Manifold Learning and Missing Data Recovery through Unsupervised Regression

Miguel Á. Carreira-Perpiñán
EECS, University of California, Merced
http://eecs.ucmerced.edu

Zhengdong Lu
Microsoft Research Asia, Beijing
http://research.microsoft.com/en-us/people/zhengdol

***Abstract*—We propose an algorithm that, given a high-dimensional dataset with missing values, achieves the distinct goals of learning a nonlinear low-dimensional representation of the data (the dimensionality reduction problem) and reconstructing the missing high-dimensional data (the matrix completion, or imputation, problem). The algorithm follows the Dimensionality Reduction by Unsupervised Regression approach, where one alternately optimizes over the latent coordinates given the reconstruction and projection mappings, and vice versa; but here we also optimize over the missing data, using an efficient, globally convergent Gauss-Newton scheme. We also show how to project or reconstruct test data with missing values. We achieve impressive reconstructions while learning good latent representations in image restoration with 50% missing pixels.**

***Keywords*-dimensionality reduction; manifold learning; missing data; matrix completion.**

In manifold learning, we observe a high-dimensional dataset $\mathbf{Y}_{D\times N} = (\mathbf{y}_1, \dots, \mathbf{y}_N)$ and want to learn a low-dimensional, latent representation of it based on a small number of degrees of freedom $L \ll D$ (the training problem). Often we also want to project a new $\mathbf{y}$ to a low-dimensional $\mathbf{x} = \mathbf{F}(\mathbf{y})$, or to map a new $\mathbf{x}$ to a high-dimensional $\mathbf{y} = \mathbf{f}(\mathbf{x})$ (the testing problem). A large number of manifold learning algorithms exist, but they typically assume that the training dataset $\mathbf{Y}$ and the test vectors are fully observed. Yet in many applications the data available may contain missing values caused by excessive noise, sensor malfunction, nonresponse to specific questions in a survey, or other reasons. For training, sometimes it is possible to extract a sizable, complete subset (where each $\mathbf{y}_n$ has no missing entries) and use a standard algorithm on it, but this still loses useful information that the partially present vectors contain. Given the effort sometimes invested in data collection, discarding partial vectors is wasteful. In other cases, keeping only the fully-observed vectors may not leave sufficient data to train at all. It then becomes necessary to use all the existing data and deal with the missing values. Even if we manage to estimate mappings $\mathbf{F}$, $\mathbf{f}$, we cannot apply them directly to a test vector with missing entries. This situation occurs in applications such as collaborative filtering (e.g. embedding a new user given some of their movie ratings in a Netflix dataset, or given some of their opinions in Opinion Space [1]) or image retrieval (restoring a new image with missing pixels and projecting it to latent space). Our goal in this paper is to propose a principled, effective nonlinear manifold learning method that deals with missing data in both training and test, and also reconstructs (imputes) it. We first briefly review existing work in dimensionality reduction and matrix completion, and introduce the Dimensionality Reduction by Unsupervised Regression approach without missing data (sec. II). We then explain our method mDRUR (sec. III), its linear version (sec. IV), and its extension to test data (sec. V), followed by experimental results (sec. VI).

I. Related Work

Many algorithms have been proposed for dimensionality reduction, but far less work exists on dimensionality reduction with missing data. Probabilistic methods provide one principled way to deal with it. Latent variable models (LVMs) such as factor analysis [2] or the generative topographic mapping [3] define a joint probability model of high- and low-dimensional variables, and the latter ones are considered as missing data during maximum-likelihood estimation with the EM algorithm. In principle, this allows considering the unobserved high-dimensional values as missing variables too and having EM deal with them. However, integrating (even approximately) over the corresponding posterior distribution for each data point is cumbersome and computationally costly. In practice, this often limits nonlinear LVMs to using Gaussian mixtures and a very small latent dimensionality. One exception is the Laplacian eigenmaps latent variable model [4], where training amounts to a spectral problem that yields a kernel density estimate, but it assumes no missing data. In our approach, instead of expensive marginalization and conditioning operations we have far more efficient optimizations that besides decouple significantly, and using arbitrarily many latent dimensions or nonlinear mappings poses no special problem; we demonstrate this with RBF mappings in 9D latent spaces.

In unsupervised regression, which goes back to early work in factor analysis, one takes the latent coordinates $\mathbf{x}_n$ of each data point $\mathbf{y}_n$ as parameters to be estimated together with the reconstruction mapping $\mathbf{f}$ that maps latent points to data space. These methods extend trivially to missing data, since their least-squares error function takes the form $\min_{\mathbf{f},\mathbf{X}} \sum_{n=1}^{N} \|\mathbf{y}_n - \mathbf{f}(\mathbf{x}_n)\|^2 = \sum_{n,d=1}^{N,D} (y_{dn} - f_d(\mathbf{x}_n))^2$ so one can simply ignore the terms for which y_{dn} is missing. Linear functions $\mathbf{f}$ have been used in the homogeneity analysis literature [5], where this approach is

called "missing data deleted". Nonlinear functions $\mathbf{f}$ have been used recently (neural nets [6]; Gaussian processes for collaborative filtering [7]). Although simple and faster if there are many missing values, this returns only $\mathbf{f}$ (no $\mathbf{F}$) and $\mathbf{X}_{L\times N} = (\mathbf{x}_1, \dots, \mathbf{x}_N)$, and reconstructs each $\mathbf{y}$ (or, rather, its missing values) implicitly as $\mathbf{f}(\mathbf{x})$. This has several disadvantages (see fig. 3): (1) since $\mathbf{x}$ has only L degrees of freedom, for complex data (e.g. images) the reconstruction $\mathbf{y} = \mathbf{f}(\mathbf{x})$ can have low quality; (2) it treats the vector output of $\mathbf{f}(\mathbf{x})$ inconsistently at training and testing, as some components are used (for the missing y_ds) and the rest discarded; (3) lacking entirely the missing terms in the error function ignores useful information, as having estimates of the missing y_{dn} provides feedback about the goodness of $\mathbf{f}$—specially if domain knowledge provides a good initial $\mathbf{Y}$. A final drawback even if all $\mathbf{y}_n$ are observed [8] is that lacking $\mathbf{F}$ in the objective function often distorts the latent space: two nearby $\mathbf{y}$ points can have distant $\mathbf{x}$s. These problems are caused by having $\mathbf{f}$ both *represent the manifold* and *reconstruct the data*. Our proposed approach addresses this in a new, principled way, by separating these two different roles: we consider the missing ys as parameters with the same status as $\mathbf{X}$, $\mathbf{f}$ or $\mathbf{F}$, and optimize over them (note that, without $\mathbf{F}$, optimizing over $\mathbf{y}$ makes no difference); and this carries over naturally to missing values in a test $\mathbf{y}$.

