HAL
open science

Procrustes analysis to coordinate mixtures of probabilistic principal component analyzers

Jakob Verbeek, Nikos Vlassis, Ben Krose

To cite this version:

Jakob Verbeek, Nikos Vlassis, Ben Krose. Procrustes analysis to coordinate mixtures of probabilistic principal component analyzers. [Technical Report] IAS-UVA-02, 2002, pp.18. inria-00321503

HAL Id: inria-00321503

https://inria.hal.science/inria-00321503v1

Submitted on 16 Feb 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

IAS technical report series, nr. IAS-UVA-02-01

Procrustes Analysis to Coordinate Mixtures of Probabilistic Principal Component Analyzers

J.J. Verbeek, N. Vlassis and B. Kröse
Computer Science Institute
Faculty of Science
University of Amsterdam
The Netherlands

Mixtures of Probabilistic Principal Component Analyzers can be used to model data that lies on or near a low dimensional manifold in a high dimensional observation space, in effect tiling the manifold with local linear (Gaussian) patches. In order to exploit the low dimensional structure of the data manifold, the patches need to be localized and oriented in a low dimensional space, so that 'local' coordinates on the patches can be mapped to 'global' low dimensional coordinates. As shown by [Roweis *et al.*, 2002], this problem can be expressed as a penalized likelihood optimization problem. We show that a restricted form of the Mixtures of Probabilistic Principal Component Analyzers model allows for an efficient EM-style algorithm. The Procrustes Rotation, a technique to match point configurations, turns out to give the optimal orientation of the patches in the global space. We also show how we can initialize the mappings from the patches to the global coordinates by learning a non-penalized density model first. Some experimental results are provided to illustrate the method.
Keywords: dimension reduction, feature extraction, principal manifold, selforganization

Intelligent Autonomous Systems

Submitted to: Int. Conf. on Artificial Neural Networks 2002

Contents

1 Introduction **1**

2 The density model **1**

3 A simplified algorithm due to simplified density model **2**
3.1 Objective Function 2
3.2 Optimization 4

4 Initialization from a data-space mixture **5**
4.1 Fixing the data-space mixture 5
4.2 Optimization and Initialization 6

5 Experimental results **6**
5.1 Artificial data-set 6
5.2 Images of faces 6
5.3 Robot navigation 9

6 Discussion and conclusions **9**

A GEM for constrained MPPCA **11**
A.1 Updating the loading matrices 12
A.2 Variance inside and outside the subspace 12

B A variation on weighted Procrustes analysis **13**

Intelligent Autonomous Systems
Computer Science Institute
Faculty of Science
University of Amsterdam
Kruislaan 403, 1098 SJ Amsterdam
The Netherlands
tel: +31 20 525 7461
fax: +31 20 525 7490
http://www.science.uva.nl/research/ias/

Corresponding author:
J.J. Verbeek
tel: +31 (20) 525 7515
jverbeek@science.uva.nl
http://carol.science.uva.nl/~jverbeek/

1 Introduction

With increasing sensor capabilities, powerful feature extraction methods are becoming increasingly important. Consider a robot sensing its environment with a camera yielding a stream of 100×100 pixel images, i.e. a stream of 10.000 dimensional vectors if we regard the image as a pixel intensity vector. The observations made by the robot often have a much lower intrinsic dimensionality. If we assume a fixed environment, and a robot that can rotate around its axis and drive through a room, the intrinsic dimensionality is only three. Linear feature extraction techniques are able to do a fair compression of the signal by mapping it to a much lower dimensional space. However, only in very few special cases the manifold on which the signal is generated is a linear subspace of the sensor space. This clearly limits the use of linear techniques and suggests to use non-linear feature extraction techniques.

Mixtures of Factor Analyzers (MFA) [Ghahramani and Hinton, 1996] and Mixtures of Probabilistic Principal Component Analyzers (MPPCA) [Tipping and Bishop, 1999] can be used to model such non-linear data manifolds. These methods (in fact MPPCA is a special case of MFA) provide a mapping back and forth between latent and data-space. However, these mappings only have a local applicability and are not related, i.e the coordinate systems of neighboring facor analyzers might be completely differently oriented. We cannot combine the different latent spaces, the coordinates of a point in one subspace give no infotmation on the coordinates in a neighboring subspace.

Recently, a model was proposed that integrates the local linear models into a global latent space, allowing for mapping back and forth between the latent and the data-space. The idea is that there is a linear map for each factor analyzer between the the data-space and the global latent space. This penalized log-likelihood model proposed in [Roweis *et al.*, 2002], solved by an EM-like procedure, is discussed in the next section.

Here, we show how we can remove the iterative procedure from the M-step by simplifying the density model. Furthermore, we show how we can use an 'uncoordinated' mixture model to initialize the mappings from data to latent space, providing an alternative to using an external (unsupervised) method to initialize the latent coordinates for all the data points.

In the next section, we describe the density model that is used. Then, in Section 3 we show how this density model removes one of the iterative procedures in the algorithm given in [Roweis *et al.*, 2002]. Section 4 discusses how an 'uncoordinated' mixture model can be used to initialize the mappings between latent and data-space. Some experimental results are given in 5. We end with a discussion and some conclusions in 6.

2 The density model

To model the data density in the high dimensional space we use mixtures of a restricted type of Gaussian densities. The mixture is formed as a weighted sum of its component densities, the weight of each component is called its 'mixing weight' or 'prior'. The covariance matrices of the Gaussians are constrained to be of the form:

$$\mathbf{C} = \sigma^2(\mathbf{I}_D + \rho\mathbf{\Lambda}\mathbf{\Lambda}^\top), \qquad \mathbf{\Lambda}^\top\mathbf{\Lambda} = \mathbf{I}_d, \qquad \rho > 0 \tag{1}$$

where D and d are respectively the dimension of the high-dimensional/data-space and the low-dimensional/latent space. We use $\mathbf{I}_d$ to denote the d-dimensional identity matrix. The columns of $\mathbf{\Lambda}$, in factor analysis known as the *loading matrix*, are D-dimensional vectors spanning the subspace. Directions within the subspace have variance $\sigma^2(1+\rho)$, other directions have σ^2 variance. This is as the Mixture of Probabilistic Principal Component Analyzers (MPPCA) model, with the difference that here we do not only have isotropic noise outside the subspaces

but also also isotropic variance inside the subspaces. We use this density model to allow for convenient solutions later. In Appendix A we derive a Generalized EM algorithm to find maximum likelihood solutions for this model.

