u}_i, \mathbf{z}_i) p(\mathbf{s}_i|\mathbf{u}_i, \mathbf{z}_i) p(\mathbf{u}_i|\mathbf{z}_i) p(\mathbf{z}_i) \\ &= \prod_{i=1}^{n} \prod_{j=1}^{k} \left[\pi_j p(\mathbf{x}_i|\mathbf{s}_i, j, u_{ij}, j) p(\mathbf{s}_i|u_{ij}, j) p(u_{ij}|j)\right]^{z_{ij}} \end{aligned} \tag{30}$$

with $p(\mathbf{x}_i|\mathbf{s}_i, u_{ij}, j)$ given by (28), $p(\mathbf{s}_i|u_{ij}, j) = N(0, u_{ij}^{-1}\mathbf{I})$ and $p(u_{ij}|j)$ given by (29).

The EM algorithm maximising (16) w.r.t. the parameters consist of:

- **E step**: for all i and j, calculate r_{ij} according to (18), and

$$u_{ij} = \frac{\nu_j + d}{\nu_j + (\mathbf{x}_i - \boldsymbol{\mu}_j)^T (\mathbf{A}_j \mathbf{A}_j{}^T + \sigma_j^2 \mathbf{I})^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_j)} \tag{31}$$

- **M step**: for all j, re-estimate the π_j's as in (19) and

$$\boldsymbol{\mu}_j{}' = \frac{\sum_{i=1}^{n} r_{ij} u_{ij} \mathbf{x}_i}{\sum_{i=1}^{n} r_{ij} u_{ij}} \quad \text{and} \quad \mathbf{S}_j{}' = \frac{\sum_{i=1}^{n} r_{ij} u_{ij} (\mathbf{x}_i - \boldsymbol{\mu}_j{}')(\mathbf{x}_i - \boldsymbol{\mu}_j{}')^T}{\sum_{i=1}^{n} r_{ij}} \tag{32}$$

  $\mathbf{S}_j{}'$ can then be used to find $\mathbf{A}_j{}'$ and $\sigma_j^{2\prime}$ using normal eigendecomposition as in (21), or it can be used in Tipping and Bishop's faster version of the algorithm.

The only parameters not re-estimated yet in the M step are the ν_j's. Following [20], they can be updated by equating the derivative of all terms involving ν_j in the log-likelihood to zero. A solution can then be found using a nonlinear solver. However, in all experiments in this paper ν_j was fixed at 2, $\forall j$, as we are specifically interested in robust manifold learning (for more discussion on this, see Sections 3.5.1 and 5).

It is easy to see that the changes this EM algorithm introduces w.r.t. that outlined in Section 3.2 are minimal. An additional weight u_{ij} is calculated for each sample $\mathbf{x}_i$ and model j, which is used to re-estimate the means and sample covariance matrices robustly. Besides the actual model, a useful extra output of this algorithm are exactly these weights, which for outliers will be low.

3.5 Experiments

3.5.1 Density estimation

Figure 5 demonstrates the robustness of the proposed mixture of t-subspaces on a toy data set. The data was sampled from 2 semicircles, with some added Gaussian noise. To these 200 samples q uniformly distributed noise samples were added. Mixtures of PPCAs (MPPCA) with $k = 6$, $m = 1$ and mixtures of t-distributed subspaces (MTS) with identical settings were trained on this set. For the MTS, ν_j was fixed at 2, $\forall j$.

For $q = 0$, i.e. when no noise is added, both methods give more or less similar results. The likelihood of the MPPCA model is slightly higher than that of the MTS model. Note that MTS tends to assign low probability to samples at the "edges" of the manifolds; this is due to the fact that the within-subspace probability model is t-distributed as well. Samples lying along the main axes of a subspace, but far away from its mean, will therefore be assigned low weight u_{ij}.

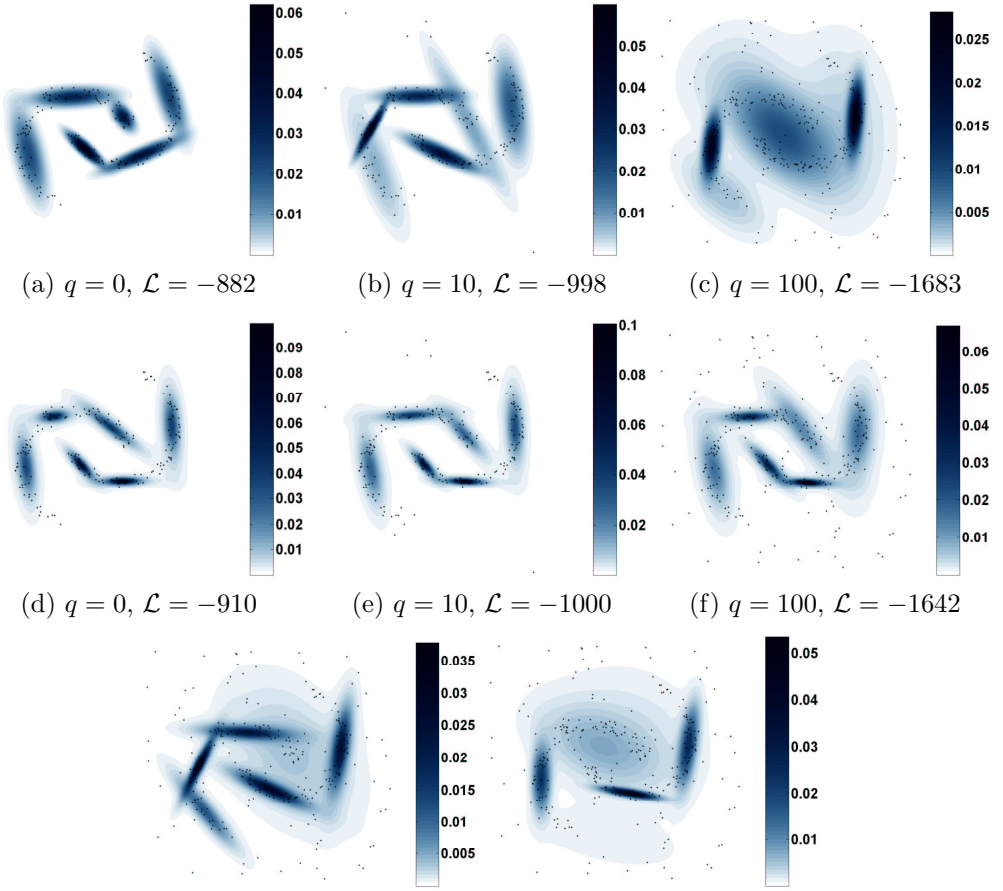
When outliers are added, MPPCA loses the manifold structure quite quickly. Already when $q = 10$, i.e. a fraction of outliers of 5%, 1 of the 6 PPCA models is used to model these. For 50% outliers, the manifold structure is lost completely. MTS is more robust: although the quality of the density estimate along the manifold deteriorates, it manages to down-weight the outliers. Furthermore, the likelihood of the MTS becomes higher than that of the MPPCA.

