Mathematical Modeling and Algorithm Application
ISSN: 3006-0842 | Vol. 10, No. 1, 2026

Research on Dimensionality Reduction Algorithms for High-Dimensional Data Based on Matrix Analysis

Qianwen Wu

Chengdu University of Technology, Yibin, Sichuan 644000, China

Abstract: More and more high dimensional data can be found in many fields including genomics, computer vision, NLP, RS etc. Therefore Dimensionality Reduction is important in most of the Machine Learning and Data analysis pipelines. This article carries out a comprehensive research on different types of dimensionality reduction algorithms which are based both in linear and nonlinear paradigms. This study includes both linear and nonlinear paradigms: In this section we have discussed about the mathematical foundations of PCA, LDA, NMF, SparsePCA with respect to consider it as singular value decomposition and eigen-decompostion for the structered matrix. Regarding non-linear methods, we consider the graph - Laplacian and manifold - geometric matrix framework that is the foundation of Isomap, LLE, Laplacian eigenmaps, t - SNE, and UMAP. And we also compare computational complexity and scalability approximation(random svds, incremental). We benchmark all algorithms on commonly used datasets(MNIST, 20newsgroups, swissroll, cifar) regarding their reconstruction quality, speed and how well they enable us to classify things. For randomized svd the randomization gives us very good approximations on a small amount of the full decomposition computing time and UMAP seems to have the best mix of scale and shape when using nonlinear approaches. Proposed some practical advice for algorithm choices according to information size, internal dimension and task necessities

Keywords: Dimensionality Reduction; Matrix Analysis; Principal Component Analysis; Singular Value Decomposition; Manifold Learning; Randomized Algorithms; High-Dimensional Data

1. Introduction

Modern science and engineering often comes in high dimensions. In genomics it may be that an RNA seq experiment looks at the expression level of many tens of thousands of genes and in computer vision an image is just a vector of 100k+ pixels, in NLP you have your documents as vectors on the order of thousands to millions. Bellman(1961) introduced “Curse of Dimensionality”, it describes how difficult is statistical & computational when there are more dimension. When dimensionality rises, data gets sparse, similarities among samples lose meaning as do many usual ways which becomes unfeasible or simply impossible[1]. Dimensionality Decrease: find some lower but condensed representation holding onto most of the highest data structures is first necessity so we can analyze them, see them & learn from them.

Matrix analysis is the right math language that help you understand what dimension reduction algorithm do. SVD will break down the data matrix into 3 structured form of matrices, Orthogonal Columns Space, Diagonal Singular Values and Orthonormal Row Space (which gives us the best low rank approx under Frobenious norm) Covariance and scatter as well as Gram and Graph Laplacian Matrixes Eigen Decomposition goes from classic PCA straight on to modern day manifold techniques [2]. Knowing about the matrix form used by all the algorithms as well is necessary to see if theories approximate reality, developing better ways for computer methods that adapt themselves according to a lot of real information.

Previous writings mostly compare algorithm by itself. By giving the same treatment to matrices we may make either quantitative / or/ even anempirical compare of ten unique algorithms (5 linear, 5nonlinear) based on them. I’m going to do like a little added-in uni matrix and analyze it in order to compare these 10 completely different ones – five linears/five non’s – with more complicated analysis and even limited benchmarks. Structure: section two is set as groundwork matrix, section three for linesars, section four is on nonlinear manifold study, section five is about computational issue and scaleable stuff, section six with benchmarks, ending it off in part seven by pointing which of these should be chosen.

2. Mathematical Foundations of Matrix-Based Dimensionality Reduction

The singular value decomposition provides the unifying mathematical framework for linear dimensionality reduction. Given a data matrix $X \in \mathbb{R}^{N\times d}$, the SVD decomposes $X = U\Sigma V^T$, where $U \in \mathbb{R}^{N\times N}$ and $V \in \mathbb{R}^{d\times d}$ are orthonormal matrices of left and right singular vectors, and Σ contains non-negative singular values $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r \geq 0$. The Eckart-Young-Mirsky theorem guarantees that the rank-k truncation $X_k = U_k \Sigma_k V_k^T$ is the best rank-k approximation to X in both the Frobenius and spectral norms [3]. This optimality property makes truncated SVD the cornerstone of PCA, latent semantic analysis, and a wide range of matrix factorisation methods for dimensionality reduction.

