Dimensionality Reduction by Unsupervised Regression

Miguel Á. Carreira-Perpiñán, EECS, UC Merced

http://faculty.ucmerced.edu/mcarreira-perpinan

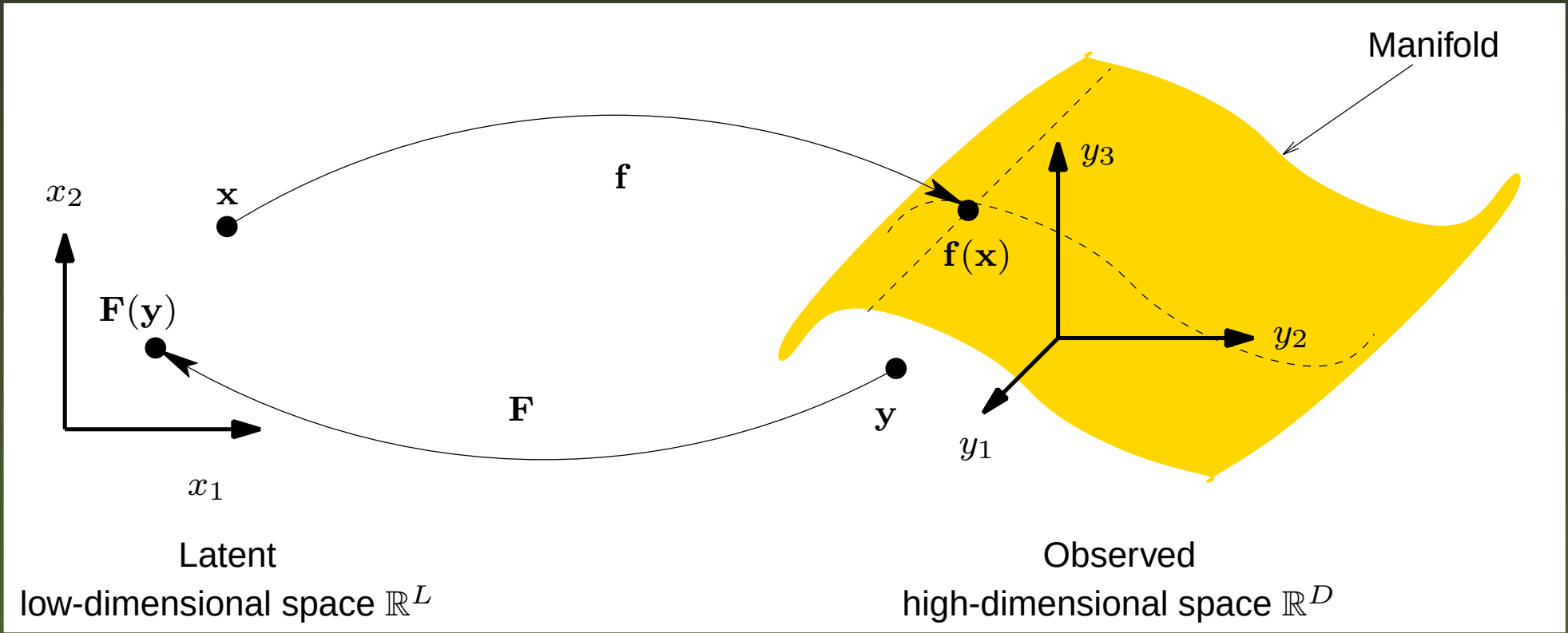
Zhengdong Lu, CSEE, OGI

http://www.csee.ogi.edu/~zhengdon

Dimensionality reduction

Given a dataset $\mathbf{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\} \subset \mathbb{R}^D$, find (for $L \ll D$):

- dimensionality reduction mapping $\mathbf{x} = \mathbf{F}(\mathbf{y}),\ \mathbf{y} \in \mathbb{R}^D$
- reconstruction mapping $\mathbf{y} = \mathbf{f}(\mathbf{x}),\ \mathbf{x} \in \mathbb{R}^L$



Dimensionality reduction methods

- Autoencoders: fit parametric mappings $\mathbf{f}$, $\mathbf{F}$ (neural nets) so

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

  Other methods provide only one mapping, either $\mathbf{f}$ or $\mathbf{F}$ (RPM, GPLVM, etc.).
  But: local optima, no density $p(\mathbf{x}, \mathbf{y})$.

- Latent variable models (GTM, LELVM): generative, estimate mappings and density $p(\mathbf{x}, \mathbf{y})$ by maximum likelihood on $p(\mathbf{y}) = \int p(\mathbf{y}|\mathbf{x})p(\mathbf{x})\, d\mathbf{x}$ (requires marginalisation).
  But: local optima, scale badly with dimension L (except LELVM).

- Spectral methods (Isomap, LLE, LE): based on neighbourhood graph, global optimum given by eigenvalue problem, often can unfold convoluted manifolds.
  But: no mappings, only latent coordinates $\mathbf{X}$ for training points $\mathbf{Y}$ (out-of-sample problem).

Dimensionality Reduction by Unsupervised Regression

DRUR formulation: given a dataset $\mathbf{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\} \subset \mathbb{R}^D$:

- Introduce latent coord. $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \subset \mathbb{R}^L$ as free params.
- Variational problem on $\mathbf{f}, \mathbf{F}, \mathbf{X}$:

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

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

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

$E_{\mathbf{f}}(\mathbf{X}, \mathbf{f})$ and $E_{\mathbf{F}}(\mathbf{X}, \mathbf{F})$ are regression problems coupled by a common, unobserved $\mathbf{X}$.

