CS 6220, Fall 2017 Oct 3, 2017

Lecture 13

Lecturer: Anil Damle *Scribe: Qinru Shi, Qiantong Xu, Lily Wang*

1 Introduction

In the last lecture, we learned about fixed rank matrix approximation methods. However, in practice, we rarely know the appropriate target rank in advance. In some cases, we might over-estimate the rank we need. In other cases, the singular values of the matrix may decay slowly, while the error guarantee depends on the sum of the rest of the singular values. Hence, today we are going to talk about a fixed-error approximation methods that takes error tolerance as the parameter and adaptively increase the rank of approximation until the target accuracy is achieved. We will also introduce the idea of recalibration that deal with matrices with flat singular spectrum. Then we will talk about a different method of matrix factorization called CUR factorization.

2 Randomized fixed error approximation

In this section, we consider the following fixed-error matrix approximation problem. Assume we are given a $m \times n$ matrix A and want to find l and $Q^{(l)}$ such that $Q^{(l)}$ has l orthonormal columns and $\|(I - Q^{(l)}Q^{(l)T})A\|_2 \leq \epsilon$ with probability $1 - 10^{-r}$.

In order to handle fixed-error approximation problems, we first need a fast and reliable way of estimating error. In the last lecture, we learned the following lemma.

Lemma 1. *Given r Gaussian random vectors $\omega^{(i)}$, $i = 1, \cdots, r$. Then,*

$$\|(I - QQ^T)A\|_2 \leq 10\sqrt{\frac{2}{\pi}} \max_{i=1,\cdots,r} \|(I - QQ^T)A\omega^{(i)}\|_2 \tag{1}$$

with probability $1 - 10^{-r}$.

This gives us a cheap way to check accuracy. Also note that the Gaussian random vectors can be used to construct low rank basis. Hence, the natural idea is to combine the lemma with a randomized method that incrementally constructs the basis, so that we get free error estimates along the way. We have learned about such a algorithm in last lecture. In stead of randomly picking a $n \times l$ Gaussian random matrix, we modify the algorithm to pick a $n \times 1$ Gaussian random vector every time and gradually increase the rank of the basis and stop when we reach the target accuracy.

The formal statement of the algorithm is shown in Algorithm 1.

Note that for any $i = 1, 2, \cdots$, we have

$$\|(I - Q^{(i)}Q^{(i)T})A\|_2 \leq \|(I - Q^{(i-1)}Q^{(i-1)T})A\|_2,$$

so the error of our approximation steadily decreases. Also note that the norm of $\tilde{q}^{(i)}$ is in the right hand side of (1). Hence, with r consecutive $\|\tilde{q}^{(i)}\|_2 \leq \epsilon/(10\sqrt{\frac{2}{\pi}})$, we can apply Lemma 1 and conclude that, with probability at least $1 - 10^{-r}$, $\|(I - Q^{(l)}Q^{(l)T})A\|_2 \leq \epsilon$.

Algorithm 1 Adaptive Randomized Range Finder

$Q^{(0)} \leftarrow [\,]$
for $i = 1, 2, \cdots$ **do**
  Draw a $n \times 1$ Gaussian random vector $\omega^{(i)}$.
  $y^{(i)} \leftarrow A\omega^{(i)}$
  $\tilde{q}^{(i)} \leftarrow (I - Q^{(i-1)}Q^{(i-1)T})y^{(i)}$
  $q^{(i)} \leftarrow \tilde{q}^{(i)}/\|\tilde{q}^{(i)}\|_2$ ▷ normalize
  $Q^{(i)} \leftarrow [Q^{(i-1)}\ q^{(i)}]$
  if r consecutive $\|\tilde{q}^{(i)}\|_2 \leq \epsilon/(10\sqrt{\frac{2}{\pi}})$ **then**
    break
$l \leftarrow i, Q^{(l)} \leftarrow Q^{(i)}$.