The same model can be rephrased using hidden variables $\mathbf{z}$, which we use to denote 'internal' coordinates of the subspaces. We scale the coordinates $\mathbf{z}$ such that:

$$p(\mathbf{z} \mid s) = \mathcal{N}(0, \mathbf{I}_d), \tag{2}$$

where $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes the Gaussian density, centered at $\boldsymbol{\mu}$ and with covariance matrix $\boldsymbol{\Sigma}$. The internal coordinates allow us to clearly express the link to the global latent space, for which we denote coordinates with $\mathbf{g}$. All mixture components/subspaces have their own orthogonal mapping to the global space, parameterized by a translation $\boldsymbol{\kappa}$ and a matrix $\mathbf{A}$, i.e. $p(\mathbf{g} \mid \mathbf{z}, s) = \delta(\boldsymbol{\kappa}_s + \mathbf{A}_s\mathbf{z})$, where $\delta(\cdot)$ denotes the distribution with mass 1 at the argument. We use p_s to denote the mixing weight of mixture component/subspace s and $\boldsymbol{\mu}_s$ to denote its mean or location. The generative model then looks like:

$$p(\mathbf{x}) = \sum_s p_s \mathcal{N}(\boldsymbol{\mu}_s, \sigma_s^2(\mathbf{I}_D + \rho_s \boldsymbol{\Lambda}_s \boldsymbol{\Lambda}_s^\top)) = \sum_s p_s \int_{\mathbf{z}} \mathrm{d}\mathbf{z} \;\; p(\mathbf{x} \mid \mathbf{z}, s) p(\mathbf{z} \mid s) \tag{3}$$

$$p(\mathbf{x} \mid \mathbf{z}, s) = \mathcal{N}(\boldsymbol{\mu}_s + \sqrt{\rho_s}\sigma_s \boldsymbol{\Lambda}_s \mathbf{z}, \sigma_s^2 \mathbf{I}_D) \tag{4}$$

$$p(\mathbf{g}) = \sum_s p_s \mathcal{N}(\boldsymbol{\kappa}_s, \mathbf{A}_s \mathbf{A}_s^\top) = \sum_s p_s \mathcal{N}(\boldsymbol{\kappa}_s, \alpha_s^2 \sigma_s^2 \rho_s \mathbf{I}_d). \tag{5}$$

We put an extra constraint on the projection matrices:

$$\mathbf{A}_s = \alpha_s \sigma_s \sqrt{\rho_s} \mathbf{R}_s, \qquad \mathbf{R}_s^\top \mathbf{R}_s = \mathbf{I}_d, \qquad \alpha_s > 0. \tag{6}$$

The matrix $\mathbf{R}_s$ implements only rotations plus reflections due to the orthonormality constraint. The generative model reads: first 'nature' picks a subspace according to the prior distribution $\{p_s\}$. A subspace generates internal coordinates $\mathbf{z}$, which on the one hand give rise to *hidden* global coordinates $\mathbf{g}$ (via the translation $\boldsymbol{\kappa}_s$ and the rotation plus scaling $\mathbf{A}_s$) and on the other hand to *observable* data $\mathbf{x}$ (via the translation $\boldsymbol{\mu}_s$ and the orthogonal mapping $\sqrt{\rho_s}\sigma_s\boldsymbol{\Lambda}_s$) plus some noise with covariance $\sigma^2\mathbf{I}_D$. Figure 1 illustrates the model. Note that the model assumes that locally there is a linear correspondence between the data space and the latent space. Furthermore, note that the densities $p(\mathbf{g} \mid \mathbf{x}, s)$ and $p(\mathbf{x} \mid \mathbf{g}, s)$ are Gaussian densities and hence that $p(\mathbf{g} \mid \mathbf{x})$ and $p(\mathbf{x} \mid \mathbf{g})$ are mixtures of Gaussians. In the next section we discuss how this density model differs from [Roweis *et al.*, 2002].

3 A simplified algorithm due to simplified density model

The goal is, given observable data $\{\mathbf{x}_n\}$, to find a good density model in the data-space *and* mappings $\{\mathbf{A}_s, \boldsymbol{\kappa}_s\}$ that give rise to 'consistent' estimates for the hidden $\{\mathbf{g}_n\}$. With consistent we mean that if a point $\mathbf{x}$ in the data-space is well modeled by two subspaces, then the corresponding estimates for its latent coordinate $\mathbf{g}$ should be close to each other, i.e. the subspaces should 'agree' on the corresponding $\mathbf{g}$.

3.1 Objective Function

To measure the level of agreement, one can consider for all data points how uni-modal the distribution $p(\mathbf{g} \mid \mathbf{x})$ is. This idea was also used in [Vlassis *et al.*, 2002], there the goal was to find orhtogonal projections for supervised data, such that the projections from high to low dimension preserve the manifold structure. In [Roweis *et al.*, 2002] it is shown how the double objective

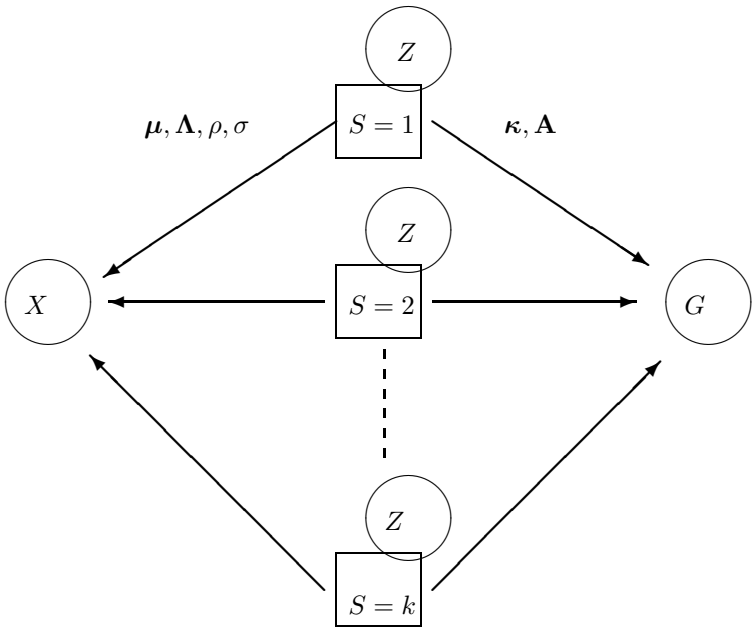

Figure 1: The generative model: $\mathbf{x}$ and $\mathbf{g}$ are independent given $s, \mathbf{z}$.

of likelihood and uni-modality can be implemented as a penalized log-likelihood optimization problem.

Let $Q(\mathbf{g} \mid \mathbf{x}_n)$ denote a Gaussian approximation of $p(\mathbf{g} \mid \mathbf{x}_n) = \sum_s p(\mathbf{g} \mid \mathbf{x}_n, s) p_{ns} = \sum_s p(\mathbf{g}, s \mid \mathbf{x}_n)$, with $p_{ns} = p(s \mid \mathbf{x}_n)$. We define:

$$Q(\mathbf{g}, s \mid \mathbf{x}_n) = Q(s \mid \mathbf{x}_n) Q(\mathbf{g} \mid \mathbf{x}_n), \tag{7}$$

where $Q(\mathbf{g} \mid \mathbf{x}_n) = \mathcal{N}(\mathbf{g}_n, \Sigma_n)$ and $Q(s \mid \mathbf{x}_n) = q_{ns}$. As a measure of uni-modality we can use a sum of Kullback-Leibler divergences:

$$\sum_{ns} \int_{\mathbf{g}} Q(\mathbf{g}, s \mid \mathbf{x}_n) \log \left[\frac{Q(\mathbf{g}, s \mid \mathbf{x}_n)}{p(\mathbf{g}, s \mid \mathbf{x}_n)} \right] \mathbf{dg} = \tag{8}$$

$$\sum_n D_{KL}(\{q_{ns}\} \parallel \{p_{ns}\}) + \sum_s q_{ns} \mathcal{D}_{ns}, \tag{9}$$

where $\mathcal{D}_{ns} = D_{KL}(Q(\mathbf{g} \mid \mathbf{x}_n) \parallel p(\mathbf{g} \mid \mathbf{x}_n, s))$. The total objective function, combining log-likelihood and the penalty term, then becomes:

$$\mathbf{\Phi} = \sum_n \log p(\mathbf{x}_n) - D_{KL}(\{q_{ns}\} \parallel \{p_{ns}\}) - \sum_s q_{ns} \mathcal{D}_{ns} \tag{10}$$

$$= \sum_{ns} \int_{\mathbf{g}} \mathrm{d}\mathbf{g} \;\; Q(\mathbf{g}, s \mid \mathbf{x}_n) [- \log Q(\mathbf{g}, s \mid \mathbf{x}_n) + \log p(\mathbf{x}_n, \mathbf{g}, s)] \tag{11}$$

