PARAMETRIC DIMENSIONALITY REDUCTION BY UNSUPERVISED REGRESSION

Miguel Á. Carreira-Perpiñán[1] and **Zhengdong Lu**[2]

[1]EECS, University of California, Merced [2]ICES, University of Texas, Austin

1 Abstract

We introduce a parametric version (pDRUR) of the recently proposed Dimensionality Reduction by Unsupervised Regression algorithm. pDRUR alternately minimizes reconstruction error by fitting parametric functions given latent coordinates and data, and by updating latent coordinates given functions (with a Gauss-Newton method decoupled over coordinates). Both the fit and the update become much faster while attaining results of similar quality, and afford dealing with far larger datasets (10^5 points). We show in a number of benchmarks how the algorithm efficiently learns good latent coordinates and bidirectional mappings between the data and latent space, even with very noisy or low-quality initializations, often drastically improving the result of spectral and other methods.

2 Dimensionality reduction by unsupervised regression, nonparametric version (nDRUR) (Carreira-Perpiñán & Lu, CVPR 2008)

Given a dataset $\mathbf{Y}_{D\times N} = (\mathbf{y}_1, \dots, \mathbf{y}_N)$, we want to learn mappings $\mathbf{F}$: $\mathbf{y} \to \mathbf{x}$ (dimensionality reduction) and $\mathbf{f}$: $\mathbf{x} \to \mathbf{y}$ (reconstruction) between data points $\mathbf{y} \in \mathbb{R}^D$ and latent points $\mathbf{x} \in \mathbb{R}^L$ with $L < D$. Some methods learn the latent projections $\mathbf{X}_{L\times N} = (\mathbf{x}_1, \dots, \mathbf{x}_N)$ but not the mappings (e.g. spectral methods). Autoencoders learn both mappings to minimise the reconstruction error $E(\mathbf{f}, \mathbf{F}) = \sum_{n=1}^N \|\mathbf{y}_n - \mathbf{f}(\mathbf{F}(\mathbf{y}_n))\|^2$, but they are exceedingly slow to train ($\mathbf{f} \circ \mathbf{F}$ is a deep network with at least 3 layers of hidden units) and very prone to bad local optima.

Nonparametric dimensionality reduction by unsupervised regression (nDRUR)

Unfold the autoencoder reconstruction error by introducing auxiliary variables $\mathbf{X}$ (unsupervised regression):

$$\min_{\mathbf{X},\mathbf{f},\mathbf{F}} E(\mathbf{X}, \mathbf{f}, \mathbf{F}) = E_{\mathbf{f}}(\mathbf{X}, \mathbf{f}) + E_{\mathbf{F}}(\mathbf{X}, \mathbf{F}) \quad \begin{cases} E_{\mathbf{f}}(\mathbf{X}, \mathbf{f}) = \sum_{n=1}^N \|\mathbf{y}_n - \mathbf{f}(\mathbf{x}_n)\|^2 + \lambda_{\mathbf{f}} R_{\mathbf{f}}(\mathbf{f}) \\ E_{\mathbf{F}}(\mathbf{X}, \mathbf{F}) = \sum_{n=1}^N \|\mathbf{x}_n - \mathbf{F}(\mathbf{y}_n)\|^2 + \lambda_{\mathbf{F}} R_{\mathbf{F}}(\mathbf{F}) \end{cases}$$

and minimise variationally over $(\mathbf{X}, \mathbf{f}, \mathbf{F})$ by alternating optimisation:

- Over $\mathbf{X}$: nonlinear over NL parameters (gradient descent or conjugate gradients).
- Over $(\mathbf{f}, \mathbf{F})$: global optimum given by two separate regressions (basis function expansion at $\{\mathbf{x}_n\}$ and $\{\mathbf{y}_n\}$, resp.).