The results obtained in Figures 5(d)-(f) were obtained when ν_j was fixed to 2, for all models. If ν_j is set higher, robustness is gradually lost; Figure 5(g) illustrates this for higher settings (here ν_j was fixed at 5). When ν_j is learned, after initialisation at 5, the result (Figure 5(h)) is nearly as poor as that of MPPCA.

3.5.2 Handwritten digit recognition

The MPPCA and MTS models were also compared on a classification problem, handwritten digit recognition. The database used was a pre-processed version of the NIST database. The original NIST database digits [30] were resized to 16×16 pixel images (preserving the aspect ratio), put upright, normalised for pencil-width and rescaled in grey value to $[-1, 1]$ [6]. A data set of 1000 training samples per class was thus created, as well as a test set containing 1000 samples per class.

Global principal component analysis on the training set left 51 dimensions. In this 51D space, mixture models were trained on each individual digit, for various settings of the number of subspaces, k, and the number



(a) $q = 0$, $\mathcal{L} = -882$ (b) $q = 10$, $\mathcal{L} = -998$ (c) $q = 100$, $\mathcal{L} = -1683$

(d) $q = 0$, $\mathcal{L} = -910$ (e) $q = 10$, $\mathcal{L} = -1000$ (f) $q = 100$, $\mathcal{L} = -1642$

(g) $q = 100$, $\mathcal{L} = -1634$ (h) $q = 100$, $\mathcal{L} = -1664$

Figure 5: Density estimates on a simple 2D toy dataset of 200 samples distributed along 2 1D curves, with some added Gaussian noise and q added uniformly distributed noise samples: (a)-(c) MPPCA, (d)-(f) MTS (both $k = 6$, $m = 1$). (g) MTS with ν_j fixed at 5, $\forall j$. (h) Same, but with ν_j initialised at 5, $\forall j$, and optimised by the EM algorithm. For the latter, after training the ν_j's ranged between 18.8 and 20.6. All runs of the EM algorithm were randomly initialised.

$m =$	4	8	12	16	20
$k = 2$	4.23 (0.10)	3.42 (0.07)	2.94 (0.04)	2.79 (0.08)	2.64 (0.05)
4	3.36 (0.15)	2.54 (0.09)	2.27 (0.10)	2.31 (0.11)	2.34 (0.11)
8	3.14 (0.14)	2.50 (0.08)	2.57 (0.13)	2.81 (0.24)	4.00 (2.99)
12	4.09 (3.03)	3.66 (2.99)	4.08 (2.95)	3.50 (0.12)	3.87 (0.20)
16	5.93 (6.31)	3.94 (2.97)	4.48 (3.05)	5.16 (2.99)	5.07 (0.52)

(a)

$m =$	4	8	12	16	20
$k = 2$	4.48 (0.09)	3.03 (0.09)	2.56 (0.06)	2.55 (0.02)	2.70 (0.05)
4	3.21 (0.08)	2.39 (0.08)	2.14 (0.05)	2.02 (0.10)	2.18 (0.06)
8	2.67 (0.13)	2.13 (0.08)	2.02 (0.13)	1.89 (0.12)	2.15 (0.12)
12	2.36 (0.12)	2.02 (0.08)	2.20 (0.59)	2.08 (0.18)	2.25 (0.20)
16	2.54 (0.62)	2.09 (0.11)	2.31 (0.40)	2.41 (0.29)	2.58 (0.27)

(b)

Table 1: Performance of (a) MPPCA and (b) MTS models with different settings for the number of subspaces k and number of dimensions per subspace, m. Numbers are % error on a test set, average and standard deviation over 10 different initialisations of the algorithm.

of dimensions per subspace, m. All experiments were repeated for 10 different initialisations, by the k-means algorithm. The covariance matrices $\mathbf{C}_j = \mathbf{A}_j {\mathbf{A}_j}^T + \sigma_j^2 \mathbf{I}$ found in each iteration of the EM algorithm were regularised by adding $10^{-3}\mathbf{I}$, to prevent the likelihood from becoming infinite. For the MTS models, the ν_j's were fixed at 2; no attempt has been made to optimise this setting. Table 1 presents the classification results obtained using MPPCA and MTS models, obtained by Bayesian classification using the densities found for each digit.

The table demonstrates that the proposed MTS models perform slightly better, for individual settings of k and m, than MPPCA models. The minimum error reached is also lower, at 1.89% vs. 2.27%. Another interesting observation is that, where MPPCA has a clear optimal model for $k = 4$ and $m = 12 - 16$, MTS performance decreases more gracefully with different parameter settings. Even for very large models (e.g. $k = 16$, $m = 20$) the error is only 2.58%, as compared to 5.07% for MPPCA. This clearly indicates that MTS is less likely to assign high probability to regions that do not contain data.

