Stat260/CS294: Randomized Algorithms for Matrices and Data

Homework 2

Instructor: Michael Mahoney *Due: November 13, 2013*

Problem 1

Here, we will consider one approach for extending the ideas underlying the least-squares algorithm we discussed in class to non-tall matrices. Let $A \in \mathbb{R}^{n \times d}$ matrix, where both n and d are large, and where $\text{rank}(A) = k$ exactly, and let $B \in \mathbb{R}^{n \times t}$. Consider the problem

$$\min_{X \in \mathbb{R}^{n \times t}} \|AX - B\|,$$

where $\|\cdot\|$ is a unitarily-invariant matrix norm. The solution to this problem is $X_{opt} = A^+ B$, and here we consider approximating X_{opt} with the solution of a sketched problem, where the sketching matrix $Z \in \mathbb{R}^{n \times r}$. The sketching matrix Z could be a random sampling or random projection matrix, but for now assume only that $\text{rank}(Z^T U) = k$, i.e., the $r \times k$ matrix $Z^T U$ had full rank, where the $n \times k$ matrix U consists of the top k left singular vectors of A, i.e., all the singular vectors associated with nonzero singular values. Recall that the min-length solution to the sketched problem can be expressed as

$$\text{argmin}_{X \in \mathbb{R}^{n \times t}} \|Z^T AX - Z^T B\| = (Z^T A)^+ Z^T B.$$

(a) First, show that

$$\|A(Z^T A)^+ Z^T B - B\|_\xi \leq \|U^\perp {U^\perp}^T B\|_\xi + \|(U^T Z)^+ Z^T U^\perp {U^\perp}^T B\|_\xi,$$

where $U^\perp$ is an $n \times (n-k)$ orthogonal matrix spanning the complement of $\text{span}(A) = \text{span}(U)$. Establish this result for both $\xi = 2, F$, i.e., for both the spectral and Frobenius norm. Comment on the structure of the proof and how this result would generalize to other unitarily-invariant matrix norms.

(b) Next, show that

$$\|A(Z^T A)^+ Z^T B - B\|_\xi \leq \|U^\perp {U^\perp}^T B\|_\xi + \|U^T Z Z^T U^\perp {U^\perp}^T B\|_\xi + \max_i |\sigma_i(Z^T U) - \sigma_i^{-1}(Z^T U)| \|Z^T U^\perp {U^\perp}^T B\|_\xi.$$

Again, comment on the structure of the proof and how this result would generalize to other unitarily-invariant matrix norms.

(c) Next, assume that Z is a random sampling matrix like we discussed in class, but don't make any assumptions about the probabilities, i.e., let them be arbitrary. Provide a bound for $\|Z^T U^\perp {U^\perp}^T B\|_F$, first in expectation and then with probability at least $1 - \delta$ by using Markov's inequality.

(d) Next assume that Z is a random sampling matrix, where the importance sampling probabilities depend on the leverage scores of A, which recall is exactly rank k. Provide a bound on $\|U^T Z Z^T U^\perp {U^\perp}^T B\|_\xi$, first in expectation and then with probability at least $1 - \delta$ by using Markov's inequality. Do this by extending the approximate matrix multiplication algorithm we discussed in class, but note that the sampling probabilities only depend on information in U^T and not in $U^\perp {U^\perp}^T B$.

(e) Next, provide a bound on $\max_i |\sigma_i(Z^T U) - \sigma_i^{-1}(Z^T U)|$, by modifying the bound we derived in class.

(f) Finally, put all these results together to show that the solution to the original problem, when $\xi = 2, F$, i.e., when the error is measured with respect to the spectral and Frobenius norm, can be approximated to $(1 \pm \epsilon)$ relative error, if leverage score importance sampling probabilities are used. Be precise about the number of samples that need to be chosen, etc. to get $(1 \pm \epsilon)$ error, with probability at least $1 - \delta$.

Problem 2

In the class, we considered random sampling algorithms for low-rank approximation, and in particular an additive-error column-sampling algorithm that involved sampling with respect to an importance sampling distribution proportional to the Euclidean norms squared of the columns of the input matrix, and a relative-error column-sampling algorithm that involved sampling with respect to an importance sampling distribution proportional to the leverage scores, relative to a low-rank space, of the input matrix. Here, we consider the extension of these ideas to random projection algorithms. In particular, given as input an $m \times n$ matrix A and a rank parameter k, construct a $n \times \ell$ random projection matrix Π, where $\ell \gtrsim \alpha k/\epsilon^2$ for some constant α, and compute $B = A\Pi$.

(a) By applying the matrix perturbation analysis we used in the additive-error random sampling algorithm as well as Johnson-Lindenstrauss ideas applied to the rows of A, show that

$$\|A - P_{B_k}A\|_F \leq \|A - P_{U_k}A\|_F + \epsilon\|A\|_F,$$

where P_{B_k} is a projection matrix onto the best rank-k approximation to B, and where P_{U_k} is a projection matrix onto the top k left singular vectors of A.

