Dimensionality Reduction by Unsupervised Regression

Miguel Á. Carreira-Perpiñán
EECS, University of California, Merced
mcarreira-perpinan@ucmerced.edu

Zhengdong Lu
CSEE, OGI, Oregon Health & Science University
zhengdon@csee.ogi.edu

Abstract

We consider the problem of dimensionality reduction, where given high-dimensional data we want to estimate two mappings: from high to low dimension (dimensionality reduction) and from low to high dimension (reconstruction). We adopt an unsupervised regression point of view by introducing the unknown low-dimensional coordinates of the data as parameters, and formulate a regularised objective functional of the mappings and low-dimensional coordinates. Alternating minimisation of this functional is straightforward: for fixed low-dimensional coordinates, the mappings have a unique solution; and for fixed mappings, the coordinates can be obtained by finite-dimensional nonlinear minimisation. Besides, the coordinates can be initialised to the output of a spectral method such as Laplacian eigenmaps. The model generalises PCA and several recent methods that learn one of the two mappings but not both; and, unlike spectral methods, our model provides out-of-sample mappings by construction. Experiments with toy and real-world problems show that the model is able to learn mappings for convoluted manifolds, avoiding bad local optima that plague other methods.

We can classify most nonlinear dimensionality reduction methods into three categories: latent variable models, mapping-based methods and spectral methods. *Latent variable models* (LVMs), such as the generative topographic mapping (GTM) [2], are the most ambitious: they aim at estimating the joint density $p(\mathbf{x}, \mathbf{y})$ of the observed ($\mathbf{y}$) and latent ($\mathbf{x}$) variables (i.e., they are generative models), and from this they can define mappings for dimensionality reduction $\mathbf{F}(\mathbf{y}) = \mathrm{E}\{\mathbf{x}|\mathbf{y}\}$ and reconstruction $\mathbf{f}(\mathbf{x}) = \mathrm{E}\{\mathbf{y}|\mathbf{x}\}$ (i.e., the mean of $p(\mathbf{x}|\mathbf{y})$ and $p(\mathbf{y}|\mathbf{x})$, respectively), and deal with missing data. Unfortunately, both parameter estimation (by maximum likelihood) and dimensionality reduction require the marginal observed-data distribution $p(\mathbf{y}) = \int p(\mathbf{y}|\mathbf{x})p(\mathbf{x})\,d\mathbf{x}$, which for nonlinear methods is intractable unless the latent space is discretised. This means the methods are limited to low latent dimensionalities (since the grid size grows exponentially with it). They are also prone to getting stuck in bad local optima, because the likelihood is very wiggly, particularly when the data manifold is highly nonlinear and convoluted, e.g. as in a spiral or Swiss roll (an exception, LELVM, is discussed in sec. 4). While the likelihood function does contain local optima that are good approximations to the manifold, it contains far more that are not, and one nearly always ends up in one of those if initialising the parameters with random values or with the principal components of the data.

The prototype *mapping-based method* is the autoencoder, perhaps the most direct approach to nonlinear dimensionality reduction (and one of the earliest). Autoencoders do not estimate the density (i.e., they are not generative models), but estimate parametric mappings for dimensionality reduction $\mathbf{F}$ and reconstruction $\mathbf{f}$ that minimise the reconstruction error of the data $\{\mathbf{y}_n\}_{n=1}^N$:

$$E(\mathbf{f}, \mathbf{F}) = \textstyle\sum_{n=1}^N \|\mathbf{y}_n - \mathbf{f}(\mathbf{F}(\mathbf{y}_n))\|^2. \qquad (1)$$

The mappings are typically neural networks with many parameters, trained with backpropagation. Autoencoders have the same local optima problem as LVMs, since the reconstruction error is very nonconvex. Besides, training deep nets (which theoretically can model complex relationships) has been difficult (though a recent approach [5] has been proposed to improve this). However, autoencoders have no problem using latent spaces of any dimensionality, since they do not model its density. Other mapping-based methods [13, 6, 8, 10, 7, 11] are reviewed in section 4.

A more recent class of methods are *spectral manifold learning methods* such as Laplacian eigenmaps (LE) [1], LLE [12] or Isomap [14]. These give up entirely with estimating either mappings or densities and instead directly estimate the coordinates $\mathbf{X}$ of the latent points. Typically, they set up a quadratic objective derived from a neighbourhood graph and solve for its leading eigenvectors, projecting on which yields $\mathbf{X}$. They have important advantages over LVMs and autoencoders: they have no local optima, yet they often succeed in recovering good embeddings for difficult problems such as the spiral or Swiss roll (thanks to the neighbourhood graph). And they can use any dimension L for the latent space by simply computing L eigenvectors.

Our approach is closest to autoencoders. We formulate a conceptually straightforward objective function that deals symmetrically with both dimensionality reduction and reconstruction mappings, but explicitly introduces as auxiliary parameters the latent coordinates (unsupervised regression). The latter allow the use of a good initialisation provided by the embedding of a spectral method. The training algorithm alternates between solving a variational problem for the mappings—which yields a radial basis function (RBF) expansion at the data and latent points—and a finite-dimensional problem for the latent points. Our method avoids several of the problems that plague other methods: it can work with any dimension (observed or latent), it provides mappings both ways (but no densities), and it is less prone to bad local optima. We describe the method formulation and algorithm in section 1, discuss computational issues in section 2, demonstrate the method in a difficult toy problem and real-world problems in section 3, and discuss related work in section 4.

1. Dimensionality Reduction by Unsupervised Regression (DRUR)

Consider a dataset $\{\mathbf{y}_n\}_{n=1}^N \subset \mathbb{R}^D$ of N points in D dimensions, represented as a $D \times N$ matrix $\mathbf{Y} = (\mathbf{y}_1, \dots, \mathbf{y}_N)$, and assume they live in a manifold of dimension $L < D$. We want to estimate mappings $\mathbf{F} : \mathbb{R}^D \to \mathbb{R}^L$ for dimensionality reduction and $\mathbf{f} : \mathbb{R}^L \to \mathbb{R}^D$ for reconstruction. Call $\mathbb{R}^D$ and $\mathbb{R}^L$ (or appropriate subsets thereof) the observed (or data) space and the latent space, respectively. Let us introduce as auxiliary parameters the low-dimensional coordinates in latent space of points $\{\mathbf{x}_n\}_{n=1}^N \subset \mathbb{R}^L$, represented as an $L \times N$ matrix $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_N)$, and respectively associated with the observed points. We define a method of *Dimensionality Reduction by Unsupervised Regression (DRUR)* to estimate the mappings as minima of the following variational problem:

$$\min_{\mathbf{X},\mathbf{f},\mathbf{F}} E(\mathbf{X},\mathbf{f},\mathbf{F}) = E_{\mathbf{f}}(\mathbf{X},\mathbf{f}) + E_{\mathbf{F}}(\mathbf{X},\mathbf{F}) \quad (2)$$

$$E_{\mathbf{f}}(\mathbf{X},\mathbf{f}) = \textstyle\sum_{n=1}^N \|\mathbf{y}_n - \mathbf{f}(\mathbf{x}_n)\|^2 + \lambda_{\mathbf{f}} \|\mathcal{D}_{\mathbf{f}}\mathbf{f}\|^2 \quad (3)$$

$$E_{\mathbf{F}}(\mathbf{X},\mathbf{F}) = \textstyle\sum_{n=1}^N \|\mathbf{x}_n - \mathbf{F}(\mathbf{y}_n)\|^2 + \lambda_{\mathbf{F}} \|\mathcal{D}_{\mathbf{F}}\mathbf{F}\|^2 \quad (4)$$

where $\mathbf{X} \in \mathbb{R}^{D \times N}$ (finite-dimensional) and $\mathbf{f}$ and $\mathbf{F}$ belong to appropriate function spaces (infinite-dimensional). The norms $\|\cdot\|$ are, respectively, the 2–norm for vectors and a functional norm for $\mathbf{f}$ and $\mathbf{F}$. That is, each of $E_{\mathbf{f}}$ and $E_{\mathbf{F}}$ is formulated as a regularised least-squares regression problem: $E_{\mathbf{f}}$ for fitting $(\mathbf{X},\mathbf{Y})$ assuming $\mathbf{X}$ as inputs and $\mathbf{Y}$ as outputs, and $E_{\mathbf{F}}$ for fitting $(\mathbf{Y},\mathbf{X})$. If $\mathbf{X}$ were known then these would be supervised regression problems, but $\mathbf{X}$ is unknown, thus the name for our method. We use quadratic regularisers based on differential operators $\mathcal{D}_{\mathbf{f}}$ and $\mathcal{D}_{\mathbf{F}}$ with regularisation parameters $\lambda_{\mathbf{f}},\ \lambda_{\mathbf{F}} \geq 0$. Non-quadratic regularisers or loss functions other than the quadratic error may be used as well. We could also add a regularisation term $E_{\mathbf{X}}(\mathbf{X})$ if desired, e.g. to impose temporal or spatial structure (known from prior knowledge) in the latent space.