One potential problem of this method is that as i increases, $\|\tilde{q}^{(i)}\|$ may become extremely small as the singular values of A approach 0. This makes the normalization step (i.e. finding the direction of $\tilde{q}^{(i)}$) very unreliable since we can only do finite accuracy arithmetics. To solve this problem, we can re-orthogonalize $q^{(i)}$ by projecting $q^{(i)}$ onto range$(Q^{(i-1)})^{\perp}$. After the normalization step, we can add a line $q^{(i)} \leftarrow (I - Q^{(i-1)}Q^{(i-1)T})q^{(i)}$ and normalize $q^{(i)}$ again. [1]

Another problem of this method is that the error estimate we used is very crude, so the actual accuracy of the result may be a lot higher than the target accuracy. In fact, there exists a better estimate with similar computational cost. See [2] for more details.

3 Randomized subspace iteration

The method described above works well for matrices with fast decaying singular values. However, it still works poorly for matrices with a relatively flat singular spectrum. To solve this issue, we introduce the idea of recalibration.

First, we look at a variation of the power iteration method called the subspace iteration method that finds the largest k eigenvalues of a symmetric $n \times n$ matrix A. The method was developed by K.J. Bathes [4]. For more information on this method, see [5].

The pseudo code of the methods are shown in Algorithm 2.

Algorithm 2 Subspace iteration for symmetric matrix $A^{n \times n}$

Pick a random matrix Z with shape $n \times k$
for $i = 1, 2, \cdots$ **do**
  $Y^{(i)} \leftarrow AZ$.
  Conduct QR factorization on $Y^{(i)}$, $Y^{(i)} = Q^{(i)}R^{(i)}$
  $Z \leftarrow Q^{(i)}$ ▷ cols are eigenvectors of the largest K eigenvalues
Output Z.

After a sufficient number of iterations, the columns of Z will be the eigenvectors to the largest k eigenvalues of A. We can see that the algorithm is very similar to the power iteration method that find the largest eigenvalue. The difference here is that we are forcing the columns of Z to be orthogonal so that we get the first k eigenvalues.

Now, we come back to the problem of low-rank approximating $m \times n$ matrix A with slowly decaying σ's. How do we apply the power iteration method above to A? The idea is to "recalibrate"

A so that the singular values decay faster. Instead of working directly with A, we consider $B = (AA^T)^q A$ for some small q.

Assume we have the SVD decomposition of $A = U\Sigma V^T$, then we can see that

$$B = (AA^T)^q = (U\Sigma U^T)^q U\Sigma V^T = U\Sigma^{2q+1}V^T.$$

Thus, the singular vectors of B are the same as A, but the singular values of B are

$$\sigma_j(B) = \sigma_j(A)^{2q+1}, \quad \text{for } j = 1, 2, 3, \cdots,$$

which will decay considerably faster.

Now we can replace A with B in Algorithm 2 and get Algorithm 3. This algorithm is introduced in [6].

Algorithm 3 Randomized subspace iteration for low rank matrix $A^{m\times n}$

Given an m by n matrix A, a target rank k, oversampling p, and power q.
Draw an $n \times (k+p)$ Gaussian random matrix Ω
Form $Y_0 = A\Omega$, and compute QR factorization $Y_0 = Q_0 R_0$
for $j = 1, 2, \cdots, q$ **do**
  Form $\widetilde{Y}_j = A^T Q_{j-1}$, and compute QR factorization$\widetilde{Y}_j = \widetilde{Q}_j \widetilde{R}_j$
  Form $Y_j = A\widetilde{Q}_j$, and compute QR factorization $Y_j = Q_j R_j$
$Q = Q_q \approx$ range A

Note that we never directly apply B to vectors. The reason is that we are trying to avoid getting ill-conditioned matrices and losing smaller singular values. If the arithmetic accuracy of our computation is μ, then the information of singular values smaller than $\mu^{1/(2q+1)}\|A\|$ will be lost when we apply B directly [1]. By orthogonalizing at every step, we keep the condition number at 1 avoid this scenario.

One problem of this algorithm is that it takes $2q+1$ times more matrix multiplication and QR factorization than the original fixed-ranked approximation algorithm, but it reduces the error exponentially faster.

Another potential problem of this algorithm is that it is still fixed rank. We can solve it by incorporating the error estimation method we used in Algorithm 1 into the power iteration method and incrementally increase the rank.