The form of (11) shows that we can view this as a variational approach to fit the mixture model where we consider both s and $\mathbf{g}$ as hidden variables. We approximate the true $p(\mathbf{g}, s \mid \mathbf{x})$ with the 'simple' $Q(\mathbf{g}, s \mid \mathbf{x})$. The terms involving the mixture model p, represent the expected likelihood. The penalty term makes the distributions $Q(\mathbf{g}, s \mid \mathbf{x}_n)$ mimmic the distributions $p(\mathbf{g}, s \mid \mathbf{x}_n)$.

Our density model differs with that of [Roweis *et al.*, 2002] in two aspects:

1. isotropic noise model outside the subspaces (as opposed to diagonal covariance matrix)

2. isotropic variance in subspace (as opposed to general Gaussian).

As a result, orthogonal mappings from the subspaces to the latent space can be absorbed in the matrices $\mathbf{\Lambda}$, since for orthogonal matrices $\mathbf{R}$ we have $\mathbf{\Lambda}\mathbf{R}^\top(\mathbf{\Lambda}\mathbf{R}^\top)^\top = \mathbf{\Lambda}\mathbf{\Lambda}^\top$, i.e. the density model does not change if we replace $\mathbf{\Lambda}$ with $\mathbf{\Lambda}\mathbf{R}^\top$. Also, using our density model it turns out that to optimize $\mathbf{\Phi}$ with respect to Σ_n, it should be of the form[1] $\Sigma_n = \beta_n^{-1}\mathbf{I}_d$. Therefore we work with β_n from now on.

Using our density model we can rewrite the objective function (up to some constants) as:

$$\mathbf{\Phi} = \sum_{ns} q_{ns}\Bigg[-\frac{d}{2}\log\beta_n - \log q_{ns} - \frac{e_{ns}}{2\sigma_s^2} - \frac{v_s}{2}\Big[d\beta_n^{-1} + \frac{\mathbf{g}_{ns}^\top\mathbf{g}_{ns}}{\rho_s+1}\Big] \tag{12}$$

$$-D\log\sigma_s + \frac{d}{2}\log\frac{v_s}{\rho_s+1} + \log p_s\Bigg], \tag{13}$$

where we used the following abbreviations:

$$\mathbf{g}_{ns} = \mathbf{g}_n - \boldsymbol{\kappa}_\mathbf{s}, \qquad \mathbf{x}_{ns} = \mathbf{x}_n - \boldsymbol{\mu}_\mathbf{s}, \qquad e_{ns} = \| \mathbf{x}_{ns} - \alpha_s^{-1}\mathbf{\Lambda}_s\mathbf{R}_s^\top\mathbf{g}_{ns} \|^2, \qquad v_s = \frac{\rho_s+1}{\sigma_s^2\rho_s\alpha_s^2}. \tag{14}$$

3.2 Optimization

Next, we give an EM-style algorithm to optimize $\mathbf{\Phi}$, it is a simplified version of the algorithm provided in [Roweis *et al.*, 2002]. The simplifications are: (i) the iterative process to solve for the $\mathbf{\Lambda}_s, \mathbf{A}_s$ is no longer needed; an exact update is possible and (ii) the E-step no longer involves matrix inversions.

The same manner of computation is used: in the E-step, we compute the uni-modal distributions $Q(s, \mathbf{g} \mid \mathbf{x}_n)$, parameterized by $\beta_n, \mathbf{g}_n$ and q_{ns}. Let $\langle\mathbf{g}_n\rangle_s = \mathrm{E}_{p(\mathbf{g}|\mathbf{x}_n,s)}[\mathbf{g}]$ denote the expected value of $\mathbf{g}$ given $\mathbf{x}_n$ and s. We used the following identities in the E-step update equations:

$$\langle\mathbf{g}_n\rangle_s = \boldsymbol{\kappa}_s + \mathbf{R}_s\mathbf{\Lambda}_s^\top\mathbf{x}_{ns}\alpha_s\rho_s/(\rho_s+1), \tag{15}$$

$$\mathcal{D}_{ns} = \frac{v_s}{2}[d\beta_n^{-1} + \| \mathbf{g}_n - \langle\mathbf{g}_n\rangle_s \|^2] + \frac{d}{2}[\log\beta_n - \log v_s]. \tag{16}$$

The approximating distributions Q can be found by iterating the fixed-point equations given below. Due to the form of the update for q_{ns} it seems to make sense to initialize the q_{ns} at p_{ns}.

$$\beta_n = \sum_s q_{ns}v_s, \qquad \mathbf{g}_n = \beta_n^{-1}\sum_s q_{ns}v_s\langle\mathbf{g}_n\rangle_s, \qquad q_{ns} = \frac{p_{ns}\exp -\mathcal{D}_{ns}}{\sum_{s'} p_{ns'}\exp -\mathcal{D}_{ns'}}. \tag{17}$$

In the M-step, we update the parameters of the mixture model: $p_s, \boldsymbol{\mu}_s, \boldsymbol{\kappa}_s, \Lambda_s R_s, \sigma_s, \rho_s$. Again we use some compacting notation:

$$C_s = \sum_n q_{ns} \| \mathbf{g}_{ns} \|^2, \qquad E_s = \sum_n q_{ns}e_{ns}, \qquad G_s = d\sum_n q_{ns}\beta_n^{-1} \tag{18}$$

Then, the update equations are:

$$p_s = \frac{\sum_n q_{ns}}{\sum_{ns'} q_{ns'}}, \qquad \boldsymbol{\kappa}_s = \frac{\sum_n q_{ns}\mathbf{g}_n}{\sum_n q_{ns}}, \qquad \boldsymbol{\mu}_s = \frac{\sum_n q_{ns}\mathbf{x}_n}{\sum_n q_{ns}}, \tag{19}$$

$$\alpha_s = \frac{C_s + G_s}{\sum_n q_{ns}(\mathbf{g}_{ns}^\top\mathbf{R}_s\mathbf{\Lambda}_s^\top\mathbf{x}_{ns})}, \qquad \rho_s = \frac{D(C_s+G_s)}{d(\alpha_s^2E_s + G_s)}, \qquad \sigma_s^2 = \frac{E_s + \rho_s^{-1}\alpha_s^{-2}[C_s + (\rho_s+1)G_s]}{(D+d)\sum_n q_{ns}} \tag{20}$$

[1] Once we realize that the matrices V_s in [Roweis *et al.*, 2002] are of the form $c\mathbf{I}_d$ with our density model, it can be seen easily by setting $\partial\mathbf{\Phi}/\partial\mathbf{\Sigma}_n = 0$ that $\mathbf{\Sigma}_n = \beta^{-1}\mathbf{I}_d$.