Until recently, traditional dimensionality reduction with missing data has been unrelated with the problem of matrix completion, which arises commonly in data mining, machine learning and computer vision. Much recent work on matrix completion stems from compressed sensing, whose success assumes the signal is sparse on a certain basis, i.e., a low-rank assumption of the matrix, so that its rows (or columns) lie on a low-dimensional subspace. Our work generalizes this linear assumption and instead assumes the data live on a smooth, nonlinear manifold with low intrinsic dimensionality. Research on compressed sensing on manifolds is limited. Baraniuk and Wakin [9] point out that when the curvature of the manifold is small enough, a random projection from the data space is likely to keep the metric of the manifold with small distortion, but do not give an algorithm to recover the manifold structure; Chen et al. [10] do give a practical algorithm that assumes the manifold to be predetermined, and cannot be applied to partially observed data. Singular value projection (SVP) [11] is a scalable optimization scheme recently proposed for rank minimization with affine constraints, of which low-rank matrix completion is a special case. It iterates between a SVD step (to ensure the low-rank structure) and a data-feeding step (to incorporate the information of observed entries). SVP has theoretical bounds for the exact completion case, and empirically is effective on real-world problems, such as MovieLens data. Wang et al. [12] propose a denoising approach that generalizes matrix completion to local low-rank manifolds, but does not recover the nonlinear manifold explicitly.

II. Dimensionality Reduction by Unsupervised Regression (DRUR)

Given fully observed data points $\mathbf{Y}_{D\times N}$, Dimensionality Reduction by Unsupervised Regression [8], [13] minimizes the following objective function alternately over mappings $\mathbf{f}$, $\mathbf{F}$ and latent coordinates $\mathbf{X}_{L\times N}$ (where $R_{\mathbf{f}}(\mathbf{f})$, $R_{\mathbf{F}}(\mathbf{F})$ are quadratic regularizers):

$$\min_{\mathbf{X},\mathbf{f},\mathbf{F}} E(\mathbf{X},\mathbf{f},\mathbf{F}) = E_{\mathbf{f}}(\mathbf{X},\mathbf{f}) + E_{\mathbf{F}}(\mathbf{X},\mathbf{F}) \quad (1)$$

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

$$E_{\mathbf{F}}(\mathbf{X},\mathbf{F}) = \textstyle\sum_{n=1}^{N} \|\mathbf{x}_n - \mathbf{F}(\mathbf{y}_n)\|^2 + \lambda_{\mathbf{F}} R_{\mathbf{F}}(\mathbf{F}). \quad (3)$$

E can be seen as unfolding the reconstruction error of an autoencoder $\sum_{n=1}^{N} \|\mathbf{y}_n - \mathbf{f}(\mathbf{F}(\mathbf{y}_n))\|^2$ by introducing auxiliary variables $\mathbf{X}$. This has several advantages: the optimization for fixed $\mathbf{X}$ results in two separate regressions over $\mathbf{f}$ and $\mathbf{F}$, which are shallower architectures than the autoencoder $\mathbf{f}\circ\mathbf{F}$ and thus easier to optimize; and the search space enlarged with $\mathbf{X}$ allows to use a good initialization from a spectral method and facilitates escaping local optima. DRUR learns very good embeddings in practice, thanks to the bidirectional constraints imposed by $\mathbf{f}$ (that does not like to map close inputs $\mathbf{x}$ to faraway outputs $\mathbf{y}$) and $\mathbf{F}$ (that does not like to map close inputs $\mathbf{y}$ to faraway outputs $\mathbf{x}$); the value of using both mappings $\mathbf{F}$, $\mathbf{f}$ as opposed to just one was extensively demonstrated in [8]. The mappings may be taken nonparametric or parametric. In the latter case, the optimization over $\mathbf{X}$ decouples over each point, so we have N optimizations over L parameters each (instead of one over NL), which besides are amenable to a form of the Gauss-Newton method that avoids ill-conditioning; this gives an orders-of-magnitude speedup [13]. Scalability is paramount when reconstructing missing data in a high-dimensional $\mathbf{y}$-space, so we focus on the parametric version (pDRUR), but our approach applies to the nonparametric one too.

III. Missing Data with DRUR (mDRUR)

Consider now the dimensionality reduction problem where the data matrix $\mathbf{Y}_{D\times N}$ has missing values: each column vector $\mathbf{y}_n$ can have from 0 to $D-1$ missing components (for now, ignore the case where all D components are missing, which means we do not have a vector $\mathbf{y}_n$ at all). We represent the pattern of missing values in $\mathbf{Y}$ with a binary matrix $\mathbf{M}_{D\times N}$, where $m_{dn} = 1$ if y_{dn} is observed and 0 otherwise. Since our algorithm will need initial estimates of the missing $\mathbf{Y}$ values, $m_{dn} = 0$ will indicate that y_{dn} is the initial value. In the equations below, we write $\mathbf{Y}_0$ to mean the missing entries of $\mathbf{Y}$ (so $\mathbf{Y}$ contains $\mathbf{Y}_0$ and the present values), and likewise $\mathbf{y}_{n,0}$ for column n of $\mathbf{Y}$.

