Fast online SVD revisions for lightweight recommender systems

Matthew Brand*

Abstract

The singular value decomposition (SVD) is fundamental to many data modeling/mining algorithms, but SVD algorithms typically have quadratic complexity and require random access to complete data sets. This is problematic in most data mining settings. We detail a family of sequential update rules for adding data to a "thin" SVD data model, revising or removing data already incorporated into the model, and adjusting the model when the data-generating process exhibits nonstationarity. We also leverage the SVD to estimate the most probable completion of incomplete data. We use these methods to model data streams describing tables of consumer×product ratings, where fragments of rows and columns arrive in random order and individual table entries are arbitrarily added, revised, or retracted at any time. These purely online rules have very low time complexity and require a data stream cache no larger than a single user's ratings. We demonstrate this scheme in an interactive graphical movie recommender that predicts and displays ratings/rankings of thousands of movie titles in real-time as a user adjusts ratings of a small arbitrary set of probe movies. The system "learns" as it is used by revising the SVD in response to user ratings. Users can asynchronously join, add ratings, add movies, revise ratings, get recommendations, and delete themselves from the model.

Keywords: collaborative filtering; singular value decomposition; online updating; real-time recommending; incomplete data.

1 The problem

Recommender systems use a small sample of customer preferences to predict likes and dislikes over a much wider set of products. The prediction function is estimated from a tabular database of customer×product scores. In this paper, we will focus on movie ratings. It is not unusual for these tables to be enormous ($O(10^3) - O(10^7)$ rows and columns) but mostly empty, with most scores unknown. In fielded systems, the table is constantly changing, with rows, columns, and individual scores constantly being added, revised, or censored. These edits may arrive asynchronously from many distributed sources. Efficient methods for warehousing, updating, and accessing such tables remain active issues in database and datastructure research. Estimating a reasonably efficient, compact, and accurate prediction function is an even harder problem that has attracted much attention in the data mining and machine learning communities. Nearest-neighbor methods, which effectively match against raw data, have remained popular and effective despite high search costs and limited predictivity. More sophisticated prediction methods are often defeated by very high data dimensionality, high computational costs of model fitting, and the inability to adapt to new or retracted data. Moreover, with very sparsely populated tables, the data is often insufficient to support accurate parameter estimates in these models. Typically a dense subset of the table is constructed from the responses of a focus group, and the prediction function is extrapolated from that.

The very high dimensionality of this and other data mining problems has motivated explorations of multilinear models such as the thin singular value decomposition (thin SVD), both as a compressed representation of the data and as a basis for predictions via linear regression. Linear regression models generally have lower sample complexity per parameter than nonlinear and nonparametric models, and can thus be expected to show better generalization. The SVD and related eigenvalue decomposition (EVD) lie at the heart of thousands of data-analysis algorithms, where they are used for dimensionality reduction, noise suppression, clustering, factoring, and model-fitting. Several well-known recommender systems are based on the SVD/EVD [5, 14, 17, 11]. Unfortunately, computing an SVD of a very large dataset is an impractical affair, requiring complete data, run-time quadratic in the dataset size, and in-memory storage of the entire dataset. Many algorithms have been proposed to deal with some but not all of these problems. Adapting to new or retracted data is also an issue, though it is well understood how to append [6, 16, 15, 22, 3] or delete [21] entire columns or rows, provided that they are complete.

In this paper we contemplate the following *purely online* data mining scenario: Data arrives asynchronously (in no particular order) as *fragments* of rows and columns. Rows, columns, and fragments thereof may be added, changed, or retracted in any order. The ultimate size of the data matrix is unknown. The collected data will typically be very sparse, but missing values cannot be presumed to be zeros. The task is to compute the best running estimate of a thin rank-r SVD

*Mitsubishi Electric Research Labs, Cambridge, Massachusetts, USA

of the true data matrix, without any storage or caching of incoming data, and make recommendations (from predicted missing values) from this SVD on demand.

To this end we develop an exact rank-1 update that provides very fast additions, deletions, and element-wise edits—fast enough for linear-time construction of the whole SVD. When faced with missing data, we use an imputative update that maximizes the probability of correct generalization. We demonstrate experimentally that these methods are much faster and at least as predictive as offline SVD approaches reported in the data mining literature. The rank-1 updates are the main theoretical contribution of this paper; the main practical contribution is a demonstration of their use in a full-scale movie recommending system with a graphical user interface in which users move sliders to rate movies and see all other movies re-rated and ranked with millisecond response times. The system "learns" from user's query ratings, again, in real-time.

2 The SVD in data mining

The singular value decomposition factors a matrix $\mathbf{X}$ into two orthogonal matrices $\mathbf{U}, \mathbf{V}$ and a diagonal matrix $\mathbf{S} \doteq \text{diag}(\mathbf{s})$, such that $\mathbf{USV}^\top = \mathbf{X}$ and $\mathbf{U}^\top\mathbf{XV} = \mathbf{S}$. The elements of $\mathbf{s}$ are called singular values and the columns of $\mathbf{U}, \mathbf{V}$ are called the left and right singular vectors, respectively. If we sign and arrange these matrices such that the values on the diagonal of $\mathbf{S}$ are nonnegative and in descending order and the first nonzero element in each column of $\mathbf{U}$ is also positive, then the SVD is unique (ignoring any zero singular values). The SVD has the optimal truncation property: If we discard all but the r largest singular values and the corresponding singular vectors, the product of the resulting thinned matrices $\mathbf{U}'\mathbf{S}'\mathbf{V}'^\top \approx \mathbf{X}$ is the best rank-r approximation of $\mathbf{X}$ in the least-squares sense. This is called a *thin* SVD. For this reason, the matrix $\mathbf{U}'^\top\mathbf{X} = \mathbf{S}'\mathbf{V}'^\top$—the projection of $\mathbf{X}$ onto r orthogonal axes specified by the columns of $\mathbf{U}'$—is an excellent reduced-dimension representation of the data.

If $\mathbf{X}$ is a tabulation of consumer×product affinity scores, then the subspace spanned by the columns of $\mathbf{U}'$ can be interpreted as a r-dimensional *consumer taste space*, where individuals are located according to the similarity of their tastes. The relationship between a user's ratings (represented as a column vector $\mathbf{c}$) and his/her taste-space location $\mathbf{p}$ is simply $\mathbf{p} = \mathbf{U}'^\top\mathbf{c}$ and $\mathbf{c} \approx \mathbf{U}'\mathbf{p}$, where the approximation is squared-error-optimal for an r-dimensional model. If $\mathbf{c}$ is the n^{th} column in the original $\mathbf{X}$, then $\mathbf{p}$ is the n^{th} row in $\mathbf{V}'\mathbf{S}'$. If $\mathbf{c}$ is incomplete, various imputation methods (discussed below) can estimate $\mathbf{p}$ and thence a completion of $\mathbf{c}$; this is the basis of SVD-based recommending. One can also identify people with similar tastes by their Euclidean distant to $\mathbf{p}$ in taste space (e.g., [11, 17]). Similarly, the *product taste space* $\mathbf{V}'$ contains products arranged by what sorts of people like them. Often these spaces are useful for subsequent analyses such as clustering, visualization, market segmentation, and pricing of new products.