We can see intuitively that the two regression terms $E_{\mathbf{f}}$ and $E_{\mathbf{F}}$ compete with each other: $E_{\mathbf{f}}$ likes to separate points in $\mathbf{X}$ from each other so $\mathbf{f}$ can more easily interpolate $(\mathbf{X},\mathbf{Y})$ and reduce the error $\|\mathbf{Y} - \mathbf{f}(\mathbf{X})\|$ ($\sigma_{\mathbf{x}}$ is fixed so $\mathbf{G}_{\mathbf{f}} \to \mathbf{I}$; see later and the continuity argument below). However, $E_{\mathbf{F}}$ likes to drive all $\mathbf{X}$ to $\mathbf{0}$ since then $\mathbf{F}$ can smoothly interpolate $(\mathbf{Y},\mathbf{X})$ (by making $\mathbf{F} \equiv \mathbf{0}$). As shown in the experiments, while dropping either $E_{\mathbf{f}}$ or $E_{\mathbf{F}}$ often does not result in good mappings, the full DRUR objective function converges to an intermediate solution for $\mathbf{X}$ that often does result in good mappings and improves the spectral initialisation. However, apart from these two extremes (separated/clustered $\mathbf{X}$), E does have many other local optima, which the spectral initialisation of $\mathbf{X}$ helps to avoid.

Another good argument for the joint use of both mappings is distance preservation. Loosely speaking, a continuous function $\mathbf{f}$ (where continuity here is controlled by the regularisation parameter $\lambda_{\mathbf{f}}$) has the property that if two inputs $\mathbf{x},\ \mathbf{x}'$ are close then their outputs $\mathbf{f}(\mathbf{x}),\ \mathbf{f}(\mathbf{x}')$ will be close, or equivalently, if two outputs $\mathbf{f}(\mathbf{x}),\ \mathbf{f}(\mathbf{x}')$ are far then their inputs $\mathbf{x},\ \mathbf{x}'$ will be far. If E only depended on $\mathbf{f}$ and $\mathbf{X}$ but not on $\mathbf{F}$ (as is essentially the case in RPM [13] and GPLVM [6, 7]), then nothing prevents having close outputs $\mathbf{f}(\mathbf{x}),\ \mathbf{f}(\mathbf{x}')$ whose inputs $\mathbf{x},\ \mathbf{x}'$ should be close but are placed far apart during training, and this indeed occurs (section 3). In DRUR, both $\mathbf{f}$ and $\mathbf{F}$ are continuous (particularly for strong regularisation). If $\mathbf{f}$ and $\mathbf{F}$ were inverses of each other, we would have that "$\mathbf{x},\ \mathbf{x}'$ close $\Leftrightarrow$ $\mathbf{f}(\mathbf{x}),\ \mathbf{f}(\mathbf{x}')$ close", or equivalently "$\mathbf{x},\ \mathbf{x}'$ far $\Leftrightarrow$ $\mathbf{f}(\mathbf{x}),\ \mathbf{f}(\mathbf{x}')$ far". If they are approximate inverses (and the DRUR objective functional encourages this indirectly) then we should still get a good preservation of distances, and indeed we see that in our experiments.

The fact that E is partially separable suggests minimising E by coordinate descent on $(\mathbf{f},\mathbf{F})$ and $(\mathbf{X})$: $\min_{\mathbf{f},\mathbf{F}} E(\mathbf{X},\mathbf{f},\mathbf{F})$ (adaptation step) and $\min_{\mathbf{X}} E(\mathbf{X},\mathbf{f},\mathbf{F})$ (projection step). We deal with them separately.

Adaptation step For fixed $\mathbf{X}$, the solution of the variational problem

$$\min_{\mathbf{f},\mathbf{F}} E(\mathbf{X},\mathbf{f},\mathbf{F}) = \min_{\mathbf{f}} E_{\mathbf{f}}(\mathbf{X},\mathbf{f}) + \min_{\mathbf{F}} E_{\mathbf{F}}(\mathbf{X},\mathbf{F}) \quad (5)$$

exists and is unique under certain conditions on the function space and the regularisers (see appendix A). Both mappings $\mathbf{f}, \mathbf{F}$ have the form of a radial basis function expansion centred at each of the data points ($\mathbf{x}_1, \dots, \mathbf{x}_N$ or $\mathbf{y}_1, \dots, \mathbf{y}_N$):

$$\mathbf{f}(\mathbf{x}) = \textstyle\sum_{n=1}^N \mathbf{a}_n g(\mathbf{x} - \mathbf{x}_n),\ \mathbf{F}(\mathbf{y}) = \sum_{n=1}^N \mathbf{b}_n G(\mathbf{y} - \mathbf{y}_n)$$

where $g(\mathbf{x},\mathbf{x}')$ and $G(\mathbf{y},\mathbf{y}')$ are the Green's functions of the self-adjoint operators $\mathcal{D}_{\mathbf{f}}^*\mathcal{D}_{\mathbf{f}}$ and $\mathcal{D}_{\mathbf{F}}^*\mathcal{D}_{\mathbf{F}}$, respectively. The

vector coefficients $\{\mathbf{a}_n, \mathbf{b}_n\}_{n=1}^N$, written as matrices $\mathbf{A} = (\mathbf{a}_1, \ldots, \mathbf{a}_N)$ of $D \times N$ and $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_N)$ of $L \times N$, are the unique minimisers of the quadratic problems

$$\min_{\mathbf{A}} \|\mathbf{Y} - \mathbf{A}\mathbf{G}_\mathbf{f}\|^2 + \lambda_\mathbf{f} \operatorname{tr}\left(\mathbf{A}\mathbf{G}_\mathbf{f}\mathbf{A}^T\right) \quad (6)$$

$$\min_{\mathbf{B}} \|\mathbf{X} - \mathbf{B}\mathbf{G}_\mathbf{F}\|^2 + \lambda_\mathbf{F} \operatorname{tr}\left(\mathbf{B}\mathbf{G}_\mathbf{F}\mathbf{B}^T\right) \quad (7)$$

where $\|\mathbf{A}\|^2 = \operatorname{tr}\left(\mathbf{A}\mathbf{A}^T\right)$ is the Frobenius norm, and

$$\|\mathcal{D}_\mathbf{f}\mathbf{f}\|^2 = \textstyle\sum_{n,m=1}^N (\mathbf{a}_m^T\mathbf{a}_n) g(\mathbf{x}_m - \mathbf{x}_n) = \operatorname{tr}\left(\mathbf{A}\mathbf{G}_\mathbf{f}\mathbf{A}^T\right)$$
$$\|\mathcal{D}_\mathbf{F}\mathbf{F}\|^2 = \textstyle\sum_{n,m=1}^N (\mathbf{b}_m^T\mathbf{b}_n) G(\mathbf{y}_m - \mathbf{y}_n) = \operatorname{tr}\left(\mathbf{B}\mathbf{G}_\mathbf{F}\mathbf{B}^T\right)$$

and the $N \times N$ symmetric Gram matrices are

$$\mathbf{G}_\mathbf{f} = (g(\mathbf{x}_n - \mathbf{x}_m))_{nm}\,,\ \mathbf{G}_\mathbf{F} = (G(\mathbf{y}_n - \mathbf{y}_m))_{nm}\,. \quad (8)$$