The MTS models not only give better performance, but also a slightly higher likelihood on both the training set and the test set, although with

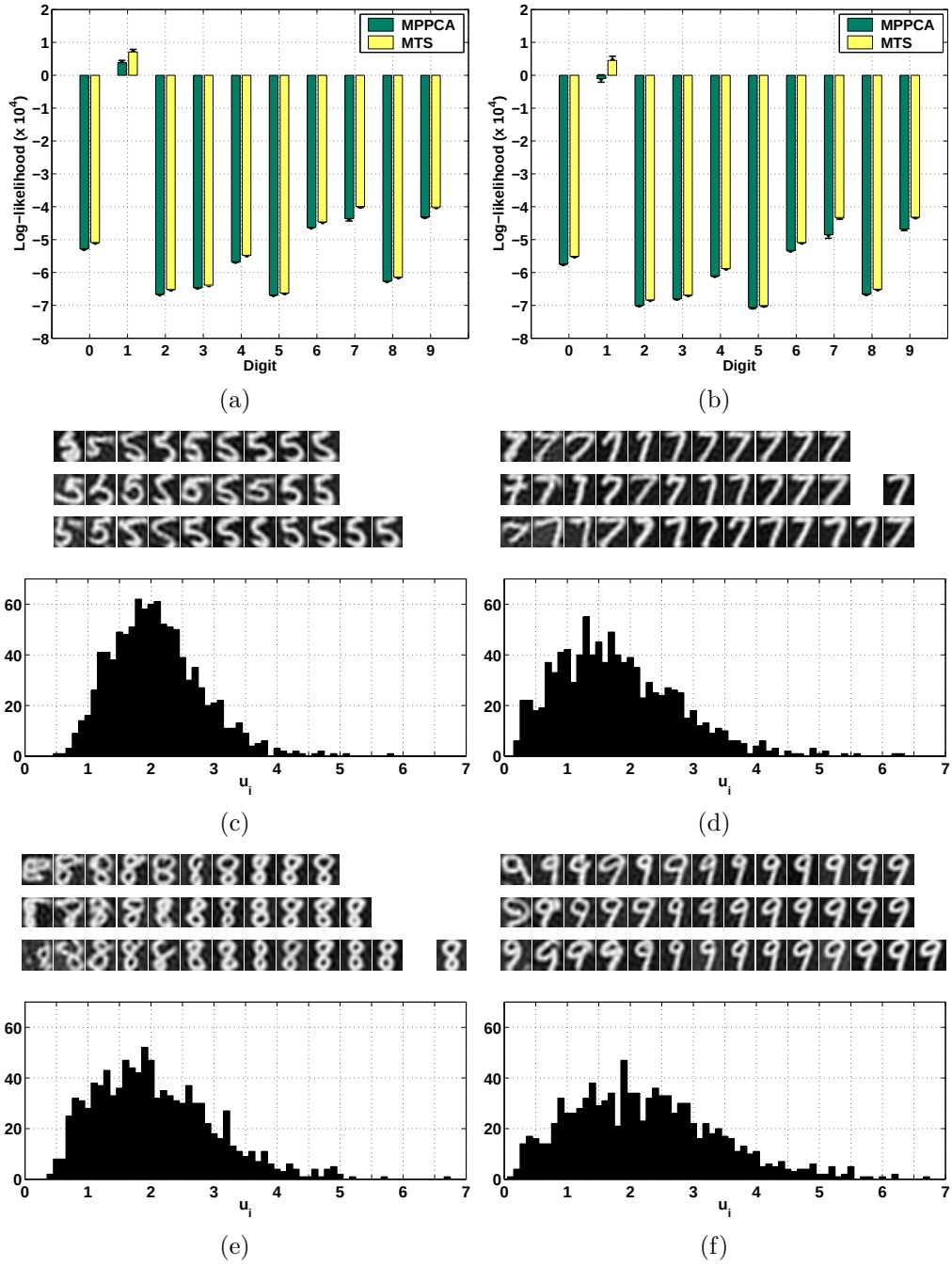


Figure 6: Average log-likelihood of (a) the training set and (b) the test set over 10 experiments, per digit, for MPPCA and MTS models ($k = 4$, $m = 12$). Error bars indicate standard deviation. (c)-(f) Histograms of $\sum_{j=1}^{k} u_{ij}$ for MTS models ($k = 4$, $m = 12$) trained on digits "5", "7", "8" and "9". Images above the histogram indicate typical images having weights directly below.

fixed ν_j's they have the same number of free parameters as MPPCA models. This is illustrated in Figure 6(a)-(b). The difference is largest for digits with little natural variation in appearance, i.e. digits "0", "1", "4", "7" and "9". For digits with a larger amount of variation the likelihood difference is not as pronounced. This indicates that MPPCA tends to over-estimate variance due to the presence of outliers; in other words, by assigning low weight to outliers, MTS obtains a tighter fit around the manifold, while not overfitting.

Finally, Figures 6(c)-(f) show some histograms of $\sum_{j=1}^{k} u_{ij}$, the cumulative weight assigned to samples by an MTS mixture with $k = 4$,$m = 12$. Outliers have been correctly given low weight, and "prototypical" digits have been given high weight. Simple thresholding of the cumulative weight makes for an easy outlier removal procedure.

4 Robust manifold learning

4.1 Robust locally linear coordination (RLLC)

The robust subspace mixture model developed above can be applied in LLC. However, it will not directly lead to good results. Figures 7(a)-(c) show what happens when the MPPCA used before is replaced by an MTS (compare to Figure 2). Although the result it slightly better, the left edge is still "curled up". To see why this should be the case, note that there are two effects which may lead to poor embeddings when outliers are present:

- the mixture-of-subspaces may be fitted poorly, which for MTS is much less of a problem than for MPPCA;
- all samples, outlier or not, are still treated equally in the LLE step of the LLC algorithm.

The last point is caused by the fact that, although LLC takes the responsibilites into account, these are still normalised to sum to 1. So, even though a sample might be ill-represented by any model, it will still play as large a role in the reconstruction as any other sample.

