Probabilistic Matrix Analysis

Hekmat Taherinejad, Rahim Tariverdi, Hamidreza Behjoo, Kodirjon Akhmedov

December 21, 2019

Contents

1 Appetizer

Singular Value Decomposition (SVD) is a useful framework for dimensionality reduction tasks. Consider Principal Component Analysis (PCA), which, given a centered matrix of m observations and p covariates $X \in R^{m \times p}$, computes $W \in R^{p \times l}$ and $\Lambda \in R^{l \times l}$ such that $X^T X \approx W \Lambda W^T$. This maps the original observations from a p-dimensional space into an l-dimensional space, for $l \leq p$, thus performing dimensionality reduction. One algorithm for computing a reduced PCA is via a low-rank matrix approximation. Take $X \approx U_r \Sigma_r V_r^T$, and then take $X^T X \approx (U_r \Sigma_r V_r^T)^T (U_r \Sigma_r V_r^T) = V_r \Sigma_r^2 V_r^T$ by the orthogonality of U_r. Thus computing the SVD of A immediately yields the PCA of A.

These techniques have a huge number of applications in machine learning, data mining, bioinformatics, and even political science – from predicting hashtags from the content of a status by factorizing their co-occurrence matrix, to understanding political voting patterns by factorizing the matrix of legislators and bills.

Unfortunately, computing the SVD can be extremely time-consuming for the large-scale problems. Thus, we turn to randomized methods which offer significant speedups over classical methods.

2 Introduction

Indeed, matrix decompositions are often the workhorse algorithms for scientific computing applications in the areas of applied mathematics, statistical computing, and machine learning. Despite our ever-increasing computational power, the emergence of large-scale datasets has severely challenged our ability to analyze data using traditional matrix algorithms. Moreover, the growth of data collection is far outstripping computational performance gains. The computationally expensive singular value decomposition (SVD) is the most ubiquitous method for dimensionality reduction, data processing and compression. The concept of randomness has recently been demonstrated as an effective strategy to easing the computational demands of low-rank approximations from matrix decompositions such as the SVD, thus allowing for a scalable architecture for modern big data applications.

The basic idea of probabilistic matrix algorithms is to employ a degree of randomness in order to derive a smaller matrix from a high-dimensional matrix, which captures the essential information. Thus, none of the randomnes should obscure the dominant spectral information of the data as long as the input matrix features some low-rank structure. Then, a deterministic matrix factorization algorithm is applied to the smaller matrix to compute a near-optimal low-rank approximation. The principal concept is sketched in Figure 1.

3 Probabilistic framework for low-rank approximations

In the following, we advocate the probabilistic framework, formulated by Halko et al. (2011), to compute a near-optimal low-rank approximation. Conceptually,

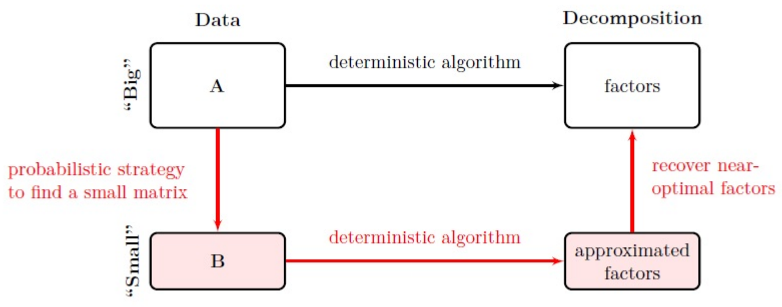


Figure 1: First, randomness is used as a computational strategy to derive a smaller matrix B from A. Then, the low-dimensional matrix is used to compute an approximate matrix decomposition. Finally, the near-optimal (high-dimensional) factors may be reconstructed.

this framework splits the computational task into two logical stages:

- Stage A: Construct a low dimensional subspace that approximates the column space of A. This means, the aim is to find a matrix $Q \in R^{m\times k}$ with orthonormal columns such that $A \approx QQTA$ is satisfied.
- Stage B: Form a smaller matrix $B = Q^T A \in R^{k\times n}$, i.e., restrict the high-dimensional input matrix to the low-dimensional space spanned by the near-optimal basis Q. The smaller matrix B can then be used to compute a desired low-rank approximation.

The first computational stage is where randomness comes into the play, while the second stage is purely deterministic. In the following, the two stages are described in detail.

3.1 The generic randomized algorithm

Stage A: Computing the near-optimal basis

First, we aim to find a near-optimal basis Q for the matrix A such that $A \approx QQ^T A$ is satisfied. The desired target rank k is assumed to be $k \ll \min{(m, n)}$. Specifically, $P := QQ^T$ is a linear orthogonal projector. A projection operator corresponds to a linear subspace, and transforms any vector to its orthogonal projection on the subspace. This is illustrated in Figure 2, where a vector x is confined to the column space col(A).