Our approach to extend pDRUR to missing data consists of optimizing its objective function over the missing $\mathbf{Y}_0$ values too (we use Frobenius norms for matrices throughout,

input $\mathbf{M}_{D\times N}$: binary indicator matrix (0 = missing)
input $\mathbf{Y}_{D\times N}$: observed values & initial missing values
input $\mathbf{X}_{L\times N}$: initialized by a spectral method
Fit parametric mappings $\mathbf{f}$: $(\mathbf{X},\mathbf{Y})$, $\mathbf{F}$: $(\mathbf{Y},\mathbf{X})$
repeat
  Project-reconstruct: **for** $n = 1,\ldots,N$
    $(\mathbf{y}_{n,0},\mathbf{x}_n)$ = approx. Gauss-Newton min. of (7)
  Adapt: approximately fit $\mathbf{f}$: $(\mathbf{X},\mathbf{Y})$, $\mathbf{F}$: $(\mathbf{Y},\mathbf{X})$
until convergence
return $\mathbf{Y}_0,\mathbf{X},\mathbf{F},\mathbf{f}$

Figure 1. Missing-data DRUR algorithm.

and $\mathbf{f}(\mathbf{X})$ means $\mathbf{f}$ applied vectorwise to each column $\mathbf{x}$):

$$E(\mathbf{Y}_0,\mathbf{X},\mathbf{f},\mathbf{F}) = E_{\mathbf{f}}(\mathbf{Y}_0,\mathbf{X},\mathbf{f}) + E_{\mathbf{F}}(\mathbf{Y}_0,\mathbf{X},\mathbf{F}) \quad (4)$$

$$E_{\mathbf{f}}(\mathbf{Y}_0,\mathbf{X},\mathbf{f}) = \|\mathbf{Y}-\mathbf{f}(\mathbf{X})\|^2 + \lambda_{\mathbf{f}}\|\mathbf{f}\|^2 \quad (5)$$

$$E_{\mathbf{F}}(\mathbf{Y}_0,\mathbf{X},\mathbf{F}) = \|\mathbf{X}-\mathbf{F}(\mathbf{Y})\|^2 + \lambda_{\mathbf{F}}\|\mathbf{F}\|^2. \quad (6)$$

The role of $\mathbf{F}$ is fundamental: without it (using $E_{\mathbf{f}}$ alone), optimizing over $\mathbf{Y}_0$ yields a trivial solution $\mathbf{Y}_0 = (\mathbf{f}(\mathbf{X}))_0$ corresponding to the "missing data deleted" method discussed earlier. We apply alternating optimization to this mDRUR objective function cyclically over $(\mathbf{X},\mathbf{Y}_0) \rightarrow (\mathbf{f},\mathbf{F})$ (see fig. 1). Empirically we find that other orders, such as $\mathbf{X} \rightarrow \mathbf{Y}_0 \rightarrow (\mathbf{f},\mathbf{F})$ or $\mathbf{Y}_0 \rightarrow \mathbf{X} \rightarrow (\mathbf{f},\mathbf{F})$, are consistently slower (all converge though). We deal with each of the two optimization problems separately. At each step we globally rescale $\mathbf{X}$ (and the RBF width of $\mathbf{f}$ and weights of $\mathbf{F}$) so the errors $E_{\mathbf{f}}$ and $E_{\mathbf{F}}$ are always comparable.

A. Projection-Reconstruction Step: wrt $(\mathbf{X},\mathbf{Y}_0)$

If optimizing over $(\mathbf{X},\mathbf{Y}_0)$ (also over $\mathbf{X}$ or $\mathbf{Y}_0$ separately), eq. (4) decouples so there are N independent optimizations each over at most $D+L$ variables $(\mathbf{x}_n,\mathbf{y}_{n,0})$. This is crucial with some of our applications, e.g. where $\mathbf{y}_n$ is a high-dimensional image with missing entries. Consider one such term (omitting the subindex n for simplicity):

$$E(\mathbf{x},\mathbf{y}_0) = \|\mathbf{y}-\mathbf{f}(\mathbf{x})\|^2 + \|\mathbf{x}-\mathbf{F}(\mathbf{y})\|^2 \quad (7)$$

where $\mathbf{x}\in\mathbb{R}^L$, $\mathbf{y}\in\mathbb{R}^D$, $L<D$, and we partition $\mathbf{y}$ as $\mathbf{y} = \binom{\mathbf{y}_0}{\mathbf{y}_1}$ (possibly reordering indices) so that $\mathbf{y}_1$ is constant (the present components), and $\mathbf{y}_0$ is a variable to be optimized over (the missing components). We can then obtain the gradient and Hessian of E wrt $(\mathbf{x},\mathbf{y}_0)$ as follows:

$$\frac{\partial E}{\partial \mathbf{x}} = 2\left(-\mathbf{J}_{\mathbf{f}}^T(\mathbf{y}-\mathbf{f}(\mathbf{x})) + \mathbf{x} - \mathbf{F}(\mathbf{y})\right) \quad (8)$$

$$\frac{\partial E}{\partial \mathbf{y}_0} = 2\left(-\mathbf{J}_{\mathbf{F},0}^T(\mathbf{x}-\mathbf{F}(\mathbf{y})) + \mathbf{y}_0 - (\mathbf{f}(\mathbf{x}))_0\right) \quad (9)$$

$$\frac{\partial^2 E}{\partial \mathbf{x}^2} = 2\left(\mathbf{I} + \mathbf{J}_{\mathbf{f}}^T\mathbf{J}_{\mathbf{f}} - \mathbf{H}_{\mathbf{f}}^T(\mathbf{y}-\mathbf{f}(\mathbf{x}))\right) \quad (10)$$

