SEMINAR PAPER

Randomized Linear Algebra

Lecturer

Gerhold Stefan

Student

Kovanusic Kristina

Abstract

The following seminar paper is based on “Lecture Notes on Randomized Linear Algebra” which is written by Michael W. Mahoney. The main focus is on randomized matrix multiplication, least square approximation as well as low-rank matrix approximation. Randomized sampling techniques have recently proved capable of efficiently solving many standard problems in linear algebra and enabling computations at scales far larger than what was previously possible.

Contents

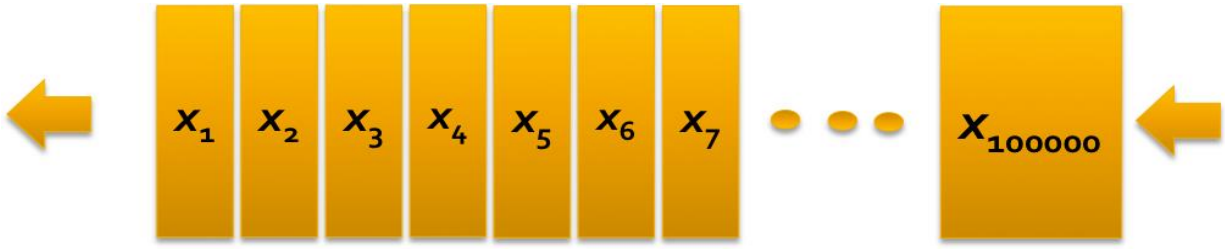
1 Introduction

This seminar paper will cover recent developments in randomized matrix algorithms of interest in large-scale machine learning and statistical data analysis applications. We will focus on basic algorithms for fundamental matrix problems such as matrix multiplication, least-squares regression, low-rank matrix approximation, which use randomization in some nontrivial way.

The area, which we will cover, goes by the name Randomized Numerical Linear Algebra (RandNLA) or Randomized Linear Algebra (RLA). Randomized matrix algorithms typically use randomness to perform random sampling to highlight structural properties of interest, a small number of rows, columns or elements from the matrix, or a random projection, projecting the original data to a much lower dimension. Much of the interest of this randomized matrix algorithms arises since many data analysis methods call these algorithms either as black boxes or use similar ideas with their analysis.

1.1 Models of data

When processing large-scale data (in particular streaming data), we desire some methods that can be performed with: a few (one or two) passes of data, limited memory (so impossible to store all data) or with low computational complexity.



Matrices are the most useful way to model the data, but they are not the only way to model data. Here are some other ways data can be modeled:

- Turing machine
- Database table
- Strings
- Graphs

Those models are not inconsistent and it is often helpful to model the data in different ways, depending on what one is interested in doing. When we want to model the data, we need to know their categorization. There are:

• Small: A data set is small if you can look at the data and easily find solutions to problem of interest.

• Medium: A data set is medium-sized if it fits into RAM and one can easily run computations of interest in a reasonable length of time and get answers to questions of interest.

• Large: A data set is large if it doesn't fit into RAM and/or one can't relatively-easily run computations of interest.

1.2 Differences in perspective

Our first Theme is approximating matrix multiplication but, before we proceed, I wanted to present what are the differences between traditional perspective and RandNLA perspective on matrix multiplication. Say that we have an $m \times n$ matrix A and an $n \times p$ matrix B, and assume that we are interested in computing their product AB.

• **Traditional perspective on matrix multiplication:** This is the well-known way to compute the product AB, and it is done with the usual three-loop algorithm. In this case, one views an element of AB as an *inner product between a row of A and a column of B.*

• **RandNLA perspective on matrix multiplication:** A less obvious and not so "popular" way is to view the product AB as a sum of n terms, each of which is an *outer product between a column of A and a row of B.* Since this perspective would be our focus, here is what we can do: we can try to construct some sort of "sketch" of the columns of A and the rows of B and represent those sketches as matrices C and R, than approximate the product AB by the product CR.