Hence, $\mathbf{A}$ and $\mathbf{B}$ are the solutions of the linear systems $\mathbf{A}(\mathbf{G}_\mathbf{f} + \lambda_\mathbf{f}\mathbf{I}) = \mathbf{Y}$ and $\mathbf{B}(\mathbf{G}_\mathbf{F} + \lambda_\mathbf{F}\mathbf{I}) = \mathbf{X}$, and the variational problem for fixed $\mathbf{X}$ becomes a finite-dimensional quadratic minimisation problem with a unique solution. For illustration purposes we choose for both $\mathbf{f}$ anf $\mathbf{F}$ the motion coherence theory operator [15], whose Green's function is a Gaussian with scale parameter $\sigma > 0$. Thus $g(\mathbf{x} - \mathbf{x}_n) = \exp\left(-\|\mathbf{x} - \mathbf{x}_n\|^2 / 2\sigma_\mathbf{x}^2\right)$ and $G(\mathbf{y} - \mathbf{y}_n) = \exp\left(-\|\mathbf{y} - \mathbf{y}_n\|^2 / 2\sigma_\mathbf{y}^2\right)$. This yields positive definite Gram matrices $\mathbf{G}_\mathbf{f}$, $\mathbf{G}_\mathbf{F}$. The role of $\sigma_\mathbf{x}$ and $\sigma_\mathbf{y}$ is to control sensitivity to noise in the $\mathbf{X}$ and $\mathbf{Y}$ spaces, respectively, by setting the scale at which interactions betwen points are deemed significant.

Projection step For fixed $\mathbf{f}$ and $\mathbf{F}$, we have the following minimisation over $\mathbf{X}$:

$$\min_{\mathbf{X}} \textstyle\sum_{n=1}^N \|\mathbf{y}_n - \mathbf{f}(\mathbf{x}_n)\|^2 + \sum_{n=1}^N \|\mathbf{x}_n - \mathbf{F}(\mathbf{y}_n)\|^2 = \min_{\mathbf{X}} \|\mathbf{Y} - \mathbf{A}\mathbf{G}_\mathbf{f}\|^2 + \|\mathbf{X} - \mathbf{B}\mathbf{G}_\mathbf{F}\|^2 \quad (9)$$

where $\mathbf{G}_\mathbf{f}$ depends nonlinearly on $\mathbf{X}$. This is intuitively equivalent to inverting $\mathbf{f}$ (i.e., $\mathbf{x}_n = \mathbf{f}^{-1}(\mathbf{y}_n)$) while trying to respect $\mathbf{x}_n = \mathbf{F}(\mathbf{y}_n)$, and does lead in practice to $\mathbf{f}^{-1} \approx \mathbf{F}$ on the data manifold. The solution cannot be obtained in closed form because of the term $\|\mathbf{y}_n - \mathbf{f}(\mathbf{x}_n)\|^2$ (in fact the objective function has local minima) and requires nonlinear minimisation. While this step can be prone to local minima, we can use two things in our favour: (1) We can use a good initialisation $\mathbf{X}$ from the embedding produced by a spectral manifold learning algorithm such as LE, LLE or Isomap. Naturally, the success of these depends on obtaining a good neighbourhood graph, which may require carefully setting its parameters (e.g. k for a k-nearest-neighbour graph). (2) In projection steps after the initial one, the term $\|\mathbf{x}_n - \mathbf{F}(\mathbf{y}_n)\|$ (whose influence is comparable to that of the term $\|\mathbf{y}_n - \mathbf{f}(\mathbf{x}_n)\|$) provides a strong constraint, since the optimal $\mathbf{x}_n$ must be close to $\mathbf{F}(\mathbf{y}_n)$.

Simply fitting $\mathbf{f}$ to $(\mathbf{X}, \mathbf{Y})$ and $\mathbf{F}$ to $(\mathbf{Y}, \mathbf{X})$ (thus avoiding the projection-adaptation iteration) has been proposed (e.g. [12]) as a way to define out-of-sample mappings for spectral methods. However, our experiments show that *full joint optimisation of $E(\mathbf{X}, \mathbf{f}, \mathbf{F})$ over $\mathbf{X}$, $\mathbf{f}$ and $\mathbf{F}$ considerably improves the spectral embedding*, eliminating folds in the mappings and local clustering in $\mathbf{X}$—to which spectral methods are prone because of the boundary effects induced by the neighbourhood graph, and because of local variations in the density of points $\mathbf{Y}$.

Summary Putting all together, our now finite-dimensional objective function is, in matrix form:

$$E(\mathbf{X}, \mathbf{A}, \mathbf{B}; \lambda_\mathbf{f}, \lambda_\mathbf{F}, \sigma_\mathbf{x}, \sigma_\mathbf{y}) = \|\mathbf{Y} - \mathbf{A}\mathbf{G}_\mathbf{f}\|^2 + \lambda_\mathbf{f} \operatorname{tr}\left(\mathbf{A}\mathbf{G}_\mathbf{f}\mathbf{A}^T\right) + \|\mathbf{X} - \mathbf{B}\mathbf{G}_\mathbf{F}\|^2 + \lambda_\mathbf{F} \operatorname{tr}\left(\mathbf{B}\mathbf{G}_\mathbf{F}\mathbf{B}^T\right) \quad (10)$$

with matrices $\mathbf{Y}_{D\times N}$, $\mathbf{X}_{L\times N}$, $\mathbf{A}_{D\times N}$, $\mathbf{B}_{L\times N}$, and $N \times N$ Gram matrices $\mathbf{G}_\mathbf{f}$ (dependent on $\mathbf{X}$) and $\mathbf{G}_\mathbf{F}$ (constant) given in eq. (8). The user parameters $(\lambda_\mathbf{f}, \lambda_\mathbf{F}, \sigma_\mathbf{x}, \sigma_\mathbf{y})$ have a natural, intuitive role, controlling (1) the level of smoothness of the mappings ($\lambda_\mathbf{f}$, $\lambda_\mathbf{F}$) and (2) the sensitivity to noise ($\sigma_\mathbf{x}$, $\sigma_\mathbf{y}$), much as the bandwidth of a kernel density estimator (a roughly good σ value is the average interpoint distance). Fig. 1 gives the DRUR algorithm.

Note that we do not need an additional parameter $\mu > 0$ to weigh the $\mathbf{F}$–error wrt the $\mathbf{f}$–error, because this introduces a redundant symmetry in E:

$$E(\mathbf{X}, \mathbf{A}, \mathbf{B}; \mu, \lambda_\mathbf{f}, \lambda_\mathbf{F}, \sigma_\mathbf{x}, \sigma_\mathbf{y}) = \|\mathbf{Y} - \mathbf{A}\mathbf{G}_\mathbf{f}\|^2 + \lambda_\mathbf{f} \operatorname{tr}\left(\mathbf{A}\mathbf{G}_\mathbf{f}\mathbf{A}^T\right) + \mu\|\mathbf{X} - \mathbf{B}\mathbf{G}_\mathbf{F}\|^2 + \lambda_\mathbf{F} \operatorname{tr}\left(\mathbf{B}\mathbf{G}_\mathbf{F}\mathbf{B}^T\right) = E(\mu^{-\frac{1}{2}}\mathbf{X}, \mathbf{A}, \mu^{-\frac{1}{2}}\mathbf{B}; 1, \lambda_\mathbf{f}, \mu\lambda_\mathbf{F}, \mu^{-\frac{1}{2}}\sigma_\mathbf{x}, \sigma_\mathbf{y}). \quad (11)$$

DRUR becomes PCA if we restrict the mappings to be linear, and then the algorithm yields an iterative algorithm to compute PCA (by orthogonal projection); see appendix B.