Note that the above equations require E_s which in turn requires the $\mathbf{\Lambda}_s\mathbf{R}_s$ via equation (14). To find $\mathbf{R}_s\mathbf{\Lambda}_s^\top$ we have to minimize:

$$\sum_n q_{ns}e_{ns} = \sum_n q_{ns} \parallel \mathbf{x}_{ns} - \alpha_s^{-1}\mathbf{\Lambda}\mathbf{R}^\top\mathbf{g}_{ns} \parallel^2 \quad \text{or equivalently:} \quad -\sum_n q_{ns}\mathbf{g}_{ns}^\top(\mathbf{R}_s\mathbf{\Lambda}_s^\top)\mathbf{x}_{ns}. \tag{21}$$

This problem is known as the 'weighted Procrustes rotation' [Cox and Cox, 1994]. Let

$$\mathbf{C} = [\sqrt{q_{1s}}\mathbf{x}_{1s} \cdots \sqrt{q_{ns}}\mathbf{x}_{ns}][\sqrt{q_{1s}}\mathbf{g}_{1s} \cdots \sqrt{q_{ns}}\mathbf{g}_{ns}]^\top, \qquad \text{with SVD: } \mathbf{C} = \mathbf{U}\mathbf{L}\mathbf{\Gamma}^\top, \tag{22}$$

where the $\mathbf{g}_{ns}$ have been padded with zeros to form D-dimensional vectors, then the optimal $\mathbf{\Lambda}_s\mathbf{R}_s^\top$ is given by the first d columns of $U\Gamma^\top$.

4 Initialization from a data-space mixture

To initialize the optimization procedure described in the previous section, we can go two ways. The two ways correspond to starting with an E-step or with an M-step. In [Roweis *et al.*, 2002] it is proposed to start with an M-step, requiring initial estimates for the global coordinates $\mathbf{g}_n$. The initial global coordinates are typically provided by an 'external' unsupervised procedures, examples are LLE and Isomap. However, such methods might suffer from bad time complexity scaling with increasing sample-size[2]. In this section we introduce an alternative initialization, that starts the update procedure with an E-step. This requires we find a data-space mixture model first, we can use the GEM algorithm described in Appendix A. Note that the greedy initialization method described in [Verbeek *et al.*, 2001a], that builds the mixture model component-wise in order to avoid problems due to 'unlucky' random initialization of the mixture, can be directly used for the mixtue model described here.

4.1 Fixing the data-space mixture

Next, we consider how we can simplify the objective (13), if we fix the data-space mixture model. In this case the problem reduces to find appropriate mappings $\{\boldsymbol{\kappa}_s, \mathbf{A}_s\}$. Since the data-space mixture is fixed the log-likelihood term is constant and the objective reduces to the penalty term (9). If the density model is fixed, we can reduce the number of free parameters in (9) by setting $q_{ns} = p_{ns}$. This simplifies (9) to:

$$\sum_{ns} p_{ns}\mathcal{D}_{ns}. \tag{23}$$

Furthermore, we replace the Gaussian $Q(\mathbf{g} \mid \mathbf{x}_n)$ with a delta-peak distribution on its center $\mathbf{g}_n$, this removes the free parameters $\mathbf{\Sigma}_n$. This amounts to simplifying (23) further to a weighted sum of log-likelihoods:

$$\sum_{n,s} p_{ns} \log p(\mathbf{g}_n \mid \mathbf{x}_n, s) \tag{24}$$

The posterior distribution $p(\mathbf{z} \mid \mathbf{x}, s)$ on internal coordinates $\mathbf{z}$ given a data point $\mathbf{x}$ is a Gaussian and hence also the posterior $p(\mathbf{g} \mid \mathbf{x}, s)$ is a Gaussian. The covariance of the posterior $p(\mathbf{z} \mid \mathbf{x}, s)$ is $\mathbf{I}_d/(\rho_s + 1)$ and to get the covariance for $\mathbf{g}$ we only have to multiply this with the square of the scaling factor of $\mathbf{A}_s$ which yields: $\mathbf{I}_d\alpha_s^2\sigma_s^2\rho_s/(\rho_s + 1) = \mathbf{I}_d v_s^{-1}$

[2]Both LLE and Isomap scale in principle at least quadratic, due to the neighborhood graph construction.

4.2 Optimization and Initialization

Global maximization of (24) over all parameters is hard. Therefore, we maximize by alternating maximization over the global coordinate point estimates $\{\mathbf{g}_n\}$, and then over the patch locations and rotations $\{\boldsymbol{\kappa}_s, \mathbf{R}_s\}$. This leads to a local maximum of the objective. Skipping some constants, we rewrite (24) as:

$$-\frac{1}{2}\sum_{n,s} p_{ns}[2d\log\alpha + v_s \parallel \langle \mathbf{g}_n\rangle_s - \mathbf{g}_n \parallel^2]. \tag{25}$$

In Appendix B we show how to find locally optimal parameters. Note that if the mixture in the data-space is kept fixed, then all computations involve d dimensional vectors and matrices, as expected.

Finally, to initialize the mappings we take component-wise approach, using the already initialized components to initialize the next. Note that due to the invariance w.r.t. global translations and rotations of the objective (24), we can simply initialize the first component with $\mathbf{R} = \mathbf{I}_d, \alpha = 1, \boldsymbol{\kappa} = 0$. To initialize a new component s, we compute $\mathbf{R}_s, \alpha_s, \boldsymbol{\kappa}_s$ which maximize:

$$\sum_n \Big[\sum_{i\in F} p_{ni}\Big] p_{ns} \log p(\mathbf{g}_n \mid \mathbf{x}_n, s), \tag{26}$$

where F is the set of components for which we already fixed the mapping, i.e. the weights p_{ns} are rescaled according to the already initialized components. Again, the solution is found in Appendix B. Note that these weights emphasize the data 'on the border' between the subspace s and the already initialized subspaces. In order to select the next subspace to initialize, we could for example pick

$$s = \arg\max_s \sum_n p_{ns} \sum_{i\in F} p_{ni}/p_s. \tag{27}$$

This method somewhat resembles the method described in [Verbeek *et al.*, 2001b]. There, also a similar density model is learned and then fixed to compute how the different local models can be combined into a global structure. The method presented here is more robust and applicable to any latent dimensionality.

5 Experimental results

In this section we discuss three experiments to illustrate the method. The first concerns artificially generated data. The other two experiments deal with mapping sets of images to low dimensional coordinates.

5.1 Artificial data-set

As a first experiment we used an artificially generated data-set, depicted in Figure 2. We learned a mixture model with 20 components and initialized the mappings to the latent space from the data-space mixture. For all data points we also have the coordinates on the surface, so we can inspect correlation coefficients with the discovered latent coordinates. The correlation coefficient between the first latent dimension and the surface coordinates are: 0.0494 and 0.9997. For the second latent dimension these are: 0.9961 and -0.0030.

5.2 Images of faces

For this experiment we captured gray valued images of a face with a webcam. Each image of 40×40 pixels is treated as a 1600-dimensional vector, with each coordinate describing the gray value of a specific pixel.

Figure 2: An artificially generated data set, 1000 points in 3d on a 2d surface.