The key idea is to reduce the dimension via random sketching. We can do that in one of two complementary ways and those are:

• **Randomly sample:** It identifies some sort of uniformity structure and use that to construct a sampling distribution with which to construct a random sample. Often relies on information of data.

• **Random projection:** It performs a random projection which rotates/projects data onto lower dimensions. Often data-agnostic.

2 Approximating matrix multiplication

The idea is to start by considering a very simple randomized algorithm to approximate the product of two matrices. As I already mentioned, our problem is the following: given an arbitrary $m \times n$ matrix A and an arbitrary $n \times p$ matrix B, and to compute, exactly or approximately, the product AB. At first, we will use some basic well-known three-loop algorithm. The running time of this algorithm is $O(mnp)$ time, which is $O(n^3)$ time if $m = n = p$.

Algorithm 1 Vanilla three-loop matrix multiplication algorithm

Input: An $m \times n$ matrix A and an $n \times p$ matrix B
Output: The product AB

```
1: for i = 1 to m do
2:   for j = 1 to p do
3:     (AB)_ij = 0
4:     for k = 1 to n do
5:       (AB)_ik += A_ij B_jk
6:     end for
7:   end for
8: end for
9: return AB
```

The main question here is: can we solve this problem more faster? We can consider a different approach: a randomized algorithm that randomly samples columns and rows of the matrices A and B. We can think about matrix multiplication as returning a matrix that equals the sum of outer products of columns of A and the corresponding rows of B , as the sum of rank-one matrices:

$$AB = \sum_{i=1}^{n} A^{(i)} B_{(i)}$$

Where each $A^{(i)}B_{(i)} \in \mathbb{R}^{m \times n}$ is an $m \times p$ rank-one matrix, computed as the outer product of two vectors in $\mathbb{R}^n$.

With this background, here follows our **Basic Matrix Multiplication** algorithm.

Algorithm 2 The Basic Matrix Multiplication algorithm
Input: An $m \times n$ matrix A, an $n \times p$ matrix B, a positive integer c, and probabilities $\{p_i\}_{i=1}^{n}$ **Output:** Matrices C and R such that $CR \approx AB$ 1: **for** $t = 1$ to c **do** 2: Pick $i_t \in \{1, \ldots, n\}$ with probability **Pr**$[i_t = k] = p_k$, in i.i.d. trials, with replacement 3: Set $C^{(t)} = A^{(i_t)} / \sqrt{cp_{i_t}}$ and $R_{(t)} = B_{(i_t)} / \sqrt{cp_{i_t}}$ 4: **end for** 5: **return** C and R

Basically, what we want to show here is that:

$$\begin{aligned} AB &= \sum_{i=1}^{n} A^{(i)} B_{(i)} \\ &\approx \frac{1}{c} \sum_{t=1}^{c} \frac{1}{p_{i_t}} A^{(i_t)} B_{(i_t)} \\ &= CR. \end{aligned}$$

With respect to a few implementation issues, here are some things to note, when probabilities of different forms are used in the Basic Matrix Multiplication algorithm:

• Uniform sampling: one can choose which elements to keep before looking at the data, and so one can implement this algorithm in one-pass over the data.

• Nonuniform sampling: $p_i = \frac{\|A^{(i)}\| \, \|B_{(i)}\|}{\sum_{i'=1}^{n} \|A^{(i')}\| \, \|B_{(i')}\|}$

° p_i's are called – optimal sampling probabilities.

2.1 Initial results for approximating the product of two matrices

Here is the initial result for the quality of approximation of the Basic Matrix Multiplication algorithm. We can provide the following statement of the manner in which the sampling probabilities of the form Eqn. (1) are optimal.

Also, we can note how $\mathbb{E}[\|AB - CR\|_F^2]$ measures the error depends on the p_i's. Basically, they minimize the expectation of the Frobenius norm of the error, and this is equal to the sum of the variance of all of the elements of the product matrix.