$$\frac{\partial^2 E}{\partial \mathbf{y}_0^2} = 2\left(\mathbf{I} + \mathbf{J}_{\mathbf{F},0}^T\mathbf{J}_{\mathbf{F},0} - \mathbf{H}_{\mathbf{F},0}^T(\mathbf{x}-\mathbf{F}(\mathbf{y}))\right) \quad (11)$$

$$\frac{\partial^2 E}{\partial \mathbf{x}\partial \mathbf{y}_0} = -2\left(\mathbf{J}_{\mathbf{f},0}^T + \mathbf{J}_{\mathbf{F},0}\right) \quad (12)$$

where we partition the matrices (Jacobians of $\mathbf{f}$ wrt $\mathbf{x}$, and of $\mathbf{F}$ wrt $\mathbf{y}$) as $\mathbf{J}_{\mathbf{f}} = (\mathbf{J}_{\mathbf{f},0}^T\ \mathbf{J}_{\mathbf{f},1}^T)^T$ and $\mathbf{J}_{\mathbf{F}} = (\mathbf{J}_{\mathbf{F},0}\ \mathbf{J}_{\mathbf{F},1})$, and other $D\times 1$ vectors are partitioned like $\mathbf{y}$. The least-squares form of the problem suggests using a Gauss-Newton method (far more effective than gradient descent) if the dimensionality of $\mathbf{y}_0$ is not too large. We capitalize on the positive definite Hessian approximation

$$\widetilde{\mathbf{H}} = 2\Big(\mathbf{I} + \begin{pmatrix} \mathbf{J}_{\mathbf{f}}^T\mathbf{J}_{\mathbf{f}} & -\mathbf{J}_{\mathbf{f},0}^T - \mathbf{J}_{\mathbf{F},0} \\ -\mathbf{J}_{\mathbf{f},0} - \mathbf{J}_{\mathbf{F},0}^T & \mathbf{J}_{\mathbf{F},0}^T\mathbf{J}_{\mathbf{F},0} \end{pmatrix}\Big) \quad (13)$$

so the search direction $\mathbf{p}$ for optimizing over $(\mathbf{x},\mathbf{y}_0)$ is the solution of the non-sparse linear system $\widetilde{\mathbf{H}}\mathbf{p} = -\mathbf{g}$ (where $\mathbf{g}$ is the gradient). In a nonlinear least squares problem of the typical form $\min_{\mathbf{x}}\|\mathbf{y}-\mathbf{f}(\mathbf{x})\|^2$ (i.e., using $E_{\mathbf{f}}$ alone, no $\mathbf{F}$), the Gauss-Newton method has the disadvantage that the approximate Hessian of $\mathbf{f}$ has the form $2\mathbf{J}_{\mathbf{f}}^T\mathbf{J}_{\mathbf{f}}$, and thus can become singular or ill-conditioned, leading to unstable directions and steps. One then needs to correct for this, e.g. by adding an adaptive bias to its diagonal, as in the Levenberg-Marquardt method [14]; searching for a good bias involves solving multiple linear systems at each iteration. In our case, the $\mathbf{F}$-term $\|\mathbf{x}-\mathbf{F}(\mathbf{y})\|^2$, quadratic on $\mathbf{x}$, introduces an additive Jacobian $2\mathbf{I}$ that acts as an in-built regularizer. The same applies to the Jacobian wrt $\mathbf{y}_0$. This means our Gauss-Newton direction is robust without the need of any modifications; with usual line search conditions and a bounded Jacobian, global convergence is assured [14, p. 40]. See section III-C for the computational cost.

B. Adaptation Step: wrt $(\mathbf{f},\mathbf{F})$

This is as with DRUR, i.e., solving two separate regressions for $\mathbf{f}$ and $\mathbf{F}$, resp. Take the one for $\mathbf{f}$. We have used two parametric forms: linear mappings $\mathbf{f}(\mathbf{x}) = \mathbf{A}\mathbf{x}+\mathbf{a}$, and radial basis function (RBF) networks $\mathbf{f}(\mathbf{x}) = \mathbf{W}\Phi(\mathbf{x})$ with $M \ll N$ Gaussian RBFs $\phi_m(\mathbf{x}) = \exp(-\frac{1}{2}\|(\mathbf{x}-\boldsymbol{\mu}_m)/\sigma\|^2)$. For both, our regularizer is a quadratic penalty on the weights: $\|\mathbf{f}\|^2 = \|\mathbf{A}\|^2$ (linear), $\|\mathbf{W}\|^2$ (RBF). In RBFs we update the centers $\boldsymbol{\mu}_m$ by k-means, and σ by cross-validation (training the RBF on a subset of the data and testing it on the rest). The weights and biases have a unique solution given by a linear system, with a computational cost $\mathcal{O}(NM(M+D))$ in training time and $\mathcal{O}(MD)$ in memory for the RBF. The gradient of $\mathbf{f}$ wrt $\mathbf{x}$ (Jacobian $\mathbf{J}$), needed in the optimization over $(\mathbf{Y}_0,\mathbf{X})$, is $\mathbf{J}(\mathbf{x}) = \frac{1}{\sigma^2}\sum_{m=1}^M \mathbf{w}_m\phi_m(\mathbf{x})(\boldsymbol{\mu}_m-\mathbf{x})^T$.

C. Practical Considerations

mDRUR's user parameters are those parameters for the mappings (e.g. number of basis functions M and regularization $\lambda \geq 0$) that would be needed in a standard regression.

Initialization: As with any nonlinear method, local optima exist, but they are a fair price to pay for the higher quality achievable. When no domain knowledge is available, we have obtained best results by initializing $\mathbf{Y}_0$ with SVP [11] with a large rank (or with regularized linear mDRUR; both achieve very similar results); this beats using zero- or mean-imputation. Given $\mathbf{Y}_0$, we initialize $\mathbf{X}$ using a spectral method applied to $\mathbf{Y}$. Experimentally, we note mDRUR is

very good at improving an initial solution, and that even if $(\mathbf{f}, \mathbf{F})$ get stuck in a poor local optimum, the reconstructed $\mathbf{Y}_0$ can still be quite good.