input: $\mathbf{Y}_{D\times N} = (\mathbf{y}_1, \ldots, \mathbf{y}_N)$; $\lambda_\mathbf{f}, \lambda_\mathbf{F}, \sigma_\mathbf{x}, \sigma_\mathbf{y} > 0$
Obtain $\mathbf{X}_{L\times N} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$ from spectral method
$\mathbf{G}_\mathbf{f} = (g(\mathbf{x}_n - \mathbf{x}_m; \sigma_\mathbf{x}))$, $\mathbf{G}_\mathbf{F} = (G(\mathbf{y}_n - \mathbf{y}_m; \sigma_\mathbf{y}))$
$\mathbf{A} = \mathbf{Y}(\mathbf{G}_\mathbf{f} + \lambda_\mathbf{f}\mathbf{I})^{-1}$, $\mathbf{B} = \mathbf{X}(\mathbf{G}_\mathbf{F} + \lambda_\mathbf{F}\mathbf{I})^{-1}$
repeat
  Project: $\mathbf{X}$ = approximate minimiser of (9)
  Adapt: $\mathbf{G}_\mathbf{f} = (g(\mathbf{x}_n - \mathbf{x}_m; \sigma_\mathbf{x}))$
    $\mathbf{A} = \mathbf{Y}(\mathbf{G}_\mathbf{f} + \lambda_\mathbf{f}\mathbf{I})^{-1}$, $\mathbf{B} = \mathbf{X}(\mathbf{G}_\mathbf{F} + \lambda_\mathbf{F}\mathbf{I})^{-1}$
until convergence
return $\mathbf{A}$, $\mathbf{B}$, $\mathbf{X}$

Figure 1. Dimensionality Reduction by Unsupervised Regression (DRUR) algorithm.

2. Optimisation and computational cost

The overall training cost is $\mathcal{O}(N^3)$: a $\mathcal{O}(N^3)$ setup cost and a $\mathcal{O}(N^2)$ cost per step (adaptation and projection).

Adaptation step for $\mathbf{A}$, $\mathbf{B}$ This requires solving two non-sparse linear systems of $N \times N$, at a worst-case cost of $\mathcal{O}(N^3)$. The system for $\mathbf{B}$, $\mathbf{B}(\mathbf{G_F} + \lambda_\mathbf{F}\mathbf{I}) = \mathbf{X}$, has a constant coefficient matrix because $\mathbf{G_F} = (G(\mathbf{y}_n - \mathbf{y}_m))_{nm}$ depends only on $\mathbf{Y}$ and $\sigma_\mathbf{y}$. Thus, since $\mathbf{G_F} + \lambda_\mathbf{F}\mathbf{I}$ is positive definite, we can precompute its Cholesky factorisation $\mathbf{L}\mathbf{L}^T$ (where $\mathbf{L}$ is lower triangular) at a cost of $\frac{1}{6}N^3$ multiplications [4, p. 144], and then update $\mathbf{B}$ at each step by solving two triangular systems $\mathbf{Z}\mathbf{L}^T = \mathbf{X}$ and $\mathbf{B}\mathbf{L} = \mathbf{Z}$ at a comparatively negligible cost $\mathcal{O}(N^2)$. This does not cost extra memory either as we can replace $\mathbf{G_f}$ with $\mathbf{L}$.

The system for $\mathbf{A}$, $\mathbf{A}(\mathbf{G_f} + \lambda_\mathbf{f}\mathbf{I}) = \mathbf{Y}$, does have to be solved anew at each step since the coefficient matrix $\mathbf{G_f} + \lambda_\mathbf{f}\mathbf{I}$ does change. This means a worst-case cost of $\mathcal{O}(N^3)$ per step. However, since $\mathbf{G_f} + \lambda_\mathbf{f}\mathbf{I}$ is positive definite and the solution $\mathbf{A}$ changes slowly from step to step, we can obtain an approximate solution by running several iterations of linear conjugate gradients (CG) [9]. Inexact steps make sense when the exact step (which does not yield the final optimum) are costly, so we can make slightly less progress towards the optimum at a far lower computational cost. Besides, linear CG can be initialised to the solution $\mathbf{A}$ of the previous time step (which, say, Gauss elimination does not benefit from). Linear CG is based on matrix-vector products so its cost is N^2D times the number of CG iterations (far smaller than N), thus the overall cost again is negligible wrt the $\mathcal{O}(N^3)$ cost of solving the initial linear system (which must be solved exactly to ensure we do not deviate from the good initialisation provided by the spectral method). Thus, the approximate step costs $\mathcal{O}(N^2)$.

Projection step for $\mathbf{X}$ This requires nonlinear optimisation. In principle, since the objective function has the form of a least-squares problem, the practically best methods would be Gauss-Newton or Levenberg-Marquardt [9], which converge linearly but with a small rate (i.e., fast), yet use only the gradient and not the Hessian of E. These methods require solving a linear system of $NL \times NL$ where NL is the number of parameters of $\mathbf{X}$, and again would benefit from an inexact solution via linear CG. In our current implementation we use nonlinear CG [9], which is slower than Gauss-Newton but faster than gradient descent; and we initialise it from the previous step's $\mathbf{X}$, although in the future we plan to study better initialisations using also the $\mathbf{X}$ predicted by $\mathbf{F}$, namely $\mathbf{X} = \mathbf{F}(\mathbf{Y}) = \mathbf{B}\mathbf{G_F}$, which minimises the quadratic right-hand term of (9). Crucially, however, note that (attractive as this might seem because of its ease) we cannot approximate the minimisation over $\mathbf{X}$ as $\mathbf{X} \approx \mathbf{F}(\mathbf{Y})$, because this decouples the updates for $(\mathbf{X}, \mathbf{F})$ and $(\mathbf{f})$ and the points $\mathbf{X}$ shrink to $\mathbf{0}$. The cost of computing the gradient wrt $\mathbf{X}$ (and so the nonlinear CG) is $\mathcal{O}(N^2)$.

3. Experiments

We provide results in several datasets: a hard toy problem (spiral, fig. 2), which allows visualisation of the mapping $\mathbf{f}$, and real-world high-dimensional problems (motion-capture, fig. 3; face images, fig. 4). For computational efficiency, we preprocess the dataset with PCA, retaining $> 99.99\%$ of the variance (mocap: top 20 PCs; face: top 50). We compare DRUR with 4 other related dimensionality reduction methods: LELVM [3], GTM [2], GPLVM [6] and RPM [13], reviewed in section 4. We implement RPM by making $\lambda_\mathbf{F} = 0$ in DRUR to remove $\mathbf{F}$. Fig. 2**A**–**D** shows how DRUR fails to recover the spiral if initialised randomly or from the principal component of the data; other methods (e.g. GTM, not shown) also failed. It is very hard to warp a random or linear function into a spiral, which folds around itself; but its shape may be captured by a spectral method using a neighbourhood graph, such as Laplacian eigenmaps (LE), which we use throughout (using a k-nearest-neighbour graph and Gaussian-affinity edges). Fig. 2**E**,**F**,**H** shows how DRUR then recovers the spiral with as few as 50 samples; this means DRUR may be used in applications (such as motion-capture) where training data is sparse, yet the manifold is very nonlinear. Most progress during training occurs in the first 10–30 iterations. We also explored systematically (for the same, fixed spiral dataset and LE initialisation; results not shown) the range of parameters $(\lambda_\mathbf{f}, \sigma_\mathbf{x})$ and $(\lambda_\mathbf{F}, \sigma_\mathbf{y})$ for which DRUR yields good respective mappings $\mathbf{f}$, $\mathbf{F}$. DRUR recovers the spiral unless, as one would expect with any RBF mapping, $\sigma_\mathbf{x}$ is very small wrt the nearest-neighbour spacing of the initial $\{\mathbf{x}_n\}$ (when $\mathbf{f}$ can abandon the LE initialisation and reach a bad local minimum), or when $\lambda_\mathbf{f}$ is very large (when $\mathbf{f}$ becomes linear); small $\lambda_\mathbf{f}$ yield wigglier mappings. Removing $\mathbf{f}$ altogether causes $\mathbf{X}$ to shrink to $\mathbf{0}$; note the method of [10, 11] has $\mathbf{F}$ but no $\mathbf{f}$, however a prior on $\mathbf{X}$ prevents this shrinkage.