Lemma 1 *Given matrices A and B construct matrices C and R with the* Basic Matrix Multiplication *algorithm. Then:*

$$\mathbb{E}\left[(CR)_{ij}\right] = (AB)_{ij}$$

and

$$\mathrm{Var}\left[(CR)_{ij}\right] = \frac{1}{c}\sum_{k=1}^{n}\frac{A_{ik}^2 B_{kj}^2}{p_k} - \frac{1}{c}(AB)_{ij}^2$$

Proof: Fix i, j. For $t = 1, \ldots, c$, define $X_t = \left(\frac{A^{(i_t)}B_{(i_t)}}{cp_{i_t}}\right)_{ij} = \frac{A_{ii_t}B_{i_t j}}{cp_{i_t}}$.

$$\mathbb{E}[X_t] = \sum_{k=1}^{n} p_k \frac{A_{ik}B_{kj}}{cp_k} = \frac{1}{c}(AB)_{ij} \qquad \text{and} \qquad \mathbb{E}[X_t^2] = \sum_{k=1}^{n}\frac{A_{ik}^2 B_{kj}^2}{c^2 p_k}.$$

By construction $(CR)_{ij} = \sum_{t=1}^{c} X_t$, we have $\mathbb{E}\left[(CR)_{ij}\right] = \sum_{t=1}^{c}\mathbb{E}[X_t] = (AB)_{ij}$. While $(CR)_{ij}$ is the sum of c independent random variables, follows that $\mathrm{Var}\left[(CR)_{ij}\right] = \sum_{t=1}^{c}\mathrm{Var}[X_t]$. Since we know that $\mathrm{Var}[X_t] = \mathbb{E}[X_t^2] - \mathbb{E}[X_t]^2$ wa can see that

$$\mathrm{Var}[X_t] = \sum_{k=1}^{n}\frac{A_{ik}^2 B_{kj}^2}{c^2 p_k} - \frac{1}{c^2}(AB)_{ij}^2$$

And the lemma follows. □

Lemma 2 *Given matrices A and B construct matrices C and R with the* Basic Matrix Multiplication *algorithm. Then,*

$$\mathbb{E}[\|AB - CR\|_F^2] = \sum_{k=1}^{n} \frac{\left\|A^{(k)}\right\|_2^2 \left\|B_{(k)}\right\|_2^2}{cp_k} - \frac{1}{c}\|AB\|_F^2$$

and furthermore if

$$p_k = \frac{\|A^{(k)}\|_2 \|B_{(k)}\|_2}{\sum_{k'=1}^{n} \left\|A^{(k')}\right\|_2 \left\|B_{(k')}\right\|_2} \tag{1}$$

then

$$\mathbb{E}[\|AB - CR\|_F^2] = \frac{1}{c}\left(\sum_{k=1}^{n} \left\|A^{(k)}\right\|_2 \left\|B_{(k)}\right\|_2\right)^2 - \frac{1}{c}\|AB\|_F^2$$

Proof: At first, note that

$$\mathbb{E}[\|AB - CR\|_F^2] = \sum_{i=1}^{m}\sum_{j=1}^{p} \mathbb{E}\left[(AB - CR)_{ij}^2\right] = \sum_{i=1}^{m}\sum_{j=1}^{p} \mathrm{Var}\left[(CR)_{ij}\right]$$

and from Lemma 1 follows that

$$\mathbb{E}[\|AB - CR\|_F^2] = \frac{1}{c}\sum_{k=1}^{n} \frac{1}{p_k}\left(\sum_i A_{ik}^2\right)\left(\sum_j B_{kj}^2\right) - \frac{1}{c}\|AB\|_F^2$$

$$= \frac{1}{c}\sum_{k=1}^{n} \frac{1}{p_k}\left\|A^{(k)}\right\|_2^2 \left\|B_{(k)}\right\|_2^2 - \frac{1}{c}\|AB\|_F^2$$

When we use the values $p_k = \frac{\|A^{(k)}\|_2 \|B_{(k)}\|_2}{\sum_{k'=1}^{n} \left\|A^{(k')}\right\|_2 \left\|B_{(k')}\right\|_2}$, then

$$\mathbb{E}[\|AB - CR\|_F^2] = \frac{1}{c}\left(\sum_{k=1}^{n} \left\|A^{(k)}\right\|_2 \left\|B_{(k)}\right\|_2\right)^2 - \frac{1}{c}\|AB\|_F^2$$

□

At the end, we can provide the following statement where the sampling probabilities of the Eqn. (1) are optimal. They minimize the expectation of the Frobenius norm of the error and this is equal to the sum of the variances of all elements of the product matrix.