Very high dimension of the missing values: Call $D_n = \dim \mathbf{y}_{n,0}$ the number of missing variables in $\mathbf{y}_n$ for each $n = 1, \dots, N$. In some applications, such as image reconstruction, matrix completion or recommender systems, D_n can be large. While mDRUR's formulation is very efficient—each $\mathbf{y}_n$ decouples from all others—solving (or even storing the relevant matrices of) the linear system in each linear mDRUR or Gauss-Newton step becomes expensive, with a cost $\mathcal{O}((L + D_n)^3)$. We can then use the following strategies. A simple but slow option is (stochastic) gradient descent. More complicated options are Gauss-Newton-CG (which solves the linear systems approximately with the conjugate gradient method) and limited-memory BFGS (which approximates the Hessian using a relatively small number of column vectors). A much simpler strategy that we have found useful is to use an approximate, fast step where we ignore the $\mathbf{F}$-term in the error and thus set $\mathbf{y}_{n,0} = (\mathbf{f}(\mathbf{x}_n))_0$. If this decreases the DRUR error, we accept the step and optimize the error over $\mathbf{x}_n$; otherwise, we reject it and do the usual optimization over $(\mathbf{y}_{n0}, \mathbf{x}_n)$. This guarantees we decrease the error at each iteration, and is much faster with large D_n. Finally, in matrix completion problems where there may be no a priori preference over $\mathbf{Y}$ or $\mathbf{Y}^T$, we should choose the alternative with fewer rows, so the cost of each step is the smaller of the two.

D. Relation with Low-Rank Factorization

If the mappings $\mathbf{f}$, $\mathbf{F}$ can be written as a linear combination of basis functions, as is the case for linear mappings and RBF networks, then (absorbing the respective biases into $\mathbf{Y}$ and $\mathbf{X}$) the terms $\|\mathbf{Y} - \mathbf{f}(\mathbf{X})\|^2 = \|\mathbf{Y} - \mathbf{W}_\mathbf{f}\mathbf{\Phi}_\mathbf{f}(\mathbf{X})\|^2$ and $\|\mathbf{X} - \mathbf{F}(\mathbf{Y})\|^2 = \|\mathbf{X} - \mathbf{W}_\mathbf{F}\mathbf{\Phi}_\mathbf{F}(\mathbf{Y})\|^2$ are SVD-like. At convergence we obtain two approximate low-rank factorizations $\mathbf{Y} \approx \mathbf{W}_\mathbf{f}\mathbf{\Phi}_\mathbf{f}(\mathbf{X}) = \mathbf{f}(\mathbf{X})$ and $\mathbf{X} \approx \mathbf{W}_\mathbf{F}\mathbf{\Phi}_\mathbf{F}(\mathbf{Y}) = \mathbf{F}(\mathbf{Y})$ with $\mathbf{W}_\mathbf{f}$ of $D \times M_\mathbf{f}$ and $\mathbf{\Phi}_\mathbf{f}(\mathbf{X})$ of $M_\mathbf{f} \times N$, $\mathbf{W}_\mathbf{F}$ of $L \times M_\mathbf{F}$ and $\mathbf{\Phi}_\mathbf{F}(\mathbf{Y})$ of $M_\mathbf{F} \times N$, and the ranks are given by (strictly, smaller or equal than) the numbers of basis functions $M_\mathbf{f}$ and $M_\mathbf{F}$, respectively ($M_\mathbf{f}, M_\mathbf{F} \ll N$).

IV. Linear mDRUR

With linear mappings $\mathbf{f}(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{a}$, $\mathbf{F}(\mathbf{y}) = \mathbf{B}\mathbf{y} + \mathbf{b}$, the mDRUR objective function is

$$E(\mathbf{Y}_0, \mathbf{X}, \mathbf{A}, \mathbf{a}, \mathbf{B}, \mathbf{b}) = \|\mathbf{Y} - \mathbf{A}\mathbf{X} - \mathbf{a}\mathbf{1}^T\|^2 + \lambda_\mathbf{f} \|\mathbf{A}\|^2 + \|\mathbf{X} - \mathbf{B}\mathbf{Y} - \mathbf{b}\mathbf{1}^T\|^2 + \lambda_\mathbf{F} \|\mathbf{B}\|^2 \quad (14)$$

where $\mathbf{1}$ is a column vector of N ones. If we partition matrices as $\mathbf{A} = \left(\begin{smallmatrix}\mathbf{A}_0 \\ \mathbf{A}_1\end{smallmatrix}\right)$ and $\mathbf{B} = (\mathbf{B}_0 \ \mathbf{B}_1)$, and vector $\mathbf{a}$ like $\mathbf{y}$, the gradient wrt $\mathbf{x}$ and $\mathbf{y}_0$ (for a given n) is:

$$\frac{\partial E}{\partial \mathbf{x}} = 2\left(-\mathbf{A}^T(\mathbf{y} - \mathbf{A}\mathbf{x} - \mathbf{a}) + \mathbf{x} - \mathbf{B}\mathbf{y} - \mathbf{b}\right) \quad (15)$$

$$\frac{\partial E}{\partial \mathbf{y}_0} = 2\left(-\mathbf{B}_0^T(\mathbf{x} - \mathbf{B}\mathbf{y} - \mathbf{b}) + \mathbf{y}_0 - \mathbf{A}_0\mathbf{x} - \mathbf{a}_0\right) \quad (16)$$

so equating it to zero yields a linear system for the optimum (or other stationary points):

$$\left(\mathbf{I} + \begin{pmatrix} \mathbf{A}^T\mathbf{A} & -\mathbf{A}_0^T - \mathbf{B}_0 \\ -\mathbf{A}_0 - \mathbf{B}_0^T & \mathbf{B}_0^T\mathbf{B}_0 \end{pmatrix}\right) \begin{pmatrix} \mathbf{x} \\ \mathbf{y}_0 \end{pmatrix} = -\begin{pmatrix} \mathbf{A}^T\mathbf{a} - \mathbf{b} - (\mathbf{A}_1^T + \mathbf{B}_1)\mathbf{y}_1 \\ \mathbf{B}_0^T\mathbf{B}_1\mathbf{y}_1 + \mathbf{B}_0^T\mathbf{b} - \mathbf{a}_0 \end{pmatrix}$$

