Stat260/CS294: Randomized Algorithms for Matrices and Data

Lecture 6 - 09/23/2013

Lecture 6: Sampling/Projections for Least-squares Approximation

Lecturer: Michael Mahoney *Scribe: Michael Mahoney*

Warning: these notes are still very rough. They provide more details on what we discussed in class, but there may still be some errors, incomplete/imprecise statements, etc. in them.

6 Sampling/Projections for Least-squares Approximation

In the next several classes, we will be discussing RandNLA algorithms for the least-squares problem. This is a fundamental problem in linear algebra, and many of the methods in RandNLA are most easily introduced and understood in this relatively-simple setting. Here is the reading for today.

- Chapter 4 of: Mahoney, "Randomized Algorithms for Matrices and Data"
- Drineas, Mahoney, Muthukrishnan, and Sarlos, "Faster Least Squares Approximation"
- Sarlos, "Improved Approximation Algorithms for Large Matrices via Random Projections"

Today, we will start this by covering two topics.

- A brief overview of LS problems.
- A brief overview of sketching methods for LS problems.

6.1 Some general thoughts on LS problems

In many applications, we want to find an approximate solution to a problem or set of equations that, for noise reasons or whatever other reasons, does *not* have a solution, or not unrelatedly does not have a unique solution. A canonical example of this is given by the very overconstrained (i.e., overdetermined) least-squares (LS) problem, and this will be our focus for the next several classes. Some (but not all) of what we we discuss will generalize to very underconstrained (i.e., undetermined) LS problems, roughly-square LS problems, etc., but here we focus on the simple setup of very overconstrained LS problems.

Let $A \in \mathbb{R}^{n \times d}$ and $b \in \mathbb{R}^n$ be given. If $n \gg d$, i.e., there are many more rows/constraints than columns/variables, then in general there does not exist a vector x such that $Ax = b$. Basically, this is since b may have a part that sits outside the column space of A. That is, $b \in \mathbb{R}^n$, but $\text{span}(A)$ is a d-dimensional subspace of $\mathbb{R}^n$, and so with even a little noise, numerical instability, etc., there will be a part of b that is not captured as a linear combination of the columns of A.

In this case, a popular way to find th "best" vector x such that $Ax \approx b$ is to minimize the norm of the residuals, i.e., to solve $\min_{x \in \mathbb{R}^d} ||Ax - b||$, where $|| \cdot ||$ is some norm. The most popular choice is the Euclidean or ℓ_2 norm, in which case the LS problem is to minimize the sum of squares of the residaul, i.e., to solve

$$\mathcal{Z} = \min_{x \in \mathbb{R}^d} \|Ax - b\|_2 . \tag{1}$$

If we let A^+ denote the Moore-Penrose generalized inverse of A, then

$$x_{opt} = A^+ b \tag{2}$$

is the solution to the LS problem. Actually, we should note that $x = A^+ b + \xi$, where $\xi \perp \text{span}(A)$, i.e., where $\xi \in \mathbb{R}^n$ is any vector perpendicular to the column span of A, solves the LS problem given in Eqn. (1), and the solution given in Eqn. (2) actually is the minimal-ℓ_2-norm solution to the LS problem. Since we will be interested in working with this shortest or minimal-norm solution, we will call it *the* solution to the LS problem. For most of what we will talk about in the very overconstrained regression problem, worrying about having any components in this perpendicular space will not be a problem, basically since there is not a "rest of the space" to deal with. We will see, however, when we consider the extension of these ideas to low-rank approximation that we will need to be a little more careful in dealing with how the top part of the spectrum of a matrix and its sketch interact with the bottom part of the spectrum of the matrix and its sketch.

This LS problem is ubiquitous and has many well-known interpretations. A statistical interpretation is that it provides the best linear unbiased estimator to the original problem. A geometric interpretation is that the solution is simply the orthogonal projection of b onto the span(A). And so on. Note that the latter interpretation is basically a statement about the data at hand, while the former interpretation is basically a statement about models and unseen data. This parallels the algorithmic-statistical approaches we mentioned earlier. Along these lines, here are 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"? (By this, we mean, say, to machine precision.) The answer (roughly) is that the running time in the RAM model to solve the LS problem is $O(nd^2)$ time, and—as we will describe below—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? (By this, we mean that it is the optimum for some underlying statistical model.) The answer (roughly) is that it is when the data are "nice" in ways that mean that large-sample theory can be applied, e.g., that there are a large number of small components such that measure concentrates and that there are no small number of components that are particularly important or influential. As we will describe below, this can be checked with empirical statistics such as the leverage scores and with other regression diagnostics.

