CS 598: Communication Cost Analysis of Algorithms

Lecture 22: Randomized algorithms for low-rank matrix factorization and least-squares

Edgar Solomonik

University of Illinois at Urbana-Champaign

November 7, 2016

Short-term lecture plan

This lecture will start with randomized algorithms

- we will cover methods for computing tensor factorizations subsequently, since they can solve more general problems
- our presentation of randomized algorithms follows the paper
  - Halko, Martinsson, Tropp "Finding structure with randomness: stochastic algorithms for constructions approximate matrix decompositions"
- a quote from the above paper,
  "*The disappointing computational profile of Monte Carlo integration seems to have inspired a distaste for randomized approaches within the scientific community. Fortunately, there are many other types of randomized algorithms–such as the ones in this paper–that do not suffer from the same shortcomings.*"

Approximate low-rank factorizations

Given matrix $A \in \mathbb{R}^{m \times n}$, find $X \in \mathbb{R}^{m \times k}$, $Y \in \mathbb{R}^{n \times k}$ with $k \leq \min(m, n)$ so

$$||A - XY^T||_F \leq \varepsilon$$

- if $A = XY^T$ (exact low rank factorization), we can
  - obtain $A = QR\Pi$ via
    1. $[Q_1, R_1] = \mathrm{QR}(X)$
    2. $[Q_2, R, \Pi] = \mathrm{QRP}(R_1 Y^T)$
    3. $Q = Q_1 Q_2$
  - Q: how many operations does this require?
  - A: $O(mk^2 + nk^2)$
  - obtain $A = UDV^T$ via
    1. $[U_1, R] = \mathrm{QR}(X)$
    2. $[U_2, D, V] = \mathrm{SVD}(RY^T)$
    3. $U = U_1 U_2$
  - again with cost $O(mk^2 + nk^2)$
- in exact arithmetic these transformations also preserve error ε

Randomization basics

Intuition: consider a random vector w of dimension n

- also consider any basis Q for the n dimensional space
- with high probability (w.h.p) w is not orthogonal to any row of Q^T
- moreover, consider $A \in \mathbb{R}^{m \times n}$ with exact rank k
- $A = UDV^T$ where $V^T \in \mathbb{R}^{n \times k}$
- w.h.p vector w is not orthogonal to any row of V^T
- moreover $z = V^T w$ is random
- therefore $Aw = UDz$ is random linear combination of columns of UD
- now consider random matrix $W \in \mathbb{R}^{n \times k}$
- columns of $B = AW$ are random linear combinations of those in UD
- Q: will the columns of B be linearly independent?
- A: yes, w.h.p. and then B has the same span as U!

Using the basis to compute a factorization

If B has the same span as the range of A

- $[Q, R] = \mathrm{QR}(B)$ gives orthogonal basis Q for B
- $QQ^T A = QQ^T UDV^T = (QQ^T U)DV^T$, now $Q^T U$ is orthogonal and so $QQ^T U$ is a basis for the range of A
- so compute $H = Q^T A$, $H \in \mathbb{R}^{k \times n}$ and compute $[U_1, D, V] = \mathrm{SVD}(H)$
- then compute $U = QU_1$ and we have a rank k SVD of A

$$A = UDV^T$$

- matrix multiplications required $O(mnk)$ operations
- QR and SVD required $O((m+n)k^2)$ operations
- Q: why is this be preferable to QR with column pivoting?
- A: if $k \ll \min(m, n)$ the bulk of the computation is within matrix multiplication, which can be done with fewer synchronizations and higher efficiency

Randomized approximate factorization

Now lets consider the case when $A = UDV^T + E$ where $D \in \mathbb{R}^{k \times k}$ and E is a small perturbation

- E may be noise in data or numerical error
- to obtain a basis for U it is insufficient to multiply by random $B \in \mathbb{R}^{n \times k}$, due to influence of E
- however, oversampling, for instance $l = k + 10$, and random $B \in \mathbb{R}^{n \times l}$ gives good results
- a Gaussian random distribution provides particularly good accuracy
- Q: so far the dimension of B has assumed knowledge of the target approximate rank k, what could we do to find it dynamically?
- A: generate vectors (columns of B) one at a time or a block at a time, which results in a provably accurate basis