which is analogous to the Gauss-Newton linear system, and non-sparse as well. This yields the projection-reconstruction step. As for the adaptation step, the linear regressions for $\mathbf{f}$ and $\mathbf{F}$ yield

$$\mathbf{A} = (\mathbf{Y} - \mathbf{a}\mathbf{1}^T)\mathbf{X}^T(\mathbf{X}\mathbf{X}^T + \lambda_\mathbf{f}\mathbf{I})^{-1}, \quad \mathbf{a} = \overline{\mathbf{y}} - \mathbf{A}\overline{\mathbf{x}}$$
$$\mathbf{B} = (\mathbf{X} - \mathbf{b}\mathbf{1}^T)\mathbf{Y}^T(\mathbf{Y}\mathbf{Y}^T + \lambda_\mathbf{F}\mathbf{I})^{-1}, \quad \mathbf{b} = \overline{\mathbf{x}} - \mathbf{B}\overline{\mathbf{y}}$$

where $\overline{\mathbf{x}} = \mathbf{X}\mathbf{1}/N$ and $\overline{\mathbf{y}} = \mathbf{Y}\mathbf{1}/N$. Both steps define a globally convergent fixed-point iteration. With no missing data and no regularization, the global minimum of E is PCA: $\mathbf{A} = \mathbf{B}^T = \mathbf{U}_L$ (leading eigenvectors of the covariance matrix of $\mathbf{Y}$), $\mathbf{a} = \overline{\mathbf{y}}$, $\mathbf{b} = \overline{\mathbf{x}}$, and $\mathbf{X} = \mathbf{U}_L^T\mathbf{Y} - \overline{\mathbf{x}}\mathbf{1}^T$. With missing data, E has local optima; adding regularization seems to reduce their effect.

V. Projecting and Reconstructing Test Data with Missing Values

Given a test point $\mathbf{y} \in \mathbb{R}^D$ with missing values, we propose to reconstruct $\mathbf{y}$ and project it as $\mathbf{F}(\mathbf{y})$ as follows. We consider an mDRUR training problem with augmented matrices $(\mathbf{Y}\ \mathbf{y})$ and $(\mathbf{X}\ \mathbf{x})$, minimizing over $\mathbf{x}$ and $\mathbf{y}_0$ and keeping everything else fixed ($\mathbf{Y}$, $\mathbf{y}_1$, $\mathbf{X}$, $\mathbf{f}$, $\mathbf{F}$). This way, the trained model does not change, but we do constrain $\mathbf{x}$ and $\mathbf{y}_0$ to conform to it. This results in minimizing the error function (7) (and we do not need $\mathbf{Y}$ and $\mathbf{X}$, just the mappings $\mathbf{f}$ and $\mathbf{F}$), and is thus formally identical to the project-reconstruct step over the nth point $(\mathbf{x}_n, \mathbf{y}_{n,0})$. Hence, we can use the same algorithm: Gauss-Newton for RBFs, or a single linear system for linear mappings (with the computational cost described earlier). We initialize $(\mathbf{x}, \mathbf{y}_0)$ to the pair $(\mathbf{x}_n, \mathbf{y}_n)$ for which $\mathbf{y}_n$ is closest to $\mathbf{y}$ in the present variables (in some applications the user may provide better initial values based on prior information). After minimizing (7), we discard $\mathbf{x}$ and return $\mathbf{y}$ as the reconstructed input, and $\mathbf{F}(\mathbf{y})$ as its low-dimensional projection. If there are many missing values, then (7) may have multiple minima corresponding to genuinely correct reconstructions of $\mathbf{y}$; which one is found depends on the initial pair. The approach carries over to the case where we are given a latent point $\mathbf{x}$ with missing values, but this is practically less useful. This approach is consistent with the existing mDRUR model in that, if the test point coincides with one of the training points (and has the same missing values) then its reconstruction, projection and $\mathbf{x}$ value coincide with those found during training (as follows from the fact that the error function was the same in both cases). Our approach seamlessly accomodates arbitrary patterns of missing values.

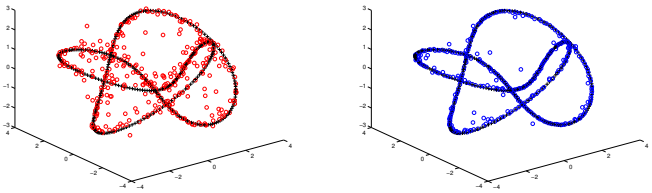

Figure 2. 100D trefoil data: recovered $\mathbf{Y}$ mapped back to 3D (original: black $+$, SVP: red $\circ$, mDRUR: blue $\circ$).

VI. Experiments

We apply mDRUR on missing data tasks with different scale, dimensionality, and characteristics. We assume the observed entries are uniformly distributed across $\mathbf{Y}$. While this may not represent well certain matrix completion tasks with other distributions (e.g. power-law [15]), it serves as a fair test bed for the various manifold learning algorithms we try. These are: the low-rank matrix completion model SVP, as linear baseline; the nonlinear PCA (nlPCA) of [6] (using their code), which is an unsupervised regression method using only $\mathbf{f}$ (a neural net); and mDRUR (with $\mathbf{Y}$ initialized from SVP). The results for linear mDRUR (not shown) were very similar to those of SVP, and consistently worse than mDRUR's over different proportions of missing data. We always report the result with best parameter choices for competing models (SVP: rank; nlPCA: number of latent variables and hidden nodes). We report the reconstruction error wrt the ground truth $\|\mathbf{Y}_{\text{rec}} - \mathbf{Y}\|$, where $\mathbf{Y}_{\text{rec}}$ is the $\mathbf{Y}$ reconstructed from partial observations.