- Alternating minimisation over $(\mathbf{X})$ and $(\mathbf{f}, \mathbf{F})$:
  - Adaptation: min. E over $(\mathbf{f}, \mathbf{F})$ for fixed $\mathbf{X}$ (two regressions)
  - Projection: min. E over $(\mathbf{X})$ for fixed $\mathbf{f}$, $\mathbf{F}$
- Initialise $\mathbf{X}$ to a spectral embedding.

DRUR training: adaptation step

- Adaptation step: decouples into two separate regressions:

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

Unique solution: RBF mappings centred at the points $\mathbf{X}$, $\mathbf{Y}$:

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

with Gram matrices $\mathbf{G_f} = (g(\mathbf{x}_n - \mathbf{x}_m))_{nm}$, $\mathbf{G_F} = (G(\mathbf{y}_n - \mathbf{y}_m))_{nm}$ of $N \times N$ and coefficients $\{\mathbf{a}_n, \mathbf{b}_n\}$ given by linear systems:

$$\min_{\mathbf{A}} \|\mathbf{Y} - \mathbf{A}\mathbf{G_f}\|^2 + \lambda_{\mathbf{f}} \operatorname{tr}\left(\mathbf{A}\mathbf{G_f}\mathbf{A}^T\right) \Longrightarrow \mathbf{A}(\mathbf{G_f} + \lambda_{\mathbf{f}}\mathbf{I}) = \mathbf{Y}$$

$$\min_{\mathbf{B}} \|\mathbf{X} - \mathbf{B}\mathbf{G_F}\|^2 + \lambda_{\mathbf{F}} \operatorname{tr}\left(\mathbf{B}\mathbf{G_F}\mathbf{B}^T\right) \Longrightarrow \mathbf{B}(\mathbf{G_F} + \lambda_{\mathbf{F}}\mathbf{I}) = \mathbf{X}$$

We use Gaussian RBFs: $\begin{cases} g(\mathbf{x}-\mathbf{x}_n) = \exp\left(-\|\mathbf{x}-\mathbf{x}_n\|^2/2\sigma_{\mathbf{x}}^2\right) \\ G(\mathbf{y}-\mathbf{y}_n) = \exp\left(-\|\mathbf{y}-\mathbf{y}_n\|^2/2\sigma_{\mathbf{y}}^2\right). \end{cases}$

DRUR training: projection step

- Projection step: for fixed $\mathbf{f}$, $\mathbf{F}$, a nonlinear minimisation over $\mathbf{X}$:

$$\min_{\mathbf{X}} \sum_{n=1}^{N} \|\mathbf{y}_n - \mathbf{f}(\mathbf{x}_n)\|^2 + \sum_{n=1}^{N} \|\mathbf{x}_n - \mathbf{F}(\mathbf{y}_n)\|^2 =$$

quadratic on $\mathbf{X}$

$$\min_{\mathbf{X}} \|\mathbf{Y} - \mathbf{A}\mathbf{G}_{\mathbf{f}}\|^2 + \|\mathbf{X} - \mathbf{B}\mathbf{G}_{\mathbf{F}}\|^2$$

nonlinear on $\mathbf{X}$

  This tries to make $\mathbf{f} = \mathbf{F}^{-1}$ on the data manifold, so $\mathbf{y}_n = \mathbf{f}(\mathbf{x}_n)$ and $\mathbf{x}_n = \mathbf{F}(\mathbf{y}_n)$.

- Computational cost:
  - linear on the dimensions D and L
  - cubic on the number of training points N
    $\mathcal{O}(N^3)$ setup cost, $\mathcal{O}(N^2)$ per iteration

- Dimensionality reduction and reconstruction on unseen data at runtime are fast: $\mathbf{f}$, $\mathbf{F}$ are RBF mappings

DRUR and PCA

DRUR becomes PCA if constraining the mappings to be linear:

$$\min_{\mathbf{X},\mathbf{A},\mathbf{B}} E(\mathbf{X},\mathbf{A},\mathbf{B}) = \|\mathbf{Y}-\mathbf{A}\mathbf{X}\|^2 + \|\mathbf{X}-\mathbf{B}\mathbf{Y}\|^2$$

$$\text{s.t. } \mathbf{A}^T\mathbf{A} = \mathbf{I},\ \mathbf{B}\mathbf{B}^T = \mathbf{I}.$$

- Adaptation: $\mathbf{A} = \mathbf{Y}\mathbf{X}^+,\ \mathbf{B} = \mathbf{X}\mathbf{Y}^+$, reorthogonalise $\mathbf{A}$, $\mathbf{B}$
- Projection: $\mathbf{X} = \frac{1}{2}(\mathbf{A}^T + \mathbf{B})\mathbf{Y}$

which converges to PCA:

- $\mathbf{A} = \mathbf{B}^T = \mathbf{U}_L$ (leading L eigenvectors of $\mathrm{cov}\{\mathbf{Y}\}$)
- $\mathbf{X} = \mathbf{U}_L^T\mathbf{Y}$