Fig. 2**F1** shows how DRUR improves over the LE initialisation (see also fig. 3**A**,**B**): the distribution of $\mathbf{X}$ given by LE (blue) is quite nonuniform, with local bunching caused by sampling artifacts (which often cause local folds in $\mathbf{f}$, these may be seen by zooming in fig. 2**E**) and by boundary effects in the neighbourhood graph (apparent at the spiral ends: note the sigmoid shape of $\mathbf{x}_n$ vs n in fig. 2**F1**). The final $\mathbf{X}$ from DRUR are much more uniformly distributed (note how $\mathbf{x}_n$ vs n is linear; see also fig. 3**B**), which yields a smoother $\mathbf{f}$. Fig. 2**G**,**G1** shows RPM (DRUR without $\mathbf{F}$) on the same problem; note how the latent space representation breaks into separate chunks (jumps in $\mathbf{x}_n$ vs n) which are associated with large-scale folds in $\mathbf{f}$ (fig. 2**G**). This is more noticeable in the mocap data (fig. 3**D**), where points $\mathbf{x}_n, \mathbf{x}_m$ corresponding to the same (approximate) pose $\mathbf{y}_n \approx \mathbf{y}_m$ end up separated from each other, yielding a latent space representation where consecutive cycles of running are offset from each other, and with discontinuous jumps (recall

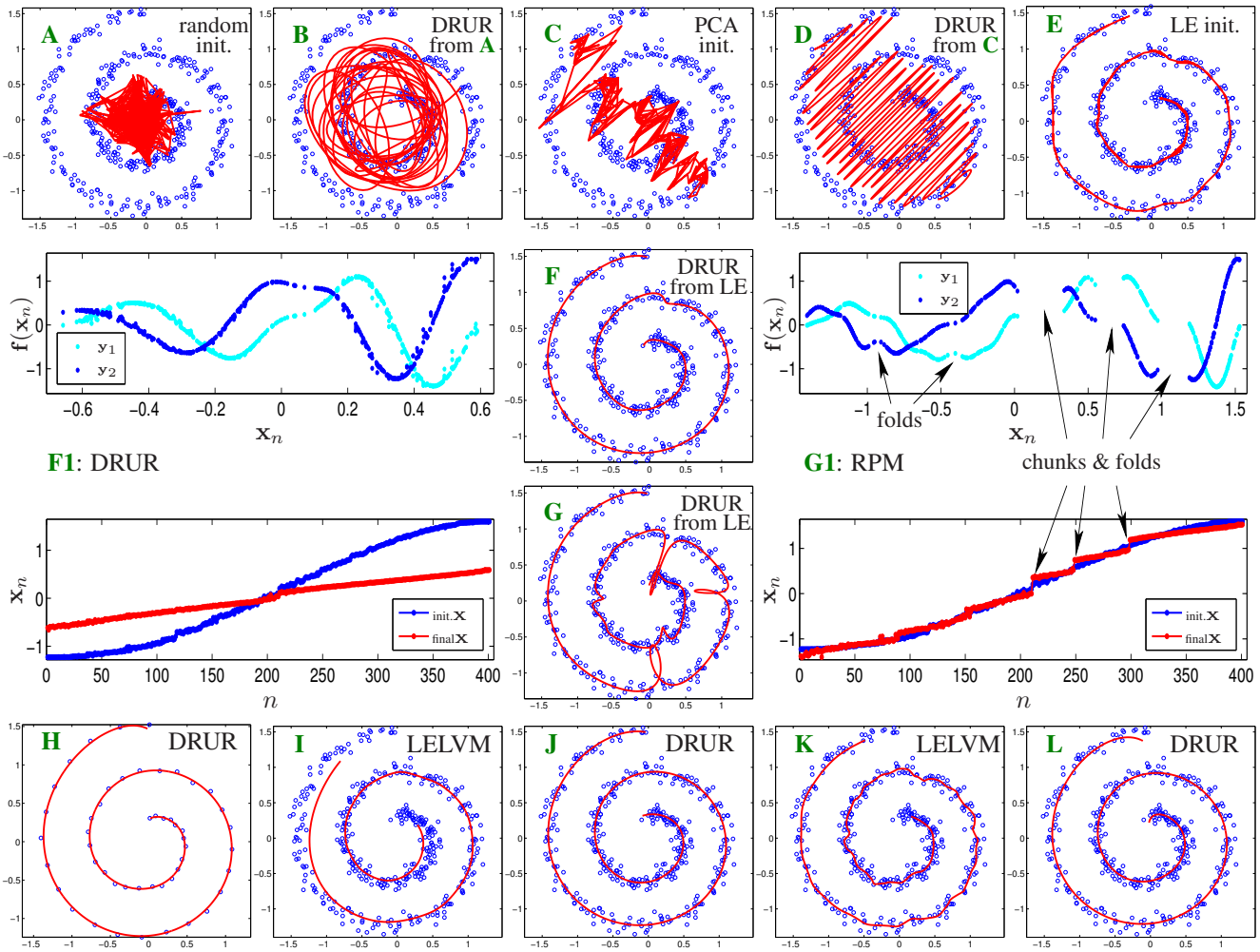

Figure 2. Noisy spiral dataset ($N = 400$ samples) and $\mathbf{f}(\mathbf{x})$ for $\mathbf{x} \in$ range of $\mathbf{X}$. **B**,**D**,**F**: DRUR results for 3 initialisations: random (**A**), PCA (**C**) and LE (**E**) ($\lambda_\mathbf{f} = 0.1$, $\lambda_\mathbf{F} = 0.1$, $\sigma_\mathbf{x} = 0.08$, $\sigma_\mathbf{y} = 0.02$). **H**: DRUR results for LE initialisation for a small sample size of $N = 50$ ($\lambda_\mathbf{f} = 0.05$, $\lambda_\mathbf{F} = 0.05$, $\sigma_\mathbf{x} = 0.05$, $\sigma_\mathbf{y} = 0.1$). **F1** (for DRUR in panel **F**): upper plot, $\mathbf{f}(\mathbf{x}_n) = (\mathbf{y}_1, \mathbf{y}_2)$; lower plot, initial (LE, blue) vs final (red) $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$. **G**,**G1**: like **F**,**F1** but for RPM (same parameters as DRUR but $\lambda_\mathbf{F} = 0$, initialised from LE). Note the folds in $\mathbf{f}$ and chunks in $\mathbf{X}$. **I**: LELVM result for $\sigma_\mathbf{x} = 0.32$, reconstruction error $= 0.336$. **J**: DRUR with $\sigma_\mathbf{x} = 0.32$, $\sigma_\mathbf{y} = 0.08$, $\lambda_\mathbf{f} = 0.1$, $\lambda_\mathbf{F} = 0.1$, reconstruction error $= 0.010$. **K**: LELVM with $\sigma_\mathbf{x} = 0.08$, reconstruction error $= 0.039$. **L**: DRUR with $\sigma_\mathbf{x} = 0.08$, $\sigma_\mathbf{y} = 0.02$, $\lambda_\mathbf{f} = 1$, $\lambda_\mathbf{F} = 0.1$, reconstruction error $= 0.007$. Note how DRUR not only provides a mapping that recovers the spiral, but also improves over the initial LE embedding (removing folds and stretching the spiral ends).

the continuity argument from section 1). In contrast, the DRUR latent space (fig. 3**B**) does collect the loops together in a smooth way, with no jumps. Fig. 3**B** shows the reason: $\mathbf{F}(\mathbf{y}_n)$ (red circles) $\approx \mathbf{x}_n$ (blue), i.e., we achieve $\mathbf{f} \approx \mathbf{F}^{-1}$ over the manifold. The sequence of stickmen shows how the learned $\mathbf{f}$ can be used to synthesise realistic, smooth motion. In sum, minimising the reconstruction error jointly over $(\mathbf{X}, \mathbf{f}, \mathbf{F})$ improves the initial $\mathbf{X}$ from LE and ensures a consistent, smooth representation in latent space. GPLVM [6] obtained similar results to RPM in this and other sequences (consistent with the fact that both estimate $\mathbf{f}$ and $\mathbf{X}$, but not $\mathbf{F}$), see fig. 3**C** and fig. 2 in [7]: the latent space contains offset loops, nonuniformities and discontinuous jumps that worsen the spectral embedding $\mathbf{X}$.