One way to ignore samples in the reconstruction is to remove (or down-weight) their connection to the other samples. In other words, if a sample is reliable (an inlier), it should be reconstructed by its neighbours; if it is unreliable (an outlier), it should be reconstructed *only by itself*. This means outliers can be mapped anywhere, as they will have no influence on the mapping of other samples.

Given a reliability measure f_i for each sample $\mathbf{x}_i$, $0 \leq f_i \leq 1$ (where a value of 1 indicates a reliable sample), the second step of LLE (2) thus

changes to minimising:

$$\begin{aligned}\varepsilon_{II}(\mathbf{Y}) &= \sum_{i=1}^{n} \parallel \mathbf{y}_i - \left[(1-f_i)\mathbf{y}_i + f_i \sum_{j=1}^{q} W_{i\Omega_{ij}} \mathbf{y}_{\Omega_{ij}}\right] \parallel^2 \\ &= \sum_{i=1}^{n} \parallel f_i \mathbf{y}_i - f_i \sum_{j=1}^{q} W_{i\Omega_{ij}} \mathbf{y}_{\Omega_{ij}} \parallel^2 \\ &= \sum_{i=1}^{n} f_i^2 \parallel \mathbf{y}_i - \sum_{j=1}^{q} W_{i\Omega_{ij}} \mathbf{y}_{\Omega_{ij}} \parallel^2 \end{aligned} \tag{33}$$

or, in matrix terms,

$$\varepsilon_{II}(\mathbf{Y}) = \mathrm{tr}\left(\mathbf{Y}^T \mathbf{F}^T (\mathbf{I} - \mathbf{W})^T (\mathbf{I} - \mathbf{W}) \mathbf{F} \mathbf{Y}\right) \tag{34}$$

where $\mathbf{F}$ contains the f_i's on its diagonal and zeroes elsewhere. In the same way, for LLC expression (7) changes to minimising:

$$\mathrm{tr}\left(\mathbf{P}^T \mathbf{F}^T \mathbf{G} \mathbf{F} \mathbf{P}\right) \tag{35}$$

with the same constraints as before: $\frac{1}{n}\mathbf{1}^T\mathbf{U}\mathbf{P} = 0$, $\frac{1}{n}\mathbf{P}^T\mathbf{U}^T\mathbf{U}\mathbf{P} = \mathbf{P}^T\mathbf{H}\mathbf{P} = \mathbf{I}$.

An outcome of the MTS model is, for each sample, the set of weights assigned to it by each of the models, u_{ij}. The total weight $\sum_{j=1}^{k} u_{ij}$ can therefore be seen as an unnormalised, global measure of reliability, where high values indicate reliable samples. This value can easily be normalised by using a transformation, e.g.,

$$f_i = 1 - \exp\left(-\sum_{j=1}^{k} u_{ij}\right) \tag{36}$$

to fit the description of the reliability measure needed. Ofcourse, once any mixture model (e.g. an MPPCA) is fitted, the estimated probability $p(\mathbf{x}_i)$ could also be used as a reliability measure. However, the model fit itself still needs to be robust, and with MTS the weight measure comes for free.

Summing up, a robust LLC can be obtained by:

- fitting an MTS instead of an MPPCA model;
- using the resulting weights as reliabilities (36) in minimising (35).

This version of LLC will be called *robust LLC*, or RLLC.

4.2 Experiments

4.2.1 3D toy data

RLLC was applied to the 3D sine-wave data set used before. The result is shown in Figures 7(d)-(f). Obviously, the embedding is better than that using unweighted LLE (Figures 7(a)-(c)) and, again, better than ordinary LLE. Outliers are mapped randomly in the embedded space, but do not distort the mapping of the real manifold.

RLLC can handle higher noise levels than ordinary LLC, but still breaks down earlier than ordinary LLE does. This is illustrated in Figure 8. For a moderate noise level (25 points, i.e. 5%) LLC still finds a good mapping. However, for 50 points (10%) the mapping can be good, but can also come out bad (the folded mapping of Figure 8(h)) for different realisations of the noise. For this noise level, ordinary LLE manages to find a mapping which, though highly skewed, is still recognisable as a 2D manifold.

Different parameter settings may, ofcourse, lead to better results. For this example however, changing k is not really an option: if it is set too small (i.e. $k \leq 4$) the manifold structure will be lost; if it is set too large (i.e. $k \geq 8$), one or more models will start fitting noise. It was decided not to change other parameters, such as the regularisation parameters r_{LLE} and r_{PCA}, as this does not reflect what one would do in practical problems.

4.2.2 Rotating face images

Finally, RLLC was applied to the face data set. It gave better results (Figures 9-10) than ordinary LLC, in that the manifold seems to have been unfolded. Only for small values of k, e.g. $k = 10$, there still seems to be some folding at the edges (indicated by the more dense areas in the mapping).

Unlike for LLC, for RLLC k-means initialisation was feasible. For larger k it does not seem to have much influence; only for $k = 10$ is the result slightly different.

5 Conclusions

This paper discussed global manifold learning by locally linear coordination (LLC). This method is an interesting combination of fitting a mixture of local subspace models and an embedding procedure. LLC solves two problems at once: the computational problems of locally linear embedding, and the lack of a global mapping of a mixture of subspaces. However, it was shown to be highly sensitive to the presence of outliers.