DRUR algorithm

input: $\mathbf{Y}_{D\times N} = (\mathbf{y}_1, \ldots, \mathbf{y}_N)$; $\lambda_\mathbf{f}, \lambda_\mathbf{F}, \sigma_\mathbf{x}, \sigma_\mathbf{y} > 0$

initialise: $\mathbf{X}_{L\times N} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$ from spectral method
$\mathbf{G_f} = (g(\mathbf{x}_n - \mathbf{x}_m; \sigma_\mathbf{x}))$, $\mathbf{G_F} = (G(\mathbf{y}_n - \mathbf{y}_m; \sigma_\mathbf{y}))$
$\mathbf{A} = \mathbf{Y}(\mathbf{G_f} + \lambda_\mathbf{f}\mathbf{I})^{-1}$, $\mathbf{B} = \mathbf{X}(\mathbf{G_F} + \lambda_\mathbf{F}\mathbf{I})^{-1}$

repeat

Project: $\mathbf{X} =$ approximate minimiser of $E(\mathbf{X}, \mathbf{f}, \mathbf{F})$

Adapt: $\mathbf{G_f} = (g(\mathbf{x}_n - \mathbf{x}_m; \sigma_\mathbf{x}))$
$\mathbf{A} = \mathbf{Y}(\mathbf{G_f} + \lambda_\mathbf{f}\mathbf{I})^{-1}$, $\mathbf{B} = \mathbf{X}(\mathbf{G_F} + \lambda_\mathbf{F}\mathbf{I})^{-1}$

until convergence

return $\mathbf{A}$, $\mathbf{B}$, $\mathbf{X}$

User parameters:

- Regularisation of RBF mappings: $\lambda_\mathbf{f}$, $\lambda_\mathbf{F}$ (smoothness)
- Width of RBF mappings: $\sigma_\mathbf{x}$, $\sigma_\mathbf{y}$ (scale in inputs)