Fig. 4 shows DRUR on a high-dimensional face image dataset, using a 2D latent space. As is sometimes the case with spectral methods, the LE embedding (fig. 4**A**) is poor, possibly because of an imperfect neighbourhood graph. It contains degenerate 1D tendrils, folds, and an uneven distribution with large gaps and strong compression at the boundaries. The $\mathbf{X}$ after DRUR training (fig. 4**B**) are tremendously improved, practically filling in the cleft between the tendrils and eliminating the boundary effects. We obtained a very similar improvement with the digit–2

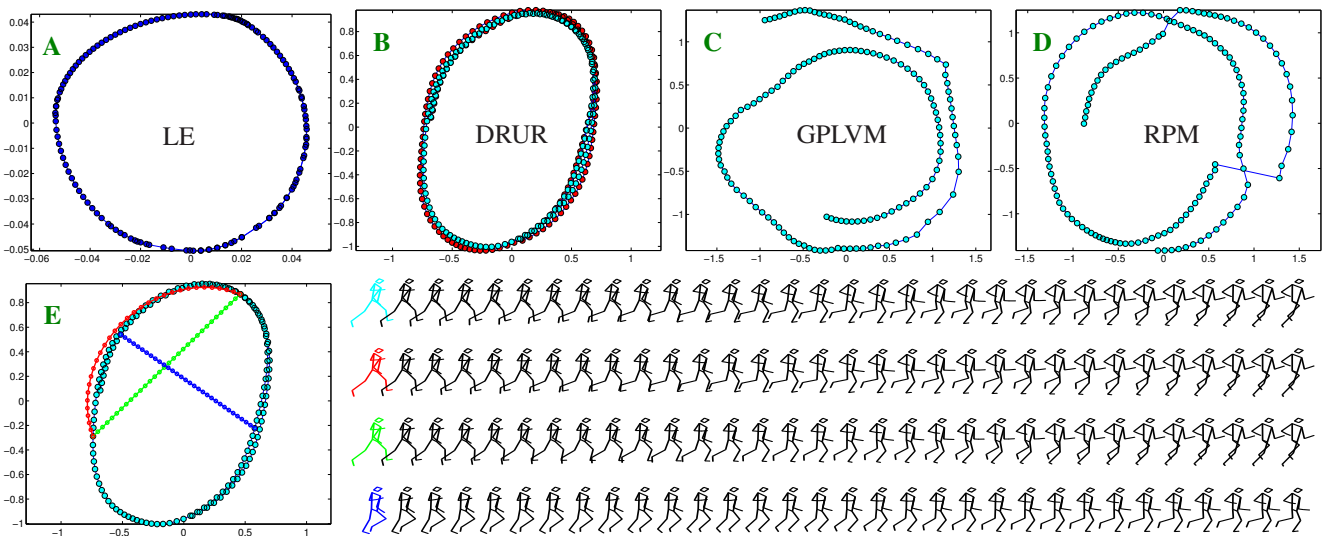

Figure 3. CMU motion capture running sequence `09_01` (we obtained similar results with other sequences): 148 samples (containing almost 2 running cycles), with data from the first 50 sensors, so the pose $\mathbf{y}$ is 150–dimensional. All methods were tested using a 2D latent space. **A**: embedding $\mathbf{X}$ from LE. **B**: $\mathbf{X}$ (red circles) and $\mathbf{F}(\mathbf{Y})$ (cyan) obtained by DRUR from the LE initialisation ($\lambda_{\mathbf{f}} = 0.8$, $\lambda_{\mathbf{F}} = 0.5$, $\sigma_{\mathbf{x}} = 0.8$, $\sigma_{\mathbf{y}} = 0.8$, trained for 100 iterations). Note wrt **A** the more uniform distribution of $\mathbf{X}$. **E**: 3 colour-coded trajectories in latent space for DRUR and corresponding stickmen recovered with $\mathbf{f}(\mathbf{x})$ (cyan: subset of the pose training data, for reference); none of the stickmen were in the training set. **C**: $\mathbf{X}$ from GPLVM (run for 1 000 iterations). **D**: $\mathbf{X}$ from RPM (same parameters as DRUR but $\lambda_{\mathbf{F}} = 0$). Both GPLVM and RPM dislocate the latent representation provided by the spectral (LE) initialisation.

dataset [14] (not shown). The latent space (fig. 4**C**) roughly represents left-right pose as Y axis and lighting direction as X axis. DRUR smoothly reconstructs a continuous out-of-sample path in latent space, yielding an animated movie (e.g. varying the left-right pose). The reconstructed faces look slightly blurred, but it seems very hard to capture such detailed structure with only 2 degrees of freedom and a small training set.

Finally, we compare DRUR with latent variable models (i.e., defining a density $p(\mathbf{x}, \mathbf{y})$ in both latent and observed space). GTM (not shown) failed with the mocap data, while LELVM succeeded with it and the spiral. The reason is that, essentially, LELVM defines a kernel density estimate with centres $(\mathbf{x}_n, \mathbf{y}_n)$ with $\mathbf{X}$ given by LE, rather than gridding the latent space, and the mappings $\mathbf{f}$, $\mathbf{F}$ are the corresponding Nadaraya-Watson estimators. However, DRUR does have 2 advantages over LELVM: the improvement over the initial $\mathbf{X}$ provided by LE, and the fact that LELVM's mappings are convex sums (see section 4). The latter can be seen at the ends of the spiral in fig. 2**I**,**K**, which LELVM cannot reach, unlike DRUR (fig. 2**J**,**L**), so DRUR achieves a lower reconstruction error when using the same $\sigma_{\mathbf{x}}$ as LELVM.

4. Related work

DRUR is most closely related to mapping-based methods for dimensionality reduction, in particular autoencoders. These methods use different combinations of two elements of the triplet $(\mathbf{X}, \mathbf{f}, \mathbf{F})$ (latent points, reconstruction mapping, dimensionality reduction mapping), but only DRUR uses all three (eq. (2)). Comparing eqs. (1) and (2), we see that autoencoders eliminate $\mathbf{x}_n = \mathbf{F}(\mathbf{y}_n)$ and use a parametric function (typically a neural net) instead of a RBF expansion. Eliminating $\mathbf{X}$ reduces the number of parameters, but makes the estimation prone to bad local optima. Essentially, the problem with autoencoders is not of representation ability (parameter values exist for $\mathbf{f}$, $\mathbf{F}$ that approximate well the manifold), but of search (in an objective function plagued with bad local minima, it is very hard to reach the good ones from most initial points). Instead, *DRUR solves an easier search (because of the ability to use a good initialisation for* $\mathbf{X}$*) in an augmented space* $(\mathbf{X}, \mathbf{f}, \mathbf{F})$. DRUR generalises several recent methods that use explicit coordinates for the latent points $\mathbf{X}$ (possibly initialised by a spectral method), but estimate only one of the two mappings. The distance-preservation argument of section 1 suggests that jointly estimating both mappings is really necessary and this is confirmed by our experiments. Regularised principal manifolds (RPM) [13] estimates only $\mathbf{X}$ and $\mathbf{f}$, which has the form of a parametric RBF whose centres are fixed by the user (not specified in [13]). This has the problem that, as $\mathbf{X}$ change, the fixed RBF centres may not track well the distribution of $\mathbf{X}$ and thus result in a bad mapping; in this paper, we have chosen to use $\mathbf{X}$ as RBF centres. Like RPM, the Gaussian Process Latent Variable Model (GPLVM) [6]