Lemma 3 *Sampling probabilities* $\{p_i\}_{i=1}^n$ *of the form Eqn. (1) minimizes* $\mathbb{E}[\| AB - CR \|_F^2]$.

Proof: To prove that this choice for the p_k's minimizes $\mathbb{E}[\| AB - CR \|_F^2]$ define the function

$$f(p_1, \dots, p_n) = \sum_{k=1}^{n} \frac{1}{p_k} \left\| A^{(k)} \right\|_2^2 \left\| B_{(k)} \right\|_2^2$$

Which characterizes the dependence of $\mathbb{E}[\| AB - CR \|_F^2]$ on the p_k's. To minimize f subject to $\sum_{k=1}^n p_k = 1$, introduce the Lagrange multiplier λ and define the function

$$g(p_1, \dots p_n) = f(p_1, \dots p_n) + \lambda \left(\sum_{k=1}^{n} p_k - 1 \right)$$

We have at the minimum that

$$0 = \frac{\partial g}{\partial p_i} = \frac{-1}{p_i^2} \left\| A^{(i)} \right\|_2^2 \left\| B_{(i)} \right\|_2^2 + \lambda$$

Thus,

$$p_i = \frac{\left\| A^{(i)} \right\|_2 \left\| B_{(i)} \right\|_2}{\sqrt{\lambda}} = \frac{\left\| A^{(i)} \right\|_2 \left\| B_{(i)} \right\|_2}{\sum_{i'=1}^{n} \left\| A^{(i')} \right\|_2 \left\| B_{(i')} \right\|_2}$$

Where the second equality comes from solving for $\sqrt{\lambda}$ in $\sum_{k=1}^{n-1} p_k = 1$. That these probabilities are a minimum follows since $\frac{\partial^2 g}{\partial p_i^2} > 0 \ \ \forall i$ s.t. $\left\| A^{(i)} \right\|_2^2 \left\| B_{(i)} \right\|_2^2 > 0$.

□

3 Least-squares Approximation

In this chapter, we will discuss about RandNLA algorithms for the least-squares regression. Least-squares approximation is a technique to find an approximate solution to a system of linear equations that has no exact solution. This is a fundamental problem of interest in linear algebra, and many of this methods in RandNLA are mostly easily introduced and understood in this relatively-simple setting. With recent data explosion, traditional approach is no longer suitable while working with large datasets, instead, randomized algorithms become popular in addressing this issue. In so many applications, we want to find an approximate solution to a problem or set of equations that, for a bunch of reasons, does not have a solution, or does not have a unique solution.

We are beginning with:

Let $A \in \mathbb{R}^{n\times d}$ and $b \in \mathbb{R}^n$ be given. If $n \gg d$, in which case there are many more rows than columns, than in general there does not exist a vector x such that $Ax = b$. This is because b may have some part that sits outside the column space of A. When we have this case, the most popular way to find "best" vector x such that $Ax \approx b$ is to minimize the norm of the residuals, to solve $\min_{x\in\mathbb{R}^d}\|Ax - b\|$, where $\|\cdot\|$ is some norm.

Before we proceed, we need to mention two basic questions that people are interested in when considering LS problems:

• Algorithmic question: *How long does it take to solve the LS problem "exactly"*? The answer is $O(nd^2)$ time - the running time in the RAM model to solve the LS problem, and this can be accomplished with one of a variety of direct or indirect methods.

• Statistical question: *When is solving the LS problem the "right" thing to do*? The answer here is - that it is when the data are "nice" in the way that means that large-sample theory can be applied, that there are a large number of components that are particularly important or influential. That can be checked with empirical statistics such as the leverage scores and with other regression diagnostics.