This has several advantages:

- It capitalises on the ability of spectral methods to provide a good initial $\mathbf{X}$.
- Optimising over $(\mathbf{X}, \mathbf{f}, \mathbf{F})$ jointly far improves the initial $\mathbf{X}$, eliminating folds, local clustering, boundary effects, etc.
- Penalising errors in both the latent and data spaces encourages $\mathbf{f}$ and $\mathbf{F}$ to be inverses of each other on the data manifold and far improves over having just $\mathbf{f}$ (which does not penalise projecting close $\mathbf{y}$s to distant $\mathbf{x}$s).

However, the nonparametric mappings, while appropriate with sparse data, do not scale:

- Training is $\mathcal{O}(N^3)$: (Gram) linear systems of $N \times N$ for $(\mathbf{f}, \mathbf{F})$, nonlinear optimisation over NL parameters for $\mathbf{X}$.
- Testing is $\mathcal{O}(N)$: $(\mathbf{f}, \mathbf{F})$ are basis function expansions over $\{\mathbf{x}_n\}$ and $\{\mathbf{y}_n\}$, respectively.

Many computer vision applications (articulated pose tracking, image retrieval, etc.) need faster mappings trained on large datasets.

3 Parametric DRUR (pDRUR)

By forcing $\mathbf{f}$, $\mathbf{F}$ to be in a parametric family:

- The optimisation over $\mathbf{X}$ decouples into N separate low-dim. optimisations each over L parameters, and affords a particularly robust version of the Gauss-Newton method.
- The optimisation over $\mathbf{f}$, $\mathbf{F}$ is faster.
- $\mathbf{f}$, $\mathbf{F}$ are faster at testing time.

The disadvantages:

- We restrict a bit the flexibility of $\mathbf{f}$, $\mathbf{F}$.
- Model selection needed (number of RBFs or hidden units, regularisation parameter); but no different from model selection in a standard regression setting.
- The adaptation step may have local optima.

input $\mathbf{Y}_{D\times N} = (\mathbf{y}_1, \dots, \mathbf{y}_N)$
Obtain $\mathbf{X}_{L\times N} = (\mathbf{x}_1, \dots, \mathbf{x}_N)$ from a spectral method
Fit parametric mappings $\mathbf{f}$ to $(\mathbf{X}, \mathbf{Y})$ and $\mathbf{F}$ to $(\mathbf{Y}, \mathbf{X})$
repeat
  Project: **for** $n = 1, \dots, N$
    $\mathbf{x}_n$ = approximate minimizer of (1) with Gauss-Newton
  end
  Adapt: approximately fit parametric mappings $\mathbf{f}$, $\mathbf{F}$
until convergence
return $\mathbf{f}$, $\mathbf{F}$, $\mathbf{X}$

4 Projection step: optimization over $\mathbf{X}$

The minimisation over $\mathbf{X}$ separates over each $\mathbf{x}_n$ because the form of $\mathbf{f}$, $\mathbf{F}$ does not depend on $\mathbf{X}$ (in nDRUR, $\{\mathbf{x}_n\}$ are the centres of the basis functions of $\mathbf{f}$). Consider one such problem:

$$\min_{\mathbf{x}\in\mathbb{R}^L} E_n(\mathbf{x}) = \|\mathbf{y}_n - \mathbf{f}(\mathbf{x})\|^2 + \|\mathbf{x} - \mathbf{F}(\mathbf{y}_n)\|^2 \tag{1}$$

with gradient and Hessian (where $\mathbf{J}$ is the Jacobian of $\mathbf{f}$)

$$\nabla E_n(\mathbf{x}) = 2\big(-\mathbf{J}(\mathbf{x})^T(\mathbf{y}_n - \mathbf{f}(\mathbf{x})) + \mathbf{x} - \mathbf{F}(\mathbf{y}_n)\big)$$

$$\nabla^2 E_n(\mathbf{x}) = 2\Big(\mathbf{I} + \mathbf{J}(\mathbf{x})^T\mathbf{J}(\mathbf{x}) - \sum_{d=1}^D (y_{dn} - f_d(\mathbf{x}))\nabla^2 f_d(\mathbf{x})\Big).$$

