EECS 275 Matrix Computation

Ming-Hsuan Yang

Electrical Engineering and Computer Science
University of California at Merced
Merced, CA 95344
http://faculty.ucmerced.edu/mhyang

Lecture 22

Overview

- Algorithms for Modern Massive Data Sets (MMDS):
  - Explore algorithms for modeling and analyzing massive, high dimensional and nonlinear structured data
  - Bring together computer scientists, computational and applied mathematicians, and practitioners
- Tools: numerical linear algebra, kernel methods, multilinear algebra, randomized algorithms, optimization, differential geometry, geometry and topology, etc.
- Organized by Gene Golub et al. in 2006, 2008 and 2010
- Slides available at `mmds.stanford.edu`

Topics

- Low rank matrix approximation: theory, sampling algorithms
- Nearest neighbor algorithms and approximation
- Spectral graph theory and applications
- Non-negative matrix factorization
- Kernel methods
- Algebraic topology and analysis of high dimensional data
- Higher order statistics, tensors and approximations

Matrix factorization and applications

- Treat each data point as a vector
  - 2D image $\rightarrow$ 1D vector of intensity values
  - 3D models $\rightarrow$ 1D vector of 3D coordinates
  - Document $\rightarrow$ 1D vector of term frequency
- Massive data set
- High dimensional data
- Find a low dimensional representation using eigendecomposition
- See O'Leary's slides

Low rank matrix approximation

- SVD is great but computationally expensive for large scale problems
- Efficient randomized algorithms for low rank approximation with guaranteed error bounds
- CX algorithm [Drineas and Kanna FOCS 01]: randomly pick k columns

$$A_{m\times n} \approx C_{m\times k} X_{k\times n}$$

- CUR algorithm [Drineas and Kannan SODA 03]: randomly pick c columns and r rows

$$A_{m\times n} \approx C_{m\times c} U_{c\times r} R_{r\times n}$$

- Element-wise sampling [Achioptas and McSherry STOC 01]

$$A_{m\times n} \approx S_{m\times n},\ S_{ij} = \begin{cases} A_{ij}/p_{ij} & \text{, with probability } p_{ij} \\ 0 & \text{, with probability } 1-p_{ij} \end{cases}$$

- See Kannan's slides and Drineas' slides.

Approximating matrix multiplication

- Given an m by n matrix A and an n by p matrix B,

$$AB = \sum_{i=1}^{n} \underbrace{A^{(i)} B_{(i)}}_{\in \mathbb{R}^{m \times p}}$$

  where $A^{(i)}$ are the i-th column of A and $B_{(i)}$ is the i-th row of B
- Each term is a rank-one matrix
- Random sampling algorithm
  - fix a set of probabilities p_i $i = 1, \ldots, n$ summing up to 1
  - for $t = 1$ up to s, set $j_t = i$ where $Pr(j_t = i) = p_i$ (pick s terms of the sum with replacement w.r.t. p_i)
  - approximate AB by the sum of s terms, after scaling

$$AB \approx \frac{1}{s} \sum_{t=1}^{s} \frac{1}{p_{j_t}} \underbrace{A^{(j_t)} B_{(j_t)}}_{\in \mathbb{R}^{m \times p}}$$

Approximating matrix multiplication (cont'd)

- In matrix notation

$$A_{m\times n}B_{n\times p} \approx C_{m\times s}R_{s\times p}$$

- Create C and R i.i.d. trials with replacement
- For $t = 1$ up to s, pick a column $A^{(j_t)}$ and a row $B_{(j_t)}$ with probability

$$Pr(j_t = i) = \frac{\|A^{(i)}\|_2\|B_{(i)}\|_2}{\sum_{i=1}^{n}\|A^{(i)}\|_2\|B_{(i)}\|_2}$$

- Include $A^{(j_t)}/(sp_{j_t})^{1/2}$ as a column of C and $B_{(j_t)}/(sp_{j_t})^{1/2}$ as a row of R

Approximating matrix multiplication (cont'd)

