MERL – A MITSUBISHI ELECTRIC RESEARCH LABORATORY
http://www.merl.com

Incremental singular value decomposition of uncertain data with missing values

Matthew Brand

TR-2002-24 May 2002

Abstract

We introduce an incremental singular value decomposition (SVD) of incomplete data. The SVD is developed as data arrives, and can handle arbitrary missing/untrusted values, correlated uncertainty across rows or columns of the measurement matrix, and user priors. Since incomplete data does not uniquely specify an SVD, the procedure selects one having minimal rank. For a dense $p \times q$ matrix of low rank r, the incremental method has time complexity $O(pqr)$ and space complexity $O((p+q)r)$—better than highly optimized batch algorithms such as MATLAB's `svd()`. In cases of missing data, it produces factorings of lower rank and residual than batch SVD algorithms applied to standard missing-data imputations. We show applications in computer vision and audio feature extraction. In computer vision, we use the incremental SVD to develop an efficient and unusually robust subspace-estimating flow-based tracker, and to handle occlusions/missing points in structure-from-motion factorizations.

First circulated spring 2001.

Proceedings of the 2002 European Conference on Computer Vision (ECCV2002 Copenhagen), Springer Lecture Notes in Computer Science volume 2350.

Incremental singular value decomposition of uncertain data with missing values

Matthew Brand

Mitsubishi Electric Research Labs, 201 Broadway, Cambridge 02139 MA, USA
To appear, Proc. European Conference on Computer Vision (ECCV), May 2002.

Abstract. We introduce an incremental singular value decomposition (SVD) of incomplete data. The SVD is developed as data arrives, and can handle arbitrary missing/untrusted values, correlated uncertainty across rows or columns of the measurement matrix, and user priors. Since incomplete data does not uniquely specify an SVD, the procedure selects one having minimal rank. For a dense $p \times q$ matrix of low rank r, the incremental method has time complexity $O(pqr)$ and space complexity $O((p+q)r)$—better than highly optimized batch algorithms such as MATLAB's `svd`(). In cases of missing data, it produces factorings of lower rank and residual than batch SVD algorithms applied to standard missing-data imputations. We show applications in computer vision and audio feature extraction. In computer vision, we use the incremental SVD to develop an efficient and unusually robust subspace-estimating flow-based tracker, and to handle occlusions/missing points in structure-from-motion factorizations.

1 Introduction

Many natural phenomena can be faithfully modeled with multilinear functions, or closely approximated as such. Examples include the combination of lighting and pose [20] and shape and motion [12,3] in image formation, mixing of sources in acoustic recordings [6], and word associations in collections of documents [1,23]. Multilinearity means that a matrix of such a phenomenon's measured effects can be factored into low-rank matrices of (presumed) causes. The celebrated singular value decomposition (SVD) [8] provides a bilinear factoring of a data matrix $\mathbf{M}$,

$$\mathbf{U}_{p\times r}\,\mathrm{diag}(\mathbf{s}_{r\times 1})\mathbf{V}^{\top}_{r\times q} \overset{\mathrm{SVD}r}{\longleftarrow} \mathbf{M}_{p\times q}, \quad r \leq \min(p,q) \tag{1}$$

where $\mathbf{U}$ and $\mathbf{V}$ are unitary orthogonal matrices whose columns give a linear basis for $\mathbf{M}$'s columns and rows, respectively. For low-rank phenomena, $r_{\text{true}} \ll \min(p,q)$, implying a parsimonious explanation of the data. Since r_{true} is often unknown, it is common to wastefully compute a large $r_{\text{approx}} \gg r_{\text{true}}$ SVD and estimate an appropriate smaller value $r_{\text{empirical}}$ from the distribution of singular values in $\mathbf{s}$. All but $r_{\text{empirical}}$ of the smallest singular values in $\mathbf{s}$ are then zeroed to give a "thin" truncated SVD that closely approximates the data. This forms the basis of a broad range of algorithms for data analysis, dimensionality reduction, compression, noise-suppression, and extrapolation.

The SVD is usually computed by a batch $O(pq^2 + p^2q + q^3)$ time algorithm [8], meaning that all the data must be processed at once, and SVDs of very large datasets are essentially unfeasible. Lanczos methods yield thin SVDs in $O(pqr^2)$ time [8], but r_{true} should be known in advance since Lanczos methods are known to be inaccurate for the smaller singular values [1]. A more pressing problem is that the SVD requires *complete* data, whereas in many experimental settings some parts of the measurement matrix may be missing, contaminated, or otherwise untrusted. Consequently, a single missing value forces the modeler to discard an entire row or column of the data matrix prior to the SVD. The missing value may be imputed from neighboring values, but such imputations typically mislead the SVD away from the most parsimonious (low-rank) decompositions.

We consider how an SVD may be updated by adding rows and/or columns of data, which may be missing values and/or contaminated with correlated (colored) noise. The size of the data matrix need not be known: The SVD is developed as the data comes in and handles missing values in a manner that minimizes rank. The resulting algorithms have better time and space complexity than full-data batch SVD methods and can produce more informative results (more parsimonious factorings of incomplete data). In the case of dense low-rank matrices, the time complexity is linear in the size and the rank of the data—$O(pqr)$—while the space complexity is sublinear—$O((p+q)r)$.

2 Related work

SVD updating has a literature spread over three decades [5,4,1,10,7,23] and is generally based on Lanczos methods, symmetric eigenvalue perturbations, or identities similar to equation 2 below. Zha and Simon [23] use such an identity but their update is approximate and requires a dense SVD. Chandrasekaran et alia [7] begin similarly but their update is limited to single vectors and is vulnerable to loss of orthogonality. Levy and Lindenman [14] exploit the relationship between the QR-decomposition and the SVD to incrementally compute the left singular vectors in $O(pqr^2)$ time; if p, q, and r are known in advance and $p \gg q \gg r$, then the expected complexity falls to $O(pqr)$. However, this is also vulnerable to loss of orthogonality and results have only been reported for matrices having a few hundred columns.