Cost analysis of randomized low-rank factorization

From previous lecture, the BSP cost of QR with column pivoting is

$$T_{\mathrm{QRP}}(m,n,k,P) = O\left(\frac{mnk}{P}\cdot\gamma + k\sqrt{\frac{mn}{P}}\cdot\beta + k\sqrt{\frac{P}{mn}}\log(P)^2\cdot\alpha\right)$$

which can be obtained by selecting $p_r/p_c = m/n$ and $b = n/(p_c\log^2(P))$

- $T_{\mathrm{QRP}}(n,n,k,P) = O\left(\frac{n^2k}{P}\cdot\gamma + \frac{nk}{\sqrt{P}}\cdot\beta + (k/n)\sqrt{P}\log(P)^2\cdot\alpha\right)$
- the cost of the randomized algorithm for is

$$T_{\mathrm{MM}}(m,n,k,P) + T_{\mathrm{QR}}(m,k,k,P) = O\left(\frac{mnk}{P}\cdot\gamma + \left(\frac{mnk}{P}\right)^{2/3}\cdot\beta + \left(\frac{Pk}{m}\right)^{2/3}\log(P)\cdot\alpha\right)$$

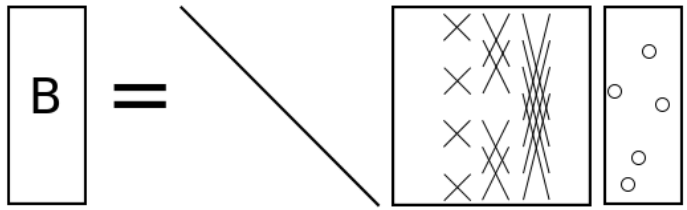
- assuming that we factorize the basis by QR and $k\times k$ SVD of R

Exploiting structured randomization

We can lower the number of operations needed by the randomized algorithm by generating B so that AB can be computed more rapidly

- there are different ways to generate B in this way, most look like

$$B = DFR$$



- D is diagonal with elements randomly chosen from some space
- F can be applied to a vector in $O(n \log(n))$ operations
  - can be actual discrete Fourier transform
  - can also be real, for instance Hadamard transform $H_{2n} = \begin{bmatrix} H_n & H_n \\ H_n & -H_n \end{bmatrix}$
- R is $p \approx k$ columns of the $n \times n$ identity matrix
- we can then compute AB with $O(mn \log(n))$ operations (if $m > n$)
  - in fact $O(mn \log(k))$ if a subsampled FFT algorithm is used

Cost of structured randomized factorization

Instead of matrix multiplication, we apply m FFTs of dimension n

- Q: if $m > P$, how much communication is required in BSP?
- A: each FFT is independent, so a transpose, $O(mn/P)$
- so we have the following total cost

$$O\left(\frac{mn\log(n)}{P}\cdot\gamma+\frac{mn}{P}\cdot\beta\right)+T_{\text{QR}}(m,k,k,P)=O\left(\frac{mn\log(n)+mk^2}{P}\right.$$
$$\left.+\left[\frac{mn}{P}+\left(\frac{mk^2}{P}\right)^{2/3}\right]\cdot\beta+\left(\frac{Pk}{m}\right)^{2/3}\log(P)\cdot\alpha\right)$$

  assuming $m > n$
- this is lower by a factor of $(n/k)^{2/3}$ with respect to the previous randomized version
- we should be able to lower communication cost by transposing A so that $m \leq n$ (but we need $\max(m+n)k^2/P$ computation)

Short pause

Least squares

Given $A \in \mathbb{R}^{m \times n}$, $m \geq n$, would like to approximate

$$y = \operatorname*{argmin}_x ||Ax - b||_2$$

by finding x within relative error ε, so that

$$||Ax - b||_2 - ||Ay - b||_2 \leq \varepsilon ||Ay - b||_2$$

- QR-based methods require $O(mn^2)$ computation
- randomization yields costs

$$O(mn \log(n) + mn \log(1/\varepsilon) + n^3)$$

Rokhlin–Tygert (2008)

Again leverages subsampled Fourier-Transform (SRFT) or similar structured random matrix