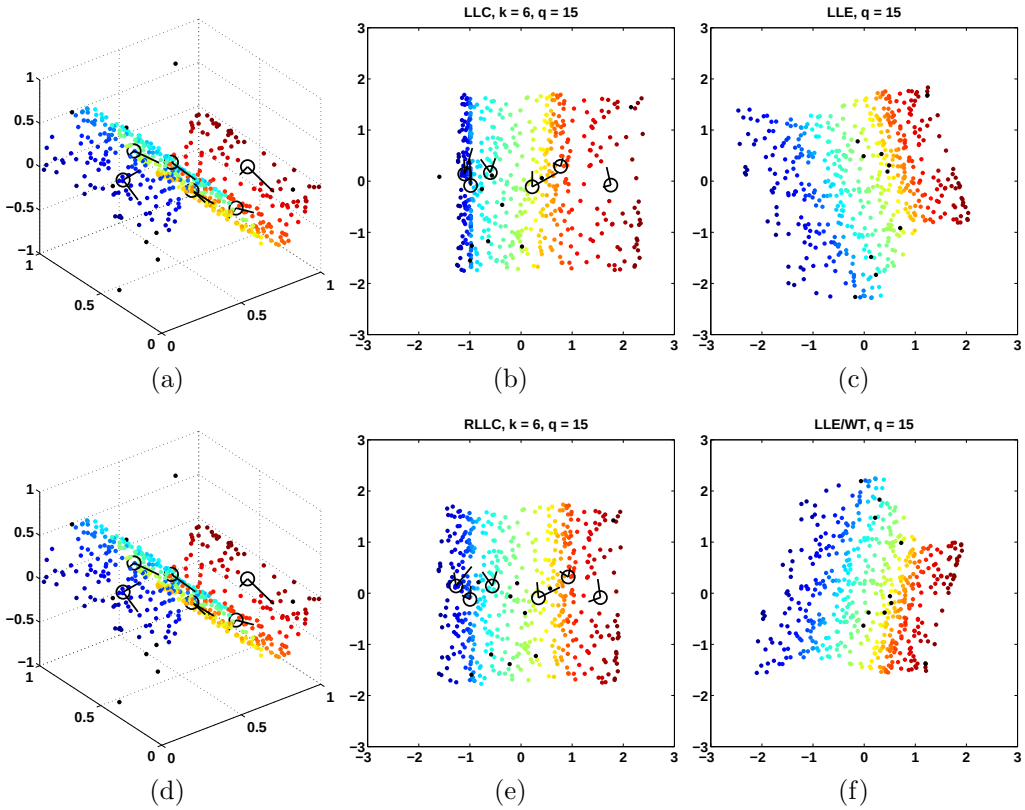


Figure 7: 500 samples on a 3D sine wave with added Gaussian noise, $\sigma = 0.01$ and 10 uniform outliers (left column) mapped to 2D using (R)LLC based on MTS (middle column) and LLE (right column). (a)-(c) LLC. (d)-(f) RLLC. Other parameters: $k = 6$, $q = 15$, $m = 2$, r_{LLE} automatic, $r_{MTS} = 0$, $\nu = 2$.

A method for robust manifold learning by EM was then presented, based on earlier work on mixtures of probabilistic PCA subspaces and mixtures of t-distributions. It is less suitable for density estimation in the absence of outliers, as it tends to ignore the "edges" of the data set. However, in the presence of outliers, it is much more robust than mixtures of PCA subspaces. As a consequence, it is useful for description and classification of high-dimensional data, such as images. The model was compared to mixtures of probabilistic PCAs on a handwritten digit recognition problem, and was shown to give good results.

Finally, this robust subspace mixture model was applied in a reformulation of LLC, in which individual samples are weighted by their reliability. The resulting robust LLC (RLLC) was demonstrated to give better results than ordinary LLC, even on data sets in which no obvious outliers were present.