The SVD is most informative when the data has first been translated so that it is centered on the origin. In that case, the SVD can be interpreted as a Gaussian covariance model of the data that captures correlations between consumer's tastes. Centering allows proper Bayesian inference about typicality and missing values, as well as statistical tests to verify Gaussianity.

Centering aside, the main practical impediment to using a thin SVD is the cost of computing it. State-of-the-art methods are typically based on Lanczos or Ritz-Raleigh iterations. Run-time can can be linear in the number of nonzero elements in the data, but these methods require multiple passes through the entire dataset to converge, and are not suitable in online settings. Sequential SVD updating algorithms have focussed on modifying a known SVD of data $\mathbf{X}$ to obtain an SVD of the data with an appended column ($[\mathbf{X}, \mathbf{c}]$). We recently introduced an exact (closed-form) update rule [3] that can build a rank-r SVD of a low rank $p \times q$ matrix through sequential updates in linear time[1] ($O(pqr)$ time for the entire matrix), making a single pass through the columns of the matrix, meaning that the data need not be stored. It can be shown that the method introduced in this paper contains this as a special case and inherits its very favorable performance. The rule given here generalizes to provide the remaining operations needed for a true online system: downdating (removing rows and columns) revising (changing selected values in a row or column), and recentering. In an online setting, keeping the SVD model centered is an acute problem, because the mean of the data is constantly drifting, partly due to sample variation and, more importantly, due to long-run nonstationarities such as changing market dynamics and drifting tastes in a population.

3 Background

SVD updating has a literature spread over three decades [7, 6, 2, 10, 1, 8, 15, 22, 3] and is generally based on iterative Lanczos or Ritz-Raleigh methods, or relationships between SVDs in a subspace and the full data SVD. The last category includes some very fast methods, but they are often approximate [1] and/or vulnerable to loss of orthogonality [22, 8, 15]. For example: Berry *et alia* [1] propose to project the problem into a previously estimated low-rank subspace, but the resulting updates ignore any component of new data that lies outside that subspace. Levy and Lindenman [15] show that one can incrementally compute the left singular vectors in $O(pqr^2)$ time; if p, q, and r are known in advance

[1] For a $p \times q$ matrix $\mathbf{X}$ and desired SVD of rank $r \leq O(\sqrt{\min(p,q)})$, our updates perform the entire SVD in $O(pqr)$ time–purely linear in the size of the inputs and outputs. Other updating algorithms are quadratic in the size of the outputs and/or must make multiple passes through the data and/or must be iterated to convergence.

operation	known	desired	$\mathbf{a}$	$\mathbf{b}^\top$
update	$\mathbf{USV}^\top = [\mathbf{X}, \mathbf{0}]$	$\mathbf{U'S'V'}^\top = [\mathbf{X}, \mathbf{c}]$	$\mathbf{c}$	$[0, \cdots, 0, 1]$
downdate	$\mathbf{USV}^\top = [\mathbf{X}, \mathbf{c}]$	$\mathbf{U'S'V'}^\top = \mathbf{X}$	$-\mathbf{c}$	$[0, \cdots, 0, 1]$
revise	$\mathbf{USV}^\top = [\mathbf{X}, \mathbf{c}]$	$\mathbf{U'S'V'}^\top = [\mathbf{X}, \mathbf{d}]$	$\mathbf{d} - \mathbf{c}$	$[0, \cdots, 0, 1]$
recenter	$\mathbf{USV}^\top = \mathbf{X}$	$\mathbf{U'S'V'}^\top = \mathbf{X} - \mathbf{m1}^\top$	$-\mathbf{m}$	$\mathbf{1}^\top \doteq [1, \cdots, 1]$

Table 1: Database operations expressed as rank-1 modifications of an SVD $\mathbf{USV}^\top = \mathbf{X}$ to give $\mathbf{U'S'V'}^\top = \mathbf{X} + \mathbf{ab}^\top$.

and $p \gg q \gg r$, then the expected complexity falls to $O(pqr)$. However, orthogonality can decay quickly and results have only been reported for matrices having a few hundred columns.

None of this literature contemplates missing values, except insofar as they can be treated as zeros (e.g., [2]). 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., [20]). 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 [17].

In the special case where a matrix $\mathbf{M}$ is nearly dense, its normalized scatter matrix $\Sigma_{m,n} \doteq \langle \mathbf{M}_{i,m}\mathbf{M}_{i,n} \rangle_i$ ($\langle\cdot\rangle_i$=expectation w.r.t. known values in row i) may be fully dense due to fill-in. A popular heuristic interprets Σ's eigenvectors as $\mathbf{M}$'s right singular vectors [13]. It can be shown that this is strictly incorrect; there may not be any imputation of the missing values that is consistent with Σ's eigenvectors[2]. For the very sparse problems that we will consider, this approach is mooted by the fact that Σ is also incomplete and its eigenvectors undefined.

4 Modifying the SVD

Updating, downdating, revising, and recentering are all instances of rank-1 modifications: Given column vectors $\mathbf{a}, \mathbf{b}$ and a known SVD $\mathbf{USV}^\top = \mathbf{X}$, what is the SVD of $\mathbf{X} + \mathbf{ab}^\top$? Table 1 illustrates all cases. Typically $\mathbf{b}$ is a binary vector indicating which columns should be modified, and $\mathbf{a}$ is derived from contains update or downdate values ($\mathbf{c}$), revision values ($\mathbf{d}$), or a mean value ($\mathbf{m}$) which should be subtracted from all columns. These operations are summarized in table 1. In appendix A we show how any such low-rank modification can be solved via operations in the low-dimensional subspaces specified by the known SVD. The basic strategy is that the *new* SVD can be expressed as a product of the *old* subspaces (slightly augmented) and a not-quite diagonal core matrix, which can be rediagonalized by left and a right rotation. These are small-matrix operations, and thus fast. Applying the opposite rotations to the augmented old subspaces gives a new SVD. Even this step can be made fast by accumulating the small rotations over many updates instead of applying them to the large subspace matrices. Thus for a rank-r thin SVD, the dominant computations scale in r, which is typically very small relative to the size of the data.

It can be shown that the special case of the rank-1 rule giving SVD updates is algebraically equivalent to (and simpler than) an update rule we recently introduced in a related computer vision paper [3]. There we showed that through careful management of the computation, the update rule can build an exact SVD of a $p \times q$ rank-r matrix in purely linear $O(pqr)$ time when the rank is small relative to the size, specifically when $r \sim O(\sqrt{\min(p,q)})$. This is borne out empirically in figure 1, where a new implementation of our method is compared against a commercial Lanczos implementation. When the matrix has rank$> r$, our method (and any thin SVD algorithm) necessarily gives an approximation: Each update will increase the rank of the SVD by 1, until a user-specified ceiling is reached. At this point the update ceases being exact because the last singular value will have to be dropped, giving the optimal fixed-rank approximation. Typically this singular value has tiny mass and the approximation errors will cancel out over many updates, so that in practise our method often has numerical accuracy competitive with Lanczos methods [3]. Because our method is so fast, one can always build a rank-$2r$ model and use the rank-r submodel, which is typically accurate to machine precision. It should also be noted that when mining datasets of elicited responses such as user ratings, the values themselves are notoriously unreliable (users show poor repeatability, with ratings wandering up to 40% of the scale from day to day[3]), so a good low-rank approximation of the data has higher probability of generalization than a medium-rank model that perfectly reconstructs the data.