Figure 3: Images selected at equally space intervals in the discovered latent space.

Looking from left to rigth In the first experiment we used a set of 100 images of a face where there was only one degree of freedom in the data: the face was rotated from left to right when capturing the images. First we mapped the data to a 5 dimensional sub-space with PCA, capturing over 68% of the total variance in the data. Then we learned a coordinated mixture of principal component analyzers, the latent dimensionality was set to one and we used eight mixture components. To initialize, we used the method described in Section 4. The data-space mixture was learned with the greedy method of [Verbeek *et al.*, 2001a], this method builds the mixture component per component to avoid bad initialization of the parameters. In figure 3 we show some of the used images, they are ordered according to the 1-d latent representation found by our method.

Two degrees of freedom Next, we consider a data set similar to the previous, except that the face now has a second degree of freedom, namely looking up and down. First we learned a coordinated mixture model with 1000 randomly selected images from a set of 2000 images of 40×40 pixels each. Again, we used a PCA projection, in this case to 22 dimensions, preserving over 70% of the variance in the data set. We used a latent dimensionality of two and 20 mixture components.

Here we initialized the coordinated mixture model by clamping the latent coordinates $\mathbf{g}_n$ at coordinates found by Isomap and clamping the β_n at small values for 50 iterations. The q_{ns} were initialized at random values, and updated from the start. After the first 50 iterations, we also updated the β_n and $\mathbf{g}_n$. The obtained coordinated mixture model was used to map the remaining 1000 images, the 'test-set' into the latent space. For each image $\mathbf{x}_n$ we can compute $p(\mathbf{g} \mid \mathbf{x}_n)$, as described in Section 3 this distribution can be approximated with a single Gaussian $Q_n = \arg\min_Q D_{KL}(Q \parallel p(\mathbf{g} \mid \mathbf{x}_n))$ with a certain mean and variance. We used the mean of Q_n as the latent coordinate for an image.

In Figure 4 we show the latent coordinates for the test-set. We observe that a sensible 2-dimensional parametrization is found in the training data set, that generalizes to the test set. To illustrate the discovered parametrization further, two examples of linear traversal of the latent space are given in Figure 5.

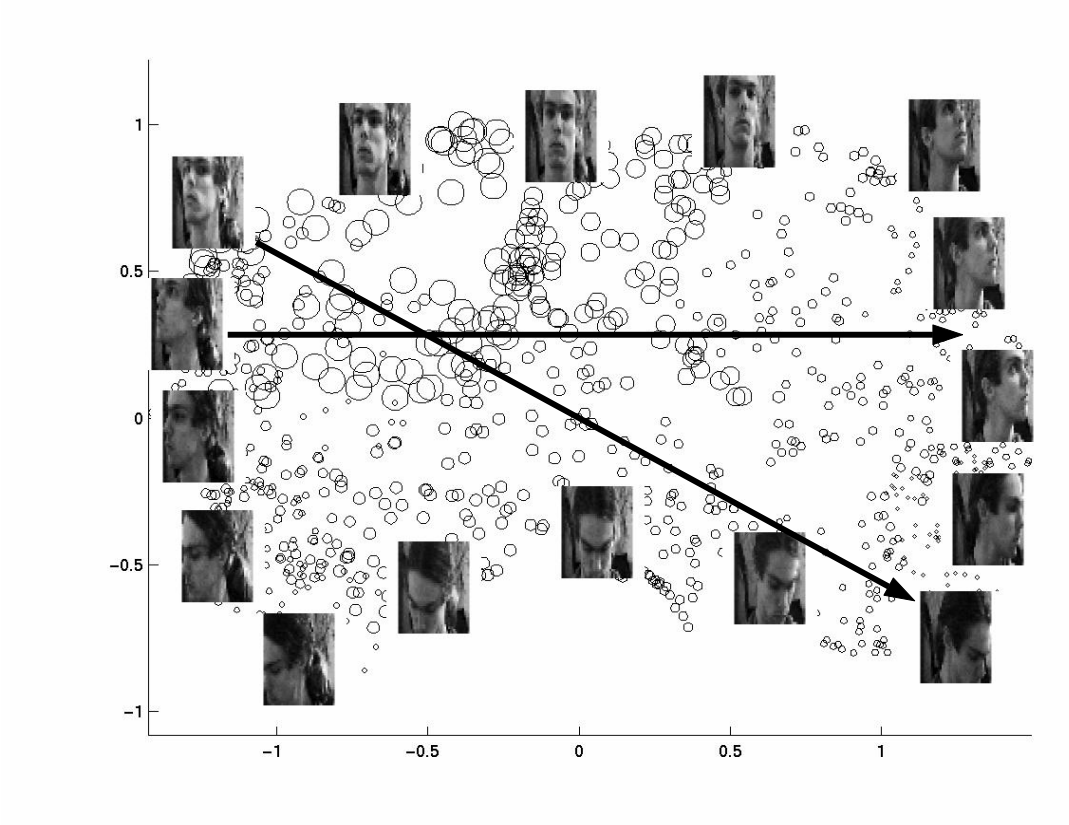

Figure 4: The latent coordinates for the test set. Each circle represents a latent coordinate (its center) and the uncertainty β on it (the radius of the circle is $\beta^{-1/2}$). For some coordinates we displayed the corresponding image at the coordinate.

Figure 5: Examples of linear traversal of the latent space. The followed trajectories correspond to the arrows in Figure 4.

Figure 6: An example of a panoramic image as used by the robot.

5.3 Robot navigation

Here we consider images obtained from a omni-directional camera[3] mounted on a mobile robot. These images are warped to gray-scale panoramic images of 256×64 pixels, an example of such an image is given in Figure 6. The robot was placed at many locations in an office, while the orientation of the robot was kept fixed. At each location the position of the robot was recorded together with an image taken at that position.

First we consider data obtained from a rectangular area of an office, approximately $8 \times 1\frac{1}{2}$ meters wide. The robot recorded images on a 10×10 centimeter grid, in the corridor we obtained 970 images. A uniformly random selected subset of 500 of those images were used to train the coordinated mixture model, with latent dimension two and 20 mixture components. Since we have 'supervised' data here, we can set and keep fixed the $\mathbf{g}_n$ here at the recorded positions of the robot. In case of significant errors in the recorded positions we might only use them for initialization. Prior to training the model, we projected the images to a 15 dimensional space capturing over 70% of the variance in the total 16.384 dimensions.

We mapped the test set to the latent space as described in the previous section. The covariance of the error between the true location and the found latent coordinates is almost diagonal with standard deviations of 13 and 17 centimeters in respectively the horizontal and vertical direction. The ellipse in Figure 7 illustrates the covariance structure, the axes of the ellipse are parallel to the eigenvectors of the covariance matrix, and their lengths equal to square root of the corresponding eigenvalues.

The same experiment was repeated with a data from a larger region, see the right pannel in Figure 7. Here the images were projected on a 50-dimensional space and 50 mixture components were used. From the total 2435 images 1700 were used for training and 735 for testing. In this experiment, the standard deviation of the error along the eigenvectors of the covariance matrix was 15 and 17 centimeters.

Although these results are good, considering that only linear models are used, we found it very hard to get good environment representation when not using the supervised position information.