3.1 Methods for solving LS problems

With respect to the question of how long does it takes to solve LS problems, there are some methods that we can use. Those methods are called *direct* methods or *iterative* methods.

By *direct* methods for solving LS problems, we have:

• Cholesky decomposition: If A is a full rank and well-conditioned, then one can use the Cholesky decomposition to compute an upper triangular matrix T such that $A^T A = T^T T$, and then one can solve the normal equations $T^T T x = A^T b$.

•QR decomposition: A little bit slower method, but numerically more stable, especially if A is rank-deficient or ill-conditioned, involves computing a QR decomposition $A = QR$ and than solving $Rx = Q^T b$.

•SVD: More expensive but better still if A is very ill-conditioned, involves computing the SVD, $A = U\Sigma V^T$, where this is the economical SVD (things that are zeroed-out by singular values are not included).

The complexity of all these methods is $O(nd^2)$.

Next to *direct* and *iterative* methods there are also well-known *Strassen-like* methods for matrix multiplication. Those algorithms can also be applied to rectangular LS problems, with similar improvements in worst-case running time. Since they are never used in practice, we won't focus on them, but because they are of theoretical interest, it was worth mentioning them.

3.2 Randomized sketches for LS problems

To see how to solve the LS problem, we can define a function

$$f(x) = \|Ax - b\|_2^2 = (Ax - b)^T (Ax - b).$$

To find the minimizer of this function, we can set the derivate equals to zero, s.t. $\frac{\partial f}{\partial x} = 0$, noting that the second derivate is positive. Then we get $A^T Ax - A^T b = 0$, which is the normal equation, $A^T Ax = A^T b$. If A has full column rank, than $A^T A$ is square and has full rank, and this is a $d \times d$ system of linear equations with solution:

$$x_m = (A^T A)^{-1} A^T b.$$

Here, we will deal with so-called linear sketches. This means that we can write the operation of the sketch as a linear function. Note, that random projection matrices and random sampling matrices satisfy this. The advantages of working with linear sketches include that we can take advantage of linear theory stuff and it is easy to update sketches.

Now, let M be an arbitrary sketching matrix and we are going to apply to A and b to construct a sketch. When forming a sketch, the original LS problem

$$\mathcal{Z} = \min_{x \in \mathbb{R}^d} \|Ax - b\|_2,$$

would be replaced with a sketched LS problem

$$\tilde{\mathcal{Z}} = \min_{x \in \mathbb{R}^d} \|M(Ax - b)\|_2.$$

We want to ask what are the properties that M needs to satisfy s.t.:

$$\tilde{x}_m \approx x_m$$

$$\|A\tilde{x}_m\|_2 \approx \|Ax_m - b\|_2.$$

The second requirement is a statement where one might not even be able to obtain "certificate" for the solution. The bound on the vector achieving the optimal solution is typically of greater interest in NLA, where the vector is used form something downstream such as classification. For matrix extensions of these ideas, we usually want results for the objective, since we will measure quality by norm reconstruction. Here are two very important structural conditions.

Let $A = U\Sigma V^T = U_A \Sigma_A V_A^T$ be the SVD of A, and let $b^{\perp} = U_A U_A^T A$ and note that $\mathcal{Z} = \|Ax_m - b\|_2 = \|b^{\perp}\|_2$. Then follow two conditions:

• **Condition 1:**

$$\sigma_{min}(MU_A) \geq 1/\sqrt{2}$$

• **Condition 2:**

$$\|U_A M^T M b^{\perp}\|_2^2 \leq \frac{\varepsilon}{2} \mathcal{Z}^2.$$

The main lemmas for those structural conditions would be our next thing to do. They say that if we have a sketched matrix M that satisfies those two conditions, then we have a relative-error approximation to the solution of the LS problem. We are now interested in quality-of-approximation.