[2] Evidently this has been overlooked in the literature, so we offer a short proof: Take a dense matrix $\mathbf{M}$ with one element set to zero, and compute its normalized scatter matrix as if the element were missing. The difference between the normalized scatter and $\mathbf{M}$'s covariance will be zeroes except for one diagonal element and the row and column containing it. If there is an imputation consistent with the scatter, then the off-diagonal elements of this row (or column) should be the product of the imputed value and the other values in the vector. However, the corresponding elements of the scatter matrix are computed without regard to other values of the vector, and so no such relationship holds.

[3] Joseph Konstan, personal communication.

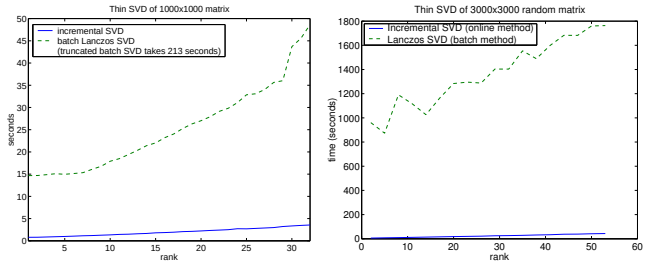

Figure 1: Run-time of sequential SVD updating (blue solid line) versus batch Lanczos (green dashed line), as a function of the number of singular vector/value triplets computed from a random matrix. Each datapoint represents the average of 100 trials. In each trial, both algorithms correctly factor the same dense matrix. The sequential update shows clear linear scaling and speed advantages. The experiment graphed at left employed low-rank matrices; at right, full-rank matrices having reasonable low-rank approximations. Our method shows similar speed advantages over other updating algorithms (e.g., [8, 15, 16]), but produces more accurate results. Experiments were performed in Matlab 6 on an AlphaServer with a 600MHz CPU and 10G RAM.

We stress that *all* of the operations introduced in this paper—not just the update—have low computational complexity and work in a pure streaming-data setting, with no data warehousing and low storage overhead. The updates require only the current SVD matrices, the index of the current user or product, and the vector containing the new ratings. Even for data tables with thousands of rows and columns, the updates can keep everything on the CPU's on-board memory cache, making for very fast performance.

5 Imputation and prediction

Ratings tables are typically incomplete; most entries are unknown. Missing values present a serious problem for data mining algorithms based on matrix factorizations because the decompositions are not uniquely defined: Even if a single value is missing, there is a continuous orbit of SVDs consistent with the remaining known entries. The imputation problem—how to predict missing values—plays a key role in computing the SVD and in making recommendations.

The literature is rich with proposed imputation schemes. Most perform a small SVD of a submatrix that is dense, regress against the SVD to impute missing values in some adjoining part of the submatrix, perform a larger SVD of the dense and imputed values, and repeat until all missing values are filled in. This can have quartic complexity, and the result is very sensitive to the choice of regression method and order of imputations. Other methods are based upon the expectation-maximization algorithm and have similar complexity and sensitivity to initial conditions.

Approximation theory teaches that when learning in an online setting with a finite memory, some sensitivity to data ordering is unavoidable. The strategy to minimize this sensitivity in sequential updating algorithms is to select updates that have the highest probability of correctly generalizing, usually by controlling the complexity of the model while maximizing the probability of the data.

Our approach exploits the fact that the (squared) singular values and left singular vectors comprise an EVD of the data's covariance matrix. Under the generic assumption that the data is normally but anisotropically distributed, the SVD can be interpreted as a Gaussian model of the data density, and SVD updating as sequential updating of that density. Adding a complete vector is equivalent to updating the density with a point; adding an incomplete vector is equivalent to updating the density with a subspace whose axes correspond to the unknown elements of the vector. If the SVD is thin, then the imputed point may be further constrained to lie in the intersection of the data subspace and the missing value subspace. Naive imputation schemes such as linear regression (e.g., [12, 17]) essentially choose the point in the missing value subspace or intersection subspace that is closest to the origin, essentially assuming that some unknowns are zero-valued. Such imputations are not likely to be true, and generally reduce the predictivity of the model if incorporated into the SVD.

Clearly imputation requires some prior or learned knowledge. In appendix B we introduce a fast imputative update for use when some of the values in a ratings vector $\mathbf{c}$ are unknown but assumed to lie within a known range—a commonplace in ratings data. The solution is exact in the sense that the updated SVD will reconstruct a matrix $\mathbf{X}$ whose covariance statistics exactly match those obtained by integrating over the uncertain values with a uniform prior. However, the bounds only add information if they are asymmetric about zero; otherwise we find that this imputative scheme actually slightly underperforms a probabilistic scheme introduced in [3] and modified here for data mining.

We suggest that that imputation should be informed by the density of previously processed data. It was shown in [3] that if one considers all points in the intersection subspace according to their likelihood vis-a-vis the data density with a uniform prior, then the posterior mean estimate of the missing values is given by choosing the point that lies the fewest standard deviations from the origin. This is illustrated in figure 2. The calculation is quite simple and is given in appendix B.1. For SVD updating the full imputation is not needed, just the subspace coordinates of the imputed point. It was also shown that this imputation greedily minimizes the growth of the rank of the SVD, which is the complexity of the model [3]. This imputation was developed to predict the location of occluded features in computer vision problems, and happens to give good results in data mining tasks as well.

The imputed ordinates are essentially predictions of how

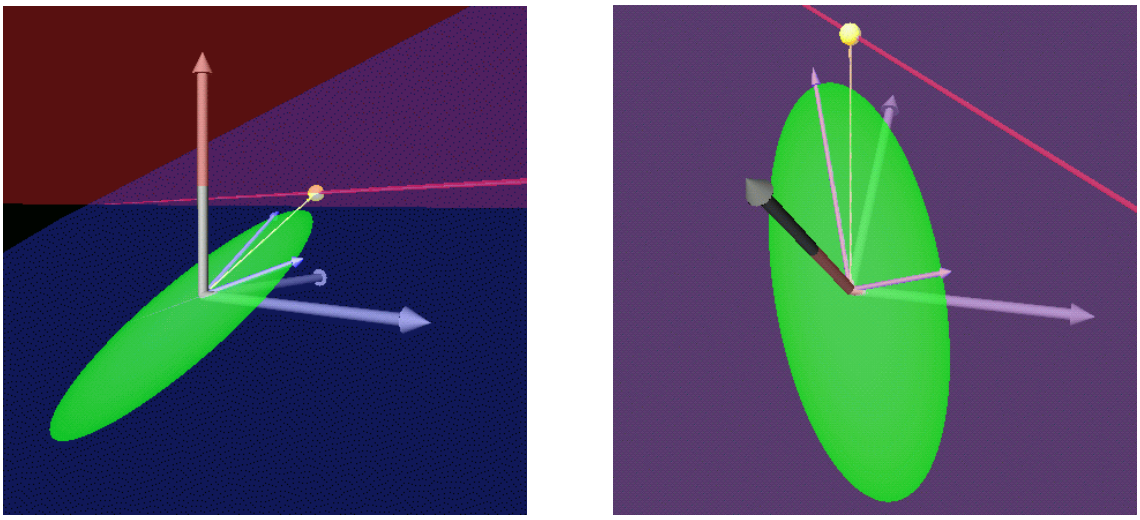

Figure 2: Illustration of imputation in 3-space viewed from side (left) and top (right). The SVD currently specifies a 2-dimensional subspace indicated by the blue (tilted) plane; its axes (eigenvectors) are indicated by the small arrows and its 1 standard deviation probability isocontour is indicated by the green ellipse. The incomplete vector $\mathbf{c} = [?, ?, 2]$ is depicted by the red (level) plane. The intersection of the data subspace and the completion subspace is the 1D space depicted as a red line. On that line, the yellow point is the most probable imputation, having smallest Mahalonobis (but not Euclidean) distance to the density peak at the origin.

a consumer would rate all products, equivalent to a linear mix of the ratings of all other consumers, weighted by the correlations between their ratings and the few known ratings of the current consumer. Thus the same machinery is used for imputing and recommending.