- The input matrices are given in "sparse unordered representation," e.g., their non-zero entries are presented as triples (i, j, A_{ij}) in any order
- The expectation of CR (element-wise) is AB
- The nonuniform sampling minimizes the variance of this estimator
- Easy to implement the sampling algorithm in two phases
- If the matrices are dense the algorithm runs in $O(smp)$ time instead of $O(nmp)$ time
- Require only $O(sm + sp)$ memory space
- Does not tamper with the sparsity of the matrices
- For the above algorithm

$$E(\|AB - CR\|_{2,F}) \leq \frac{1}{\sqrt{s}} \|A\|_F \|B\|_F$$

- With probability at least $1 - \delta$

$$\|AB - CR\|_{2,F} \leq \frac{O(\log(1/\delta))}{\sqrt{s}} \|A\|_F \|B\|_F$$

Special case: $B = A^\top$

- If $B = A^\top$, then the sampling probabilities are

$$Pr(\text{picking } i) = \frac{\|A^{(i)}\|_2^2}{\sum_{i=1}^{n} \|A^{(i)}\|_2^2} = \frac{\|A^{(i)}\|_2^2}{\|A\|_F^2}$$

- Also $R = C^\top$, and the error bounds are

$$E(\|AA^\top - CC^\top\|_{2,F}) \leq \frac{1}{\sqrt{s}}\|A\|_F^2$$

- Improvement for the spectral norm bound for the special case

$$E(\|AA^\top - CC^\top\|_2) \leq \frac{4\sqrt{\log s}}{\sqrt{s}}\|A\|_F\|A\|_2$$

- The sampling procedure is slightly different; s columns/rows are kept in expectation, i.e., column i is picked with probability

$$Pr(\text{picking } i) = \min(1, \frac{s\|A^{(i)}\|_2^2}{\|A\|_F^2})$$

Approximating SVD in $O(n)$ time

- The complexity of computing SVD of a m by n matrix A is $O(\min(mn^2, m^2n))$ (e.g., using Golub-Kahan algorithm)
- The top few singular vectors/values can be approximated faster using Lanczos/Arnoldi methods
- Let A_k be rank k approximation of A
- A_k is a matrix of rank k such that $\|A - A_k\|_{2,F}$ is minimized over all rank k matrices
- Approximate SVD in linear time $O(m + n)$
  - sample c columns from A and rescale to form the $m \times c$ matrix C
  - compute the $m \times k$ matrix H_k of the top k left singular vectors C
- Structural theorem: For any probabilities and number of columns

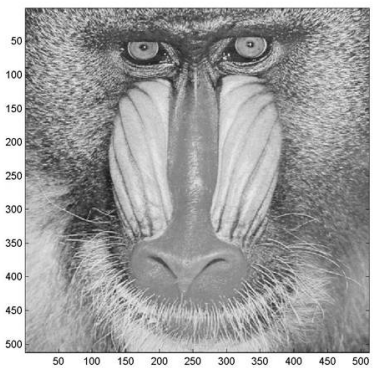
$$\|A - H_k H_k^\top A\|_{2,F}^2 \leq \|A - A_k\|_{2,F}^2 + 2\sqrt{k}\|AA^\top - CC^\top\|_F$$

- Algorithmic theorem: If $p_i = \|A^{(i)}\|_2^2 / \|A\|_F^2$ and $c \leq 4\eta^2 k/\varepsilon^2$, then

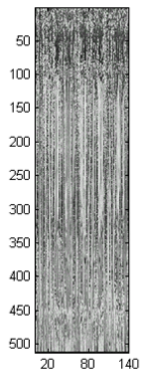
$$\|A - H_k H_k^\top A\|_{2,F}^2 \leq \|A - A_k\|_{2,F}^2 + \varepsilon\|A\|_F^2$$