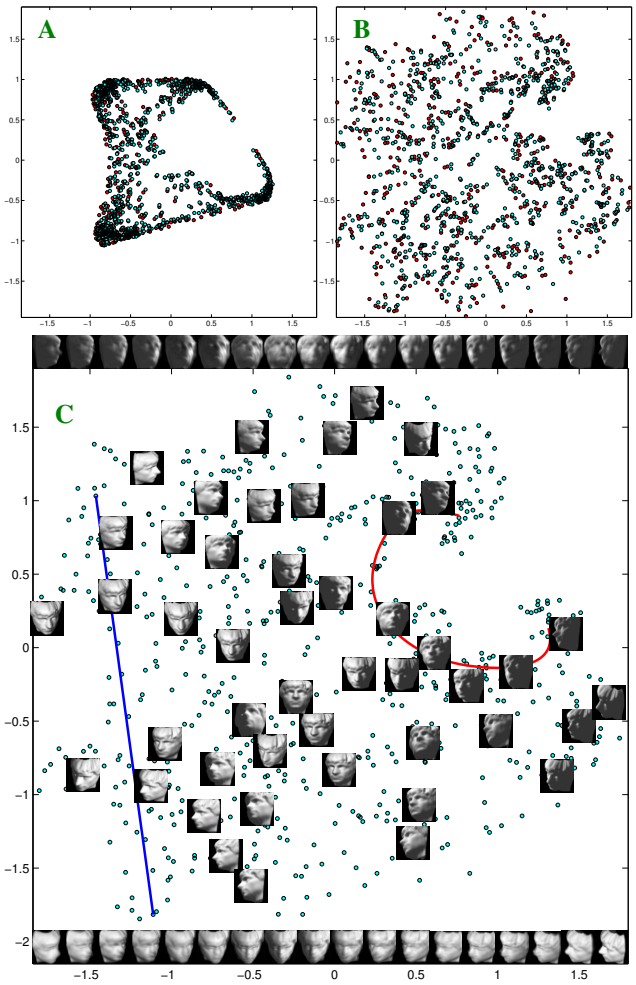

Figure 4. Faces database from [14]: 698 images of $64 \times 64 = 4096$ pixels. DRUR results using a 2D latent space, LE initialisation ($\lambda_{\mathbf{f}} = 0.2$, $\lambda_{\mathbf{F}} = 0.2$, $\sigma_{\mathbf{x}} = 0.3$, $\sigma_{\mathbf{y}} = 0.3$, trained for 200 iterations). **A**: embedding $\mathbf{X}$ from LE (red circles) and $\mathbf{F}(\mathbf{Y})$ (cyan) obtained by DRUR at initialisation. **B**: $\mathbf{X}$, $\mathbf{F}(\mathbf{Y})$ after DRUR training. **C**: subset of data images plotted on the latent space, 2 colour-coded trajectories and corresponding reconstruction with $\mathbf{f}(\mathbf{x})$ (2 rows of images, none of which were in the training set).

also estimates only $\mathbf{X}$ and $\mathbf{f}$, which has the form of a Gaussian process function, and use a quadratic regulariser on $\mathbf{X}$ as well. Training is costly (each gradient step is $\mathcal{O}(N^3)$), although approximate techniques for fast Gaussian process regression may be used. Neither RPM nor GPLVM learn $\mathbf{F}$, so if this is needed (e.g. for dimensionality reduction, to get the $\mathbf{x}$ that a new $\mathbf{y}$ maps to) the user has to solve a difficult problem to invert $\mathbf{y} = \mathbf{f}(\mathbf{x})$. With DRUR, we simply use $\mathbf{x} = \mathbf{F}(\mathbf{y})$. Lacking $\mathbf{F}$ can cause RPM and GPLVM to worsen the embedding $\mathbf{X}$ provided by the spectral method (fig. 3) while DRUR actually improves it. In [7], it was noted that GPLVM fails to preserve distances in $\mathbf{X}$; they proposed to eliminate $\mathbf{X}$, replacing it with a back-constraint function of $\mathbf{Y}$ (e.g. a neural net). However, this effectively turns GPLVM into an autoencoder. Meinicke et al. [8] estimate only $\mathbf{X}$ and $\mathbf{f}$, which has the form of a nonparametric kernel regression mapping, and also need constraints on $\mathbf{X}$. Motivated by a tracking application, Rahimi et al. [10, 11] estimate only $\mathbf{X}$ and $\mathbf{F}$ (a RBF expansion), but keep $\mathbf{X}$ from shrinking to $\mathbf{0}$ by adding a prior on $\mathbf{X}$ (which encourages temporal structure) as well as fixing by hand the latent coordinates for a few points. As we can see, the lack of either $\mathbf{f}$ or $\mathbf{F}$ means some of these methods need a (quadratic) regulariser on $\mathbf{X}$ to prevent degenerate $\mathbf{X}$; this is not necessary in DRUR, although it could be incorporated easily (without complicating the projection step), e.g. to encourage temporal or spatial structure in the latent space.

Finally, we consider nonlinear latent variable models, defining a density $p(\mathbf{x}, \mathbf{y})$ in both latent and observed space, and mappings $\mathbf{F}(\mathbf{y}) = \mathrm{E}\{\mathbf{x}|\mathbf{y}\}$ and $\mathbf{f}(\mathbf{x}) = \mathrm{E}\{\mathbf{y}|\mathbf{x}\}$ (note GPLVM is not in this class, as it is not a generative model; it defines neither a density in latent space nor $p(\mathbf{x}|\mathbf{y})$ nor $\mathbf{F}(\mathbf{y})$). DRUR is related to the recently proposed Laplacian Eigenmaps Latent Variable Model (LELVM) [3]. This is perhaps the only nonlinear LVM to overcome the problems mentioned in the introduction of integrating $p(\mathbf{x}, \mathbf{y})$ over a latent space of arbitrary dimension, and of local optima. LELVM is a nonparametric out-of-sample extension of Laplacian eigenmaps based on a semi-supervised learning argument. Essentially, it obtains $\mathbf{X}$ from LE and then builds $p(\mathbf{x}, \mathbf{y})$ as a kernel density estimate with centres $\{(\mathbf{x}_n, \mathbf{y}_n)\}$, from which both mappings $\mathbf{F}(\mathbf{y}) = \mathrm{E}\{\mathbf{x}|\mathbf{y}\}$ and $\mathbf{f}(\mathbf{x}) = \mathrm{E}\{\mathbf{y}|\mathbf{x}\}$ (which are nonparametric kernel regression mappings) are derived; e.g. $\mathbf{f}(\mathbf{x}) = \sum_{n=1}^{N} p(n|\mathbf{x})\mathbf{y}_n$ with $p(n|\mathbf{x}) \propto G((\mathbf{x} - \mathbf{x}_n)/\sigma_{\mathbf{x}})$. These mappings are convex sums ($\sum_{n=1}^{N} p(n|\mathbf{x}) = 1$), so they can only map $\mathbf{x}$ to the convex hull of $\{\mathbf{y}_n\}$, and $\mathbf{y}$ to the convex hull of $\{\mathbf{x}_n\}$, causing distortions on the dataset boundaries. Compared to DRUR, LELVM is fast to train, as it fixes $\mathbf{X}$ to the Laplacian eigenmaps embedding (i.e., it does not optimise over $\mathbf{X}$), and it provides also a density. But as shown in our experiments, *DRUR improves the* $\mathbf{X}$ *distribution* (by optimising over $\mathbf{X}$), and its mappings are not convex sums, avoiding boundary distortions and achieving a lower reconstruction error.

5. Conclusion

We have given a clean, symmetric formulation of the joint estimation of dimensionality reduction and reconstruction mappings to minimise a regularised reconstruction error through the use of auxiliary variables—the latent coordinates. The latter allow the use of a good initialisation (provided by a spectral manifold learning method) that prevents falling into one of the many bad local optima of the objective—as happens with autoencoders. Further, the experiments show the final solution not only gives out-of-

sample mappings for the spectral embedding, but also noticeably improves typical defects of the latter (folds and distortions around the data boundaries). DRUR recovers mappings for convoluted manifolds and can work with spaces (observed and latent) of any dimension. Learning both mappings is useful for several applications, such as dimensionality reduction, synthesis and tracking. The training cost is linear in the dimension and cubic in the number of data points, since the form of the mappings is a RBF expansion centred at the data points. However, once trained, DRUR is fast to use on new points (by simply applying a RBF). Reducing the training cost, perhaps using techniques for fast Gaussian process regression, is a topic of future research.

Acknowledgements This work was partially supported by NSF CAREER award IIS–0754089. CMU data: `http://mocap.cs.cmu.edu` (created with funding from NSF EIA–0196217).