In moving from continuous-valued computer vision problems to bounded-value data mining problems, we have noted a potential pathology of the of density-based imputation: It is possible (though rare) that the intersection space is very far from the origin, in which case a large but improbable vector will be added to the SVD. If it is known *a priori* that such vectors are impossible (e.g., values are bounded to lie in some small range), then such constraints can often be expressed as a Gaussian prior and a *maximum a posterior* imputation made via least-squares methods. In the context of a thin SVD, this is equivalent to assuming that all of the truncated singular values have $\varepsilon > 0$ mass instead of zero mass (equivalent to the Bayesian formulation of principal components analysis; see [19]). In this case the imputed vector will be much smaller but will lie slightly outside the taste-space, requiring either an exact rank-1-increasing update or an approximate fixed-rank update.

6 The bootstrapping problem

The more mass in the singular values, the more constrained the imputation is by previous inputs, and therefore the better the estimated SVD. This poses a problem in the beginning, when the SVD has almost no mass because the first few users have rated few items. In our application, users typically rate less than 1% of all items, so there is little chance than any item has been rated by two users until many users have used the system. One way to work around this is to warehouse the first few hundred submitted ratings, then re-order the rows and columns of this matrix so that it is dense in one corner. This can be done by rapid sorting in $O(pq \log pq)$ time. The SVD can be "grown" out of this corner by sequential updating with partial rows and columns. When it finally becomes necessary to impute values (the algorithm has run out of complete partial rows and columns), these imputations are reasonably well constrained. This scheme defers imputations until they can be well constrained by previously incorporated data. It also enables factorings of extremely sparse datasets. Figure 3 illustrates this method with a synthetic toy problem and shows that it compares favorably to *Matlab*'s batch thin SVD (with column-average imputations) in terms of rank, log-volume, flop count, and even numeric accuracy. In the next section we apply it to a real full-scale data-mining problem.

7 Application to collaborative filtering

The collaborative filtering problem takes an extremely sparse array of consumer $\times$ product scores and asks for predictions of the missing scores. Sarwar *et alia* [17] collected a 93.7% empty matrix containing ratings of 1650 movies on a 1-5 scale by 943 individuals, and split it 80%/20% into training and test sets. They filled missing elements in the training matrix with the average rating for each movie, centered the matrix, then computed a series of thin, sparse Lanczos SVDs.

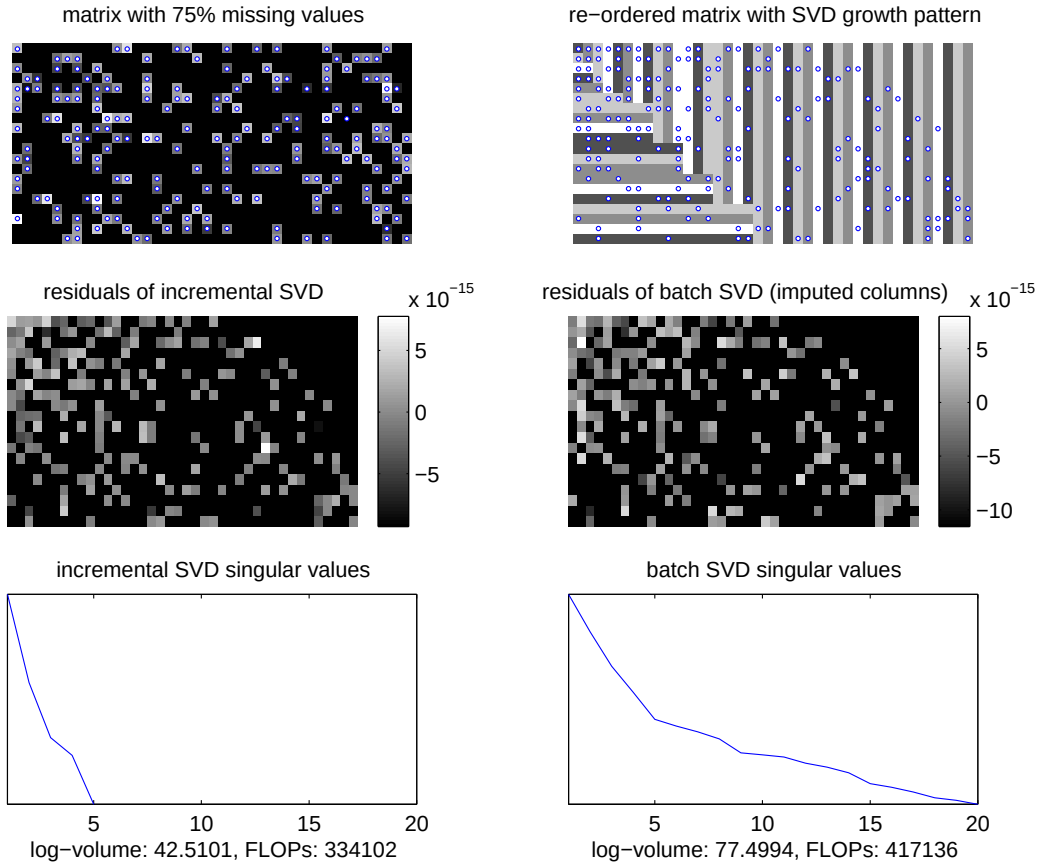

Figure 3: Bootstrapping an imputative SVD of a mostly unknown matrix. TOP LEFT: A rank-8 20×40 matrix with 75% of its entries randomly obliterated (set to *NaN*). Dots indicate entries having values. TOP RIGHT: The matrix is rearranged and partitioned into partial rows and columns for incremental SVD. The pattern of bars shows the order of updates. MIDDLE AND BOTTOM LEFT: Incremental SVD yields a rank-5 decomposition that reconstructs the surviving data to machine precision. MIDDLE AND BOTTOM RIGHT: A batch SVD of the matrix with EM-imputed entries requires all 20 singular values to reconstruct the surviving data to machine precision. Truncating the batch SVD to rank 8 amplifies the residuals by 10^{14} (not shown). The bottom graphs show how much of the volume of the associated Gaussians is explained by each singular value/vectors triplet.

They found that a rank-14 basis best predicted the test set as measured by average absolute error (MAE) and mean-squared error.