None of this literature contemplates missing or uncertain values, except insofar as they can be treated as zeros (e.g., [1]), which is arguably incorrect. In batch-SVD contexts, missing values are usually handled via subspace imputation, using an expectation-maximization-like procedure: Perform an SVD of all complete columns, regress incomplete columns against the SVD to estimate missing values, then re-factor and re-impute the completed data until a fixpoint is reached (e.g., [21]). This is extremely slow (quartic time) and only works if very few values are missing. It has the further demerit that the imputation does not minimize effective rank. Other heuristics simply fill missing values with row- or column-means [19].

In the special case where a matrix $\mathbf{M}$ is nearly dense, its normalized scatter matrix $\Sigma_{m,n} \doteq \langle \mathbf{M}_{i,m}\mathbf{M}_{i,n}\rangle_i$ may be fully dense due to fill-in. In that case Σ's eigenvectors are $\mathbf{M}$'s right singular vectors [13]. However, this method does not lead to the left singular vectors, and it often doesn't work at all because Σ is frequently incomplete as well, with undefined eigenvectors.

3 Updating an SVD

We begin with an existing rank-r SVD as in equation 1. We have a matrix $\mathbf{C}_{p\times c}$ whose columns contain additional multivariate measurements. Let $\mathbf{L} \doteq \mathbf{U}\backslash\mathbf{C} = \mathbf{U}^\top\mathbf{C}$ be the projection of $\mathbf{C}$ onto the orthogonal basis $\mathbf{U}$, also known as its "eigen-coding." Let $\mathbf{H} \doteq (\mathbf{I} - \mathbf{U}\mathbf{U}^\top)\mathbf{C} = \mathbf{C} - \mathbf{U}\mathbf{L}$ to be the component of $\mathbf{C}$ orthogonal to the subspace spanned by $\mathbf{U}$. ($\mathbf{I}$ is the identity matrix.) Finally, let $\mathbf{J}$ be an orthogonal basis of $\mathbf{H}$ and let $\mathbf{K} \doteq \mathbf{J}\backslash\mathbf{H} = \mathbf{J}^\top\mathbf{H}$ be the projection of $\mathbf{C}$ onto the subspace orthogonal to $\mathbf{U}$. For example, $\mathbf{J}\mathbf{K} \xleftarrow{\text{QR}} \mathbf{H}$ could be a QR-decomposition of $\mathbf{H}$. Consider the following identity:

$$\begin{aligned}\begin{bmatrix}\mathbf{U}\ \mathbf{J}\end{bmatrix}\begin{bmatrix}\text{diag}(\mathbf{s}) & \mathbf{L}\\ 0 & \mathbf{K}\end{bmatrix}\begin{bmatrix}\mathbf{V}\ 0\\ 0\ \mathbf{I}\end{bmatrix}^\top &= \begin{bmatrix}\mathbf{U}\ (\mathbf{I}-\mathbf{U}\mathbf{U}^\top)\mathbf{C}/\mathbf{K}\end{bmatrix}\begin{bmatrix}\text{diag}(\mathbf{s}) & \mathbf{U}^\top\mathbf{C}\\ 0 & \mathbf{K}\end{bmatrix}\begin{bmatrix}\mathbf{V}\ 0\\ 0\ \mathbf{I}\end{bmatrix}^\top\\ &= \begin{bmatrix}\mathbf{U}\,\text{diag}(\mathbf{s})\mathbf{V}^\top\ \mathbf{C}\end{bmatrix} \;=\; \begin{bmatrix}\mathbf{M}\ \mathbf{C}\end{bmatrix}\end{aligned} \tag{2}$$

Like an SVD, the left and right matrices in the product are unitary and orthogonal. The middle matrix, which we denote $\mathbf{Q}$, is diagonal with a c-column border. To update the SVD we must diagonalize $\mathbf{Q}$. Let

$$\mathbf{U}'\,\text{diag}(\mathbf{s}')\mathbf{V}'^\top \xleftarrow{\text{SVD}} \mathbf{Q} \tag{3}$$

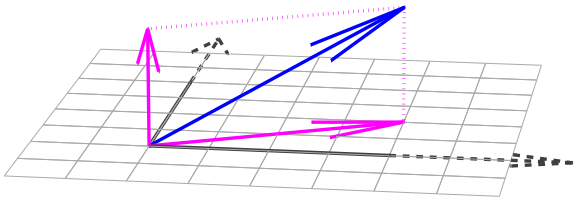

Fig. 1. A vector is decomposed into components within and orthogonal to an SVD-derived subspace. The parallel component causes the singular vectors to be rotated (see figure 2), while the orthogonal component increases the rank of the SVD.

$$\mathbf{U}'' \leftarrow \begin{bmatrix}\mathbf{U} & \mathbf{J}\end{bmatrix}\mathbf{U}'; \quad \mathbf{s}'' \leftarrow \mathbf{s}'; \quad \mathbf{V}'' \leftarrow \begin{bmatrix}\mathbf{V} & 0\\ 0 & \mathbf{I}\end{bmatrix}\mathbf{V}' \tag{4}$$

Then the updated SVD is

$$\mathbf{U}''\,\mathrm{diag}(\mathbf{s}'')\mathbf{V}''^{\top} = \begin{bmatrix}\mathbf{U}\,\mathrm{diag}(\mathbf{s})\mathbf{V}^{\top} & \mathbf{C}\end{bmatrix} = \begin{bmatrix}\mathbf{M} & \mathbf{C}\end{bmatrix}. \tag{5}$$

The whole update procedure takes $O((p+q)r^2 + pc^2)$ time[1], spent mostly in the subspace rotations of equation 4. To add rows one simply swaps $\mathbf{U}$ for $\mathbf{V}$ and $\mathbf{U}''$ for $\mathbf{V}''$.

In practice, some care must be taken to counter numerical error that may make $\mathbf{J}$ and $\mathbf{U}$ not quite orthogonal. We found that applying modified Gram-Schmidt orthogonalization to $\mathbf{U}$ when the inner product of its first and last columns is more than some small ε away from zero makes the algorithm numerically robust. A much more efficient scheme will be developed below.

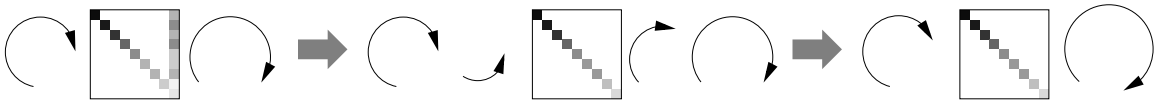

Fig. 2. Visualization of the SVD update in equation 2. The quasi-diagonal $\mathbf{Q}$ matrix at left is diagonalized and the subspaces are counter-rotated to preserve equality.