Example of randomized SVD

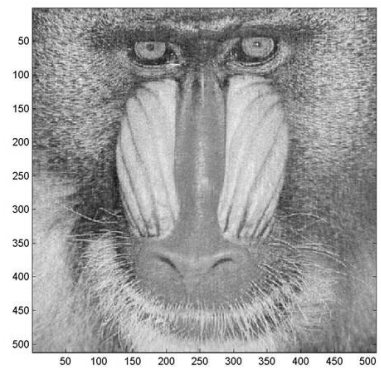
- Compute the top k left singular vectors of matrix C and restore them in the 512-by-k matrix H_k



Original matrix A



After sampling columns C



$H_k H_k^\top A$

Potential problems with SVD

- Structure in the data is *not* respected by mathematical operations on the data:
  - Reification: maximum variance directions
  - Interpretability: what does a linear combination of 6000 genes mean?
  - Sparsity: destroyed by orthogonalization
  - Non-negativity: is a convex but not linear algebraic notion
- Does there exist "better" low-rank matrix approximation?
  - "better" structure properties for certain applications
  - "better" at respecting relevant structure
  - "better" for Interpretability and informing intuition

CX matrix decomposition

- Goal: Find $A_{m\times n} \approx C_{m\times \tilde{c}} X_{\tilde{c}\times n}$ so that $A - CX$ small in some norm
- One way to approach this is

$$\min_{X\in\mathbb{R}^{c\times n}} \|A - CX\|_F = \|A - C(C^{\dagger}A)\|_F$$

  where $C^{\dagger}$ is the pseudoinverse of C
- SVD of A: $A_k = U_k \Sigma_k V_k^{\top}$ where A_k is of $m \times n$, U_k of $m \times k$, Σ of $k \times k$, and $V_k^{\top}$ of $k \times n$
- Subspace sampling: V_k is an orthogonal matrix containing the top k left singular vectors of A
- The columns of V_k are orthonormal vectors, but the rows of V_k, denoted by $(V_k)_{(i)}$ are not orthonormal vectors
- Subspace sampling in $O(SVD_k(A))$ time

$$\forall i = 1, 2, \ldots, n, \quad p_i = \frac{\|(V_k)_{(i)}\|_2^2}{k}$$

Relative error CX

- Relative error CX decomposition
  - compute the probabilities p_i
  - for each $i = 1, 2, \ldots, n$, pick the i-th column of A with probability $\min(1, cp_i)$
  - let C be the matrix containing the sampled columns

Theorem

For any k, let A_k be the best rank k approximation to A. In $O(SVD_k(A))$ we can compute p_i such that if $c = O(k \log k/\varepsilon^2)$ then with probability at least $1 - \delta$

$$\begin{aligned} \min_{X \in \mathbb{R}^{c \times n}} \|A - CX\|_F &= \|A - CC^{\dagger}A\|_F \\ &\leq (1+\varepsilon)\|A - A_k\|_F \end{aligned}$$

CUR decomposition

- Goal: Find $A_{m\times n} \approx C_{m\times c} U_{c\times r} R_{r\times n}$ so that $\|A - CUR\|$ is small in some norm
- Why: After making two passes over A, one can compute provably C, U, and R and store them (sketch) instead of A of $O(m+n)$ vs. $O(mn)$
- SVD of $A_{m\times n} = U_{m\times p}\Sigma_{p\times p}V^{\top}_{p\times n}$ where $\text{ran}(A) = p$
- Exact computation of the SVD takes $O(\min(mn^2, m^2n))$ and the top k left/right singular vectors/values can be computed from Lanczos/Arnoldi methods
- Rank k approximation $A_k = U_k\Sigma_{k\times k}V_k^{\top}$ where $\Sigma_{k\times k}$ is a diagonal matrix with top k singular values of A
- Note that the columns of U_k are linear combinations of all columns of A, and the rows of $V_k^{\top}$ are linear combinations of all rows of A

The CUR decomposition

- Sampling columns for C: use CX algorithm to sample columns of A
  - C has $\tilde{c}$ columns in expectation

$$\underset{m \times \tilde{c}}{C} = \underset{m \times \rho}{U_C} \underset{\rho \times \rho}{\Sigma_C} \underset{\rho \times \tilde{c}}{V_C^\top}$$