We obtained their data and repeated their procedure with similar (but not identical) numerical results: The rank-15 basis was marginally better than their reported result, with an MAE of 0.7914 and a standard deviation (SD) of $\sigma = 0.9960$. The difference is possibly due to different test/train splits. We then applied our incremental imputative SVD to the raw training dataset and found a 5-dimensional subspace that had even better prediction accuracy of 0.7910 MAE, 1.0811 SD (see figure 4). In each of 5 disjoint train/test splits, the incremental algorithm produced a compact 4- or 5- dimensional basis that predicted at least as well as the best (but much larger) Lanczos-derived basis. These scores are competitive with published reports of the performance of nearest-neighbor based systems on the same dataset (see [17]), which makes the subspace approach appealing because of its lower overhead for storage, updates, and predictions.

Not surprisingly, an incremental SVD of the whole database indicated that the five largest singular values account for most of the variance in the data. In all cases the resulting predictor is within 1 rating point of the true value more than 80% of the time and within 2 points more than 99% of the time. This is probably more than accurate enough, as raters often exhibit day-to-day inconsistencies of 1-2 points when asked to rate the same movies on different days. The incremental algorithm also has the practical advantages of being faster (0.5 GFLOPs versus 1.8 GFLOPs), and opens the way to fast online updating of the SVD as new

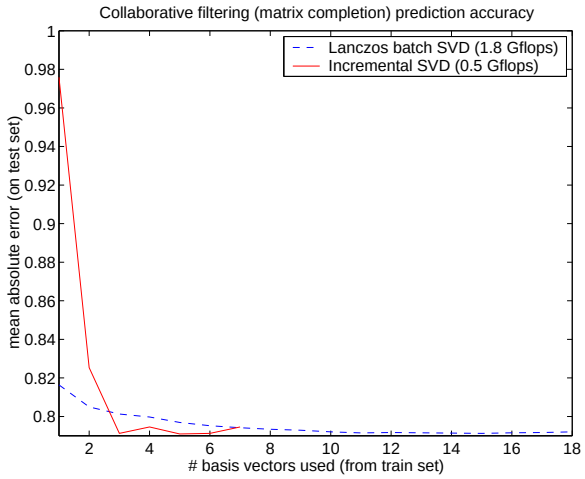

Figure 4: Prediction error of a held-out test set from thin SVDs of a training set with 93.7% of all values missing. The incremental method finds a 5-dimensional basis that bests the 15-dimensional basis found by the Lanczos method (after column imputations).

movies or viewers are added to the database. (Collaborative filtering systems typically require large overnight computations to incorporate new data.)

7.1 The instant movie recommender Because the SVD basis is small and the computations are lightweight, we found it practical to implement a real-time interactive collaborative filtering system in java, shown in figure 5. To query the system, a user selects a small number of movies by dragging their titles into a rating panel, where each can be rated by moving a pointer along a sliding scale. Imputed ratings of all the other movies and a sorted list of recommendations are updated and displayed in real-time: As the slider moves, sliders next to all other movie titles move. It takes an average of 6 milliseconds to re-rate all movies, so the user experiences instantaneous feedback. One advantage of instant animated visual feedback is that the user can see how strongly the predicted rating of any movie is correlated or anticorrelated with that of the movie whose rating is currently being varied. Second, if the system makes a prediction that the user disagrees with, she can drag that movie into the rating panel and correct the system. A few such iterations quickly yields a robust (overconstrained) estimate of the user's location in taste space, leading to improved recommendations and a more informative ratings vector to be incorporated into the SVD. We find that users naturally engage in this kind of interaction.

When the user is done, her ratings can be sent back to a central server for immediate updating of the SVD basis. A user can also obtain a persistent identifier (e.g., her column index) so that she can come back and review, revise, or remove her ratings. The demo version pictured in figure 5 combines the recommender and the SVD operations in a standalone java application; SVD updates average 50 milliseconds. Some advantages of this approach are 1) real-time interactivity; 2) all of the computation for recommendations is done on the client's computer; 3) the basis (SVD) is constantly updated; 4) the user community "grows" the basis by adding new movies and users.

The recommending engine and interface are lightweight enough that they could be served as a javascript web-page. It is even practical to combine them with the SVD update in a web-served client: Instead of using a central server to update the basis, users could simply broadcast their ratings to each other and do autonomous updates. This offers the possibility of totally decentralized collaborative filtering.

8 Discussion

In this paper we introduced family of rank-1 SVD revision rules and showed that they efficiently allow a thin SVD to "mimic" database operations on tables of consumer×product scores: adding, deleting, and revising rows, columns, and fragments thereof. In addition the SVD can be recentered as the mean of the data stream drifts. All operations have low time and storage complexity, and run fast enough for collaborative filtering and recommending to be a real-time graphical interaction with the model of communal tastes. We also introduced a new imputation rule and revised a highly successful probabilistic imputation rule in light of constraints on the range of values that can be imputed.

8.1 Changing tastes A particularly interesting issue is how to handle nonstationarity in online learning systems. In a fielded system, it is likely that a repeat user will add new ratings and change old ones. This is accommodated through revisions and recenterings of the SVD, which are desirable because tastes change and the SVD should not be anchored to "stale" ratings. These operations are exact (iff the table truly has rank r), yielding a model which gives equal weight to the most recent opinions of all users. However, if tastes really do change, then older ratings should be discounted, because the taste subspace is nonstationary and should be tracked over time. For large-sample processes, this is properly accommodated by gradually forgetting the past: In the context of sequential SVD updating, the singular values would be made to decay exponentially $\mathbf{S} \leftarrow \lambda\mathbf{S}$ $(0 < \lambda < 1)$ on each update so that the weight of experience (literally, the mass of the singular values) does not grow to overwhelm the weight of new information (literally, the norm of a new ratings vector). When and how fast we may allow singular values to decay is an important question that we hope to answer in a future paper with the tools of large deviation theory.