Lemma 4 *Consider the overconstrained least squares approximation problem and let the matrix $U_A \in \mathbb{R}^{n \times d}$ contain the top d left singular vectors of A. Assume that the matrix M satisfies Conditions 1 and 2 form above, for some $\varepsilon \in (0,1)$. Then, the solution vector $\tilde{x}_m$ to the least squares approximation problem satisfies:*

$$\|A\tilde{x}_m - b\|_2 \leq (1+\varepsilon)\mathcal{Z}.$$

Proof: Let us first rewrite the down-scaled regression problem induced by M as

$$\begin{aligned} \min_{x \in \mathbb{R}^d} \|Mb - MAx\|_2^2 &= \min_{y \in \mathbb{R}^d} \|M(Ax_m + b^{\perp}) - MA(x_m + y)\|_2^2 \\ &= \min_{y \in \mathbb{R}^d} \|Mb^{\perp} - MAy\|_2^2 \\ &= \min_{z \in \mathbb{R}^d} \|Mb^{\perp} - MU_A z\|_2^2. \end{aligned} \quad (2)$$

Now, let $\tilde{x}_m \in \mathbb{R}^d$ be such that $U_A z_m = A(\tilde{x}_m - x_m)$ and note that z_m minimizes the Eqn. (2). The latter fact follows since

$$\|Mb^{\perp} - MA(\tilde{x}_m - x_m)\|_2^2 = \|Mb^{\perp} + M(b - b^{\perp}) - MA\tilde{x}_m\|_2^2 = \|Mb - MA\tilde{x}_m\|_2^2$$

We have that:

$$(MU_A)^T MU_A z_m = (MU_A)^T Mb^{\perp}$$

Taking the norm of both sides and observing that under Condition 1 we have $\sigma_i((MU_A)^T MU_A) = \sigma_i^2(MU_A) \geq 1/\sqrt{2}$, and for all i it follows that

$$\|z_m\|_2^2/2 \leq \|(MU_A)^T MU_A z_m\|_2^2 = \|(MU_A)^T Mb^{\perp}\|_2^2.$$

Using Condition 2 we observe that

$$\|z_m\|_2^2 \leq \varepsilon \mathcal{Z}^2.$$

We will rewrite the norm of the residual vector as

$$\begin{aligned}\|b - A\tilde{x}_m\|_2^2 &= \|b - Ax_m + Ax_m - A\tilde{x}_m\|_2^2 \\ &= \|b - Ax_m\|_2^2 + \|Ax_m - A\tilde{x}_m\|_2^2 \\ &= \mathcal{Z}^2 + \|U_A z_m\|_2^2 \\ &\leq \mathcal{Z}^2 + \varepsilon \mathcal{Z}^2.\end{aligned}$$

The first claim of the lemma follows because $\sqrt{1+\varepsilon} \leq 1 + \varepsilon$.

□

Lemma 5 *Same setup as the previous lemma. Then*

$$\|x_m - \tilde{x}_m\|_2 \leq \frac{1}{\sigma_{\min}(A)} \sqrt{\varepsilon} \mathcal{Z}$$

Proof: Using the same notation as in the proof in previous lemma, follows that:

$A(x_m - \tilde{x}_m) = U_A z_m$. With taking the norm of both side of this expression, we are having that

$$\begin{aligned}\|x_m - \tilde{x}_m\|_2^2 &\leq \frac{\|U_A z_m\|_2^2}{\sigma_{\min}^2(A)} \\ &\leq \frac{\varepsilon \mathcal{Z}^2}{\sigma_{\min}^2(A)}.\end{aligned}$$

Taking a square root, the second claim follows.

□

By making further assumption on b, we can connect error bound with $\|x_m\|_2$.

Lemma 6 Suppose that $\|U_A U_A^T b\| \geq \gamma \|b\|_2$, for some $0 < \gamma \leq 1$. Under Conditions 1 and 2, solution $\tilde{x}_m$ to subsample LS problem obeys

$$\|x_m - \tilde{x}_m\|_2 \leq \sqrt{\varepsilon}\, \kappa(A) \sqrt{\gamma^{-2} - 1} \|x_m\|_2.$$

• $\|U_A U_A^T b\| \geq \gamma \|b\|_2$, says a nontrivial fraction of energy of b lies in range (A).