A. Solution of the adaptation step

Functional (5) separates. Consider the variational problem

$$\min_{\mathbf{f}} E_{\mathbf{f}}(\mathbf{X},\mathbf{f}) = \sum_{n=1}^{N} \|\mathbf{y}_n - \mathbf{f}(\mathbf{x}_n)\|^2 + \lambda_{\mathbf{f}} \|\mathcal{D}_{\mathbf{f}}\mathbf{f}\|^2 \quad (12)$$

with $\mathbf{f}: \mathbb{R}^L \to \mathbb{R}^D$. The Euler-Lagrange equation for $\mathbf{f}$ is

$$\sum_{n=1}^{N} \delta(\mathbf{x}-\mathbf{x}_n)(\mathbf{f}(\mathbf{x})-\mathbf{y}_n) + \lambda_{\mathbf{f}} \mathcal{D}_{\mathbf{f}}^* \mathcal{D}_{\mathbf{f}} \mathbf{f}(\mathbf{x}) = \mathbf{0} \quad (13)$$

whose solution has the form $\mathbf{f}(\mathbf{x}) = \sum_{n=1}^{N} \mathbf{a}_n g(\mathbf{x}-\mathbf{x}_n)$, where $g(\mathbf{x},\mathbf{x}')$ is the Green's function of the self-adjoint operator $\mathcal{D}_{\mathbf{f}}^* \mathcal{D}_{\mathbf{f}}$. The optimal vector coefficients $\{\mathbf{a}_n\}_{n=1}^N$ must satisfy a self-consistent equation (by substituting $\mathbf{f}(\mathbf{x})$ back into (13)) which implies in matrix form that they are the solutions of the linear system $\mathbf{A}(\mathbf{G}_{\mathbf{f}} + \lambda_{\mathbf{f}}\mathbf{I}) = \mathbf{Y}$ with Gram matrix $\mathbf{G}_{\mathbf{f}} = (g(\mathbf{x}_n - \mathbf{x}_m))_{nm}$. For the motion coherence theory operator [15] ($\mathcal{D}^m$: scalar operator $\mathcal{D}^{2m}\mathbf{f} = \nabla^{2m}\mathbf{f}$ for even indices, vector operator $\mathcal{D}^{2m+1}\mathbf{f} = \nabla(\nabla^{2m}\mathbf{f})$ for odd indices, ∇: gradient, ∇^2: Laplacian):

$$\|\mathcal{D}_{\mathbf{f}}\mathbf{f}\|^2 = \int_{\mathbb{R}^L} \sum_{m=0}^{\infty} \frac{\sigma^{2m}}{m!2^m} \|\mathcal{D}^m \mathbf{f}(\mathbf{x})\|^2 \, d\mathbf{x} \quad (14)$$

the Green's function is Gaussian $g(\mathbf{x},\mathbf{x}') = \exp\left(-\frac{1}{2}\|\frac{\mathbf{x}-\mathbf{x}'}{\sigma}\|^2\right)$. The derivation for $E_{\mathbf{F}}$ is analogous.

B. Linear mappings: PCA

DRUR becomes PCA if we restrict the mappings $\mathbf{f}$, $\mathbf{F}$ to be linear. Let $\mathbf{f}(\mathbf{X}) = \mathbf{A}\mathbf{X}$ and $\mathbf{F}(\mathbf{Y}) = \mathbf{B}\mathbf{Y}$ with matrices $\mathbf{X}_{L\times N}$, $\mathbf{Y}_{D\times N}$ and full-rank matrices $\mathbf{A}_{D\times L}$ and $\mathbf{B}_{L\times D}$ ($\mathrm{rank}\,(\mathbf{A}) = \mathrm{rank}\,(\mathbf{B}) = L$). We can impose orthogonality constraints w.l.o.g. and formulate the following constrained minimisation problem:

$$\min_{\mathbf{X},\mathbf{A},\mathbf{B}} E(\mathbf{X},\mathbf{A},\mathbf{B}) = \|\mathbf{Y}-\mathbf{A}\mathbf{X}\|^2 + \|\mathbf{X}-\mathbf{B}\mathbf{Y}\|^2$$
$$\text{s.t. } \mathbf{A}^T\mathbf{A} = \mathbf{I},\ \mathbf{B}\mathbf{B}^T = \mathbf{I}. \quad (15)$$

Note that this objective function is not quadratic in $\mathbf{X}$, $\mathbf{A}$ and $\mathbf{B}$ jointly, since the terms $\mathbf{A}\mathbf{X}$ and $\mathbf{B}\mathbf{Y}$ are not linear. However, if doing coordinate minimisation in $(\mathbf{X})$ and $(\mathbf{A},\mathbf{B})$, those terms are linear and the objective is quadratic (thus having a unique solution). The gradient of E is:

$$\frac{\partial E}{\partial \mathbf{X}} = 2\big((\mathbf{I} + \mathbf{A}^T\mathbf{A})\mathbf{X} - (\mathbf{A}^T + \mathbf{B})\mathbf{Y}\big) \quad (16)$$

$$\frac{\partial E}{\partial \mathbf{A}} = 2(\mathbf{A}\mathbf{X}\mathbf{X}^T - \mathbf{Y}\mathbf{X}^T) \quad \frac{\partial E}{\partial \mathbf{B}} = 2(\mathbf{B}\mathbf{Y}\mathbf{Y}^T - \mathbf{X}\mathbf{Y}^T). \quad (17)$$

Equating it to $\mathbf{0}$ we obtain a fixed-point iteration:

Adaptation $\mathbf{A} = \mathbf{Y}\mathbf{X}^+$, $\mathbf{B} = \mathbf{X}\mathbf{Y}^+$ followed by reorthogonalisation of $\mathbf{A}$ and $\mathbf{B}$

Projection $\mathbf{X} = \frac{1}{2}(\mathbf{A}^T + \mathbf{B})\mathbf{Y}$

with pseudoinverses $\mathbf{X}^+ = \mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1}$, $\mathbf{Y}^+ = \mathbf{Y}^T(\mathbf{Y}\mathbf{Y}^T)^{-1}$. The algorithm converges to the global minimum $\mathbf{A} = \mathbf{B}^T = \mathbf{U}_L$ and $\mathbf{X} = \mathbf{U}_L^T\mathbf{Y}$ where the $D \times L$ matrix $\mathbf{U}_L$ contains the L principal components (i.e., $\mathbf{Y} = \mathbf{U}\mathbf{S}\mathbf{V}^T$ is the SVD of $\mathbf{Y}$). In non-generic cases the algorithm may get stuck in the saddle points of E, corresponding to $\mathbf{U}_L$ being made up of other combinations of eigenvectors, but will leave them under a random perturbation. The solution is unique up to multiple singular values of $\mathbf{Y}$.

References

[1] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, June 2003.

[2] C. M. Bishop, M. Svensén, and C. K. I. Williams. GTM: The generative topographic mapping. *Neural Computation*, 10(1):215–234, Jan. 1998.

[3] M. Á. Carreira-Perpiñán and Z. Lu. The Laplacian Eigenmaps Latent Variable Model. In *AISTATS*, 2007.

[4] G. H. Golub and C. F. van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, third edition, 1996.

[5] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 2006.

[6] N. Lawrence. Probabilistic non-linear principal component analysis with Gaussian process latent variable models. *JMLR*, 6, 2005.

[7] N. Lawrence and J. Quiñonero. Local distance preservation in the GP-LVM through back constraints. In *ICML*, 2006.

[8] P. Meinicke, S. Klanke, R. Memisevic, and H. Ritter. Principal surfaces from unsupervised kernel regression. *IEEE Trans. PAMI*, 27(9):1379–1391, Sept. 2005.

[9] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer-Verlag, New York, 1999.

[10] A. Rahimi, T. Darrell, and B. Recht. Learning appearance manifolds from video. In *CVPR*, 2005.

[11] A. Rahimi and B. Recht. Estimating observation functions in dynamical systems using unsupervised regression. In *NIPS*, volume 19, pages 1113–1120, 2007.

[12] L. Saul and S. Roweis. Think globally, fit locally: Unsupervised learning of low dimensional manifolds. *JMLR*, 2003.

[13] A. J. Smola, S. Mika, B. Schölkopf, and R. C. Williamson. Regularized principal manifolds. *JMLR*, 1, 2001.

[14] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, Dec. 22 2000.

[15] A. L. Yuille and N. M. Grzywacz. A mathematical analysis of the motion coherence theory. *IJCV*, 3, 1989.