We will return to both of these points in detail below. In particular, in terms of running time, we should be thinking about algorithms that run in $o(nd^2)$ time, and the role of the statistical leverage scores which have been used in regression diagnostics in our worst case algorithms will be particularly important.

To see how to solve the LS problem , we can define a function $f(x) = \|Ax - b\|_2^2 = (Ax-b)^T(Ax-b)$, and to find the minimizer of this function, we can set the derivative equal to zero, $\frac{\partial f}{\partial x} = 0$, noting

that the second derivative is positive (or SPD, if the matrix A has full column rank). Then we get $A^TAx - A^Tb = 0$, which is just the normal equations,

$$A^TAx = A^Tb. \tag{3}$$

If A has full column rank, then A^TA is square and has full rank, and this is a $d \times d$ system of linear equations with solution

$$x_{opt} = (A^TA)^{-1}A^Tb. \tag{4}$$

Of course, forming and solving the normal equations in this way is typically not recommended, but it at least provides a form for the solution. But, in particular, note that this means that $b^{\perp} \equiv b - Ax_{opt}$ is orthogonal to span(A), i.e., ${b^{\perp}}^TA = 0$, or equivalently since span(U) = span(A), where U is an orthogonal basis for span(A) computed from the SVD or a QR decomposition, we have that ${b^{\perp}}^TU = 0$.

With respect to the question of how long it takes to solve LS problems, one can use so-called direct methods or so-called iterative methods. Here is a rough outline of *direct methods* for solving LS problems.

- Cholesky decomposition: If A is full rank and well-conditioned, then one can use the Cholesky decomposition to compute an upper triangular matrix R such that $A^TA = R^TR$, and then one can solve the normal equations $R^TRx = A^Tb$.
- QR decomposition: Somewhat slower but more numerically stable, especially if A is rank-deficient or ill-conditioned, involves computing a QR decomposition $A = QR$ and then solving $Rx = Q^Tb$.
- SVD: Somewhat more expensive but better still if A is very ill-conditioned, involves computing the SVD, $A = U\Sigma V^T$, where this is the thin or economical SVD (i.e., things that are zeroed-out by singular values are not included), in which case $x_{opt} = V\Sigma^{-1}U^Tb$.

The complexity of all of theses methods is $O(nd^2)$. That is, although the numerical properties differ and the constant factors differ, all three classes of algorithms asymptotically take a constant times nd^2 time. In most cases, using QR is a good tradeoff—but note that in certain large-scale applications, the usual rules, e.g., don't form the normal equations or don't compute the SVD, don't always hold. In addition, for most of what we will do we are not interested in these details, since we will be computing a randomized sketch and then calling a traditional algorithm as a black box, and so we will treat all of these as similar in that sense.

Another broad class of algorithm for solving LS and other problems are *iterative methods*. We will return to these later. But here we simply note that many of them boil down to conjugate gradient ideas, and (e.g., with CGNR) they typically have a running time that is something like $O(\kappa(A)\text{nnz}(A)\log(1/\epsilon))$ time. In particular, the running time depends on the error parameter as $O(\log(1/\epsilon))$, and not poly$(1/\epsilon)$, time.

A third broad class of algorithms for solving LS problems use the recursive structure of well-known *Strassen-like methods* for matrix multiplication—basically, those algorithms can also be applied to rectangular LS problems, with similar improvements in worst-case running time. These are of theoretical interest, and thus they are worth mentioning, but are never used in practice, and so we won't focus on them.

6.2 Deterministic and randomized sketches for LS Problems

At a high level, RandNLA algorithms—in general but in particular when applied to LS problems—do one of two thing.

- Construct a sketch (with a random sampling or random projection procedure) and solve the subproblem on that sketch with a traditional black box NLA algorithm.
- Construct a sketch (with a random sampling or random projection procedure) and use that sketch as a preconditioner for a traditional black box NLA algorithm on the original problem.

In both cases, the randomized algorithms interface with traditional NLA algorithms (in two different ways), and so let's start by asking "What are properties of sketches that lead to good solutions?"

Thus, for the rest of today, we won't specify whether our sketches are deterministic or randomized. We will be interested in properties of some sketch and how they relate to necessary/sufficient conditions to get good approximate solutions to the original LS problem. Roughly, we will show certain conditions, and later we will show that one can construct sketches via random sampling/projection that satisfy those conditions quickly. More generally, it is worth keeping in mind what are properties of the sketches and linear algebra versus what are properties of the randomization.