A. Synthetic Example: 100D Trefoil Data

The original trefoil dataset has 377 points in 3D. We first embed them in a 100D space through a random linear mapping and obtain $\mathbf{Y}_{100\times 377}$, then we randomly select 7% entries from $\mathbf{Y}$ as observed (93% missing). For mDRUR, we use the first $L = 3$ principal components of the SVP reconstruction as initial $\mathbf{X}$, $M_{\mathbf{f}} = 50$, $M_{\mathbf{F}} = 10$, $\lambda_{\mathbf{f}} = \lambda_{\mathbf{F}} = 0.02$. Fig. 2 shows each method's resulting $\mathbf{Y}$ mapped back to the original 3D space. Besides learning the nonlinear manifold ($\mathbf{X}$ not shown), mDRUR's reconstruction drastically denoises the wandering points produced by SVP back to the 1D trefoil manifold. This is consistent with the reconstruction error: SVP 15.25, mDRUR 6.60.

B. MNIST Digit '7'

We consider all the 6 265 '7's in the MNIST dataset, and declare as missing 50% pixels selected uniformly. We rearrange each 28×28 image into a 784D vector (destroying their 2D structure), yielding high-dimensional data $\mathbf{Y}_{784\times 5000}$ for training and 1 265 testing images; each y_{dn} value is a grayscale in $[0, 256]$. Methods: SVP: rank 18; nlPCA: neural net, $12 \times 10 \times 784$ units (i.e., $L = 12$); mDRUR: $L = 9$, $M_{\mathbf{f}} = 1\,000$, $\lambda_{\mathbf{f}} = 0.01$, $M_{\mathbf{F}} = 50$, $\lambda_{\mathbf{F}} = 0.1$, initial $\mathbf{Y}$ from SVP, initial $\mathbf{X}$ from Laplacian eigenmaps on $\mathbf{Y}$. Unlike the trefoil data, where we know beforehand that the manifold is 1D and closed, the MNIST data might have several disconnected manifolds with different intrinsic dimensionality. Still, as shown in fig. 3, we achieve a fairly good image restoration by exploiting *only the manifold structure of the data* (since we removed the 2D image structure). Visually SVP and nlPCA do a decent job, but mDRUR is the clear winner, consistent with the error: SVP 51 000, nlPCA 97 000, mDRUR 45 000. Restoring a test image with mDRUR takes 0.48 s in a modern PC (of which 25% is spent in finding the initial $(\mathbf{x}_n, \mathbf{y}_n)$), with an average pixel error of 21 ± 56 grayscales (over 256). Again, the fact that the reconstructed images look like well-formed 7s (note in the bottom panel of fig. 3, second row, that some unusual strokes that were missing have been removed in the reconstruction) suggests that mDRUR has captured fundamental aspects of the manifold. This experiment also demonstrates the power of reconstructing $\mathbf{y}_0$ by making it a free parameter separate from $\mathbf{f}$ and $\mathbf{F}$ in mDRUR, rather than being the output of $\mathbf{f}$ (without $\mathbf{F}$) as in nlPCA: the latter reconstructions are visibly blurred and display artifacts. After all, it is impossible to reconstruct very accurately an arbitrary 784D '7' image from only 9 or 12 degrees of freedom. The role of $\mathbf{f}$ is to smooth, not interpolate, the (noisy) data.

In a similar experiment with a more obvious low-dimensional manifold structure, we created rotated versions of an MNIST digit '3' every 4 degrees (a closed 1D nonlinear manifold in 784D) and made 40% pixels randomly missing. The errors for SVP (rank 6), nlPCA ($L = 2$) and mDRUR ($L = 2$) were 29.1, 34.9 and 20.8, respectively.

VII. Conclusion

We have shown how to learn a Dimensionality Reduction by Unsupervised Regression model for high-dimensional data with missing values, where knowledge about the high-dimensional data constrains the low-dimensional representation, and vice versa. By introducing parameters $(\mathbf{f}, \mathbf{F})$ and $\mathbf{Y}_0$, we separate two distinct roles that previous algorithms conflated into one: learning a nonlinear, low-dimensional representation of the data, and recovering the missing data. This idea applies in a unified way to training and testing, and to arbitrary patterns of missing data. Our work also provides a new approach to the problem of matrix completion, by reconstructing the data subject to implicit, local low-rank and smoothness constraints. mDRUR improves over other dimensionality reduction and matrix completion methods, and performs well with large proportions of missing data, even if few (or any) vectors are complete. Nonlinear manifold learning with missing data is a difficult and overlooked problem. Our improved results make a case for going beyond linear and spectral methods.

Acknowledgments

Work supported by NSF CAREER award IIS–0754089.