6 Discussion and conclusions

Discussion In the previous section we showed examples of application of the model to visual data. Both supervised and unsupervised data sets were used. The mains reason for applying this model to supervised data instead of other regression methods are the compact decription of the mapping between the high and low dimensional space, the probabilistic framework in which it is stated and the simplicity in terms of computational effort of the mapping.

One might consider applying this method to partially supervised data sets (only a limited number of measurements is provided together with a 'supervised' latent coordinate). This possibility forms a line of further research.

Another future line of research is the use of the decribed density model to get an idea of the

[3] In fact, a normal camera looks in vertical direction onto a parabolic mirror mounted above the camera.

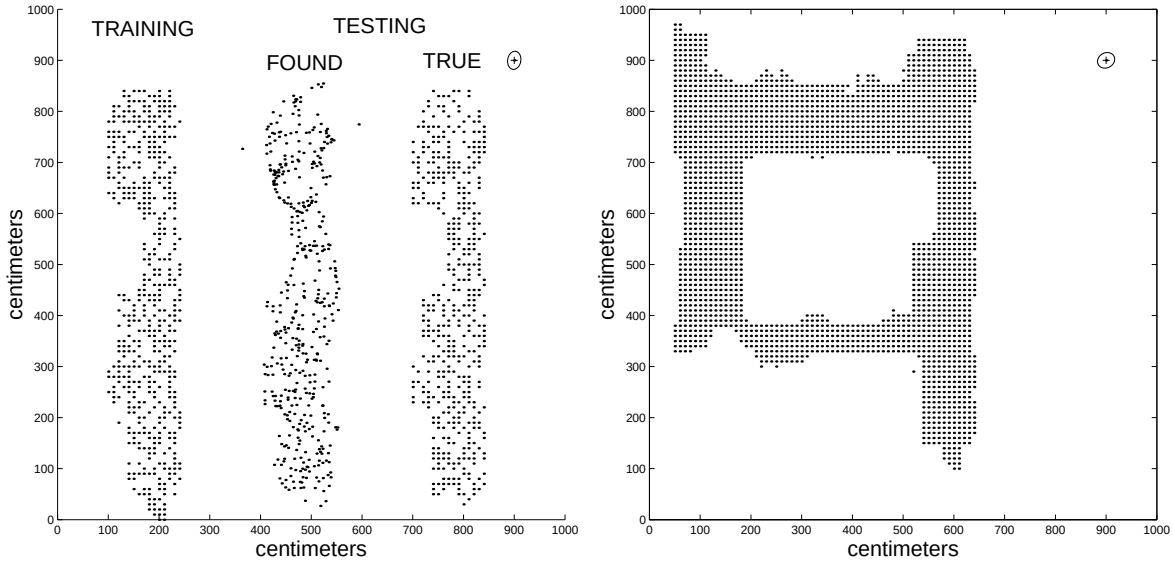

Figure 7: Positions in the robot navigation task. The training locations are shown left, the found locations for the test set are shown in the middle with the original locations right. The ellipse (top right) shows the covariance structure in the error. The right pannel shows the locations and error covariance for the second experiment.

(local) latent dimensionality. If we treat the latent dimensionality a unknown, we could try to estimate it by checking at the M step in an EM procedure for every component which latent dimensionality maximizes log-likelihood. As opposed to th MPPCA anf MFA density models, with our density model the covariance matrices do not form a nested family. With MPPCA and MFA the covariance matrices $\mathcal{C}_d$ for latent dimensionality d are included in the covariance matrices for latent dimensionality $d' > d$: $\mathcal{C}_d \subset \mathcal{C}_{d'}$. Hence with MPPCA and MFA, increasing the latent dimensionality can only increase the likelihood.

An important issue, not addressed here, is that is many cases where we collect data from a systems with only few degrees of freedom we actually collect one or more *sequences* of data. If we assume that the system can vary its state only in continuous manner, these sequences should correspond to paths on the manifold of observable data. This fact might be exploited to find low dimensional embeddings of the manifold.

Conclusions We showed how a special case of the density model used in [Roweis *et al.*, 2002] leads to a more efficient algorithm to coordinate probabilistic local linear descriptions of a data manifold. The M-step can be computed at once, the iterative procedure to find solutions for a Riccati equation is no longer needed. Furthermore, the update equations do not involve matrix inversions anymore. However, still d singular values of a $D \times D$ matrix have to be found.

We prosposed an alternative initialization scheme for the mapping from high to low dimension, that exploits the structure of a mixture model in the data space. We showed experimental results of this method discovering succesfully the structure in a data set of images of a face that rotates over its length axis. We showed how we can use the coordinated mixture model to map supervised data (robot navigation) and data for which we have uncertain latent coordinates (the 2 degrees of freedom facial images).

References

[Cox and Cox, 1994] T.F. Cox and M.A.A. Cox. *Multidimensional Scaling*. Number 59 in Monographs on statistics and applied probability. Chapman & Hall, 1994.

[Ghahramani and Hinton, 1996] Z. Ghahramani and G.E. Hinton. The EM Algorithm for Mixtures of Factor Analyzers. Technical Report CRG-TR-96-1, University of Toronto, 1996.

[Roweis *et al.*, 2002] S. Roweis, L. Saul, and G.E. Hinton. Global coordination of local linear models. In T.G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14, pages x–y. MIT Press, 2002.

[Tipping and Bishop, 1999] M.E. Tipping and C.M. Bishop. Mixtures of probabilistic principal component analysers. *Neural Computation*, 11(2):443–482, 1999.

[Verbeek *et al.*, 2001a] J. J. Verbeek, N. Vlassis, and B. Kröse. Greedy Gaussian mixture learning for texture segmentation. In A. Leonardis and H. Bischof, editors, *ICANN'01, Workshop on Kernel and Subspace Methods for Computer Vision*, pages 37–46, Vienna, Austria, August 2001. Available at: `http://www.science.uva.nl/ ~jverbeek`.

[Verbeek *et al.*, 2001b] J.J. Verbeek, N. Vlassis, and B.J.A. Kröse. A soft k-segments algorithm for principal curves. In *Proc. Int. Conf. on Artificial Neural Networks*, pages 450–456, Vienna, Austria, August 2001.

[Vlassis *et al.*, 2002] N. Vlassis, Y. Motomura, and B. Kröse. Supervised dimension reduction of intrinsically low-dimensional data. *Neural Computation*, 14(1):191–215, January 2002.