For supervised learning, LDA uses the eigendecomposition of between-class and within-class scatter matrices. In manifold learning, the normalized graph Laplacian $L = D^{-1/2}(D - W)D^{-1/2}$, where W is the adjacency matrix of nearest neighbors and D is the degree matrix, provides a discrete approximation to the Laplace–Beltrami operator on a smooth manifold[4]. L's eigenvectors with small positive eigenvalues but not zero gives coordinates where locals neighbourhoods did a good job at minimising the square (like it would in laplacian eignmaps, or some others). Using NMF we push the boundary even more by also forcing factorization on both matrices to be non negative, yielding very interpretable parts

based representations for faces, document - term matrices with additive structure and so forth.

3. Linear Dimensionality Reduction Methods

Linear dimensionality reduction techniques try to find a projection matrix W ∈ R^d×k such that the projected representations Z=XW keep some criterion (variance, class separability, sparsity, how well it can be reconstructed) in the k-dimension target space. We can observe the comparison results of these 5 frequently used linear model from Table 1 in terms of basic Matrix format and Target function for minimization, Supervision and limitations. From a matrix point of view we can see that there are some interesting relationships: PCA, SparsePCA and Random Projection work with the full data matrix or the Grammatrix of the data instead; LDA works with ratios between different scattermatrices; NMF replaces the unconstrainedSVD by the constrainedbiconvexoptimisation.

Table 1. Comparative Overview of Linear Dimensionality Reduction Methods

Method	Matrix Basis	Optimisation Objective	Supervised?	Key Limitation
PCA	Covariance / SVD	Maximise variance (L_2)	No	Linear only; sensitive to outliers
LDA	Scatter matrices	Maximise class separability	Yes	At most C-1 dimensions (C = classes)
NMF	Non-negative factors	Minimise reconstruction (L_1)	No	Non-unique solution; slow convergence
Sparse PCA	Penalised covariance	Variance + sparsity (LASSO)	No	Computationally expensive at scale
Random Proj.	Random Gaussian matrix	Johnson-Lindenstrauss bound	No	No structure preserved; approx only

Note: N = number of samples; d = original dimension; k = target dimension; C = number of classes. L_2/L_1 refer to reconstruction loss norms. LASSO = Least Absolute Shrinkage and Selection Operator.

From table 1, we know that PCA, proposed in 1933 by Hotelling is still one of the most widely applied Linear Dimensionality Reduction method due to its closed form SVD solution, optimality guarantee on Gaussian Data as well as computability[1]. The most variance direction is provided by the top eigenvector of the sample covariance matrix C = (1/N)X^TX; the rest are orthogonal principal components sorted in decreasing eigenvalue order. LDA shoves things even further away in the supervision direction: It seeks to maximize over instead, where it solves this generalized eigenvalue problem S_B w = λ S_W w [2] The major shortcoming of LDA is that at most C - 1 informative projective directions can be generated where C denotes the number of class. Random projection method uses JL Lemma which does a random Linear projectoin to $O(\varepsilon^{-2} \log N)$ dimensiosn, thus keeps All PAir Distance within factor $(1 \pm \varepsilon)$—enabling extremely fast dimensionality reduction with no training phase.

Sparse PCA extends classic PCA in that it adds an L1 penalty on the loadings vectors, so the principal components are linear combinaisons of a few original variables. It's good for high dimensional biological data, every component would ideally be mapped back to an interpretable gene pathway instead of a mix of all the things we're measuring. For NMF: X ~W H Where bothW and H needs to be>= 0 and we will use multiplicative updatingrulesto factorizeinto additivitypartsthat is common instext mining,hyer spectralimage analysisthe audio source seperation[6]. And they do so at the cost of more computation ease, hence whichever app gets prioritized depends purely upon whether it's more about being easily understood or rebuilt accurately.

4. Nonlinear Dimensionality Reduction Methods

High Dimension Dataset is a nasty beast, it lives and breaths in its own intrinsic Manifold inside a larger High-Dimensional Ambient Space; if you just throw that into some random Linear Projection without much thought your original variations already on your data will be squashed and flatten over those Curves Manifold learning techniques overcome it by these sort of matrix that show this geometrical relation for each point, which is usually presented like a set of adjacent point to approximate a difference geometry between manifolds. Table 2 compares 5 other non-linear dimensionality reduction techniques on what type of matrix construct do they use, which Geometrical Property do they preserve, Time Complexity, Scalability and Main Purpose.