Gauss-Newton idea: approximate the Hessian with a positive definite matrix and apply Newton's method:

$$\nabla^2 E_n(\mathbf{x}) \approx 2(\mathbf{I} + \mathbf{J}(\mathbf{x})^T\mathbf{J}(\mathbf{x})) \Rightarrow \mathbf{p} = \big(\mathbf{I} + \mathbf{J}(\mathbf{x})^T\mathbf{J}(\mathbf{x})\big)^{-1}\big(\mathbf{J}(\mathbf{x})^T(\mathbf{y}_n - \mathbf{f}(\mathbf{x})) - \mathbf{x} + \mathbf{F}(\mathbf{y}_n)\big).$$

Now, use the search direction $\mathbf{p}$ in a backtracking line search with initial step size $\alpha_0 = 1$ (in our experiments, this step is accepted $> 99\%$ of the times, so the line search is rarely needed).
Note the $\mathbf{I}$ term in the approximate Hessian, arising from the $\mathbf{F}$–term in the DRUR objective function: it ensures the linear system is never singular (so no Levenberg-Marquardt corrections are needed). If the line search satisfies the usual conditions (e.g. Wolfe) then the method has global convergence.
Training cost: $\mathcal{O}(NL(M + L^2))$ (linear in N).

5 Adaptation step: optimization over $\mathbf{f}$, $\mathbf{F}$

Two independent minimisations (standard regressions):

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

Consider the regression over $\mathbf{f}$. We have used two models (details in paper):

- Radial basis function (RBF) network with M centres:

$$\mathbf{f}(\mathbf{x}) = \sum_{m=1}^M \mathbf{w}_m \phi_m(\mathbf{x}) + \mathbf{w} \qquad \phi_m(\mathbf{x}) = \exp\big(-\tfrac{1}{2}\|(\mathbf{x} - \boldsymbol{\mu}_m)/\sigma\|^2\big)$$

- Neural network (MLP) with a single hidden layer with M units:

$$\mathbf{f}(\mathbf{x}) = \mathbf{W}_2\sigma(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2 \qquad \sigma(x) = 1/(1 + \exp(x))$$

This is a much easier problem than training an autoencoder:

- pDRUR: 2 separate minimisations each over a network with 1 hidden layer
- autoencoder: 1 minimisation over a network with 3 hidden layers.

Cost: $\mathcal{O}(NM(D + L))$ training (linear in N), $\mathcal{O}(M(D + L))$ testing. Note $M \ll N$.

7 Conclusion

Involving both mappings $\mathbf{f}$, $\mathbf{F}$ in the objective function and allowing the latent coordinates $\mathbf{X}$ as free parameters allows to use good initialisations and to find good optima; making the mappings parametric allows to optimise very efficiently and scale up to large datasets. We encourage you to try pDRUR in your applications. Matlab code: `https://eecs.ucmerced.edu`.

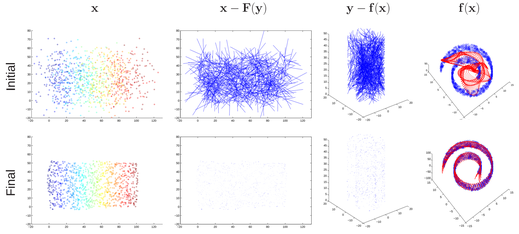
Work supported by NSF CAREER award IIS–0754089.