Readers may also be interested in our work on nonlinear

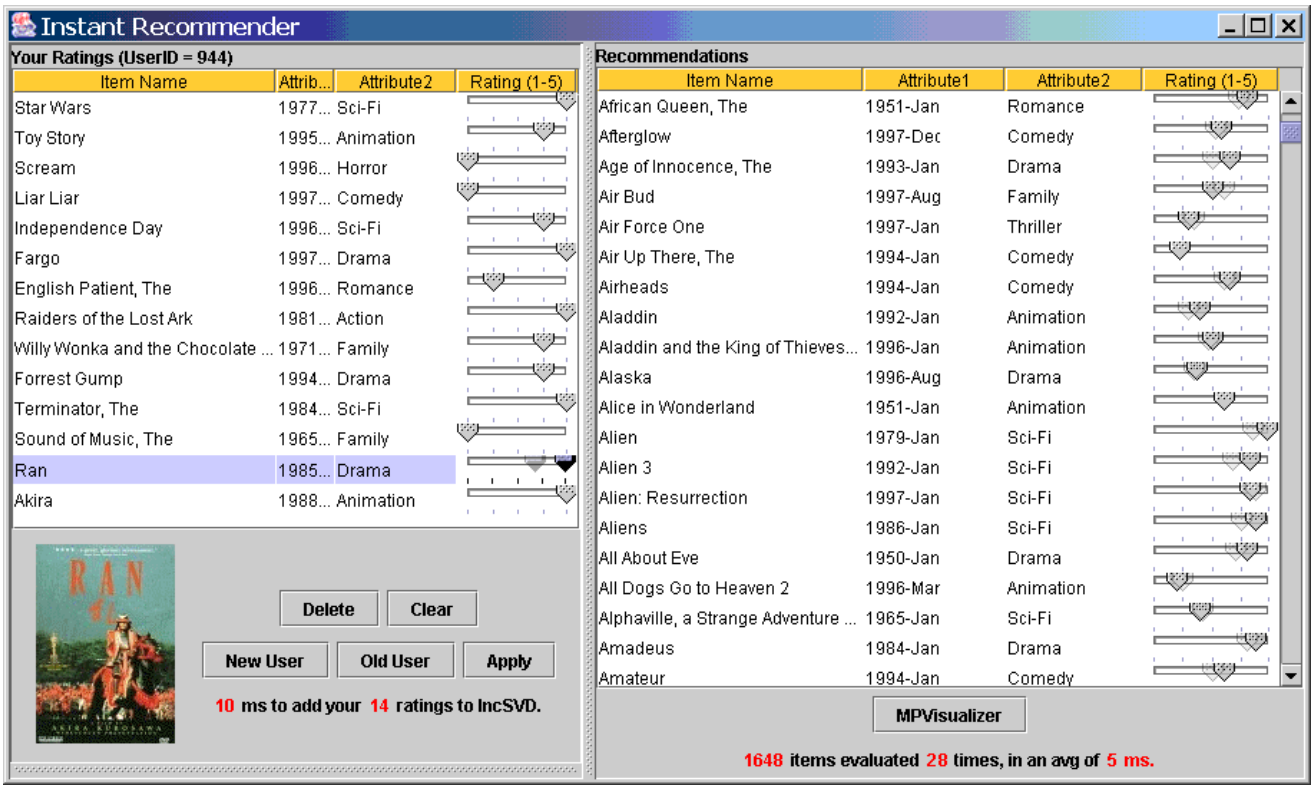

Figure 5: The Instant Movie Recommender. Moving the slider next to any movie in the left panel causes all movies on the right to be re-rated in real-time. This double image shows how ratings change as the user raises the rating of "Ran." In the user community, liking "Ran" is strongly correlated with liking "The Age of Innocence," anti-correlated with liking "Air Bud," and uncorrelated with liking "Alphaville." Movies on the right can also be sorted by recommendation score in real-time. The rating set can be changed at any time. When the user is done, her query (ratings) are used to update the basis.

analogues of the SVD, which project the data onto curved manifolds rather than flat subspaces (e.g., [4]); these are particularly useful for data visualization.

9 Acknowledgments

Andrey Rakhmanoff ported these algorithms to C. Sergei Makar, Fred Igo, and Shinsuke Azuma developed the initial demo into the browser/recommender interface shown in figure 5. GroupLens graciously provided the movie data.

References

[1] M. Berry, S. Dumais, and T. Letsche. Computational methods for intelligent information access. In *Proc. Supercomputing'95*, 1995.

[2] M. W. Berry. Large scale singular value computations. *International Journal of Supercomputer Applications*, 6:13–49, 1992.

[3] M. Brand. Incremental singular value decomposition of uncertain data. In *Proceedings, European Conference on Computer Vision*, Lecture Notes on Computer Science, pages 707–720. Springer-Verlag, 2002.

[4] M. Brand. Charting a manifold. In *Proc. NIPS-15*, 2003.

[5] S. Brin and L. Page. Anatomy of a large-scale hypertextual web search engine. In *Proc. 7th Int'l World Wide Web Conference*, 1998.

[6] J. R. Bunch and C. P. Nielsen. Updating the singular value decomposition. *Numer. Math.*, 31:111–129, 1978.

[7] P. Businger. Updating a singular value decomposition. *BIT*, 10:376–385, 1970.

[8] 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.

[9] G. Golub and A. van Loan. *Matrix Computations*. Johns Hopkins U. Press, 1996.

[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] D. Gupta and K. Goldberg. Jester 2.0: A linear time collaborative filtering algorithm applied to jokes. In *Proceedings of*

the SIGIR. ACM, 1999.
[12] W. Hill, L. Stead, M. Rosenstein, and G. Furnas. Recommending and evaluating choices in a virtual community of use. In *Human factors in computing systems Proc. CHI95*, pages 194–201, 1995.
[13] J. Jackson. *A user's guide to principal components*. Wiley, 1991.
[14] J. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proc. 9th ACM-SIAM Symposium on Discrete Algorithms*, 1998.
[15] A. Levy and M. Lindenbaum. Sequential karhunen-loeve basis extraction and its application to images. Technical Report CIS9809, Technion, 1998.
[16] M. Moonen, P. van Dooren, and J. Vandewalle. Singular value decomposition updating algorithm for subspace tracking. *SIAM Journal on Matrix Analysis and Applications*, 13(4):1015–1038, 1992.
[17] 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.
[18] G. W. Stewart. An updating algorithm for subspace tracking. *IEEE Trans. Signal Processing*, 40:1535–1541, 1992.
[19] M. Tipping and C. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society, Series B*, 21(3):611–622, 1999.
[20] O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein, and R. B. Altman. Missing value estimation methods for DNA microarrays. *BioInformatics*, 17:1–6, 2001.
[21] D. I. Witter and M. W. Berry. Downdating the latent semantic indexing model for conceptual information retreival. *The Computer Journal*, 41(1998), 1998.
[22] H. Zha and H. D. Simon. On updating problems in latent semantic indexing. *SIAM Journal on Scientific Computing*, 21(2):782–791), 1999.

A Low-rank modifications

Let $\mathbf{U}\,\mathrm{diag}(\mathbf{s})\,\mathbf{V}^\top \overset{\mathrm{SVD}_r}{\leftarrow} \mathbf{X}$ with $\mathbf{U}^\top\mathbf{U} = \mathbf{V}^\top\mathbf{V} = \mathbf{I}$ be a rank-r thin singular value decomposition (SVD) of matrix $\mathbf{X} \in \mathcal{R}^{p\times q}$. This appendix shows how to update $\mathbf{U},\mathbf{s},\mathbf{V}$ to the SVD of $\mathbf{X}+\mathbf{A}\mathbf{B}^\top$, where $\mathbf{A},\mathbf{B}$ have c columns. The original matrix $\mathbf{X}$ is not needed. Efficient rank-1 updates allow single columns (or rows) of $\mathbf{X}$ to be revised or deleted without the entire $\mathbf{V}$ (resp. $\mathbf{U}$) matrix.