- define SRFT $T \in \mathbb{R}^{l \times m}$ where $m \geq l \geq n$ (transpose of previous)
  1. compute $E = TA$ and $E = QX$, $X = R\Pi$ using column pivoting
  2. solve for v in $||Ev - Tb||_2$ using $E = QX$
  3. solve for w in $||AX^{-1}w - b||_2$ using iterative method with v as starting guess, to relative precision ε
  4. compute $x = X^{-1}w$
- key idea: AX^{-1} has low condition number w.h.p. so iterative method like CG or LSQR converges in $O(\log(1/\varepsilon))$ iterations
- if so the number of operations per step is
  1. $O(mn\log(l)) + O(n^2 l)$
  2. cheap $O(m\log(l) + nl + n^2)$
  3. $O(mn\log(1/\varepsilon))$
  4. cheap $O(n^2)$
- Tygert and Rokhlin theoretically need $l > 4n^2$ but observe that $l = 4n$ gets condition number ≤ 3 in all tests

Analysis of randomized least squares algorithm

Lets consider the case of a very tall-and-skinny matrix, with $m \geq nP$

- the QR and triangular solves needed for preconditioning are then relatively cheap
- we need to consider the initial SRFT product and the matrix-vector products in the iterative method
- we can use a 1D row-blocked layout of A to make the communication cost of matrix-vector products in the iterative method small, $O(n \log(1/\varepsilon) \cdot \beta + \log(1/\varepsilon) \cdot \alpha)$
- for the SRFT, if $n > P$, we can transpose and compute the FFT with cost

$$O(mn/P \cdot \beta + \alpha)$$

- otherwise, it makes sense to compute each FFT using all P processors at the same time
- Q: what would be the BSP communication cost then?
- A: $O(mn \log_{n/P}(n)/P \cdot \beta + \log_{n/P}(n) \cdot \alpha)$

Comparison between randomized methods

Recall the 1D QR row-recursive algorithm achieves the BSP complexity

$$O(n^2 \log(P) \cdot \beta + \log(P) \cdot \alpha)$$

- this is a bit more than the cost of SRFT when $m \approx nP$
- when m is very large the 1D algorithm may be faster than SRFT
- however, it may be possible to do SRFT faster by using the fact that the FFT is subsampled
- the SRFT may also have a relatively higher cache complexity
- a better characterization of the communication complexity of randomized least squares is an open question

Practical performance on least squares

Recent performance studies show that an algorithm based on the Tygert-Rokhlin technique can outperform LAPACK

- Avron, Maymoukov, and Toledo, "Blendenpick: supercharging LAPACK's least-squares solver", 2010
- tested dense high overdetermined ($m \gg n$) systems
- uses LSQR for iterative solver, Hadamard transforms using FFTW
- LAPACK implementation may not have used the most cache-efficient QR in this case

Leverage scores

The SRFT can be seen as a *row-mixing* algorithm

- by taking linear combinations of rows, the projection is guaranteed to captures the range well
- in fact, its possible to just extract a sample of the rows
- however, an oblivious sampling technique is not robust, for instance when a matrix column is nonzero only for one row and we don't include this row in the sample
- sampling based on *leverage scores*: $l \in \mathbb{R}^m$ provide guarantees of accuracy

$$l(i) = \sum_{i=1}^{k} U(i,k)^2$$

where $A = UDV^T$, so U are the singular vectors of A

Computing leverage scores

Obtaining leverage scores can be done by randomized projections

- can again leverage SRFT-like transforms
- see Magdon-Ismail, Mahoney, Woodruff "Fast approximation of matrix coherence and statistical leverage", 2012
- they compute all leverage scores in time $O(mn\log(n))$
- algorithm consists of the same building blocks, SRFT projection, QR/SVD on smaller matrix

Low-rank factorization for sparse matrices

If we want to obtain a low-rank approximate factorization for sparse A

- QR with pivoting or SVD are expensive and complicated due to fill
- Krylov subspace methods can be used to construct a basis for $\{x, Ax, A^2x, \ldots, A^kx\}$ and form a factorization
- randomized projections provide an attractive alternative
  - makes less sense to use SRFT than $n \times l$ Gaussian random matrix B
  - computation of AB can be done all at once, and so is more efficient in communication and synchronization than $O(k)$ steps of iterative methods
  - if high accuracy guarantees are necessary, can use power iteration

$$(AA^T)^qAB$$

in place if AB, improving accuracy exponentially with q