3.1 Automatic truncation

Define $v \doteq \sqrt{\det(\mathbf{K}^{\top}\mathbf{K})}$, which is the volume of $\mathbf{C}$ that is orthogonal to $\mathbf{U}$. If $v < \varepsilon$ for some small ε near the limits of machine precision, then $\mathbf{J}$ must have zero norm, since there is no orthogonal component (else $\mathbf{J}$

[1] An SVD of an $r \times r$ matrix would ordinarily take $O(r^3)$ time but since $\mathbf{Q}$ is a c-bordered diagonal matrix, it can be rotated into bidiagonal form in $O(cr^2)$ time [22], and thence diagonalized in an $O(r^2)$ time bidiagonal SVD [9]. If $c = 1$, the eigenvalues $\mathbf{s}'^2$ and eigenvectors $\mathbf{U}'$ of arrowhead matrix $\mathbf{Q}^{\top}\mathbf{Q}$ can be computed in $O(r^2)$ time [17]; the remaining singular vectors can also be recovered in $O(r^2)$.

is contaminated by numerical or measurement noise). In this case the noise should be suppressed by setting $\mathbf{K} \leftarrow 0$ prior to the SVD in equation 3. Since the resulting SVD will have r rather than $r+1$ singular values, equation 4 can be replaced with the truncated forms

$$\mathbf{U}'' \leftarrow \mathbf{U}\mathbf{U}'_{1:r,1:r}; \quad \mathbf{s}'' \leftarrow \mathbf{s}'_{1:r}; \quad \mathbf{V}'' \leftarrow \mathbf{V}'_{:,1:r}. \tag{6}$$

This automatically sizes the SVD is to the effective rank of the data matrix.

To explicitly suppress measurement noise, one truncates the completed update to suppress singular values below a noise threshold, derived from the user's knowledge of noise levels in the measurements.

The update procedure enables online SVDs and SVDs of datasets whose size defeats non-incremental SVD algorithms. The update can be used to add individual vectors, batches of vectors, or to merge SVDs from partitions of the data. We will now concentrate on the vector update and leverage it into linear-time and missing-value SVD algorithms.

4 Fast incremental SVD of low-rank matrices

A useful special case is when $\mathbf{c} = \mathbf{C}$ is a single column vector, for which scalar $k = \mathbf{K} = \|\mathbf{c} - \mathbf{U}\mathbf{U}^\top\mathbf{c}\|$ and vector $\mathbf{j} = \mathbf{J} = (\mathbf{c} - \mathbf{U}\mathbf{U}^\top\mathbf{c})/k$ can be computed very quickly[2]. To compute a full SVD by adding rows and/or columns, we take the first measurement $\mathbf{m} \in \mathbf{M}$ and set $\mathbf{s} \leftarrow \|\mathbf{m}\|$, $\mathbf{U} \leftarrow \mathbf{m}/\|\mathbf{m}\|$, $\mathbf{V} \leftarrow 1$. Then we iterate the update procedure above with truncation. The total procedure takes $O(pqr^2)$ time, which is essentially linear in the number of elements in $\mathbf{M}$. This can be substantially faster than the $O(pq^2)$ time of a batch SVD when the rank $r \ll q$. The advantage over the Lanczos methods is that we now have an online algorithm whose in-memory storage requirements are reduced from $O(pq)$—the size of the data—to $O(r(p+q+r))$—the size of the results.

4.1 Preserving orthogonality and reducing complexity

Because $\mathbf{U}$ and $\mathbf{V}$ are tall thin matrices, repeatedly rotating their column spaces makes loss of orthogonality through numerical error an issue. Instead of updating large matrices, we may keep $\mathbf{U}, \mathbf{V}, \mathbf{U}', \mathbf{V}'$ separate and only update the small matrices $\mathbf{U}', \mathbf{V}'$, with $\mathbf{U}$ and $\mathbf{V}$ growing strictly by appends.

In this fastest incarnation of SVD updating, we build an extended SVD,

$$\mathbf{U}_{p\times r}\mathbf{U}'_{r\times r}\,\mathrm{diag}(\mathbf{s}_{r\times 1})\mathbf{V}'^{\top}_{r\times r}\mathbf{V}^{\top}_{q\times r} \xleftarrow{\mathrm{SVD}_r} \mathbf{M}, \tag{7}$$

with orthogonal $\mathbf{U}$, $\mathbf{U}'$, $\mathbf{U}\mathbf{U}'$ and $\mathbf{V}\mathbf{V}'$. The large outer matrices are built by appending columns to $\mathbf{U}$ and rows to $\mathbf{V}$, while rotations of the subspace are handled by transforms of the much smaller $\mathbf{U}', \mathbf{V}'$ matrices. This makes the update much faster and more robust to numerical error: Let $\mathbf{Q}$ and $\mathbf{j}$ be defined as above, and let $\mathbf{A}, \mathbf{B}$ diagonalize $\mathbf{Q}$ as $\mathbf{A}\,\mathrm{diag}(\mathbf{s})\mathbf{B}^\top \xleftarrow{\mathrm{SVD}} \mathbf{Q}$. The left-hand side is updated $\mathbf{U}' \leftarrow \mathbf{U}'\mathbf{A}$ when the rank does not increase, otherwise $\mathbf{U}' \leftarrow \left[\begin{smallmatrix}\mathbf{U}' & 0\\ 0 & 1\end{smallmatrix}\right]\mathbf{A}$ and $\mathbf{U} \leftarrow [\mathbf{U}, \mathbf{j}]$. Due to its small size, $\mathbf{U}'$ loses orthogonality very slowly. Numerical error can be contained by occasionally re-orthogonalizing $\mathbf{U}'$ via modified Gram-Schmidt when the inner product of its oldest (first) column with its newest (last) column is more than some small ε away from zero.

[2] In practice, some care should be taken to order operations in computing k to get the most accurate result from floating point machines. We use $k \leftarrow \mathbf{c}^\top\mathbf{c} - 2\mathbf{L}^\top\mathbf{L} + (\mathbf{U}\mathbf{L})^\top(\mathbf{U}\mathbf{L})$.