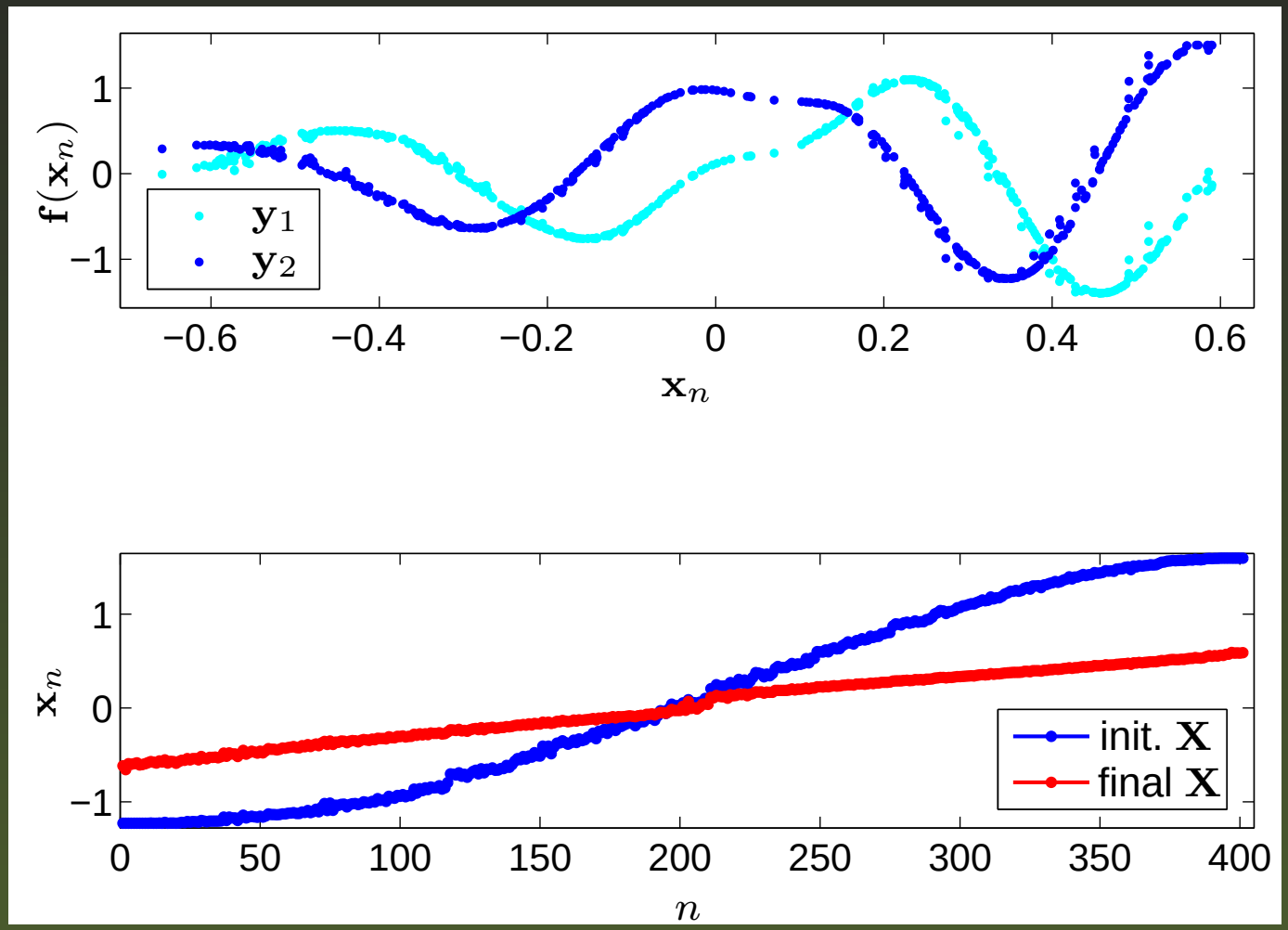
Experiments

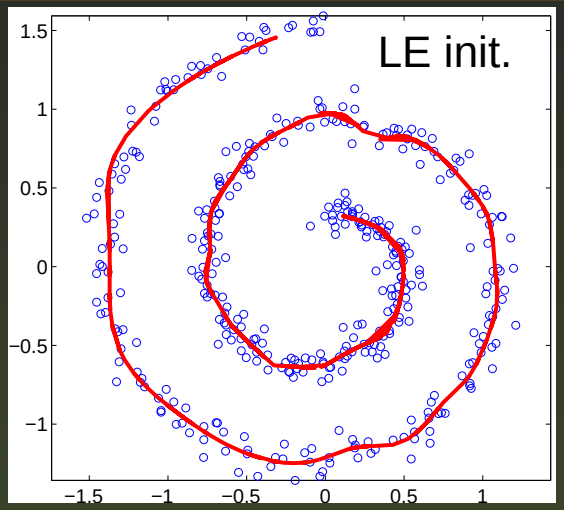
We compare DRUR with:

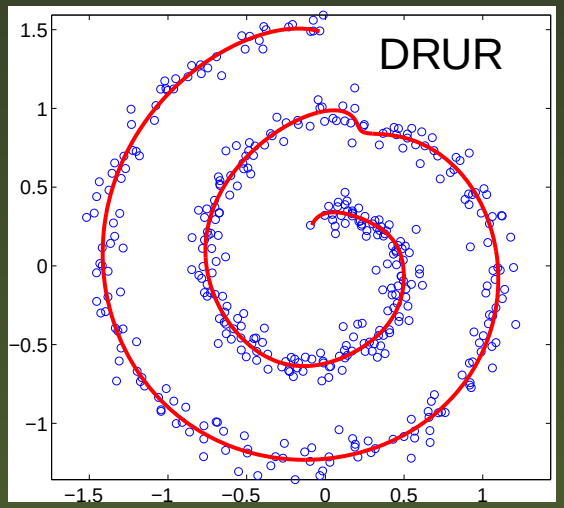
- Regularised Principal Manifolds (RPM) (Smola et al 2001): estimates $\mathbf{X}$ and $\mathbf{f}$ (RBF mapping) but not $\mathbf{F}$
- Gaussian Process Latent Variable Model (GPLVM) (Lawrence 2005): estimates $\mathbf{X}$ and $\mathbf{f}$ (GP mapping) but not $\mathbf{F}$
- Laplacian Eigenmaps Latent Variable Model (LELVM) (Carreira-Perpiñán & Lu 2007): estimates $\mathbf{f}$ and $\mathbf{F}$ (Nadaraya-Watson mappings) and density $p(\mathbf{x}, \mathbf{y})$ but not $\mathbf{X}$

We initialise all methods from the Laplacian eigenmaps (LE) embedding.