(b) By applying the analysis we used in the relative-error random sampling algorithm as well as Johnson-Lindenstrauss ideas applied to the rows of the truncated matrix consisting of the top-k singular vectors of A, show that

$$\|A - P_{B_k}A\|_F \leq (1+\epsilon)\|A - P_{U_k}A\|_F.$$

In each case, provide the tightest bounds you can for ℓ, when the random projection matrix Π is a matrix of i.i.d. Gaussians and when it is a Hadamard-based random projection. In addition, provide the tightest bound you can for the running time in each case, both for when the input matrix is dense and when the input matrix is sparse.

Problem 3

Here, we will consider the empirical performance of random sampling and random projection algorithms for approximating least-squares. You may use Matlab, or C, or R, or whatever software package you prefer to do your implementations, but be sure to describe what you used in sufficient detail that someone else could reproduce your results.

Let A be an $n \times d$ matrix, with $n \gg d$, b be an n-vector, and consider approximating the solution to $\min_x ||Ax - b||_2$. Generate the matrices A from one of three different classes of distributions introduced below.

- Generate a matrix A from multivariate normal $N(1_d, \Sigma)$, where the (i,j)th element of $\Sigma_{ij} = 2 \times 0.5^{|i-j|}$. (Refer to as GA data.)
- Generate a matrix A from multivariate t-distribution with 3 degree of freedom and covariance matrix Σ as before. (Refer to as T_3 data.)
- Generate a matrix A from multivariate t-distribution with 1 degree of freedom and covariance matrix Σ as before. (Refer to as T_1 data.)

To start, consider matrices of size $n \times d$ equal to 500×50.

(a) First, for each matrix, consider approximating the solution by randomly sampling a "small" number r of rows/elements (i.e., constraints of the overconstrained least-squares problem) in one of three ways:

uniformly at random; according to an importance sampling distribution that is proportional to the Euclidean norms squared of the rows of A; and according to an importance sampling distribution that is proportional to the leverage scores of A. In each case, plot the error as a function of the number r of samples, paying particular attention to two regimes: $r = d, d+1, d+2, \ldots, 2d$; and $r = 2d, 3d, \ldots$. Show that the behavior of these three procedures is most similar for GA data, intermediate for T_3 data, and most different for T_1 data; and explain why, and explain similarities and differences.

(b) Next, for each matrix, consider approximating the solution by randomly projecting rows/elements (i.e., constraints of the overconstrained least-squares problem) in one of two ways: a random projection matrix, in which each entry is i.i.d. $\{\pm 1\}$, with appropriate variance; and a random projection matrix, in which each entry is i.i.d. Gaussian, with appropriate variance. In each case, plot the error as a function of the number of samples, i.e., dimensions on which the data are projected, paying particular attention to two regimes: $r = d, d+1, d+2, \ldots, 2d$; and $r = 2d, 3d, \ldots$. Describe and explain similarities and differences between these three procedures for GA data, T_3 data, and T_1 data.

(c) Finally, for each matrix, consider approximating the solution by randomly projecting rows/elements (i.e., constraints of the overconstrained least-squares problem) with "sparse" projection matrices. In particular, consider a random projection matrix, in which each entry is i.i.d. either 0 or Gaussian, where the probability of 0 is q and the probability of Gaussian is $1 - q$. (Remember to rescale the variance of the Gaussians appropriately, depending on q, as we discussed in class.) For q varying from 0 to 1, in increments sufficiently small to illustrate the phenomena we discussed in class, plot the error for solving the least-squares problem. Describe and explain how this varies as a function of the number of samples, i.e., dimensions on which the data are projected, paying particular attention to two regimes: $r = d, d+1, d+2, \ldots, 2d$; and $r = 2d, 3d, \ldots$. Describe and explain similarities and differences between these three procedures for GA data, T_3 data, and T_1 data.

Next, we describe how these behave for larger problems. To do so, we will work with dense random projection algorithms in which the projection matrix consists of i.i.d. $\{\pm 1\}$ random variables, with appropriate variance. Fix a value of d, and let n increase from roughly $2d$ to roughly $100d$. The exact value of d and n will depend on your machine, your computational environment, etc., so be sure to describe what you are using. Choose a value of the oversampling parameter, etc., so that you get reasonably-good low-precision approximate solutions to the original least-squares problem. (You should expect $d \approx 500$ should work; and if you can't do the full plot to $100d$, don't worry, since the point is to get large enough to illustrate the phenomena below.)

(d) Plot the running time of the random projection algorithm versus the running time of solving the problem with a call to a QR deccomposition routine provided by your system as well as the running time of solving the problem with a call to an SVD routine provided by your system. Illustrate that, for smaller problems the random projection methods are not faster, but that for larger problems, the random projection mathods are slightly faster and/or can be used to solve larger problems than QR or SVD. Explain why is this the case, since you are not using "fast" Hadamard-based projections.