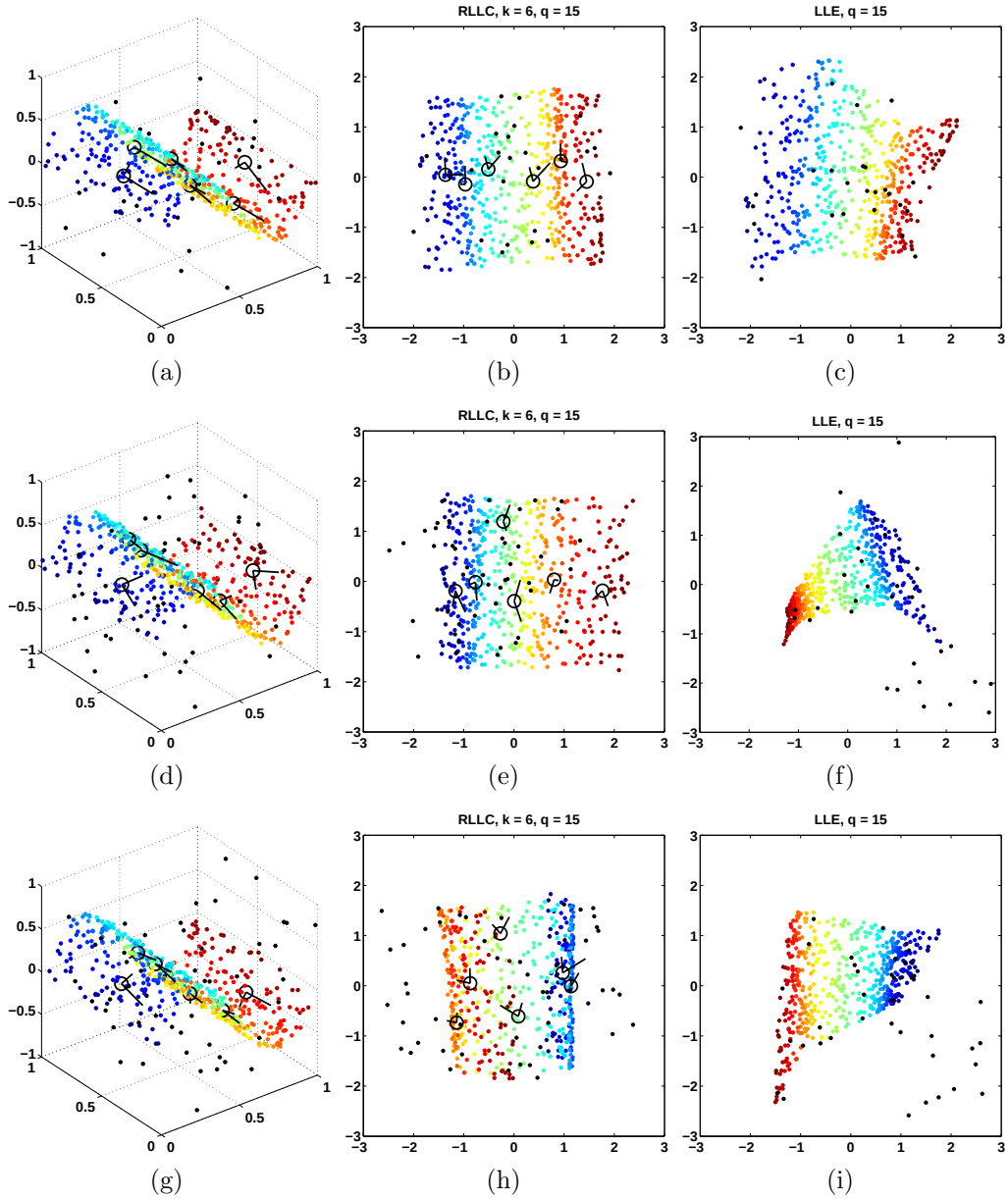


Figure 8: 500 samples on a 3D sine wave with added Gaussian noise, $\sigma = 0.01$ (left column) mapped to 2D using RLLC (middle column) and LLE (right column). (a)-(c) With 25 uniform outliers. (d)-(f) With 50 uniform (g)-(i) Same, with different initialisation. Other parameters: $k = 6$, $q = 15$, $m = 2$, r_{LLE} automatic, $r_{MTS} = 0$, $\nu = 2$.

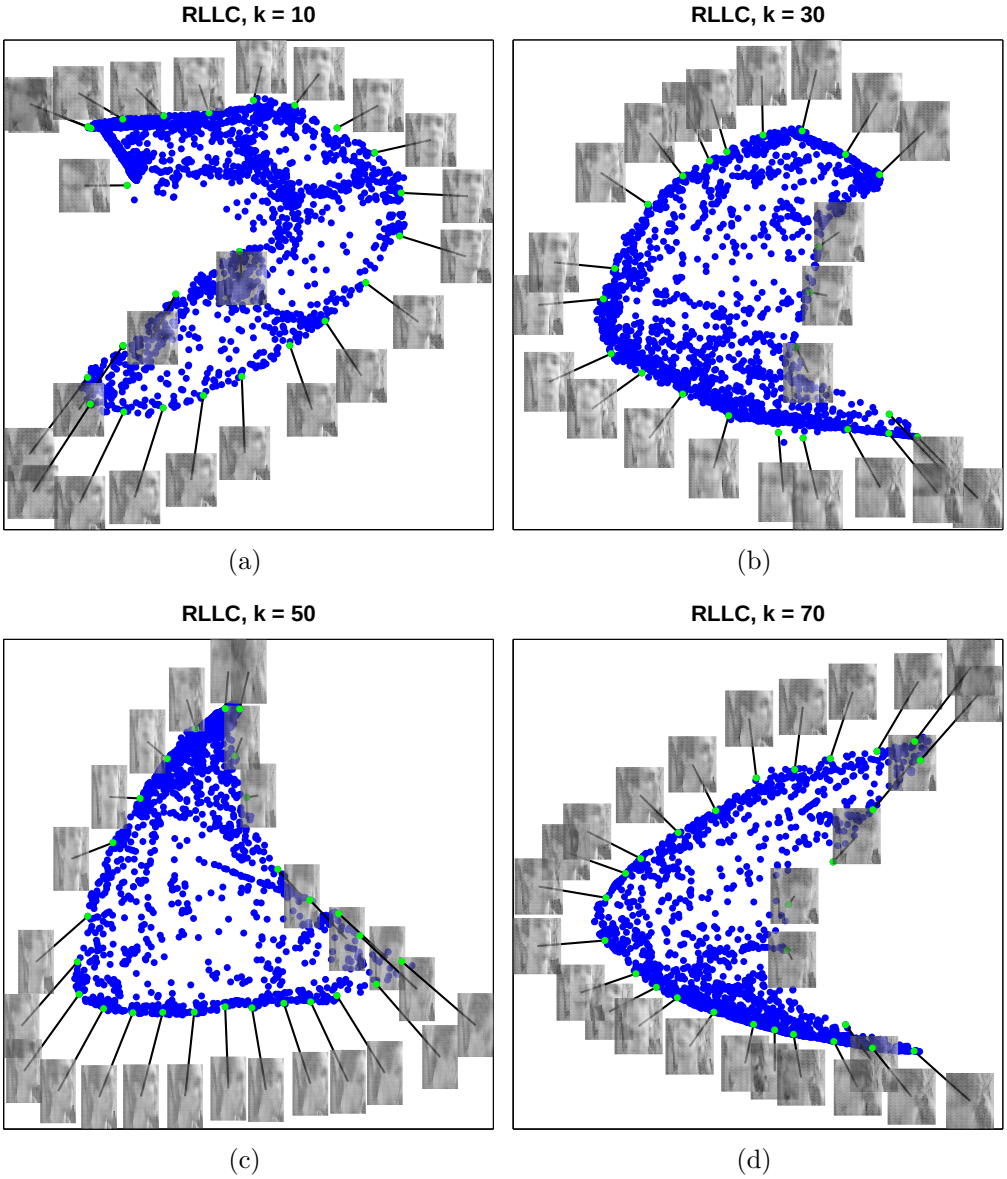


Figure 9: A set of 2000 face images, mapped down to 2D using RLLC with randomly initialised MTS, for various values of k. Before applying LLC, a global PCA mapping was used to reduce the number of dimensions to 100. Other parameters: $q = 25$, $m = 2$, r_{LLE} automatic, $r_{MTS} = r_{LLE}$, $\nu = 2$.