Experiments: spiral with DRUR

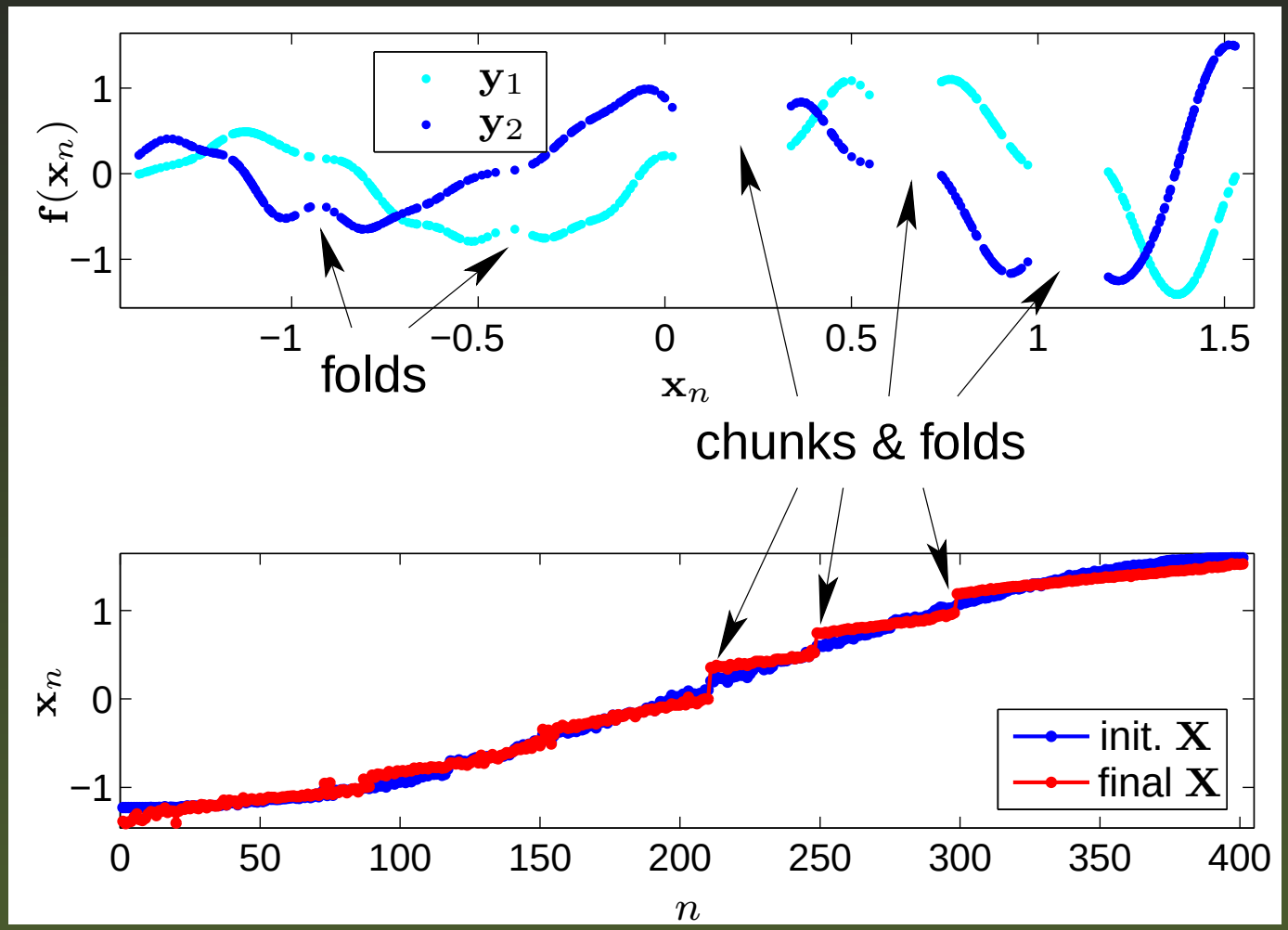


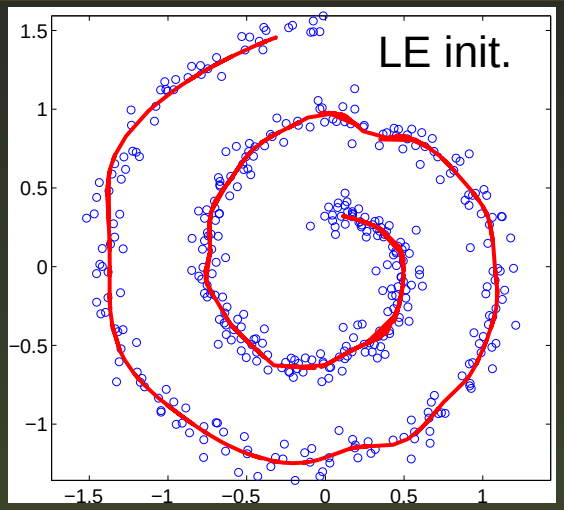


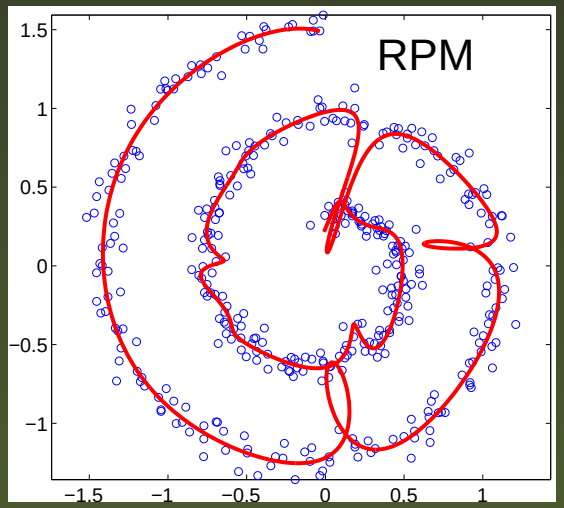


- The LE embedding has folds and boundary effects; DRUR training eliminates both, thus improving the initial embedding
- The final $\mathbf{X}$ are more uniformly distributed