A GEM for constrained MPPCA

The constrained MPPCA model, characterized by equation (1), is parameterized by $\{p_s, \theta_s\}$, the mixing weights p_s and $\theta_s = \{\boldsymbol{\mu}_s, \sigma_s, \rho_s, \boldsymbol{\Lambda}_s\}$. Let p_{ns} denote the expectation that component s generated x_n. The expected complete log-likelihood (expectation taken over the unobserved labels that indicate which component generated which data point) is then given by:

$$\langle \mathcal{L} \rangle = \sum_{ns} p(s \mid \mathbf{x}_n) \ln\{\pi_s \tilde{p}(\mathbf{x}_n; \theta_i)\}, \tag{28}$$

where we used $\tilde{p}$ to denote the density model with the new parameters. Optimization of the parameters is easy by using a Generalized EM (GEM) algorithm described below. First, we maximize (28) w.r.t. $\{\tilde{p}_s\}$ and $\{\tilde{\boldsymbol{\mu}}_s\}$, while keeping the other parameters fixed. If we write $p(s \mid \mathbf{x}_n) = p_{ns}$ this gives:

$$\tilde{\boldsymbol{\mu}}_s = \frac{\sum_n p_{ns} \mathbf{x}_n}{\sum_n p_{ns}}, \qquad \tilde{p}_s = \frac{\sum_n p_{ns}}{\sum_{ns'} p_{ns'}} \tag{29}$$

Next, we consider maximizing (28) w.r.t. the other parameters while keeping $\{\tilde{p}_s\}$ and $\{\tilde{\boldsymbol{\mu}}_s\}$ fixed. This gives us parameter values that yield even higher values for (28) and hence we are guaranteed to obtain increased log-likelihood by the GEM algorithm. Before we turn to the derivation of the GEM algorithm we note that the likelihood under component s reads:

$$p(\mathbf{x}_n \mid s) = \sigma_s^{-D} (2\pi[\rho_s + 1])^{-d/2} \exp \frac{-1}{2\sigma_s^2} [\, \| \mathbf{x}_{ns} \|^2 - \frac{\rho_s}{\rho_s + 1} \| \boldsymbol{\Lambda}^\top \mathbf{x}_{ns} \|^2 \,] \tag{30}$$

A.1 Updating the loading matrices

The weighted covariance matrix for component s is defined as:

$$\mathbf{S}_s = \frac{\sum_n p_{ns}\mathbf{x}_{ns}\mathbf{x}_{ns}^\top}{\sum_n p_{ns}}, \tag{31}$$

where we use $\mathbf{x}_{ns} = \mathbf{x}_n - \tilde{\boldsymbol{\mu}}_s$. Let $\lambda_{sj}, \mathbf{u}_{sj}$ denote the eigenvalues respectively eigenvectors of $\mathbf{S}_s$ sorted from large to small eigenvalues. We show that for $\sigma_s, \rho_s > 0$ setting $\tilde{\boldsymbol{\Lambda}}_s = [\mathbf{u}_{s1} \cdots \mathbf{u}_{sd}]$ maximizes the expected log-likelihood. To maximize (28) with respect to $\tilde{\boldsymbol{\Lambda}}_s$ for given $\tilde{\boldsymbol{\mu}}_s$, consider the terms for $\tilde{\boldsymbol{\Lambda}}_s$ in (28):

$$\sum_n p_{ns} \ln \tilde{p}(x_n; \theta_s) = -\frac{1}{2}\sum_n p_{ns}[\ln\det(2\pi\tilde{\mathbf{C}}_s) + \mathbf{x}_{ns}^\top \tilde{\mathbf{C}}_s^{-1}\mathbf{x}_{ns}]. \tag{32}$$

Note that

$$\tilde{\mathbf{C}} = \tilde{\sigma}^2(\mathbf{I}_D + \tilde{\rho}\tilde{\boldsymbol{\Lambda}}\tilde{\boldsymbol{\Lambda}}^\top) = \tilde{\sigma}^2\mathbf{T}(\mathbf{I}_D + \tilde{\rho}\mathbf{I}^*)\mathbf{T}^\top, \tag{33}$$

where $\mathbf{T} = [\mathbf{V}\mathbf{Q}]$ is a $D \times D$ matrix with pairwise orthonormal columns and $\mathbf{Q}$ an arbitrary $D \times (D-d)$ matrix that fits the orthonormality constraints. The matrix $\mathbf{I}^*$ denotes the matrix which is all zero except for the first d diagonal elements. From this is easy to see that the determinant of $\tilde{\mathbf{C}}$ is invariant for $\tilde{\boldsymbol{\Lambda}}$:

$$\det(\tilde{\mathbf{C}}) = \det(\mathbf{T})\det(\tilde{\sigma}^2(\mathbf{I}_D + \tilde{\rho}\mathbf{I}^*))\det(\mathbf{T}^\top) \tag{34}$$

$$= \det(\tilde{\sigma}^2(\mathbf{I}_D + \tilde{\rho}\mathbf{I}^*)) = (\tilde{\sigma}^2)^D(1+\tilde{\rho})^d \tag{35}$$

So maximizing (32) is equivalent to minimizing:

$$\sum_n p_{ns}\mathbf{x}_{ns}^\top\tilde{\mathbf{C}}_s^{-1}\mathbf{x}_{ns} = \tilde{\sigma}_s^{-2}\sum_n p_{ns}(\mathbf{T}^\top\mathbf{x}_{ns})^\top(\mathbf{I}_D + \tilde{\rho}\mathbf{I}^*)^{-1}\mathbf{T}^\top\mathbf{x}_{ns} \tag{36}$$

$$= \tilde{\sigma}_s^{-2}\Big[\sum_n p_{ns}\mathbf{x}_{ns}^\top\mathbf{x}_{ns} - \frac{\tilde{\rho}_s}{\tilde{\rho}_s+1}\sum_n p_{ns}\mathbf{x}_{ns}^\top\mathbf{T}^\top\mathbf{I}^*\mathbf{T}\mathbf{x}_{ns}\Big] \tag{37}$$

For $\tilde{\sigma}_s, \tilde{\rho}_s > 0$, this is equivalent to maximizing:

$$\sum_n p_{ns}(\mathbf{T}^\top\mathbf{x}_{ns})^\top\mathbf{I}^*(\mathbf{T}^\top\mathbf{x}_{ns}), \tag{38}$$

i.e. maximizing the weighted variance in the subspace spanned by the first d columns of $\mathbf{T}^\top$. This is solved by the first d eigenvectors of the weighted covariance matrix. Note that this solution for $\tilde{\boldsymbol{\Lambda}}_s$ is independent of the actual $\tilde{\sigma}_s$ and $\tilde{\rho}_s$. So after computing $\{\tilde{\boldsymbol{\mu}}_s\}$ and the $\{\tilde{p}_s\}$, we use the first d eigenvectors of the weighted covariance matrix as $\tilde{\boldsymbol{\Lambda}}_s$.

A.2 Variance inside and outside the subspace

In order to maximize (28) w.r.t. $\tilde{\sigma}_s$ and $\tilde{\rho}_s$, for fixed $\tilde{\boldsymbol{\mu}}_s, \tilde{p}_s, \tilde{\boldsymbol{\Lambda}}_s$, we first consider the terms in (28) for $\tilde{\rho}_s$:

$$\sum_n p_{ns}\ln\tilde{p}(\mathbf{x}_n;\theta_s) = -\frac{1}{2}\sum_n p_{ns}[\ln\det(\tilde{C}_s) + \mathbf{x}_{ns}^\top\tilde{C}_s^{-1}\mathbf{x}_{ns}] \tag{39}$$

$$= -\frac{d}{2}\Big[\sum_n p_{ns}\Big]\ln(\tilde{\rho}_s+1) - \frac{1}{2\tilde{\sigma}_s^2}\sum_n p_{ns}\mathbf{x}_{ns}^\top(\mathbf{I}_D + \tilde{\rho}_s\tilde{\boldsymbol{\Lambda}}_s\tilde{\boldsymbol{\Lambda}}_s^\top)^{-1}\mathbf{x}_{ns} \tag{40}$$