The concept of random projections can be used to sample the range (column space) of the input matrix A in order to efficiently construct such a orthogonal projector. Random projections are data agnostic, and constructed by first drawing a set of k random vectors $[\omega_i]_{i=1}^k$, for instance, from the standard normal distribution. Probability theory guarantees that random vectors are linearly independent with high probability. Then, a set of random projections $[y_i]_{i=1}^k$ is computed by mapping A to low-dimensional space:

$$y_i = \mathbf{A}\omega_i, \quad for\ i = 1...k \tag{1}$$

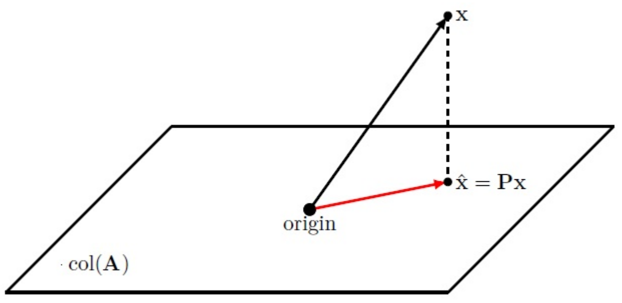


Figure 2: Geometric illustration of the orthogonal projection operator P. A vector $x \in R^m$ is restricted to the column space of A, where $Px \in col(A)$.

In other words, this process forms a set of independent randomly weighted linear combinations of the columns of A, and reduces the number of columns from n to k. While the input matrix is compressed, the Euclidean distances between the original data points are approximately preserved. Random projections are also well known as the Johnson-Lindenstrauss (JL) transform (Johnson and Lindenstrauss 1984).

Equation 3 can be efficiently executed in parallel. Therefore, let us define the random test matrix $\Omega \in R^{n \times k}$, which is again drawn from the standard normal distribution, and the columns of which are given by the vectors ω_i. The samples matrix $Q \in R^{m \times k}$, also denoted as sketch, is then obtained by post-multiplying the input matrix by the random test matrix $Y := A\Omega$ Once Y is obtained, it only remains to orthonormalize the columns in order to form a natural basis $Q \in R^{m \times k}$ This can be efficiently achieved using the QR-decomposition $Y =: QR$.

Stage B: Compute the smaller matrix

Now, given the near-optimal basis Q, we aim to find a smaller matrix $B \in R^{k \times n}$. Therefore, we project the high-dimensional input matrix A to low-dimensional space

$$B := Q^T A \tag{2}$$

Geometrically, this is a projection (i.e., a linear transformation) which takes points in a high-dimensional space into corresponding points in a low-dimensional space, illustrated in Figure 3.

This process preserves the geometric structure of the data in an Euclidean sense, i.e., the length of the projected vectors as well as the angles between the projected vectors are preserved. So finally we have:

$$A \approx QB \tag{3}$$

This decomposition is referred to as the QB decomposition. Subsequently, the smaller matrix B can be used to compute a matrix decomposition using a traditional algorithm.

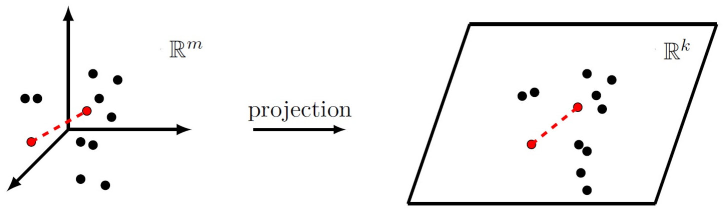


Figure 3: Points in a high-dimensional space are projected into low-dimensional space, while the geometric structure is preserved in an Euclidean sense.

3.2 Improved randomized algorithm

The basis matrix Q often fails to provide a good approximation for the column space of the input matrix. This is because most real-world data matrices do not feature a precise rank r, and instead exhibit a gradually decaying singular value spectrum. The performance can be considerably improved using the concept of oversampling and the power iteration scheme.

Oversampling
Most data matrices do not feature an exact rank. Oversampling can be used to overcome this issue by using more random projections to form the sketch, instead of just k. The intuition behind the oversampling scheme is the following. The sketch Y is a random variable, as it depends on the drawing of a random test matrix . Increasing the number of additional random projections allows one to decrease the variation in the singular value spectrum of the random test matrix, which subsequently improves the quality of the sketch.

Power iteration scheme
The second method for improving the quality of the basis Q involves the concept of power sampling iterations . Instead of obtaining the sketch Y directly, the data matrix A is first preprocessed as

$$A^{(q)} := (AA^T)^q A \tag{4}$$

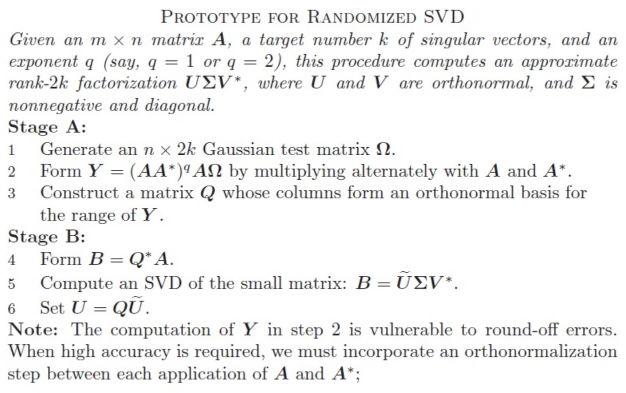
where q is an integer specifying the number of power iterations. This process enforces a more rapid decay of the singular values. Thus, we enable the algorithm to sample the relevant information related to the dominant singular values, while unwanted information is suppressed. Finally the algorithm to calculate our randomized SVD is shown in Figure 4.