Proof:

$$\begin{aligned} \mathcal{Z}^2 = & \|b\|_2^2 - \|U_A U_A^T b\|_2^2 \\ & \leq (\gamma^{-2} - 1) \|U_A U_A^T b\|_2^2 \\ & \leq (\gamma^{-2} - 1)\, \sigma_{\max}^2(A) \|x_m\|_2^2. \end{aligned}$$

This combined with Lemma 5 concludes proof.

□

4 Low-rank Matrix Approximation

Low-rank approximation of linear operators is ubiquitous in applied mathematics, scientific computing, numerical analysis, and a number of other areas. Decomposition of a matrix into low-rank matrices is a powerful tool for scientific computing and data analysis. The purpose is to obtain a low-rank matrix by decomposition of the original matrix into a product of smaller and lower-rank matrices or just by randomly projecting the matrix down to a lower-dimensional space. Such decomposition requires less storage and computational burden. The focus is on randomized methods which try as much as possible to preserve the original matrix properties by applying the subspace sampling. In many applications, randomized algorithms in terms of accuracy, stability and speed are much better than the classical decomposition algorithms. We will propose a sparse orthogonal transformation matrix to reduce the dimension of the data. The results show that compared with the most accurate methods, the transformation speed is much faster and can save a lot of memory in the case we are dealing with the huge matrices.

4.1 Basic Low-rank Matrix Approximation

The low-rank matrix approximation problem involves finding of a rank k version of a $m \times n$ matrix A, labeled as A_k, such that A_k is as "close" as possible to the best SVD approximation version of A at the same rank level.

If we have matrix $A \in \mathbb{R}^{m\times m}$, then $L = [l^1 l^2 \dots l^m] \in \mathbb{R}^{m\times m}$ are orthogonal, the (approximate) *left singular vectors* of A where $\{l^t\}_{t=1}^m \in \mathbb{R}^m$ and $R = [r^1 r^2 \dots r^n] \in \mathbb{R}^{n\times n}$ are orthogonal, the (approximate) *right singular vectors* of A where $\{r^t\}_{t=1}^n \in \mathbb{R}^n$. They are such that

$$L^T A R = \mathrm{D} = Diag(\sigma_1, \dots, \sigma_\varphi),$$

where $\mathrm{D} \in \mathbb{R}^{m\times n}$, $\varphi = \min\{m, n\}$ and $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_\varphi \geq 0$ are the singular values of A. Then we have that

$$A = LDR^T.$$

These three matrices: L, D, R construct the Singular Value Decomposition (SVD) of matrix A. The SVD is very useful, because it can reveal us the important information about the structure of a matrix. We can define some r with $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > \sigma_{r+1} = \dots = \sigma_\varphi = 0$, then is $\mathrm{rank}(A) = r$ and also $\mathrm{range}(A) = \mathrm{span}\{l^1, \dots, l^r\}$. If we let $L_r \in \mathbb{R}^{m\times r}$ to denote the matrix consisting of the first r columns of L, equivalently if we let $R_r \in \mathbb{R}^{r\times n}$ to denote the matrix consisting of the first r columns of R, and $D_r \in \mathbb{R}^{r\times r}$ denote the $r \times r$ submatrix of D, then:

$$A = L_r D_r R_r^T = \sum_{t=1}^{r} \sigma_t l^t r^{t^T}$$

• $\boldsymbol{k \leq r}$ we are defining:

$$A_k = L_k D_k R_k^T = \sum_{t=1}^{k} \sigma_t l^t r^{t^T}$$

where $A_k = L_k L_k^T A = \left(\sum_{t=1}^{k} l^t l^{t^T}\right) A$ and $A_k = A\, R_k R_k^T = A\left(\sum_{t=1}^{k} r^t r^{t^T}\right)$ s.t. A_k is the projection of A onto the space spanned by the top k singular vectors of A. The distance between A and any rank k approximation to A is minimized by A_k:

$$\min_{K \in \mathbb{R}^{m \times n}: \text{rank}(K) \leq k} \|A - K\|_2 = \|A - A_k\|_2 = \sigma_{k+1}(A)$$

and

$$\min_{K \in \mathbb{R}^{m \times n}: \text{rank}(K) \leq k} \|A - K\|_F^2 = \|A - A_k\|_F^2 = \sum_{t=k+1}^{r} \sigma_t^2(A)$$