The right side is somewhat more complicated because we are adding rows to $\mathbf{V}$ but must guarantee that $\mathbf{V}\mathbf{V}'$ is orthogonal. To do so, we will also have to calculate and update the pseudo-inverse $\mathbf{V}'^{+}$. Let r be the rank of the SVD prior to the update. When the rank increases, the right-hand side update is simply

$$\mathbf{V} \leftarrow \left[\begin{smallmatrix}\mathbf{V} & 0\\ 0 & 1\end{smallmatrix}\right], \quad \text{then } \mathbf{V}' \leftarrow \left[\begin{smallmatrix}\mathbf{V}' & 0\\ 0 & 1\end{smallmatrix}\right]\mathbf{B}, \quad \text{then } \mathbf{V}'^{+} \leftarrow \mathbf{B}^{\top}\left[\begin{smallmatrix}\mathbf{V}'^{+} & 0\\ 0 & 1\end{smallmatrix}\right]. \tag{8}$$

When the rank does not increase, we split $\mathbf{B} \rightarrow \left[\begin{smallmatrix}\mathbf{W}\\ \mathbf{w}\end{smallmatrix}\right]$ where matrix $\mathbf{W} \doteq \mathbf{B}_{(1:r,1:r)}$ is a linear transform that will be applied to $\mathbf{V}'$, and row-vector $\mathbf{w} \doteq \mathbf{B}_{(r+1,1:r)}$ is the eigen-space encoding of the new data vector. The update is

$$\mathbf{V}' =\leftarrow \mathbf{V}'\mathbf{W}, \quad \text{then } \mathbf{V}'^{+} \leftarrow \mathbf{W}^{+}\mathbf{V}'^{+}, \quad \text{then } \mathbf{V} \leftarrow \left[\begin{smallmatrix}\mathbf{V}\\ \mathbf{V}'^{+}\mathbf{w}\end{smallmatrix}\right]. \tag{9}$$

It can be verified algebraically that $\mathbf{V}_{new}\mathbf{V}'_{new}$ is identical to the first r columns of $\left[\begin{smallmatrix}\mathbf{V}_{old}\mathbf{V}'_{old} & 0\\ 0 & 1\end{smallmatrix}\right]\mathbf{B}$.

Remarkably, $\mathbf{W}^{+}$ can be computed in $O(r^2)$ time using the identity $\mathbf{W}^{+} = (\mathbf{I} + \mathbf{w}^{\top}\mathbf{w}/(1-\mathbf{w}\mathbf{w}^{\top}))\mathbf{W}^{\top}$ (when $\left[\begin{smallmatrix}\mathbf{W}\\ \mathbf{w}\end{smallmatrix}\right]^{\top}\left[\begin{smallmatrix}\mathbf{W}\\ \mathbf{w}\end{smallmatrix}\right] = \mathbf{I}$; see appendix 1 for the proof) . This can be restructured to eliminate the $O(r^3)$ matrix-matrix product in favor of an $O(r^2)$ vector-vector outer product:

$$\mathbf{W}^{+} = \mathbf{W}^{\top} + (\mathbf{w}^{\top}/(1-\mathbf{w}\mathbf{w}^{\top})))(\mathbf{w}\mathbf{W}^{\top}). \tag{10}$$

This eliminates the costliest steps of the update—rotation and re-orthogonalization of $\mathbf{U}, \mathbf{V}$—and requires that we only keep $\mathbf{U}'$ orthogonal. The time complexity falls to $O(pr^2)$ for the r rank-increasing updates and $O(pr + r^3)$ for the $q - r$ non rank-increasing updates, with an overall complexity of $O(pqr + qr^3) = O(pqr)$, assuming that the rank is small relative to the dimensionality of the samples, specifically $r = O(\sqrt{p})$. For a high-dimensional low-rank matrices, we effectively have a *linear-time* SVD algorithm.

4.2 Subspace tracking

For nonstationary data streams, the best we can do is track an evolving subspace $\mathbf{U}$. In the incremental SVD, this is neatly and inexpensively accomplished between updates by decaying the singular values $\mathbf{s} \leftarrow \gamma\mathbf{s}$; $0 < \gamma < 1$. All updates of $\mathbf{V}$ are simply dropped.

5 Missing data

Consider adding a vector $\mathbf{c}$ with missing values. In our implementation, these are indicated by setting entries in $\mathbf{c}$ to the IEEE754 floating point value *NaN* (not-a-number). Partition $\mathbf{c}$ into $\mathbf{c}_\bullet$ and $\mathbf{c}_\circ$, vectors of the known and unknown values in $\mathbf{c}$, respectively, and let $\mathbf{U}_\bullet, \mathbf{U}_\circ$ be the corresponding rows of $\mathbf{U}$. Imputation of the missing values via the normal equation

$$\hat{\mathbf{c}}_\circ \leftarrow \mathbf{U}_\circ \operatorname{diag}(\mathbf{s})(\operatorname{diag}(\mathbf{s})\mathbf{U}_\bullet^{\top}\mathbf{U}_\bullet \operatorname{diag}(\mathbf{s}))^{+}(\operatorname{diag}(\mathbf{s})\mathbf{U}_\bullet^{\top}\mathbf{c}_\bullet) = \mathbf{U}_\circ \operatorname{diag}(\mathbf{s})(\mathbf{U}_\bullet \operatorname{diag}(\mathbf{s}))^{+}\mathbf{c}_\bullet, \tag{11}$$

yields the completed vector $\hat{\mathbf{c}}$ that lies the fewest standard deviations from the origin, with respect to the density of data seen thus far ($\mathbf{X}^{+}$ denotes pseudo-inverse). Substituting equation 11 into the $\mathbf{Q}$ matrix yields

$$\mathbf{Q} = \begin{bmatrix}\operatorname{diag}(\mathbf{s}) & \mathbf{U}^{\top}\hat{\mathbf{c}}\\ 0 & k\end{bmatrix} = \begin{bmatrix}\operatorname{diag}(\mathbf{s}) & \operatorname{diag}(\mathbf{s})(\mathbf{U}_\bullet \operatorname{diag}(\mathbf{s}))^{+}\mathbf{c}_\bullet\\ 0 & \|\mathbf{c}_\bullet - \mathbf{U}_\bullet \operatorname{diag}(\mathbf{s})(\mathbf{U}_\bullet \operatorname{diag}(\mathbf{s}))^{+}\mathbf{c}_\bullet)\|\end{bmatrix}, \tag{12}$$

where $\mathbf{U}^{\top}\hat{\mathbf{c}}$ is the projection of the vector onto the left singular vectors and k is the distance of the vector to that subspace. As one might expect, with missing data it is rare that $k > 0$. In the worst case, imputation raises the per-update complexity to $O(pr^3)$, but we find in practice that the per-update run time stays closer to $O(pr^2)$, because with missing data the pseudo-inverse problem tends to be small and thus dominated by the problem of rediagonalizing $\mathbf{Q}$.