Let $\mathbf{P}$ be an orthogonal basis of $(\mathbf{I}-\mathbf{U}\mathbf{U}^\top)\mathbf{A} = \mathbf{A} - \mathbf{U}\mathbf{U}^\top\mathbf{A}$, the component of $\mathbf{A}$ orthogonal to $\mathbf{U}$, as one would obtain from the QR-decomposition

$$[\mathbf{U},\mathbf{P}]\begin{bmatrix}\mathbf{I} & \mathbf{U}^\top\mathbf{A}\\ \mathbf{0} & \mathbf{R_A}\end{bmatrix} \overset{\mathrm{QR}}{\leftarrow} [\mathbf{U},\mathbf{A}], \tag{1.1}$$

which can computed via the modified Gram-Schmidt procedure (MGS) [9, §5.2.8]. $\mathbf{R_A}$ is upper-triangular. Similarly, let $\mathbf{Q}$ be an orthogonal basis of $\mathbf{B}-\mathbf{V}\mathbf{V}^\top\mathbf{B}$. Then

$$[\mathbf{U},\mathbf{P}]^\top(\mathbf{X}+\mathbf{A}\mathbf{B}^\top)[\mathbf{V},\mathbf{Q}] \tag{1.2}$$

$$= \begin{bmatrix}\mathrm{diag}(\mathbf{s}) & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix} + [\mathbf{U},\mathbf{P}]^\top\mathbf{A}\mathbf{B}^\top[\mathbf{V},\mathbf{Q}] \tag{1.3}$$

$$= \begin{bmatrix}\mathrm{diag}(\mathbf{s}) & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix} + \begin{bmatrix}\mathbf{U}^\top\mathbf{A}\\ \mathbf{R_A}\end{bmatrix}\begin{bmatrix}\mathbf{V}^\top\mathbf{B}\\ \mathbf{R_B}\end{bmatrix}^\top. \tag{1.4}$$

The goal is to rediagonalize equation (1.4). Let $\mathbf{U}'\,\mathrm{diag}(\mathbf{s}')\,\mathbf{V}'^\top$ be the rank-$(r+c)$ SVD of the right-hand side (RHS) of equation (1.4). Then the rank-$r+c$ update of the rank-r SVD is

$$\mathbf{U}''\,\mathrm{diag}(\mathbf{s}'')\,\mathbf{V}''^\top \doteq ([\mathbf{U},\mathbf{P}]\mathbf{U}')\,\mathrm{diag}(\mathbf{s}')\,([\mathbf{V},\mathbf{Q}]\mathbf{V}')^\top \tag{1.5}$$

$$= (\mathbf{X}+\mathbf{A}\mathbf{B}^\top). \tag{1.6}$$

Note that one never needs the original data matrix $\mathbf{X}$.

A.1 Rank-1 modifications Rank-1 updates offer special efficiencies: For the updated SVD of $\mathbf{X}+\mathbf{a}\mathbf{b}^\top$, expand the MGS of equation (1.1) to obtain $\mathbf{m} \doteq \mathbf{U}^\top\mathbf{a}$; $\mathbf{p} \doteq \mathbf{a}-\mathbf{U}\mathbf{m}$; $p \doteq \sqrt{\mathbf{p}^\top\mathbf{p}} = \sqrt{\mathbf{a}^\top\mathbf{p}}$; $\mathbf{P} = \mathbf{p}/p$ and similarly $\mathbf{n} \doteq \mathbf{V}^\top\mathbf{b}$; $\mathbf{q} \doteq \mathbf{b} - \mathbf{V}\mathbf{n}$; $q \doteq \sqrt{\mathbf{q}^\top\mathbf{q}} = \sqrt{\mathbf{b}^\top\mathbf{q}}$; $\mathbf{Q} = \mathbf{q}/q$. The rightmost term in equation (1.4) is then the outer vector product

$$\begin{bmatrix}\mathbf{U}^\top\mathbf{A}\\ \mathbf{R_A}\end{bmatrix}\begin{bmatrix}\mathbf{V}^\top\mathbf{B}\\ \mathbf{R_B}\end{bmatrix}^\top = \begin{bmatrix}\mathbf{m}\\ p\end{bmatrix}\begin{bmatrix}\mathbf{n}\\ q\end{bmatrix}^\top. \tag{1.7}$$

For example, if one wanted to change the first column of $\mathbf{X}$ to $\mathbf{y}$, then $\mathbf{b} = [1,0,0,\ldots]^\top$; $\mathbf{n}^\top$ is the first row of $\mathbf{V}$; $\mathbf{a} = \mathbf{y} - \mathbf{U}\mathrm{diag}(\mathbf{s})\mathbf{n}$ is $\mathbf{y}$ minus the first column of $\mathbf{X}$; $\mathbf{m} = \mathbf{U}^\top\mathbf{y} - \mathrm{diag}(\mathbf{s})\mathbf{n}$; $\mathbf{p}' \doteq \mathbf{y} - \mathbf{U}(\mathbf{U}^\top\mathbf{y})$, etc. To append a column $\mathbf{y}$ to the SVD, append a zero column to the original SVD by appending row of zeros to $\mathbf{V}$, then update that column to $\mathbf{y}$. In this case, $\mathbf{n} = \mathbf{0}$ and $q = 1$, so equation 1.4 asks us only to rediagonalize the broken-arrow matrix

$$\begin{bmatrix}\mathrm{diag}(\mathbf{s}) & \mathbf{m}\\ \mathbf{0} & p\end{bmatrix}, \tag{1.8}$$

which can be done in $O(r^2)$ time [10].

Setting $\mathbf{y} = \mathbf{0}$ effectively downdates the SVD by zeroing the column selected by $\mathbf{b}$. In this case RHS equation (1.4) simplifies to

$$\begin{bmatrix}\mathrm{diag}(\mathbf{s}) & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix} + \begin{bmatrix}\mathbf{U}^\top\mathbf{A}\\ \mathbf{R_A}\end{bmatrix}\begin{bmatrix}\mathbf{V}^\top\mathbf{B}\\ \mathbf{R_B}\end{bmatrix}^\top \tag{1.9}$$

$$= \begin{bmatrix}\mathrm{diag}(\mathbf{s}) & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}\left(\mathbf{I} - \begin{bmatrix}\mathbf{n}\\ 0\end{bmatrix}\begin{bmatrix}\mathbf{n}\\ \sqrt{1-\mathbf{n}^\top\mathbf{n}}\end{bmatrix}^\top\right). \tag{1.10}$$

$\mathbf{P}$ is unused, and $\mathbf{Q} = (\mathbf{b}-\mathbf{V}\mathbf{n})/\sqrt{1-\mathbf{n}^\top\mathbf{n}}$ is used only if updating $\mathbf{V}$. Note that downdating the i^{th} column only requires knowing the i^{th} row of $\mathbf{V}$.

The special structure and near diagonality of RHS equations (1.4–1.10) license additional numerical efficiencies. For example, let $\mathbf{J}$ equal RHS equation (1.10). Then $\mathbf{J}\mathbf{J}^\top =$

$\mathrm{diag}(\mathbf{s})^2 - (\mathrm{diag}(\mathbf{s})\mathbf{n})(\mathrm{diag}(\mathbf{s})\mathbf{n})^\top$ is a symmetric diagonal-plus-rank-1 matrix. for such matrices it is known [18, 9, section 8.5.3] that the eigenvalues—$\mathrm{diag}(\mathbf{s}')^2$—can be found quickly via Newton iterations for the roots of $f(s_i'^2) = 1 - \sum_j \frac{s_j n_j}{s_j^2 - s_i'^2}$, while the eigenvectors—$\begin{bmatrix}\mathbf{U}' & \mathbf{0}\\ \mathbf{0} & 1\end{bmatrix}$—are proportional to $(\mathrm{diag}(\mathbf{s})^2 - s_i'^2\mathbf{I})^{-1}\mathbf{n}$. Equation (1.7) leads to a diagonal-plus-rank-2 symmetric eigenproblem, requiring more sophisticated solution methods.