Table 2. Comparative Overview of Nonlinear Dimensionality Reduction Methods

Method	Matrix Construct	Geometry Preserved	Time Complexity	Scalable?	Best Use Case
Isomap	Geodesic distance	Global manifold	$O(N^3)$	No	Swiss roll, curved surfaces
LLE	Local weight matrix	Local structure	$O(N^2d)$	No	Neighbourhood-smooth manifolds
LE	Graph Laplacian	Spectral geometry	$O(N^2)$	No	Cluster/graph-structured data
t-SNE	P/Q probability matrix	Local + clusters	$O(N^2)$	No	2D/3D visual exploration
UMAP	Fuzzy simplicial set	Local + global	O(N^1.14)	Yes	Large-scale general embedding

Note: N = number of samples; d = original dimension. Time complexities are approximate and may vary with implementation.

LE = Laplacian Eigenmaps. UMAP complexity O(N^1.14) refers to the approximate nearest-neighbour graph construction phase.

And then there is the Iso map from Tenenbaum et al. 2000, that uses some form of approximate thing with shortest path between neighbors on a geodesic distance on the manifold and then does classical MDS to it.[3] correctly flattens the swiss roll which linear pca messes up. But it's limited by the cubic in N time for Iso map due to the all pairs shortest path calculation. From the same issue of that landmark science there was a locally Linear method, Roweis and Saul (2000), where they tried to figure out what the lower dimension embedding would be such that every point was precisely one of its Knn. For which we need to obtain N by solving a sparse N×N eigenvector problem, so the LLE is also very dependent on neighborhood sizes and if there are holes within the manifold with lots of curvature.

Laplacian eigenmaps method is first put forward in 2003 by Belkin andNiyogiembedding to eigenvectors' normalization graphlaplacians for Nearest –Neighbour similarity -graph that minimize sum square of difference of embeddings of neighboring point weightedwith respect to its corresponding graph-edge weight[5]. This is the objective giving a very strict discret approx to the continousLaplace - Beltrami operator on the underlying manifold. Van der maatenand Hinton(2008)' Ts - Tnemodels high dimensionalsimilarities w/ student - t low dimensional probability as Guassian for inputminimizing their KL divergence gives us a pretty good cluster sep inthe viz, whichis only visiblein twodimensions, andit certainly doesn'tgives youany indicationof what'staking place on your fulldatasetthis algo costs O(n ** 2), so it's usually limited by samplesize ~100kor less[6]. UMAP: McInnes et al. (2018) created it. umap makes weight fuyzi simsimset representtions of the topology in data And, over overhigd & lowd topos: Cross-Entropy loss fucn. so by that way to keep better the local neighberhood strucrue and the global topologies at diferent scale through approximative neighbors anst stochastic graidents[7].

5. Computational Complexity and Scalability Analysis

5.1. Time and Space Complexity Comparison

To put the algorithms into use on a large scale, we have to do the analysis of its time and space complexity with respect to sample size N, original dimension d, target dimension k. Table 3 provides a more extensive comparison over the above dimension of all six algorithm family with the SVD variant being used, parallelizable and the data scale it's best at. This is important to practitioners, who have to pick an algorithm (or combination of algorithms) that works with their computing environment and size of the dataset; there could be a big difference in how well things work out when Nanddare both in millions.

Table 3. Time and Space Complexity of Dimensionality Reduction Algorithms

Algorithm	Time Complexity	Space Complexity	SVD Variant	Parallelisable?	Suitable Data Scale
Exact PCA	O(Nd·min(N,d))	O(d^2)	Full SVD	Partial	N, d < 10^4
Truncated SVD	O(Ndk)	O(Nk+dk)	Partial SVD	Yes	N < 10^6, d < 10^5
Randomized SVD	O(Nd log k)	O(Nk+dk)	Randomiszd	Yes	N, d > 10^6
Kernel PCA	O(N^2d+N^3)	O(N^2)	Eigendecomp	No	N < 5×10^3
Incremental PCA	O(d^2·batch)	O(dk)	Batch SVD	Yes	Streaming/online
Sparse Random Proj.	O(Nds/d)	O(Nk)	None	Yes	Very high-d (d>10^6)

Note: k = target reduced dimension; batch refers to mini-batch size for incremental PCA; s = sparsity level of the random projection matrix. Complexities are worst-case theoretical bounds; actual runtimes depend heavily on implementation, hardware, and data structure.