In this course, we will deal with so-called linear sketches. Operationally, this means that we can write the operation/action of the sketch as a linear function. Note, in particular, that random projection matrices and random sampling matrices satisfy this. The advantages of working with linear sketches include: we can take advantage of linear theory stuff (this may be obvious for NLA, but it is actually useful much more generally); and it is easy to update sketches (this matters in streaming, memory constrained environments, etc.). The disadvantages of working with linear sketches mean that we might loose something, compared to using a broader class of sketches. Note, however, note that a lot of work in ML, e.g., with kernels, say basically that tamely nonlinear stuff can be done linearly. So, this is an idea that is used more generally, and in general we will work with linear sketches. But a question worth keeping in the back of your mind is: what are metric spaces that don't embed well, either in general or with linear sketches, since those might have problems.

Now, onto the properties that we want a good sketch to have to help solve linear regression. Let's let X be an arbitrary sketching matrix, i.e., any matrix. By this, we mean an arbitrary matrix (randomized or not, tractable to compute or not, etc.) that we are going to apply to A and b to construct a sketch. For example, X could be a sampling matrix like we saw before with matrix multiplication sampling algorithms, X could be a dense projection matrix like we saw before, or X could be S^THD, THD, or other structured random projections. When forming a sketch, we will be replacing the original LS problem

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

the solution of which is $x_{opt} = A^+b$, with a sketched LS problem

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

the solution of which is $\tilde{x}_{opt} = (XA)^+Xb$. And we want to ask what are properties that X needs to satisfy s.t.

$$\begin{aligned} x_{opt} &\approx \tilde{x}_{opt} \\ \|A\tilde{x}_{opt}\|_2 &\approx \|Ax_{opt} - b\|_2 . \end{aligned}$$

Several comments are in order.

- The second requirement is a statement and is more common in TCS where one might not even be able to obtain a "certificate" for the solution. The first requirement is usually harder to get in worst-case approximation algorithm theory but usually comes for free in matrix problems with ℓ_2 objectives.
- Moreover, the bound on the vector achieving the optimal solution is typically of greater interest in NLA and ML/data applications, where the vector is used form something downstream such as classification.
- For matrix extensions of these ideas, we typically want results for the objective, since we will measure quality by norm reconstruction, rather than capturing an actual set of vectors or an actual subspace.
- For ℓ_1 and other objectives, the connections between objectives and certificates and what one can compute from the other are more tenuous.

With this in place, here are two important structural conditions. Let $A = U\Sigma V^T = U_A\Sigma_A V_A^T$ be the SVD of A, and let $b^\perp = U_A U_A^T a$ be the part of b sitting outside span(A), and note that $\mathcal{Z} = \|Ax_{opt} - b\|_2 = \left\|b^\perp\right\|_2$. Then, here are two conditions.

- **Condition I:**
$$\sigma_{min}(XU_A) \geq 1/\sqrt{2}. \tag{5}$$
- **Condition II:**
$$\left\|U_A X^T X b^\perp\right\|_2^2 \leq \frac{\epsilon}{2}\mathcal{Z}^2. \tag{6}$$

Before proceeding, let's consider these conditions; here are several comments.

- We have defined this in terms of U_A from the SVD, but we get exactly the same structural conditions for any matrix Q, e.g., from the QR decomposition. Although this issue doesn't matter here so much, it will matter more when we consider the extension of these ideas to low-rank matrix approximation problems.
- Although Condition I is that $\sigma_{min}(XU_A) \geq 1/\sqrt{2}$, i.e., we only need a lower bound on the singular values of $\sigma(XU_A)$, all of our constructions will be such that $\|1 - \sigma_i(XU_A)\| \leq 1 - 2^{-1/2}$. Thus, one should think if XU_A as an approximate isometry (or, more imprecisely, an approximate rotation). We want that X, viewed as a function $f : \mathbb{R}^n \to \mathbb{R}^r$, with $r \approx d$, is roughly a rotation/isometry. In particular, although we zero-out most of the coordinates, the mapping to the remaining are an acute perturbation with respect to the original data.
- Condition II states that $Xb^\perp = XU_A^\perp {U_A^\perp}^T b$ is still roughly orthogonal to XU_A. It is not surprising that we need a condition on the right hand side vector b, but it is surprising that we can satisfy this (with random sampling and random projection algorithms) without looking at the right hand side at all, i.e., either data-agnostic random projections or random sampling methods that only depend on information in A. Of course, in certain practical cases, one can sometimes do better by looking at the right hand side, and extensions to ℓ_1 regression, etc., typically need to look at the right hand side, although sometimes implicitly by defining an augmented matrix $\begin{bmatrix} A & -b \end{bmatrix}$ and working on that.