5.1 Minimizing rank growth

The importance of the imputation in equation 11 is that it minimizes k. We show here that this in turn controls the effective rank of the updated SVD:

Theorem 1. *Minimizing k maximizes concentration of variance in the top singular values.*

Proof. Denoting the pre-update singular values as $s_i \in \mathbf{s}$, elements of the $\mathbf{Q}$ matrix as $Q_{i,j} \in \mathbf{Q}$, and the post-update singular values of $\mathbf{Q}$ as σ_i, we compute the determinant of the new singular value matrix:

$$k \prod_i^r s_i^2 = \prod_i^{r+1} Q_{i,i}^2 = \det(\mathbf{Q}^\top \mathbf{Q}) = \prod_i^{r+1} \sigma_i^2 = \exp 2 \sum_i^{r+1} \log \sigma_i. \tag{13}$$

The second equality follows from the special sparsity structure of $\mathbf{Q}$. This shows that minimizing k is equivalent to minimizing the log-volume ($\sum_i \log \sigma_i$) of the post-update singular value matrix, which is half the log-volume of the completed data's scatter matrix. Since the amount of total variance in the singular value matrix is lower-bounded by the variance in the known data values, by the log-sum inequality, the only way to minimize the log-volume is to concentrate the variance in a few dominant singular values[3]. Consequently equation 11 minimizes growth of the effective rank in the updated SVD. QED.

In a related forthcoming paper, we show how these methods can be extended to rapidly factor very large matrices (e.g. 5000×5000) in which more that 95% of the elements are missing. In such cases the minimal rank growth property plays a very important role in guaranteeing a parsimonious model of the data. We show that this translates into considerable improvements over the state-of-the-art in genetic classification and econometric prediction tasks.

6 Uncertainty, priors, and posteriors

In experimental settings the columns of $\mathbf{M}$ are uncertain in the sense that they are samples from a distribution. When the distribution is gaussian and its covariance Σ is known (often the case in vision), the eigenbasis $\Omega\Lambda\Omega^\top \xleftarrow{\text{eig}} \Sigma$ enables a directionally weighted least-squares solution for the SVD that maximizes the likelihood $p(\mathbf{M}|\mathbf{U},\mathbf{S},\mathbf{V}) \propto e^{-\text{trace}(\mathbf{R}^\top \Sigma^{-1} \mathbf{R})}$ with respect to reconstruction residual $\mathbf{R} \doteq \mathbf{U}\,\text{diag}(\mathbf{s})\mathbf{V}^\top - \mathbf{M}$. Let $\mathbf{R}' \doteq \Lambda^{-1/2}\Omega^\top \mathbf{R}$. Then $\text{trace}(\mathbf{R}'^\top \mathbf{R}') = \text{trace}(\mathbf{R}^\top \Sigma^{-1} \mathbf{R})$, which is to say that the left-handed *certainty warp* $\Lambda^{-1/2}\Omega^\top$ rotates and scales $\mathbf{M}$ to make its uncertainty or noise model gaussian i.i.d. Therefore the problem can be solved as

$$\mathbf{U}'\,\text{diag}(\mathbf{s}')\mathbf{V}'^\top \xleftarrow{\text{SVD}} \Lambda^{-1/2}\Omega^\top \mathbf{M} \tag{14}$$

$$\mathbf{U}\,\text{diag}(\mathbf{s})\mathbf{V}''^\top \xleftarrow{\text{SVD}} \Omega\Lambda^{1/2}\mathbf{U}'\,\text{diag}(\mathbf{s}'); \qquad \mathbf{V} \leftarrow \mathbf{V}'\mathbf{V}'' \,. \tag{15}$$

Equation 14 is an SVD in the i.i.d. certainty-warped space. The product in equation 15 unwarps[4] the results and its $O(pr^2)$ SVD restores orthogonality to the unwarped results $\mathbf{U}\,\text{diag}(\mathbf{s})\mathbf{V}^\top$.

[3] Other imputation schemes minimize the entire rightmost column in $\mathbf{Q}$ (containing both the projection and the distance) thereby minimizing the trace norm $\text{trace}(\mathbf{Q}^\top \mathbf{Q}) = \sum_i^{r+1} \sigma_i^2$, which actually encourages the spread of variance over many singular values.

[4] Irani & Anandan [12] developed certainty warps for SVD but do not consider unwarping and leave the result "determined up to an unknown affine transformation."

A gaussian subspace prior $p_{\mathbf{U}}(\mathbf{U},\mathbf{S},\mathbf{V}) \propto e^{-\operatorname{trace}((\mathbf{U}-\mu_{\mathbf{U}})^\top \Sigma_{\mathbf{U}}^{-1}(\mathbf{U}-\mu_{\mathbf{U}}))}$ can be accommodated as another least-squares constraint simply by appending its own certainty-warped mean to the certainty-warped data columns, provided that all are rotated back into the same coordinate frame. equations 14–15 then become

$$\mathbf{U}'\operatorname{diag}(\mathbf{s}')\mathbf{V}'^\top \xleftarrow{\text{SVD}} [\Omega\Lambda^{-1/2}\Omega^\top\mathbf{M},\ \ \Omega_{\mathbf{U}}\Lambda_{\mathbf{U}}^{-1/2}\Omega_{\mathbf{U}}^\top\mu_{\mathbf{U}}] \tag{16}$$

$$\mathbf{V}''\mathbf{F} \xleftarrow{\text{QR}} \mathbf{V}'_{1:p,:};\ \ \mathbf{U}\operatorname{diag}(\mathbf{s})\mathbf{V}''' \xleftarrow{\text{SVD}} \Omega\Lambda^{1/2}\Omega^\top\mathbf{U}'\operatorname{diag}(\mathbf{s}')\mathbf{F}^\top;\ \ \mathbf{V} \leftarrow \mathbf{V}''\mathbf{V}''' \tag{17}$$

equation 16 calculates an SVD of data and prior, both warped into i.i.d. space. equation 17 drops the added column from $\mathbf{V}'^\top$ and reorthogonalizes in an $O(pq)$ QR downdate, then unwarps $\mathbf{U}'$ and reorthogonalizes in $O(pr^2)$ time to yield the maximum *a posteriori* (MAP) estimate.