Experiments: spiral with RPM



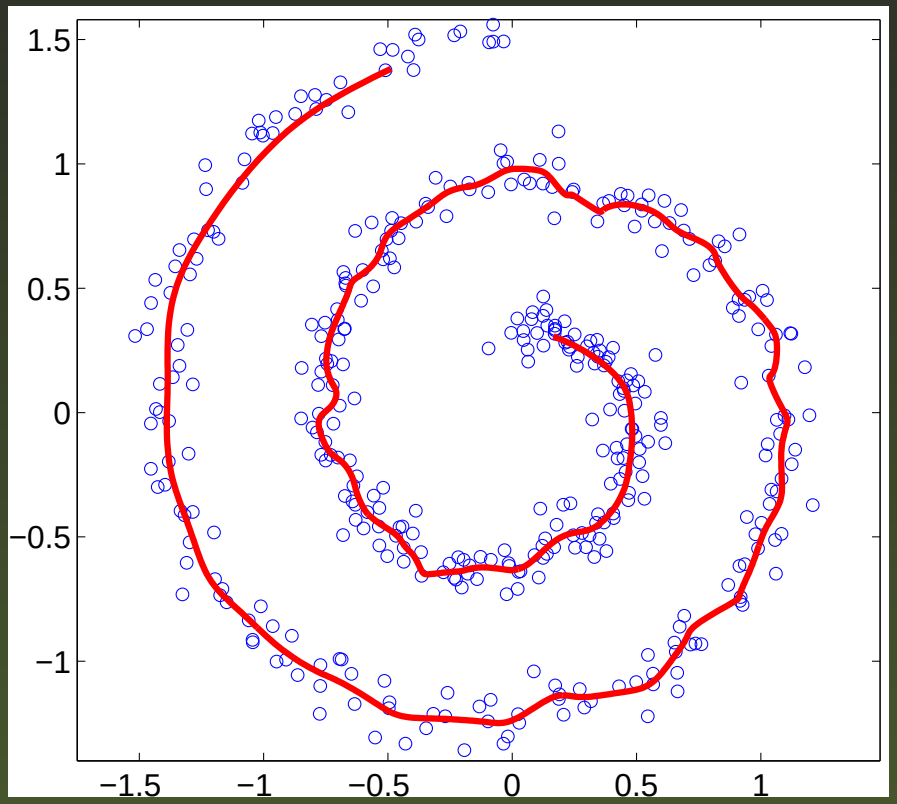




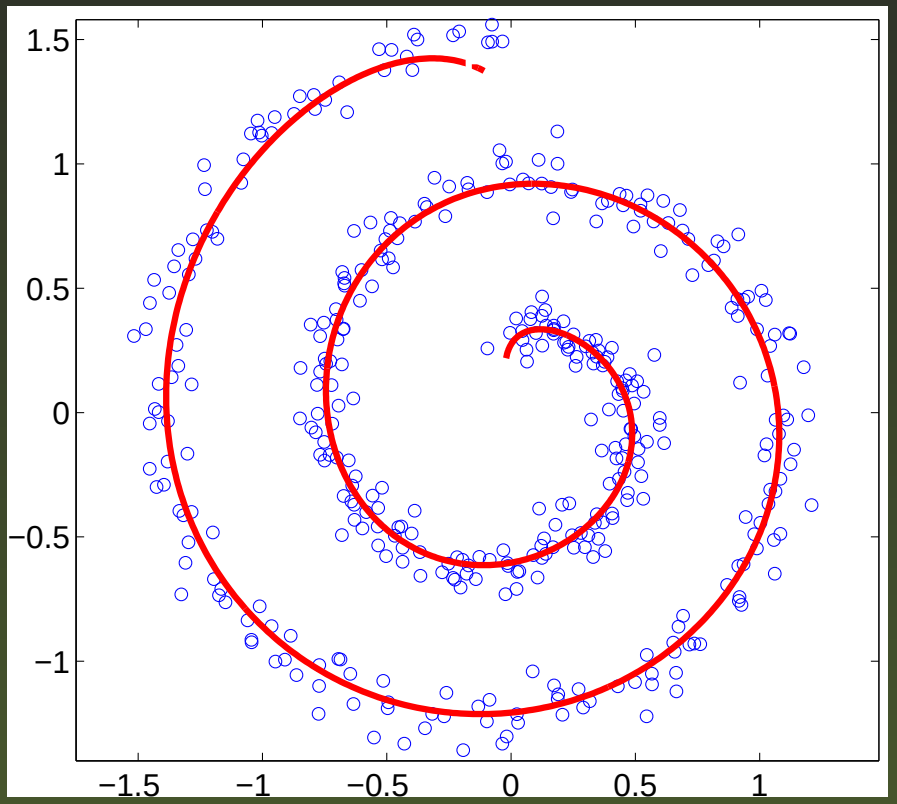
- The latent space splits into chunks & folds, actually worsening the LE embedding, because the lack of $\mathbf{F}$ means $\mathbf{x}_n$ and $\mathbf{x}_m$ can separate even if $\mathbf{f}(\mathbf{x}_n)$ and $\mathbf{f}(\mathbf{x}_m)$ are close