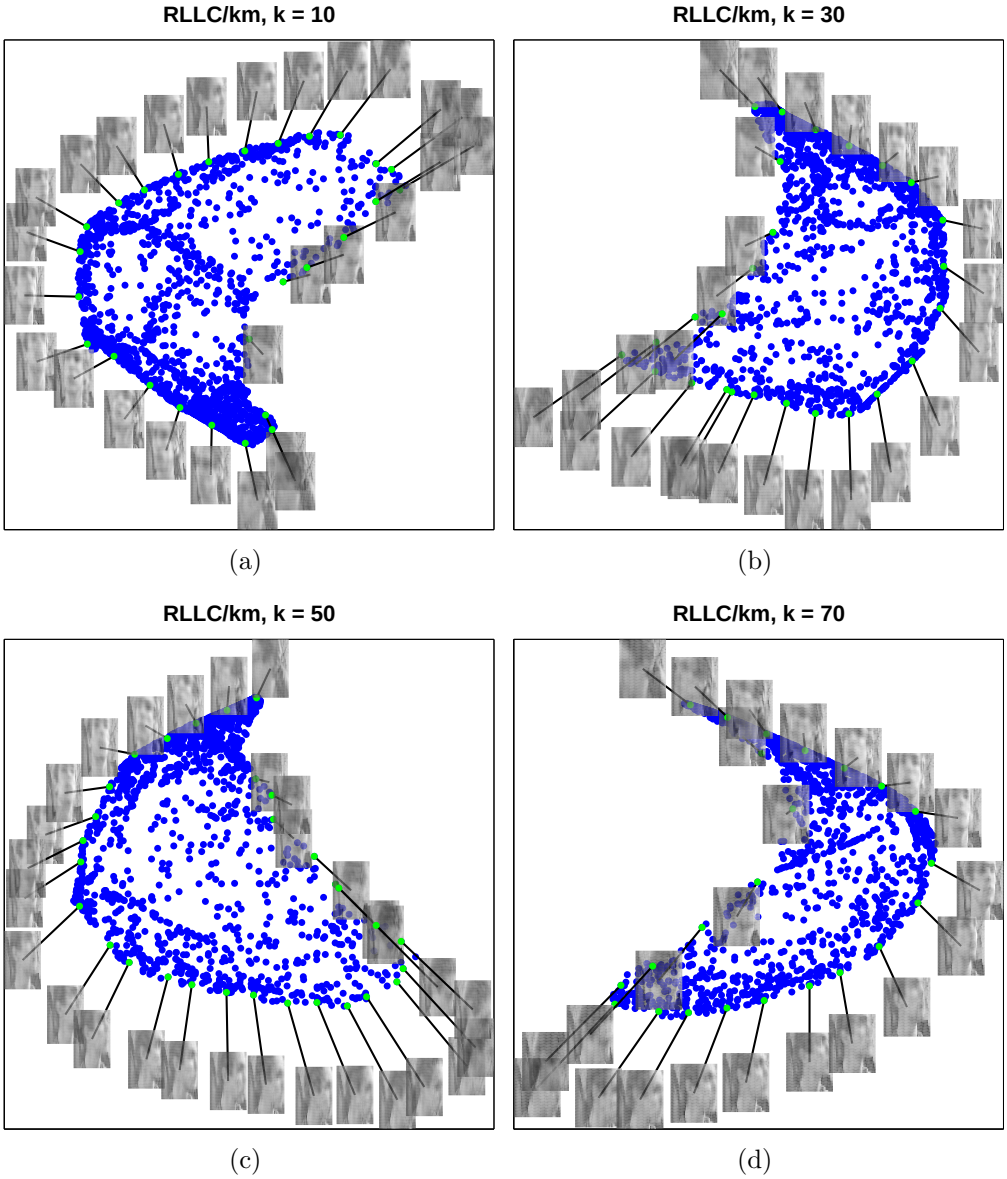


Figure 10: A set of 2000 face images, mapped down to 2D using RLLC with k-means initialised MTS, for various values of k. Before applying LLC, a global PCA mapping was used to reduce the number of dimensions to 100. Other parameters: $q = 25$, $m = 2$, r_{LLE} automatic, $r_{MTS} = r_{LLE}$, $\nu = 2$.

In future work, we intend to investigate how to set ν in an optimal way. Experiments showed that optimising it by maximum likelihood makes the method as a whole less robust. However, the choice of $\nu = 2$ in the experiments, while leading to good results, is rather arbitrary. Another interesting question is whether it will be feasible to use different in-subspace and out-of-subspace models, i.e. a Gaussian distribution for the latent variable $\mathbf{s}$ coupled with a t-distribution for the noise $\boldsymbol{\varepsilon}$. This might improve performance of MTS as a robust density estimator. A final interesting aspect is the small sample size behaviour of the various methods, which for many real-world applications is very important.

Acknowledgements

The first author would like to thank Jiři Matas and the Center for Machine Perception, of the Czech Technical University of Prague. This paper was written while he was a visiting researcher supported by the Center of Excellence EU grant ICA 1-CT-2000-70002 MIRACLE. The work was partly sponsored by the the Czech Ministry of Education under Project MSM 212300013. Thanks go to Sjaak Verbeek for providing the rotating face data set.

References

[1] C.M. Bishop, M. Svensén, and C.K.I. Williams. GTM: The generative topographic mapping. *Neural Computation*, 10(1):215–234, 1998. URL http://citeseer.nj.nec.com/bishop98gtm.html.

[2] M. Brand. Charting a manifold. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, Cambridge, MA, 2002. MIT Press. URL http://www-2.cs.cmu.edu/Groups/NIPS/NIPS2002/NIPS2002preproceedings/.

[3] K. Chang and J. Ghosh. A unified model for probabilistic principal surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(1):22–41, 2001. URL http://lans.ece.utexas.edu/~kuiyu/paper/kuiyu2001pami.ps.gz.

[4] H. Chen, P. Meer, and D. Tyler. Robust regression for data with multiple structures. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2001)*, pages 1069–1075, Los Alamitos, CA, 2001. IEEE Computer Society Press. URL http://www.caip.rutgers.edu/riul/research/papers/pdf/mulrobreg.pdf.

[5] F. De la Torre and M.J. Black. Robust principal component analysis for computer vision. In *Proc. of the International Conference on Computer Vision (ICCV'01)*, volume I, pages 362–369, 2001. URL http://www.cs.brown.edu/people/black/Papers/iccv01fdl-copyright.html.