In an incremental setting, each data vector must be warped according to $\mathbf{c}' \leftarrow \Lambda^{-1/2}\Omega^\top\mathbf{c}$ (or $\mathbf{c}' \leftarrow \Omega\Lambda^{-1/2}\Omega^\top\mathbf{c}$ if there is a prior) before it is incorporated into the SVD. After the last update, the final SVD is unwarped using equation 15 or equation 17. When the new vector $\mathbf{c}$ is both incomplete and uncertain, it cannot be warped until the missing elements are imputed. First we must *unwarp* the basis $\mathbf{U}$ to perform the imputation,

$$\hat{\mathbf{c}}_\circ \leftarrow (\Omega\Lambda^{1/2}\mathbf{U})_\circ(\Omega\Lambda^{1/2}\mathbf{U})_\bullet \backslash \mathbf{c}_\bullet), \tag{18}$$

then warp the completed $\hat{\mathbf{c}}$ and incorporate it into the SVD. Unless the covariance is diagonal, certainty warps require column-wise updating with at least some complete columns.

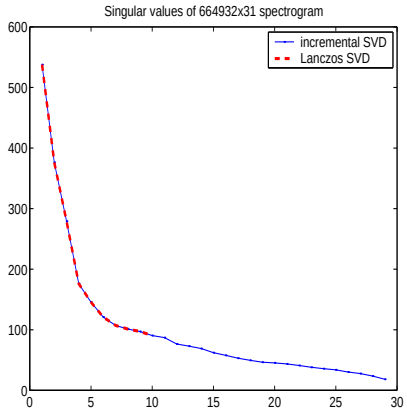

Fig. 3. Singular values from a very large SVD of dense audio data agree with the available Lanczos estimates.

7 Example applications

7.1 Eigen-coding

To test the numerical stability of our algorithm, we factored and eigen-coded a 664932×31 matrix containing a 31-band constant-Q spectrogram of roughly 2 hours of audio for a music classification study [6]. We also

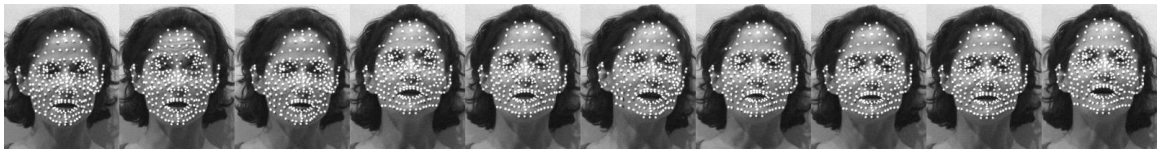

Fig. 4. Every 300th frame from a 3000-frame sequence tracked via rank-constrained optical flow with incremental SVD. Dots are superimposed on the video by the tracker.

used matlab's built-in Lanczos thin SVD to get the first 10 singular values (Lanczos estimates of small singular values are unreliable). These agreed with our results to 10 digits; the angle between the computed subspaces was a negligible 2×10^{-8} (see figure 3).

7.2 Subspace optical flow

The use of rank-constraints to regularize rigid-motion optical flow at many points in many frames was first introduced by Irani [11], and the general method has been extended to a variety of projective and motion models. Let $\mathbf{P}_{2F\times N}$ be the image projections of N points on a 3D surface viewed in F frames, arranged with horizontal and vertical projections on alternating rows. The main insight is that there is an upper rank-bound $r \geq \text{rank}(\mathbf{P})$, where r can be determined from inspection of the combined motion/projection model. Algebraically, it follows that $dr \geq \text{rank}(\mathbf{P}^{(\text{d})})$ [11,3], where the vector-transpose operator $[]^{(d)}$ partitions a matrix into d-element vertical groups and transposes the groups [15]. This connects to optical flow through the premise that intensity variations through time time are locally linear in surface motion, consequently rank constraints apply directly to measured intensity gradients $\mathbf{Y} = \mathbf{X}\frac{\text{d}}{\text{dt}}\mathbf{P}^{(2)}$, which should have rank $2r$. In this context, it is useful to compute temporal intensity variations $\mathbf{Y}_{2F\times N}$ and spatial intensity variations $\mathbf{X}_{2F\times 2F}$ in the Kanade-Lucas-Tomasi normal-flow framework, because $\mathbf{X}$ may be understood as both the *precision matrix* of the flow estimate [2] and the *covariance matrix* of the uncertainty in $\mathbf{Y}$ [3]. It follows immediately that the certainty-warp methods in section 6 give the *optimal* rank-reduction of $\mathbf{Y}$ with regard to the information in $\mathbf{X}$.

Irani's subspace optical flow algorithm sweeps a W-frame temporal window over an image sequence: In each window of frames, $\mathbf{Y},\mathbf{X}$ are measured at estimated correspondences from a reference frame. $\mathbf{Y}$ is rank-reduced to rank $2r$, then divided by $\mathbf{X}$ to estimate the flow, which is in turn rank-reduced to rank r and used to refine the correspondences. This iterates to convergence, and the window advances one frame. Many large SVDs must be computed per frame. We found that *most of this computation can be eliminated in favor of incremental* SVDs: a rank-$2r$ SVD of gradients $\mathbf{Y}$ and a rank-r SVD of correspondences $\mathbf{P}$. In fact, all that is needed are the right subspaces (singular vectors) and singular values of these two SVDs. When new measurements $\mathbf{Y}$ are made, they are incorporated into the rank-$2r$ gradient SVD, rank-reduced w.r.t. the updated subspace, divided by $\mathbf{X}$ to obtain flow, and cumulatively summed to obtain correspondences $\mathbf{P}^{(2)}$. These are then vector-transposed to $\mathbf{P}$ and similarly incorporated into the smaller rank-r SVD and rank-reduced w.r.t. its subspace.[5] When the flow has converged within a temporal window, the SVDs are permanently updated with the trusted correspondences, and the window advances.