3.3 Error Analysis and Number of Computation

In this section we give bounds on the error of the proposed randomized SVD and find number of flops required to compute it.

3.3.1 Error Analysis

we refer to the theorem 2.1 by Halko et al. (2011) and restate it here for future use.


Prototype for Randomized SVD

Given an $m \times n$ matrix $\boldsymbol{A}$, a target number k of singular vectors, and an exponent q (say, $q = 1$ or $q = 2$), this procedure computes an approximate rank-$2k$ factorization $\boldsymbol{U\Sigma V}^*$, where $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthonormal, and $\boldsymbol{\Sigma}$ is nonnegative and diagonal.

Stage A:

```
1  Generate an $n \times 2k$ Gaussian test matrix $\boldsymbol{\Omega}$.
2  Form $\boldsymbol{Y} = (\boldsymbol{AA}^*)^q \boldsymbol{A\Omega}$ by multiplying alternately with $\boldsymbol{A}$ and $\boldsymbol{A}^*$.
3  Construct a matrix $\boldsymbol{Q}$ whose columns form an orthonormal basis for
   the range of $\boldsymbol{Y}$.
```

Stage B:

```
4  Form $\boldsymbol{B} = \boldsymbol{Q}^*\boldsymbol{A}$.
5  Compute an SVD of the small matrix: $\boldsymbol{B} = \widetilde{\boldsymbol{U}}\boldsymbol{\Sigma V}^*$.
6  Set $\boldsymbol{U} = \boldsymbol{Q}\widetilde{\boldsymbol{U}}$.
```

Note: The computation of $\boldsymbol{Y}$ in step 2 is vulnerable to round-off errors. When high accuracy is required, we must incorporate an orthonormalization step between each application of $\boldsymbol{A}$ and $\boldsymbol{A}^*$;

Figure 4: The algorithm to calculate our randomized SVD.

Theorem
Suppose that A is areal $m \times n$ matrix. select an exponent q and a target number k of singular vectors, where $2 \leq k \leq 0.5 \min(m, n)$. Execute the randomized SVD algorithm to obtain a rank-$2k$ factorization of $U\Sigma V^*$ Then:

$$\mathbb{E}||A - U\Sigma V^*|| \leq [1 + 4\sqrt{\frac{2\min(m,n)}{k-1}}]^{\frac{1}{2q}}\sigma_{k+1} \tag{5}$$

where $\mathbb{E}$ denotes the expectation with respect to the random test matrix and σ_{k+1} is the $(k+1)$th singular value of A.

3.3.2 Flops Count

Number of flops required for traditional SVD is $\min(mn^2, nm^2)$, while in the randomized SVD we have $mn \log k + k^2(m+n)$. when k in much smaller than m and n it really beats traditional SVD.

4 Examples

In this section to show the usefulness of our algorithm we conduct some numerical examples and find run time and the error.

4.1 Image Recovery

This is a subsection.

4.2 Spectral Graph Partioning

As we know there are **local partition algorithms** such as random walks, PageRank and Heat Kernel and we also have **graph spectral** method in other words spectral partition algorithm. Local graph partitioning algorithm finds a small cut near the given seeds with running time depending only the size of the output. Thier usage and applications are used in many different areas for example, finding dense subgraphs, websearch, identifying communitites, locate hot spots, trace target location, and biological effect cluster and many other applications.

The Spectral Partitioning Algorithm, based on linear algebra and spectral graph theory studies how the eigenvalues of the adjacency matrix of a graph, which are purely algebraic quantities, relate to combinatorial properties of the graph.

Algorithm: Spectral Partitioning

1. Input: graph $G = (V, E)$ and vector $x \in \mathbb{R}^V$
2. Sort the vertices of V in non-decreasing order of values of entries in x, that is let $V = v_1, \ldots, v_n$ where $x_{v_1} \leq x_{v_2} \leq \ldots \leq x_{v_n}$
3. Let $i \in 1, \ldots, n-1$ be such that $h(v_1, \ldots, v_i)$ is minimal
4. Output $S = v_1, \ldots, v_i$

Graph partition methods

The most common example is spectral partitioning, where a partition is derived from approximate eigenvectors of the adjacency matrix, or spectral clustering that groups graph vertices using the eigendecomposition of the graph Laplacian matrix.
Since Laplacian matrix is symmetric its eigenvalue decomposition and SVD are the same. We compare the run time of our randomized SVD with the python built in function for the same portioning of the graph. We use facebook friendship graph (a simple one) and apply our method to it. The results our randomized SVD is *10 times* faster that python built in method.

5 Discussion

In this section you analyse and discuss your results. This section is paramount as it gives indication about the hability of the author to interpret the results and critically discuss his or her findings.

References

[1] N. Halko, P. G. Martinsson, and J. A. Tropp, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions* (SIAM Review 2011 53:2, 217-288)