4 Computing Standard Factorizations Using Low-rank Approximate Basis

Once we obtain a low-rank approximate basis of a matrix, we can use it to compute standard factorizations like SVD decomposition or QR factorization. We first show how to compute the SVD of a matrix using its low-rank approximate basis.

Assume we have a $m \times n$ matrix A and a low rank basis Q with l columns such that $\|(I - QQ^T)A\|_2 < \epsilon$. The steps of computing SVD of A are shown in Algorithm 4.

Then, the SVD decomposition of A is:

$$A = QB = Q\widetilde{U}\Sigma V^T = U\Sigma V^T.$$

Algorithm 4 Direct SVD

Compute $B \leftarrow Q^T A$. (Note that B is $l \times n$.)
Compute the SVD decomposition $B = \widetilde{U}\Sigma V^T$. (Cheap since B is small.)
Compute $U \leftarrow Q\widetilde{U}$.

Note that $\|A - QB\|_2 = \|(I - QQ^T)A\|_2 < \epsilon$, so we have $\|A - U\Sigma V^T\|_2 < \epsilon$. This means that the accuracy of this SVD decomposition is on the same level as the low-rank approximation we used. The dominating cost of this algorithm would be in computing $B = Q^T A$.[1]

The cost of this algorithm is dominated by the matrix multiplication step $B = Q^T A$ which in general costs $O(lmn)$. Hence, this method performs better when have a fast method of applying A^T to vectors.

For QR decomposition, we rename the low-rank basis to X to avoid confusion. Similar to how we compute SVD, we first form $B = X^T A$ and then compute the QR factorization of $B = Q'R$. Then, we compute $Q = XQ'$ and the QR decomposition of A is then approximately $A \approx QR$. The error and runtime analysis of this method is similar to the analysis of the SVD method.

5 Intro to CUR Factorization

The SVD decomposition allows us to represent a matrix with regard to a sequence of orthogonal vectors that has decreasing importance. However, when we have a matrix of data points, it becomes very hard for us to interpret the meaning of these orthogonal vectors. Hence, today we will start talking about another way to approximately decompose a matrix called CUR decomposition, which is very easy to interpret.

The main idea of CUR decomposition is to find rows and columns that are "significant" to the matrix and decompose the matrix using these rows and columns. Given an $m \times n$ matrix A, the CUR method decompose it as a product of 3 matrices, C, U, and R, where C consists of a small number of actual columns of A, R consists of a small number of actual rows of A, and U is chosen to make A well approximated [3]. The idea behind the selection process is to assign each row and column a *leverage score*, which signifies the importance of a given column/row in the best rank k approximation of A, then randomly sample the rows and columns using a probability distribution determined by the leverage scores. We will talk about this method in detail in the next lecture.

References

[1] Nathan Halko, Per-Gunnar Martinsson, Joel A. Tropp, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, SIAM Rev., Survey and Review section, Vol. 53, num. 2, pp. 217-288, June 2011.

[2] Edo Liberty, Franco Woolfe, Per-Gunnar Martinsson, Vladimir Rokhlin, Mark Tygert, *Randomized algorithms for the low-rank approximation of matrices*, Proc. Natl. Acad. Sci. USA, 104 (2007), pp. 20167–20172.

[3] Michael W. Mahoney and Petros Drineas, *CUR matrix decompositions for improved data analysis*, PNAS, January 20, 2009, vol.106 no.3 697-702.

[4] Klaus-Jürgen Bathe and Edward L. Wilson, *Solution Methods for Large Generalized Eigenvalue Problems in Structural Engineering*, Report UCSESM 71-20, Department of Civil Engineering, University of California, Berkeley, 1971.

[5] Klaus-Jürgen Bathe, *The subspace iteration method – revisited*, Comput Struct 2013;126:177–83.

[6] Franco Woolfe, Edo Liberty, Vladimir Rokhlin, Mark Tygert, *A fast randomized algorithm for the approximation of matrices*, Appl. Comp. Harmon. Anal., 25 (2008), pp. 335–366.