[5] Some of the efficiency is lost because the updated subspaces must be discarded until the algorithm convergest to trusted correspondences and handling the uncertainty in $\mathbf{Y}$ obliges us to make an extra SVD, but we end up doing many small $O(Nr^2)$ or $O(NWr)$ SVDs rather than many large $O(NW^2)$ SVDs per frame.

The most important advantage of this method is that as the window advances, the SVDs accumulate information about how the points have moved and thereby "learn" a good basis for the surface being tracked. The tracker gets better and better, converging faster and faster until the SVDs need not be updated at all while processing a window; new measurements can be rank-reduced merely by projection onto the subspaces and then "unprojection" back to measurement space. The SVDs may be updated at window advances, but once the scene has exhibited most of its range of motions all updates become unnecessary. To automatically switch from SVD-based rank-reduction to projection-based rank-reduction, we monitor the angle between each newly updated subspace and an old subspace. When the subspace angle falls below a small threshold ε we may safely assume that the updates are not introducing new information about the range of motions exhibited by the scene.

Figure 4 shows some frames from a 3000-image sequence tracked in this manner using nonrigid-motion rank constraints derived by [3]. Please view the accompanying video, which shows every 6th frame with synthetic tracking dots superimposed on the original images. The rank was $r = 15$ and the temporal window size was $W = 31$ frames. The Irani tracker, modified to use the same rank constraints, "falls off" the surface after the first 290 frames, but survives another 220 frames if also modified to use certainty warps as per section 6. The Irani tracker required 5 iterations per window; the incremental SVD tracker performed well with 2 iterations per window and ceased SVD updates entirely roughly 900 frames into the sequence. We also note that once the subspaces are up to full rank (r and $2r$), the incremental subspace flow algorithm works quite well with temporal window sizes as small as one frame.

It is also worth noting that if occlusions are detected (e.g., see below), the imputative update can be used to continue tracking the unoccluded points while estimating the motion of the occluded points.

7.3 Structure from motion with occlusions

The problem of occluded points in (rigid) structure-from-motion factorizations have traditionally been handled through iterative methods [18,16]. Here we use the incremental imputative SVD to solve the same problem in a nonrigid context in a single pass through the data—and lower time complexity.

The subspace tracker was used to track points on a face in 150 frames of video. A recently developed SVD-based nonrigid factorization was used to factor the 2D motions into 3D rotations, translations, 3D shape and a linear basis set of 3D shape deformations, and per-frame deformation weights [3]. The K-mode forward model is

$$\mathbf{P}_{2F\times N} = \mathbf{M}_{2F\times 3K}\mathbf{S}_{3K\times N} \oplus \mathbf{T}_{2F\times 1} = [\searrow_f \mathbf{R}_f](\mathbf{S}^{(3)}\mathbf{C}_{K\times F})^{(3)} \oplus \mathbf{T} \tag{19}$$

where **S**hape basis tensor, deformation **C**oefficients, **R**otations, and **T**ranslations can be determined from F frames of N **P**oints viewed in weak perspective. (Each $\mathbf{R}_f$ is the top two rows of a rotation matrix.) Both $\mathbf{P}$ and its precision matrix (inverse covariance matrix) $\mathbf{X}_{DF\times DF}$ are calculated from optical flow as described above.

The recovered shape was somewhat flawed because occlusions of the nostrils by the nose contaminated the tracking. The rank constraints cause the tracker to keep the nostril points below the nose tip point, even when they are occluded. This exaggerates the apparent motion of the nostrils and leading to overestimated depth estimates of the upper lip and underside of the nose. These occlusions can be detected in 2D by Delaunay-triangulating the points in a reference frame and watching for edge-crossings in the Delaunay mesh as the points move in time. The 3D factorization gives depth estimates that indicate which points are occluders and which are occluded. We estimated occlusions and near-occlusions and the corresponding entries in the matrix of tracking data were obliterated (set to *NaN*). The incomplete data was re-factored using incremental SVD with certainty warps, resulting in an improved 3D linear shape basis with a properly shaped nose and a better fit to the video (see figure 5).

Fig. 5. Video frames with profile views synthesized from a structure-from-motion analysis of the 70×80 pixel facial region. The profiles are mirror images except for differences in recovered 3D structure. The profiles on the left have poor structure between the mouth and nose because occlusion artifacts in the tracking were correlated with head nods. The profiles on the right have better shape from the tip of the nose to the top of the mouth because incomplete SVD was used to handle occlusions in the 3D reconstruction.

8 Summary

We have examined the problem of finding good low-rank subspace models of datasets that may be extremely large, partly or mostly incomplete, contaminated with colored noise, and possibly even nonstationary. By combining an update rule with careful management of numerical noise, rank-minimizing imputations, and uncertainty transforms for MAP inference (with respect to measurement noise and user priors), we developed fast, accurate, and parsimonious online SVD methods, with better time/space complexity than widely used batch algorithms. This leads to fast online algorithms for vision tasks such as recovery of eigen-spaces, semi-dense optical flow on nonrigid surfaces with occusions, and automatic handling of occlusions in structure-from-motion.

9 Acknowledgments

Thanks to M. Casey for suggesting the audio problem and providing data.