$$= -\frac{d}{2}\Big[\sum_n p_{ns}\Big]\ln(\tilde{\rho}_s+1) - \frac{1}{2\tilde{\sigma}_s^2}(\tilde{\rho}_s+1)^{-1}\sum_n p_{ns}\mathbf{x}_{ns}^\top\tilde{\boldsymbol{\Lambda}}_s\tilde{\boldsymbol{\Lambda}}_s^\top\mathbf{x}_{ns}, \tag{41}$$

where the last equality follows from (33) and we ignored some additive constants in the equalities. Setting the derivative of (41) w.r.t. $\tilde{\rho}_s$ equal to zero, and doing the same for $\tilde{\sigma}_s$ we obtain:

$$\tilde{\rho}_s + 1 = \frac{\sum_n p_{ns} \mathbf{x}_{ns}^\top \tilde{\mathbf{\Lambda}}_s \tilde{\mathbf{\Lambda}}_s^\top \mathbf{x}_{ns}}{d\tilde{\sigma}_s^2 \sum_n p_{ns}}, \qquad \tilde{\sigma}_s^2 (D + d\tilde{\rho}_s) = \frac{\sum_n p_{ns} \mathbf{x}_{ns}^\top \mathbf{x}_{ns}}{\sum_n p_{ns}}. \tag{42}$$

Combining these we obtain:

$$\tilde{\sigma}_s^2 = \frac{1}{D-d} \sum_{i=d+1}^{D} \lambda_{si}, \qquad \tilde{\rho}_s + 1 = \frac{1}{d\tilde{\sigma}_s^2} \sum_{i=1}^{d} \lambda_{si}, \tag{43}$$

where the λ_{si}'s are as before. Note that $\tilde{\sigma}_s^2$ is intuitively interpreted as the mean variance outside the subspace. Similarly, $(\tilde{\rho}_s + 1)$ equals the mean variance inside the subspace divided over $\tilde{\sigma}_s^2$.

B A variation on weighted Procrustes analysis

The quadratic form we need minimize with respect to $\{\mathbf{g}_n\}, \{\boldsymbol{\kappa}_s\}, \{\mathbf{R}_s\}, \{\alpha_s\}$ is:

$$\sum_{ns} p_{ns} [2d \log \alpha_s + c_s \alpha_s^{-2} \parallel \boldsymbol{\kappa}_s + \alpha_s \mathbf{R}_s \mathbf{z}_{ns} - \mathbf{g}_n \parallel^2], \tag{44}$$

for constants $c_s, \{\mathbf{z}_{ns}\}, \{p_{ns}\}$. Solving for the $\mathbf{g}_n$ by differentiation of (44) gives:

$$\mathbf{g}_n = \frac{\sum_s p_{ns} c_s \alpha_s^{-2} (\boldsymbol{\kappa}_s + \alpha_s \mathbf{R}_s \mathbf{z}_{ns})}{\sum_s p_{ns} c_s \alpha_s^{-2}}. \tag{45}$$

Then, keeping $\{\mathbf{g}_n\}$ fixed we solve for the other parameters. For $\boldsymbol{\kappa}_s$, this gives:

$$\boldsymbol{\kappa}_s = \frac{\sum_n p_{ns} \mathbf{g}_n}{\sum_n p_{ns}} - \frac{\sum_n p_{ns} \alpha_s \mathbf{R}_s \mathbf{z}_n}{\sum_n p_{ns}} = \bar{\mathbf{g}}_s - \alpha_s \mathbf{R}_s \bar{\mathbf{z}}_s, \tag{46}$$

i.e. the translation makes the weighted means equal, note that α_s is as given by (49). To find the rotation $\mathbf{R}$ and scaling α, we assume that the weighted means are already equal, it is easy to show that this gives the same $\mathbf{R}$ and α. To realize this we set $\tilde{\mathbf{g}}_{ns} = \mathbf{g}_n - \bar{\mathbf{g}}_s$ and $\tilde{\mathbf{z}}_{ns} = \mathbf{z}_{ns} - \bar{\mathbf{z}}_s$ and use these to find the scaling and rotation. Then, for $\mathbf{R}_s$ we need to maximize:

$$\sum_n p_{ns} \tilde{\mathbf{g}}_{ns}^\top \mathbf{R}_s \tilde{\mathbf{z}}_{ns}, \tag{47}$$

which is solved by the Procrustes rotation [Cox and Cox, 1994]. Let

$$\mathbf{C} = [\sqrt{p_{1s}} \tilde{\mathbf{g}}_{1s} \cdots \sqrt{p_{ns}} \tilde{\mathbf{g}}_{ns}][\sqrt{p_{1s}} \tilde{\mathbf{z}}_{1s} \cdots \sqrt{p_{ns}} \tilde{\mathbf{z}}_{ns}]^\top \quad \text{with SVD} \quad \mathbf{C} = \mathbf{U}\mathbf{L}\mathbf{\Gamma}^\top \tag{48}$$

then the solution is $\mathbf{R}_s = \mathbf{\Gamma}\mathbf{U}^\top$. For $\alpha > 0$, setting the derivative to zero gives:

$$\alpha_s^2 \underbrace{\frac{d \sum_n p_{ns}}{c_s}}_{a} + \alpha_s \underbrace{\sum_n p_{ns} \tilde{\mathbf{g}}_{ns}^\top \mathbf{R}_s \tilde{\mathbf{z}}_{ns}}_{b} - \underbrace{\sum_n p_{ns} \tilde{\mathbf{g}}_{ns}^\top \tilde{\mathbf{g}}_{ns}}_{c} = 0, \quad \leftrightarrow \quad \alpha = \frac{-b + \sqrt{b^2 - 4ac}}{2a} > 0. \tag{49}$$

In order to prevent degenerate solutions which collapse all data into a single point, we need to put a constraint on the scale of the latent space. Either we fix $\{\alpha_s\}$ or we could re-scale after each step such that $\sum_n \parallel \mathbf{g}_n \parallel^2 = 1$.

Copyright IAS, 2000

Acknowledgements

This research is supported by the Technology Foundation STW, applied science division of NWO and the technology program of the Ministry of Economic Affairs.

IAS reports

This report is in the series of IAS technical reports. The series editor is Stephan ten Hagen (`stephanh@science.uva.nl`). Within this series the following titles appeared:

[1] N. Vlassis . *Supervised dimension reduction of intrinsically low-dimensional data.* Technical Report IAS-UVA-00-07, Intelligent Autonomous Systems Group, University of Amsterdam, September 2000.

[2] J.J. Verbeek, N. Vlassis and B.J.A Kröse. A k-segments algorithm for finding principal curves. Technical Report IAS-UVA-00-11, Intelligent Autonomous Systems Group, University of Amsterdam, December 2000.

[3] J.J. Verbeek, N. Vlassis and B.J.A Kröse. Efficient greedy learning of Gaussian mixtures. Technical Report IAS-UVA-01-10, Intelligent Autonomous Systems Group, University of Amsterdam, May 2001.

[4] Aristidis Likas, Nikos Vlassis, and Jakob J. Verbeek *The global k-means clustering algorithm.* Technical Report IAS-UVA-01-02, Intelligent Autonomous Systems Group, University of Amsterdam, 2001.

You may order copies of the IAS technical reports from the corresponding author or the series editor. Most of the reports can also be found on the web pages of the IAS group (see the inside front page).

Intelligent Autonomous Systems