  - U_C is the orthogonal matrix containing the left singular vectors of C and ρ is the rank of C
  - Let $(U_C)_{(i)}$ denote the i-th row of U
- Sampling rows for R:
  - subspace sampling in $O(c^2 m)$ time with probability q_i

$$\forall i = 1, 2, \ldots, m \quad q_i = \frac{\|(U_C)_{(i)}\|_2^2}{\rho}$$

  - R has $\tilde{r}$ rows in expectation
- Compute U:
  - Let W be the intersection of C and R
  - Let U be a rescaled pseudoinverse of W

The CUR decomposition (cont'd)

- Put together

$$
\begin{array}{ccccc}
A & \approx & CUR & & \\
A & \approx & C & \left(\begin{bmatrix} D & & W \end{bmatrix}^{\dagger} D \right) & R \\
m \times n & & m \times \tilde{c} & \tilde{r} \times \tilde{c} & \tilde{r} \times n
\end{array}
$$

where D is a diagonal rescaling matrix and

$$U = (DW)^{\dagger} D$$

Theorem

Given C, in $O(c^2 m)$ time, one can compute q_i such that

$$\|A - CUR\|_F \leq (1 + \varepsilon)\|A - C(C^{\dagger} A)\|_F$$

holds with probability at least $1 - \delta$ if $r = O(c \log c / \varepsilon^2)$ rows

Relative error CUR

Theorem

For any k, it takes $O(SVD_k(A))$ time to construct C, U, and R such that

$$\begin{aligned} \|A - CUR\|_F &\leq (1+\varepsilon)\|A - U_k \Sigma_k V_k^\top\|_F \\ &= (1+\varepsilon)\|A - A_k\|_F \end{aligned}$$

holds with probability at least $1-\delta$ by picking

$$O(k \log k \log(1/\delta)/\varepsilon^2) \text{ columns, and}$$
$$O(k \log^2 k \log(1/\delta)/\varepsilon^6) \text{ rows}$$

where $O(SVD_k(A))$ is the time to compute the top k top left/right singular values

- Applications: Genomic microarray data, time-resolved fMRI data, sequence and mutational data, hyperspectral color cancer data
- For small k, in $O(SVD_k(A))$ time we can construct C, U, and R s.t. $\|A - CUR\|_F \leq (1+0.001)\|A - A_k\|_F$ by typically at most $k+5$ columns and at most $k+5$ rows

Element-wise sampling

- Main idea:
  - to approximate matrix A, keep a few elements of the matrix (instead of sampling rows or columns) and zero out the remaining elements
  - compute a rank k approximation to this sparse matrix (using Lanczos methods)
- Create the matrix S from A such that

$$S_{ij} = \begin{cases} A_{ij}/p_{ij} & \text{with probability } p_{ij} \\ 0 & \text{with probability } 1 - p_{ij} \end{cases}$$

- It can be shown that $\|A - S\|_2$ is bounded and the singular values of S and A are close
- Under additional assumptions the top k left singular vectors of S span a subspace that is close to the subspace spanned by the top k left singular vectors of A

Element-wise sampling (cont'd)

- Approximating singular values fast:
  - zero out a large number of elements of A, and scale the remaining ones appropriately
  - compute the singular values of the resulting sparse matrix using iterative methods
  - good choice for $p_{ij} = \frac{sA_{ij}^2}{\sum_{i,j} A_{ij}^2}$ where s denotes the expected number of elements that we seek to keep in S
  - note each element is kept or discarded independently of the others
- Similar ideas that has been used to
  - explain the success of Latent Semantic Indexing
  - design recommendation systems
  - speed up kernel computation

Element-wise sampling vs. row/column sampling

- Row/column sampling preserves subspace/structural properties of the matrices
- Element-wise sampling explains how adding noise and/or quantizing the elements of a matrix perturbs its singular values/vectors
- These two techniques should be complementary
- These two techniques have similar error bounds
- Element-wise sampling can be carried out in one pass
- Running time of element-wise sampling depends on the speed of iterative methods