References

1. Michael W. Berry. Large scale singular value computations. *International Journal of Supercomputer Applications*, 6:13–49, 1992.
2. Stan Birchfeld. Derivation of Kanade-Lucas-Tomasi tracking equation. Web-published manuscript at http://robotics.stanford.edu/~birch/klt/, 1996.
3. Matthew Brand. 3D morphable models from video. In *Proc. CVPR01*, 2001.
4. J. R. Bunch and C. P. Nielsen. Updating the singular value decomposition. *Numer. Math.*, 31:111–129, 1978.
5. P. Businger. Updating a singular value decomposition. *BIT*, 10:376–385, 1970.
6. Michael A. Casey. MPEG-7 sound-recognition tools. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(6), June 2001.
7. S. Chandrasekaran, B. S. Manjunath, Y. F. Wang, J. Winkeler, and H. Zhang. An eigenspace update algorithm for image analysis. *Graphical models and image processing: GMIP*, 59(5):321–332, 1997.
8. Gene Golub and Arthur van Loan. *Matrix Computations*. Johns Hopkins U. Press, 1996.
9. Benedikt Großer and Bruno Lang. An $O(n^2)$ algorithm for the bidiagonal SVD. Technical Report BUGHW-SC 2000/4, University of Wuppertal Department of Mathematics, 2000.
10. M. Gu and S. C. Eisenstat. A stable and fast algorithm for updating the singular value decomposition. Tech. Report YALEU/DCS/RR-966, Department of Computer Science, Yale University, New Haven, CT, 1993.
11. Michal Irani. Multi-frame optical flow estimation using subspace constraints. In *Proc. ICCV*, 1999.
12. Michal Irani and P. Anandan. Factorization with uncertainty. In *Proc. European Conf. Computer Vision*, 2000.
13. J.E. Jackson. *A user's guide to principal components*. Wiley, 1991.
14. A. Levy and M. Lindenbaum. Sequential karhunenloeve basis extraction and its application to images. Technical Report CIS9809, Technion, 1998.
15. Jan R. Magnus and Heinz Neudecker. *Matrix differential calculus with applications in statistics and econometrics*. Wiley, 1999.
16. D.D. Morris and T. Kanade. A unified factorization algorithm for points, line segments and planes with uncertainty models. In *Proc. Sixth ICCV*, pages 696–702, Bombay, 1998.
17. D.P. O'Leary and G.W. Stewart. Computing the eigenvalues and eigenvectors of arrowhead matrices. Technical Report CS-TR-2203/UMIACS-TR-89-22, University of Maryland Department of Computer Science, February 1989.
18. Conrad Poelman and Takeo Kanade. A paraperspective factorization method for shape and motion recovery. Technical Report CMU-CS-93-219, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, December 1993.

19. B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl. Application of dimensionality reduction in recommender system—a case study. In *ACM WebKDD 2000 Web Mining for E-Commerce Workshop*. ACM Press, 2000.
20. J. B. Tenenbaum and W. T. Freeman. Separating style and content with bilinear models. *Neural Computation*, 12:1247 – 1283, 2000.
21. Olga Troyanskaya, Michael Cantor, Gavin Sherlock, Pat Brown, Trevor Hastie, Robert Tibshirani, David Botstein, and Russ B. Altman. Missing value estimation methods for DNA microarrays. *BioInformatics*, 17:1–6, 2001.
22. S. van Huffel and H. Park. Efficient reduction algorithms for bordered band matrices. *Numerical Linear Algebra with Applications*, 2(2), 1995.
23. Hongyuan Zha and Horst D. Simon. On updating problems in latent semantic indexing. *SIAM Journal on Scientific Computing*, 21(2):782–791), 1999.

A Pseudo-inverse of a submatrix of an orthogonal matrix

Lemma 1. *Let* $\mathbf{B} = \left[\begin{smallmatrix}\mathbf{W}\\ \mathbf{Y}\end{smallmatrix}\right]$ *have orthogonal columns, such that* $\mathbf{B}^\top\mathbf{B} = \mathbf{W}^\top\mathbf{W} + \mathbf{Y}^\top\mathbf{Y} = \mathbf{I}$. *Then*

$$\mathbf{W}^+ = \mathbf{W}^\top + \mathbf{Y}^\top(\mathbf{I} - \mathbf{Y}\mathbf{Y}^\top)^+\mathbf{Y}\mathbf{W}^\top$$

and furthermore if $\mathbf{y} = \mathbf{Y}$ is a row vector, then

$$\mathbf{W}^+ = \mathbf{W}^\top + \frac{\mathbf{y}\mathbf{y}^\top}{1 - \mathbf{y}^\top\mathbf{y}}\mathbf{W}^\top$$

Proof. Define $\mathbf{Z} = \mathbf{Y}^\top\mathbf{Y}$.

$$\begin{aligned}
\mathbf{W}^+ &= \mathbf{W}^+\mathbf{W}^{-\top}\mathbf{W}^\top \\
&= (\mathbf{W}^\top\mathbf{W})^+\mathbf{W}^\top \\
&= (\mathbf{I} + (\mathbf{W}^\top\mathbf{W})^+ - \mathbf{I})\mathbf{W}^\top \\
&= \mathbf{W}^\top + (\mathbf{I} - \mathbf{W}^\top\mathbf{W} - \mathbf{I} + \mathbf{W}^\top\mathbf{W} + (\mathbf{W}^\top\mathbf{W})^+ - \mathbf{I})\mathbf{W}^\top \\
&= \mathbf{W}^\top + (\mathbf{Z} + ((\mathbf{W}^\top\mathbf{W})^+ - \mathbf{I})(\mathbf{I} - \mathbf{W}^\top\mathbf{W}))\mathbf{W}^\top \\
&= \mathbf{W}^\top + (\mathbf{Z} + (\mathbf{I} - \mathbf{W}^\top\mathbf{W})(\mathbf{W}^\top\mathbf{W})^+(\mathbf{I} - \mathbf{W}^\top\mathbf{W}))\mathbf{W}^\top \\
&= \mathbf{W}^\top + (\mathbf{Z} + \mathbf{Z}(\mathbf{W}^\top\mathbf{W})^+\mathbf{Z})\mathbf{W}^\top \\
&= \mathbf{W}^\top + (\mathbf{Z} + \mathbf{Z}(\mathbf{I} - \mathbf{Z})^+\mathbf{Z})\mathbf{W}^\top \\
&= \mathbf{W}^\top + \mathbf{Z}(\mathbf{I} - \mathbf{Z})^+\mathbf{W}^\top \\
&= \mathbf{W}^\top + \mathbf{Y}^\top(\mathbf{I} - \mathbf{Y}\mathbf{Y}^\top)^+\mathbf{Y}\mathbf{W}^\top,
\end{aligned}$$

where the last equality holds from the Matrix Inversion Lemma. The special case of row vector $\mathbf{y} = \mathbf{Y}$ simplifies the pseudo-inverse to a scalar inverse, giving the second form of the this lemma.