From Table 3, we can see that the exact PCA via full SVD is O(Nd×min(N, d)) with cubic time complexity in min(N, d)—which is not feasible for current large-scale datasets. And then with the advent of Randomized SVD which is introduced by work of Halko, Martinsson and Tropp(2011) which was another huge leap where instead they created a lower dimensional random sketch of the datamatrix through some number of matrix-vector multiplications and before taking thesvdofthe matrix, they tookthe svd ofthis far smaller sketch which had the same runtime asbefore butnow ran on adklgo(k). Whileit remained closetoin error tothe optimal rank k-truncationwithconstant-factoraccuracy whithhighprobability[8]. It means that randomized SVD is the go-to method when doing large scale PCA in practice: it is at the heart of the scikit-learn implementation of Truncated SVD as well as many other numerical computing libraries' LAPACK-based routines.

5.2. Randomiszd and Approximate Decomposition Strategies

The randomized SVD algorithm proceeds in two phases. In the first phase, a random Gaussian test matrix $\Omega \in \mathbb{R}^{(d\times(k+p))}$ is multiplied by X to form a sketch $Y = X\Omega$, and the column space of Y is orthonormalised via QR decomposition to obtain $Q \in \mathbb{R}^{(N\times(k+p))}$. In the second phase, a small matrix $B = Q^T X$ is formed and its exact SVD computed. This costs O(Ndk) floating-point operations in total for fixed oversampling p [8]. To speed up the algorithm, do 2 - 3 power itera-tion steps: do X multiply and then X transp. multiply in alternate order, before sketching. This zeros out the little svd values which lead to approx. err, so it is close even if s have slowly decaying svd spectrum. With incremental PCA takes advantage with online by doing it on batches to both datas, also it do an increment update for

decompositions but keep them in size of O(dk) instead of whole data set.

6. Experimental Performance Evaluation

From the side of the experiments we would also like to see what it tells us about this. Use some benchmark set on all these where first 2 are normal ones: 1. MNIST dataset contains handwritten digit image having n=60k, =784. 2. 20 newsgroups text corpus containing n=18k, d=61K and last but not the least is the popular cifar - 10 image classification data set having n=50k images(D=3072). All of them were carried out in python 3. 11 along with num, scikit-learn and umap - learn run in workstation that have intel core i9-13900k processor and 64gb ram. Table gives: 1. Reconstruction error 2. Wall clock time (s), 3. down stream linear classification using logistic - regression classifier wrt to reduce dimension rep of each alg −dataset.

Table 4. Benchmark Experimental Results for Dimensionality Reduction Algorithms

Algorithm	Dataset	Orig. Dim	Target Dim	Reconstruction Error	Runtime (s)	Classification Acc. (%)
PCA	MNIST	784	50	0.042	1.3	97.8
Sparse PCA	MNIST	784	50	0.061	18.7	97.2
Truncated SVD	20Newsgrp	61,188	100	0.038	4.1	85.4
Randomized SVD	20Newsgrp	61,188	100	0.041	0.9	84.9
Kernel PCA	Swiss Roll	3	2	0.003	22.4	—
UMAP	CIFAR-10	3072	64	—	38.2	81.6
t-SNE	CIFAR-10	3072	2	—	312.0	—

Note: Reconstruction error is reported as normalised Frobenius norm of the reconstruction residual. '—' denotes that the metric is not applicable (e.g., no reconstruction for probabilistic embeddings, no classification for purely visualisation-oriented settings). All runtimes are wall-clock seconds averaged over five independent runs.

As seen from Table 4 randomized SVD on 20 newsgroups gives reconstruction error(0.041) slightly worse compared to truncated svd(0.038). However this happens at 4.6x reduced runtime (0.9s vs 4.1s), proving that randomized approximation is fine when it comes down to accuracy loss in real world practice: And on MNIST: PCA (97.8%) with 50 components is just barely better than Sparse PCA (97.2%), and also much faster at 1.3 s vs 18.7 s. On the CIFAR-10 UMAP results in 81.6 % classification accuracy for 64 dimension out perform t-SNE on down stream learnings; t-SNE takes 312s(＞8x）of umap run time and it's 2D embeddings are not proper for downsteem classifiers[9] .Kernel PCA restores the Swiss Roll structure almost perfectly with zero reconstruction error (0.003), which shows it is faithful to geometry in curved manifolds, albeit taking 22.4s, that will get worse as N gets bigger because Kernel PCA's construction creates an $O(N^2)$ kernel matrix construction.