6 Experimental results

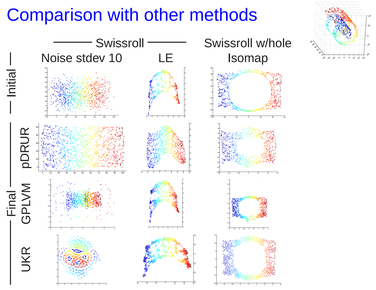
We find that pDRUR is more likely than autoencoders and other methods to converge to a good optimum from a poor initial $\mathbf{X}$. We conjecture that this is due to our effective optimization strategy and to the use of an enlarged search space over $\mathbf{X}$, which perhaps may facilitate escaping from local optima. The success of the algorithm strongly argues against out-of-sample extensions of spectral methods based on the latent coordinates $\mathbf{X}$ directly computed by the spectral method; a few pDRUR iterations (sometimes a single one) typically provide a much better $\mathbf{X}$ and consequently much better mappings.

Swiss roll

Initial $\mathbf{X}$ = true $\mathbf{X}$ + Gaussian noise with stdev 10; 30 RBFs, $\lambda_{\mathbf{f}} = \lambda_{\mathbf{F}} = 10^{-5}$; 100 iterations.

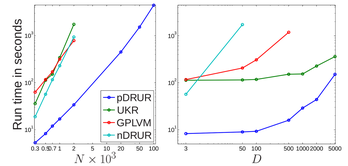


Comparison with other methods

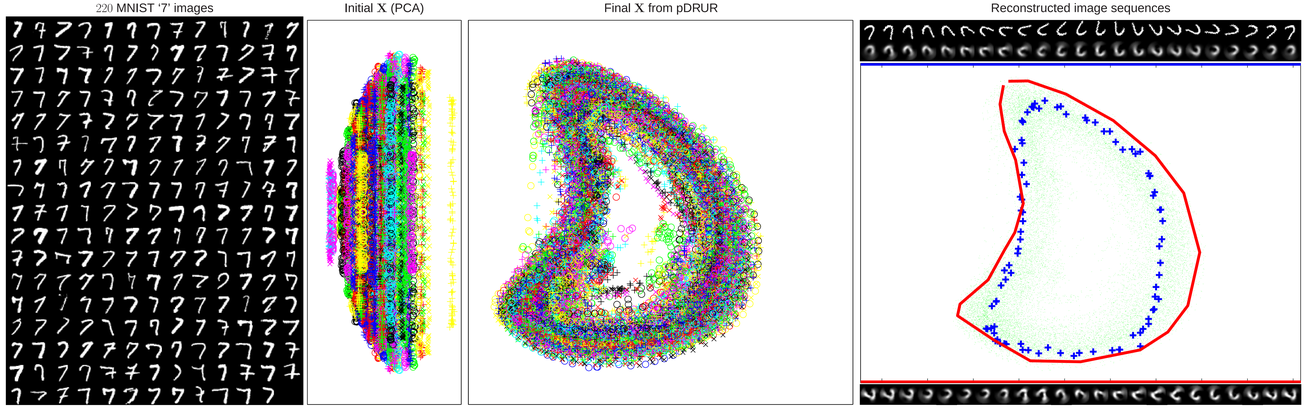


Run time

Log-log plot of the run time for different methods on a Swiss roll dataset with $N \le 10^5$ points in $D = 3$ dimensions (left) and with $N = 500$ points in $D \le 5\,000$ dimensions (right).



Rotated MNIST digit '7'



Left: 220 different MNIST 28×28 images, each rotated at 4–degree intervals totalling $N = 19\,800$ points in $D = 784$ dimensions (sample sequence above). pDRUR mapped this to 2D ($\mathbf{f}$: 5 RBFs, $\mathbf{F}$: 5 RBFs, $\lambda_{\mathbf{f}} = \lambda_{\mathbf{F}} = 0.01$, 200 iterations, $\mathbf{X}$ initialized by PCA). *Middle plots*: each rotation sequence is color- and marker-coded in latent space. *Right*: an out-of-sample path in latent space (red) and the images that $\mathbf{f}$ produces (below); and a test sequence of images, its 2D projection with $\mathbf{F}$ (blue +) and its reconstruction with $\mathbf{f} \circ \mathbf{F}$ (above).