A.2 Controlling complexity If done naively, equation 1.1 takes $O(p(r+c)^2)$ time, the rediagonalization takes $O((r+c)^3)$ time, and the updates of the subspaces in equation 1.6 takes $O((p+q)(c+r)^2)$ time. In the setting of a rank-1 update of a fixed-rank SVD, these times can be reduced to $O(pr)$, $O(r^2)$, and $O(r^3)$, respectively, by expanding the MGS, performing a sparse diagonalization, and using the following trick: Instead of performing the large multiplications $\mathbf{U}'' = [\mathbf{U}, \mathbf{p}]\mathbf{U}'$, $\mathbf{V}'' = [\mathbf{V}, \mathbf{q}]\mathbf{V}'$ prescribed by equation 1.6, we leave the SVD decomposed into 5 matrices

$$\mathbf{U}_{p\times r} \cdot \mathbf{U}'_{r\times r} \cdot \mathbf{S}_{r\times r} \cdot \mathbf{V}'^\top_{r\times r} \cdot \mathbf{V}^\top_{r\times q} \tag{1.11}$$

and only update the smaller interior matrices $\mathbf{U}', \mathbf{V}'$. In the case where $\mathbf{a}$ is contained in the subspace of $\mathbf{U}$ and similarly $\mathbf{b} \in \mathbf{V}$, $\mathbf{p}$ and $\mathbf{q}$ can be ignored and the update is exact. Otherwise the information in $\mathbf{p}$ and $\mathbf{q}$ can be expressed as appends to $\mathbf{U}$ and $\mathbf{V}$ [3, appendix], or discarded under a fixed-rank approximation. As mentioned above, it can be shown [3] that for low-rank matrices ($r \sim O(\sqrt{p})$) the entire SVD can be computed in a series of updates totaling $O(pqr)$ time.

B Missing values

Consider a nonzero rectangular volume $\mathcal{V}$ of possible updates specified by opposite corners $\mathbf{y}$ and $\mathbf{z}$. Let z_i be the i^{th} element of $\mathbf{z}$. Assuming a uniform measure in this space, the volume's second moment is

$$\Sigma_{\mathcal{V}} \doteq \mathrm{cov}(\mathcal{V}) = \Big(\int_{\mathbf{x}\in\mathcal{V}} \mathbf{x}\mathbf{x}^\top \mathrm{d}\mathbf{x}\Big)/\Big(\int_{\mathbf{x}\in\mathcal{V}} 1\,\mathrm{d}\mathbf{x}\Big), \tag{2.12}$$

where the normalizing quotient is $\prod_{i|z_i\neq y_i} |z_i - y_i|$. Here the origin is taken to be the data mean, so equation (2.12) is interpreted as a covariance. Any dimension in which $y_i = -z_i$ can be dropped (drop element y_i from $\mathbf{y}$ and similarly for z_i); symmetric bounds are uninformative, forcing the imputed value in that dimension to be 0. Similarly, drop dimensions for which $y_i = z_i$; no imputation is needed. Expanding the integrals, we find that diagonal elements of $\Sigma_{\mathcal{V}}$ are $(y_i^2 + y_i z_i + z_i^2)/3$ and off-diagonal elements are $(y_i + z_i)(y_j + z_j)/4$, or

$$\Sigma_{\mathcal{V}} = (\mathbf{y}+\mathbf{z})(\mathbf{y}+\mathbf{z})^\top/4 + \mathrm{diag}(\mathbf{w})/12, \tag{2.13}$$

where $w_i \doteq (y_i - z_i)^2$. This is a $k \times k$ diagonal-plus-rank-1 matrix where k is the number of dimensions in which $y_i \neq \pm z_i$. $\Sigma_{\mathcal{V}}$ has EVD $\mathbf{W}\Lambda\mathbf{W}^\top = \Sigma_{\mathcal{V}}$ that can be computed in $O(k^2)$ time directly from the vectors $\mathbf{y}+\mathbf{z}, \mathbf{y}-\mathbf{z}$, using the Newton method mentioned above. Updating the SVD $\mathbf{U}\mathbf{S}\mathbf{V}^\top = \mathbf{X}$ with k vectors whose missing values are set to the columns of $\mathbf{W}\Lambda^{1/2}$ will duplicate the second-order statistics of $\mathbf{X} \cup \mathcal{V}$, and therefore completes the imputative update. E.g., it is equivalent to updating the SVD with n i.i.d. samples from $\mathcal{V}$, each scaled by $\sqrt{n}$, as $n \to \infty$. A single update using just the column of $\mathbf{W}\Lambda^{1/2}$ with the largest norm will give the best single-vector approximation of the imputation.

This approach becomes more powerful when the uniform measure $\mathrm{d}\mathbf{x}$ in equation (2.12) is replaced with a more informative measure, e.g., the running estimate of data density $\mathrm{d}\mathcal{N}(\mathbf{x}|\mathbf{0}, \Sigma)$ discussed below. The integrals are solvable in closed form. If Σ is a dense (nondiagonal) covariance, even symmetric bounds become informative.

B.1 Probabilistic imputation Consider adding a vector $\mathbf{c}$ with missing values. 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

$$\begin{aligned}\hat{\mathbf{c}}_\circ &\leftarrow \mathbf{U}_\circ \mathrm{diag}(\mathbf{s})(\mathrm{diag}(\mathbf{s})\mathbf{U}_\bullet^\top \mathbf{U}_\bullet \mathrm{diag}(\mathbf{s}))^+ (\mathrm{diag}(\mathbf{s})\mathbf{U}_\bullet^\top \mathbf{c}_\bullet)\\ &= \mathbf{U}_\circ \mathrm{diag}(\mathbf{s})(\mathbf{U}_\bullet \mathrm{diag}(\mathbf{s}))^+ \mathbf{c}_\bullet,\end{aligned} \tag{2.14}$$

yields the completed vector $\hat{\mathbf{c}}$ that lies the fewest standard deviations from the density of the (centered) data, as modelled by a Gaussian density $\mathcal{N}(\mathbf{x}|\mathbf{0}, \Sigma)$, where $\Sigma \doteq \mathbf{U}_\bullet^\top \mathrm{diag}(\mathbf{s})\mathbf{U}_\bullet$ is a low-rank approximation of the covariance of the data seen thus far ($\mathbf{X}^+$ denotes Moore-Penrose pseudo-inverse). Substituting equation 2.14 into equation 1.8 yields

$$\begin{bmatrix}\mathrm{diag}(\mathbf{s}) & \mathbf{U}^\top\hat{\mathbf{c}}\\ 0 & p\end{bmatrix} \tag{2.15}$$

$$= \begin{bmatrix}\mathrm{diag}(\mathbf{s}) & \mathrm{diag}(\mathbf{s})(\mathbf{U}_\bullet \mathrm{diag}(\mathbf{s}))^+ \mathbf{c}_\bullet\\ 0 & \|\mathbf{c}_\bullet - \mathbf{U}_\bullet \mathrm{diag}(\mathbf{s})(\mathbf{U}_\bullet \mathrm{diag}(\mathbf{s}))^+ \mathbf{c}_\bullet\|\end{bmatrix}, \tag{2.16}$$

where $\mathbf{U}^\top\hat{\mathbf{c}}$ is the projection of the imputed vector onto the left singular vectors and p is the distance of the vector to that subspace. As one might expect, with missing data it rarely happens that $p > 0$.