Experimental result is some convenient practical algo picking advice. If you want to do it very big dimensionally with high accuracy use randomizedsvd because they will be way faster then the exact method but still pretty accurate for interpretable stuff when it matters like understanding the interpretation of gene-expression or a doc-term-matrix through - PCA/ NMF: -NMF look at the complex manifold by using umap instead of tsne because its more scaleable and keep all the global info [10]: Supervised set with class labels: LDA does well with unsupervised if there are a lot more classes than beginning dimensions. The intrinsic dimension of unknowable states were played an equal role alongside Eigenvalue gap, which is used alongwith graph Laplacian to make estimation.

7. Conclusion

Has been looked at completely under the one umbrella Matrix analysis like so in this particular article: High dimensional Data are studied well. From here we can see that its via SVD and the proper matrix eigendecomposition – of the covariances, scatters, Graphs Laplacian or even fuzzy tops – were the math happens on which precise linear as well as non-linear dim reduction goals depend on. The 5 analized linear methods (PCA, LDA, NMF, spPCA, Rp), the 5 non linear isomap, lle, le, t-sne, umap all have different set of tradeoffs between optmality and understandbility and compuation cost and scaliblity measured with number from 1 - 3 in every tables.

From the benchmark experiments in Table 4 we see that randomized SVD is almost exact and do the calculations orders of magnitudes faster than complete decompositions, so it can be used for the large scale linear dimensionality reduction. Non-linear amongst them all UMPAT has the best trade-off between speed and scale and retain of strcutre, and kernel pCA is still the most geoemtrically truistful for well charachterised curved manifolds of mediumsized dataset. It turns out the k is very important to select for these tasks – scree - plot style inspecting singular values spectra and cross valid accuracy are much more successful compared with just randomly picking some number.

There are some still-open research fields. Theoretical analysis of UMAP approximation guarantee, such as when does the fuzzy simplicial set construction tell the truth about the secret shape of the data, is also underway. Extend randomized matrix decomp even more to more structured HD data sets like multi mode learning, video analysis, quantumchemistry etc and go much further with randomization then we had for tensor train/tuckter structers

today. Third one is to have the diminishment be part of another learning goal, like optimizing proj matrix & your class/reg together or just thinking about it as preproc - which means maybe we get results that are actually worth doing on some task. And as high dimensional data keeps getting produced in science and engineering, matrix analysis based dimensionality reduction is going to remain both a problem and a place for ML to shine.

References

[1] Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. Journal of Educational Psychology, 24(6), 417–441. https://doi.org/10.1037/h0071325

[2] Jolliffe, I. T., & Cadima, J. (2016). Principal component analysis: a review and recent developments. Philosophical Transactions of the Royal Society A, 374(2065), 20150202. https://doi.org/10.1098/rsta.2015.0202

[3] Tenenbaum, J. B., De Silva, V., & Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. Science, 290(5500), 2319–2323. https://doi.org/10.1126/science.290.5500.2319

[4] Roweis, S. T., & Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. Science, 290(5500), 2323–2326. https://doi.org/10.1126/science.290.5500.2323

[5] Belkin, M., & Niyogi, P. (2003). Laplacian eigenmaps for dimensionality reduction and data representation. Neural Computation, 15(6), 1373–1396. https://doi.org/10.1162/089976603321780317

[6] Van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. Journal of Machine Learning Research, 9(11), 2579–2605.

[7] McInnes, L., Healy, J., & Melville, J. (2018). UMAP: Uniform manifold approximation and projection for dimension reduction. Journal of Open Source Software, 3(29), 861. https://doi.org/10.21105/joss.00861

[8] Halko, N., Martinsson, P. G., & Tropp, J. A. (2011). Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. SIAM Review, 53(2), 217–288. https://doi.org/10.1137/090771806

[9] Böhm, J. N., Berens, P., & Kobak, D. (2022). Attraction-repulsion spectrum in neighbor embeddings for dimensionality reduction. Journal of Machine Learning Research, 23(95), 1–32.

[10] Damrich, S., Berens, P., & Kobak, D. (2025). From t-SNE to UMAP with contrastive learning. International Conference on Learning Representations.