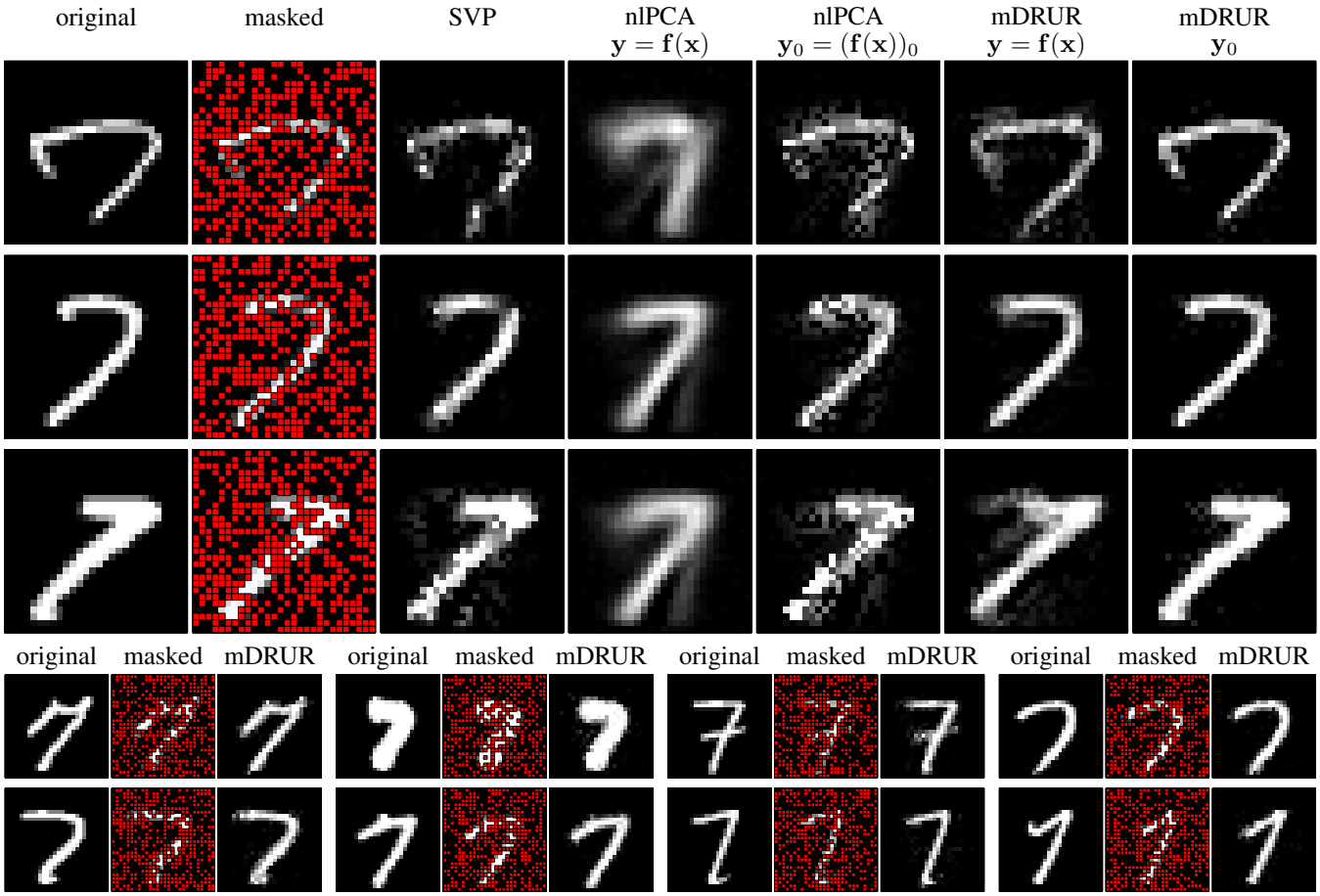

Figure 3. Restoring MNIST digits '7'. *Upper panel*: 3 training images for all methods. *Lower panel*: 8 test images with mDRUR. "Masked" shows the 50% missing pixels in red (you may need to zoom in). Grayscales clipped to $[0, 256]$. "$\mathbf{y} = \mathbf{f}(\mathbf{x})$" means reconstructing *all* pixels (missing or not) with the output of $\mathbf{f}$. "$\mathbf{y}_0 = (\mathbf{f}(\mathbf{x}))_0$" means reconstructing with the output of $\mathbf{f}$ *only* the missing pixels and leaving the rest to their present values. For mDRUR, "$\mathbf{y}_0$" means reconstructing *only* the missing pixels with the free $\mathbf{y}_0$ parameters.

REFERENCES

[1] S. Faridani, E. Bitton, K. Ryokai, and K. Goldberg, "Opinion space: A scalable tool for browsing online comments," in *CHI*, 2010, pp. 1175–1184.

[2] B. S. Everitt, *An Introduction to Latent Variable Models*, Chapman & Hall, 1984.

[3] C. M. Bishop, M. Svensén, and C. K. I. Williams, "GTM: The generative topographic mapping," *Neural Computation*, vol. 10, no. 1, pp. 215–234, Jan. 1998.

[4] M. Á. Carreira-Perpiñán and Z. Lu, "The Laplacian Eigenmaps Latent Variable Model," in *AISTATS*, 2007, pp. 59–66.

[5] A. Gifi, *Nonlinear Multivariate Analysis*. Wiley, 1990.

[6] M. Scholz, F. Kaplan, C. L. Guy, J. Kopka, and J. Selbig, "Non-linear PCA: A missing data approach," *Bioinformatics*, vol. 21, no. 20, pp. 3887–3895, Oct. 15 2005.

[7] N. D. Lawrence and R. Urtasun, "Non-linear matrix factorization with Gaussian processes," in *ICML*, 2009.

[8] M. Á. Carreira-Perpiñán and Z. Lu, "Dimensionality reduction by unsupervised regression," in *CVPR*, 2008.

[9] R. G. Baraniuk and M. B. Wakin, "Random projections of smooth manifolds," *Foundations of Computational Mathematics*, vol. 9, no. 1, pp. 51–77, Feb. 2009.

[10] M. Chen, J. Silva, J. Paisley, C. Wang, D. Dunson, and L. Carin, "Compressive sensing on manifolds using a nonparametric mixture of factor analyzers: Algorithm and performance bounds," *IEEE Trans. Sig. Proc.*, vol. 58, 2010.

[11] P. Jain, R. Meka, and I. S. Dhillon, "Guaranteed rank minimization via singular value projection," in *NIPS*, 2010.

[12] W. Wang, M. Á. Carreira-Perpiñán, and Z. Lu, "A denoising view of matrix completion," in *NIPS*, 2011.

[13] M. Á. Carreira-Perpiñán and Z. Lu, "Parametric dimensionality reduction by unsupervised regression," in *CVPR*, 2010.

[14] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed., Springer-Verlag, 2006.

[15] R. Meka, P. Jain, and I. Dhillon, "Matrix completion from power-law distributed samples," in *NIPS*, 2009.