Experiments: spiral with LELVM

LELVM



DRUR

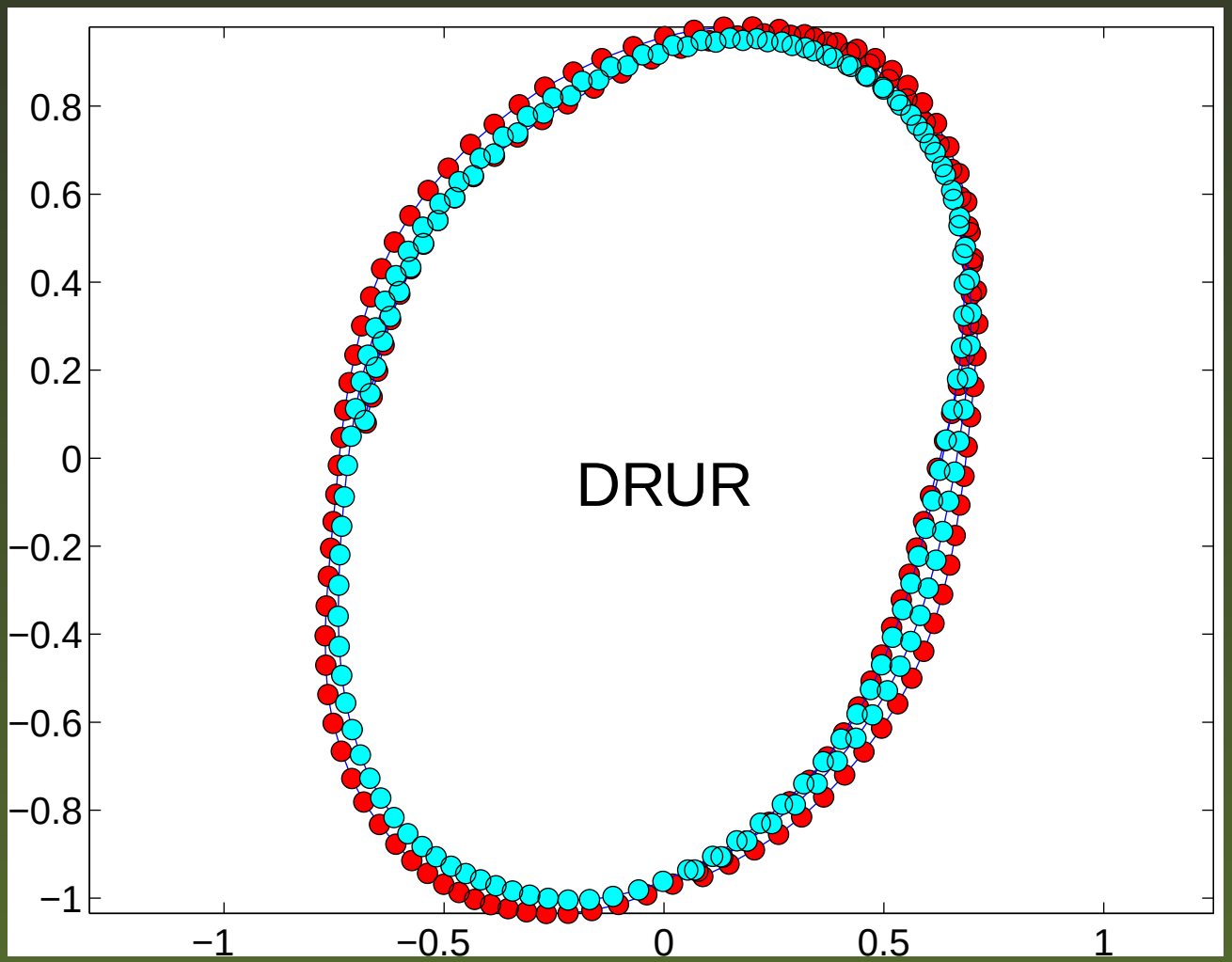


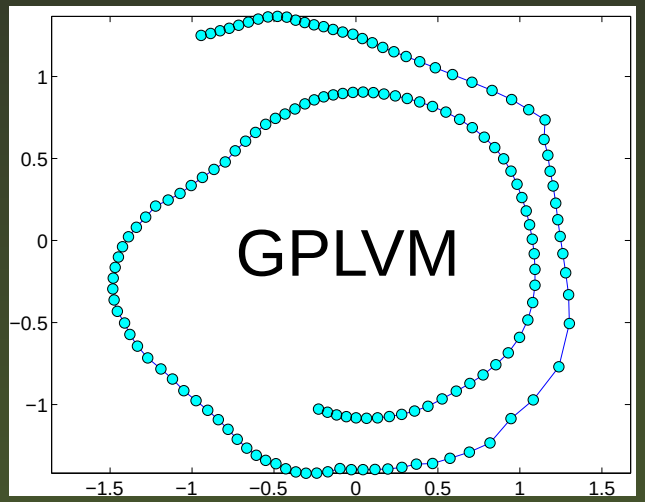
- LELVM does not modify the LE embedding
- LELVM can remove folds, but the boundary effects remain ($\mathbf{f}$, $\mathbf{F}$ are convex-sum mappings)

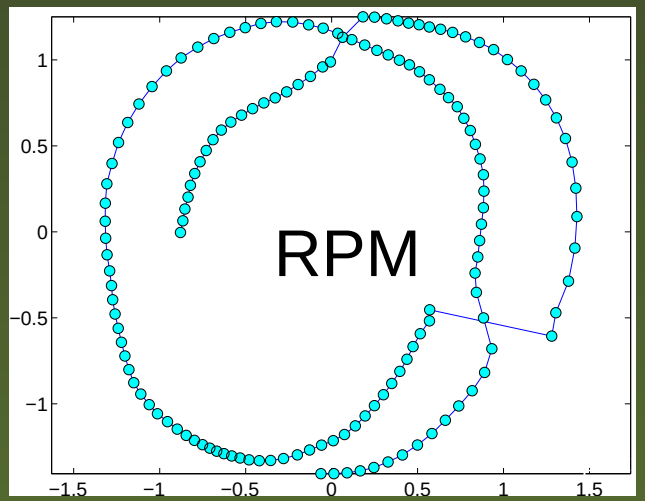
Experiments: mocap data

$\mathbf{Y}_{150\times148}$ = two cycles of running motion from the CMU mocap db:

- DRUR: smooth reconstruction (animation) of unseen poses
- RPM, GPLVM: no mapping $\mathbf{F} \Rightarrow$ disconnected latent space