Matrices for which the numerical rank k is much smaller than either m or n abound in applications. Examples include:

• Principal Component Analysis (PCA) is a basic tool in statistics and data mining. When projecting the data onto the orthogonal directions of maximal variance (principal components) we can visualize or explain the data in far fewer degrees of freedom than the ambient dimension. This amounts to computing a truncated Singular Value Decomposition.

• PCA is popular in the analysis of population genetic variation. Though problems exist with it's interpretation and more advanced nonlinear methods have been developed, PCA still remains an indispensable data analysis tool in the sciences.

• A particular application of PCA is a technique in facial recognition called Eigenfaces. Representing faces as 2-D vectors, the eigenfaces are the significant features (principal components) among the known faces in the database. Just a small number of eigenfaces are needed to span the significant variations so faces can accurately be described by a weighted sum of each eigenface. The tast of matching a face against the entire database reduces to comparing only the hand full of weights which greatly reduces the complexity of the problem.

• Estimating parameters via least squares in biology, engineering, and physics often leads to large over determined linear systems. Low rank factorization of the coefficient matrix leads to efficient solutions of the problem.

• The fast multiple method for more rapidly evaluating potential fields relies on low rank approximations of continuum operators with exponentially decaying spectra.

• Laplacian Eigenmaps arise in image processing. A few eigenvectors of a graph derived from the image provide a non-linear embeddin. This is also an example of how linear approximations are used to solve some non-linear problems of interest.

• Low cost sensor networks provide a wealth of data. Their low cost enables many sensors to be densely deployed while limiting their on board data processing capability. This inevitably results in lots of redundant data to be processed by the user.

4.2 Low-rank Approximation in Practice

There are some challenges in understanding how these theoretical ideas for randomized low-rank matrix approximation can be used in practice. For better understanding, the basic idea is to go through the practical situations and set the issues for the randomized low-rank approximation situation:

• It could be more expensive to sample $O\left(\frac{k\log(k)}{\varepsilon^2}\right)$ columns/rows and it could be difficult to do so if the constant in the big-O is left unspecified. Instead of that we could choose exactly k, or we can choose $k+p$, where p is small integer such as 5 or 10.

• In most applications, in particular in those that are interested moderate to high precision low-rank matrix approximation, numerical analysis and scientific computing applications, here are other goals of interest. Good Example for that is with given a good approximation to an orthogonal basis Q approximating A, one might want to find other types of matrix decompositions.

• When the spectrum decays some slowly but not very slowly, then it might be interested in doing some sort of power iteration, which will help the spectrum to decay more quickly and it could be of interest to incorporate this process directly into the algorithm.

4.3 Something about SVD algorithms

Excellent algorithms for computing SVDs exist, but many of them are not well suited for an emerging computational environment. Complications include:

• Multi-processor computing: CPU speed is growing slowly, but processors get cheaper all the time.

• Communication speeds improve only slowly: Communication between different levels in memory hierarchy, latency in hard drives, inter-processor communication, etc.

• The size of data sets is growing very rapidly. The cost of slow memory is falling rapidly, and information is gathered at an ever faster pace — automatic DNA sequencing, sensor networks, etc.

Bibliography

[1] https://www.princeton.edu/~yc5/orf570/randomized_linear_algebra.pdf

[2] https://edoliberty.github.io/papers/RandomIDSvdPnas.pdf

[3] https://amath.colorado.edu/faculty/martinss/2014_CBMS/Lectures/lecture05.pdf

[4] https://dl.acm.org/doi/10.1145/1536414.1536446

[5] Michael W. Mahoney, *Lecture Notes on Randomized Linear Algebra*