[6] D. de Ridder. *Adaptive methods of image processing*. PhD thesis, Delft University of Technology, Delft, 2001. URL http://www.ph.tn.tudelft.nl/~dick/publications.html.

[7] D. de Ridder and R.P.W. Duin. Locally linear embedding for classification. Technical Report PH-2002-01, Pattern Recognition Group, Dept. of Imaging Science & Technology, Delft University of Technology, Delft, The Netherlands, 2002. URL http://www.ph.tn.tudelft.nl/~dick.

[8] B.J. Frey, A. Colmenarez, and T.S. Huang. Mixtures of local linear subspaces for face recognition. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'98)*, pages 32–37, Los Alamitos, CA, 1998. IEEE Computer Society Press. URL http://www.psi.toronto.edu/~frey/papers/mfafr.abs.html.

[9] Z. Ghahramani and G.E. Hinton. The EM algorithm for mixtures of factor analyzers. Technical Report CRG-TR-96-1, University of Toronto, Toronto, Canada, 1996. URL http://www.gatsby.ucl.ac.uk/~zoubin/papers/tr-96-1.ps.gz.

[10] T. Hastie and W. Stuetzle. Principal curves. *Journal of the American Statistical Association*, 84:502–516, 1989.

[11] G.E. Hinton, P. Dayan, and M. Revow. Modelling the manifolds of images of handwritten digits. *IEEE Transactions on Neural Networks*, 8 (1):65–74, 1997. URL http://www.gatsby.ucl.ac.uk/Hinton/absps/manifold.pdf.

[12] P.J. Huber. *Robust statistics*. John Wiley & Sons, New York, NY, 1981.

[13] T. Jebara and A. Pentland. Maximum conditional likelihood via bound maximization and the CEM algorithm. In M.S. Kearns, S.A. Solla, and D.A. Cohn, editors, *Advances in Neural Information Processing Systems 11*, 1998. URL http://www.media.mit.edu/~jebara/postscript/cem.pdf.

[14] G. John. Bayesian treatment of the independent student-t linear model. *Journal of Applied Econometrics*, 8:19–40, 1993. URL http://www.biz.uiowa.edu/faculty/jgeweke/papers/paper52/paper52.pdf.

[15] O. Kouropteva, O. Okun, A. Hadid, M. Soriano, S. Marcos, and M. Pietikäinen. Beyond locally linear embedding algorithm. Technical Report MVG-01-2002, Machine Vision Group, University of Oulu, Finland, 2002. URL http://www.ee.oulu.fi/mvg/mvg.php.

[16] O. Kouropteva, O. Okun, and M. Pietikäinen. Selection of the optimal parameter value for the locally linear embedding algorithm. In *Proc. of the* 1st *Int. Conf. on Fuzzy Systems and Knowledge Discovery, Singapore*, pages 359–363, 2002. URL http://www.ee.oulu.fi/mvg/mvg.php.

[17] M. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *American Institute of Chemical Engineers Journal*, 37:223–243, 1991.

[18] A. Leonardis and H. Bischof. Robust recognition using eigenimages. *Computer Vision and Image Understanding*, 78(1):99–118, 2000. URL http://citeseer.nj.nec.com/leonardis97robust.html.

[19] G.J. McLachlan and T. Krishnan. *The EM algorithm and extensions*. John Wiley & Sons, New York, NY, 1997.

[20] D. Peel and G.J. McLachlan. Robust mixture modelling using the t distribution. *Statistics and Computing*, 10(4):339–348, 2000. URL http://www.maths.uq.edu.au/~gjm/982.ps.

[21] S.T. Roweis and L.K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000. URL http://www.sciencemag.org/content/vol290/issue5500/.

[22] S.T. Roweis, L.K. Saul, and G.E. Hinton. Global coordination of local linear models. In *Advances in Neural Information Processing Systems 14*, pages 889–896, Cambridge, MA, 2001. MIT Press. URL http://www.cs.toronto.edu/~roweis/papers/gc_final.ps.gz.

[23] L.K. Saul and D.D. Lee. Multiplicative updates for classification by mixture models. In *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2001. MIT Press. URL http://www-2.cs.cmu.edu/~nips/2001papers/psgz/AA52.ps.gz.

[24] L.K. Saul and S.T. Roweis. Think globally, fit locally: unsupervised learning of nonlinear manifolds. Technical Report MS CIS-02-18, Univ. of Pennsylvania, 2002. URL http://www.cs.toronto.edu/~roweis/papers/lle_tr02.pdf.

[25] D. Skočaj, H. Bishof, and A. Leonardis. A robust PCA algorithm for building representations from panoramic images. In *Proc. of the European Conference on Computer Vision (ECCV'02)*, volume 2353 of *Lecture Notes in Computer Science*, pages 171–178, Berlin, 2002. Springer-Verlag. URL http://cogvis.fri.uni-lj.si/learn_recog/roblearn.asp.

[26] Y.W. Teh and S.T. Roweis. Automatic alignment of local representations. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, Cambridge, MA, 2002. MIT Press. URL http://www-2.cs.cmu.edu/Groups/NIPS/NIPS2002/NIPS2002preproceedings/.

[27] J.B. Tenenbaum, V. de Silva, and J.C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290 (5500):2319–2323, 2000. URL http://www.sciencemag.org/content/vol290/issue5500/.

[28] M.E. Tipping and C.M. Bishop. Mixtures of probabilistic principal component analyzers. *Neural Computation*, 11(2):443–482, 1999. URL ftp://ftp.research.microsoft.com/users/mtipping/mppca.ps.gz.

[29] J.J. Verbeek, N. Vlassis, and B. Kröse. Coordinating Principal Component analyzers. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*, pages 914–919, Madrid, Spain, 2002. URL http://carol.wins.uva.nl/~jverbeek/pub/I0139.pdf.

[30] C.L. Wilson and M.D. Garris. Handprinted character database 3, february 1992. URL http://www.nist.gov/srd/nistsd19.htm. National Institute of Standards and Technology; Advanced Systems division.