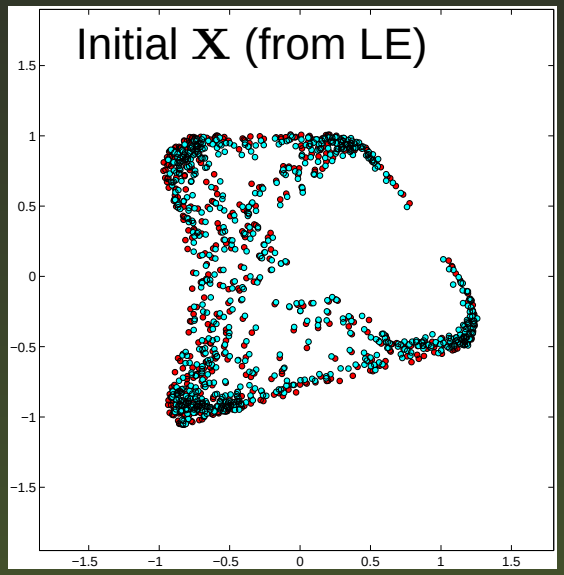


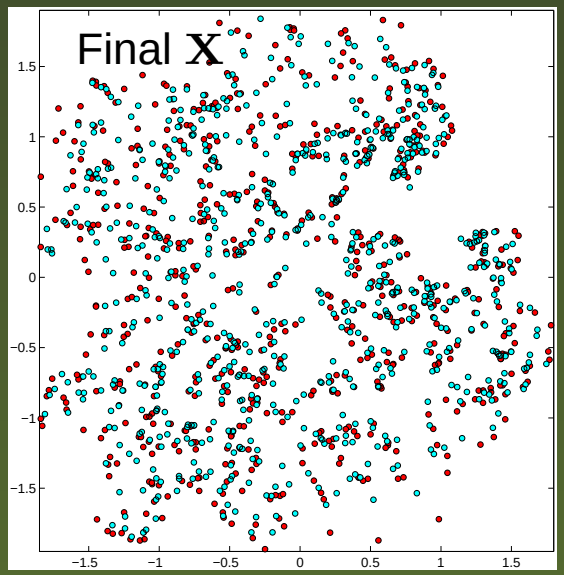


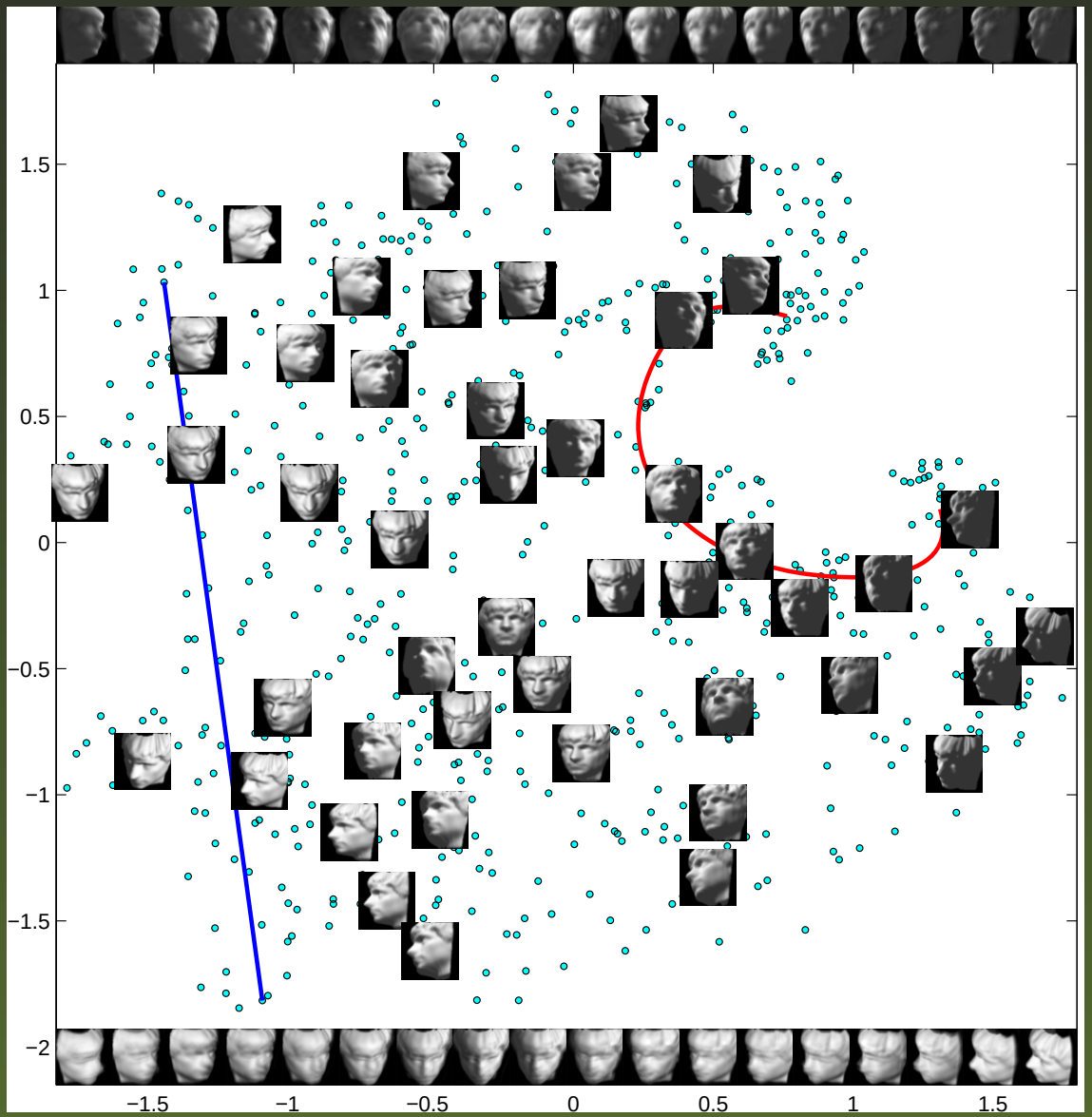


Experiments: face images

$\mathbf{Y} = 698$ face images of 64×64 pixels with varying viewpoint: DRUR significantly improves the poor original LE embedding. Similar results with digit images.







Conclusions

- Clean, symmetric formulation of dimensionality reduction in terms of the joint estimation of mappings $\mathbf{f}$, $\mathbf{F}$ and auxiliary variables $\mathbf{X}$ to minimise a regularised reconstruction error.
- Can avoid bad local optima and scales well with the dimension.
- Estimates mappings $\mathbf{f}$, $\mathbf{F}$ both ways, which are approximate inverses of each other; no pre-image problem at runtime.
- Somewhat expensive training, but fast at runtime (RBF mappings).
- Improves the spectral embedding: don't just fit $\mathbf{f}$ and $\mathbf{F}$ to $(\mathbf{X}, \mathbf{Y})$, the joint optimisation over $(\mathbf{f}, \mathbf{F}, \mathbf{X})$ is really worth.

Matlab code available online soon