Here are a few extreme cases to consider.

- Let $X = I_n$, viewed as a function $\mathbb{R}^n \to \mathbb{R}^n$. Then, $\sigma_{min}(XU_A) = \sigma_{max}(XU_A) = 1$ (since $XU_A = U_A$), and $U_A^T X^T X b^\perp = 0$. In this case, constructing X is "easy," and solving the "subproblem" is the same as the original problem and so it "hard."
- Let $X = U_A^T$, viewed as a function $\mathbb{R}^n \to \mathbb{R}^d$. Then, $\sigma_{min}(XU_A) = \sigma_{max}(XU_A) = 1$ (since $XU_A = I_d$), and $U_A^T X^T X b^\perp = 0$. In this case, constructing X is "hard," since it involves computing U_A^T which almost amounts to solving the original problem, but solving the subproblem is easy, I think. (Similar statements could be made if X was the "Q" matrix from a QR decomposition.)

So, the goal for what we will be doing will be to construct a sketch that is relatively-easy to construct and such that solving the subproblem is also relatively easy, in the sense that both take $o(nd^2)$ time.

Given all of that, here are our main lemmas for those structural conditions. Basically, these lemmas says that if we have a sketching matrix X that satisfies those two conditions, then we have a relative-error approximation to the solution to the LS problem, on both the objective and the certificate. We will do the first (a lemma about being close with respect to the objective function value) now, and we will do the next two (lemmas about being close with respect to the certificate or solution vector) next time. Note that, for these lemmas, we are interested in quality-of-approximation guarantees for a given sketching matrix X, i.e., we don't worry about the time it takes to construct X. We will get to running time considerations soon enough.

Lemma 1 *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 X satisfies conditions (5) and (6) above, i.e., Condition I and Condition II, for some $\epsilon \in (0, 1)$. Then, the solution vector $\tilde{x}_{opt}$ to the least squares approximation problem satisfies:*

$$\|A\tilde{x}_{opt} - b\|_2 \leq (1 + \epsilon)\mathcal{Z}. \tag{7}$$

Before proceeding with the proof of this lemma, here are a few comments.

- This lemma is a deterministic statement, i.e., there is no randomness, and it holds for any matrix X that satisfies those two conditions. Failure probabilities in randomized matrix algorithms will enter into the construction of X and whether X satisfies those two conditions.
- We will be mostly interested in worst-case *a priori* bounds, and we will show that X satisfies these two conditions for worst-case input; but one could easily ask for *a posteriori* bounds, by, e.g., sampling/projecting less aggressively and checking if these conditions are satisfied. We won't do this for the LS regression problem, but we will consider this approach for low-rank matrix approximation problems, and this is probably the way to implement randomized matrix algorithms more generally.

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

$$\begin{aligned}
\min_{x\in\mathbb{R}^d} \|Xb - XAx\|_2^2 &= \min_{y\in\mathbb{R}^d} \left\|X(Ax_{opt} + b^\perp) - XA(x_{opt} + y)\right\|_2^2 \quad (8)\\
&= \min_{y\in\mathbb{R}^d} \left\|Xb^\perp - XAy\right\|_2^2 \\
&= \min_{z\in\mathbb{R}^d} \left\|Xb^\perp - XU_A z\right\|_2^2. \quad (9)
\end{aligned}$$

(8) follows since $b = Ax_{opt} + b^\perp$ and (9) follows since the columns of the matrix A span the same subspace as the columns of U_A. Now, let $z_{opt} \in \mathbb{R}^d$ be such that $U_A z_{opt} = A(x_{opt} - \tilde{x}_{opt})$, and note that z_{opt} minimizes eqn. (9). The latter fact follows since

$$\left\|Xb^\perp - XA(x_{opt} - \tilde{x}_{opt})\right\|_2^2 = \left\|Xb^\perp - X(b - b^\perp) + XA\tilde{x}_{opt}\right\|_2^2 = \|XA\tilde{x}_{opt} - Xb\|_2^2.$$

XXX. THERE IS A SIGN ISSUE THERE TO FIX. Thus, by the normal equations (3), we have that

$$(XU_A)^T XU_A z_{opt} = (XU_A)^T Xb^\perp.$$

Taking the norm of both sides and observing that under condition (5) we have $\sigma_i((XU_A)^T XU_A) = \sigma_i^2(XU_A) \geq 1/\sqrt{2}$, for all i, it follows that

$$\|z_{opt}\|_2^2/2 \leq \left\|(XU_A)^T XU_A z_{opt}\right\|_2^2 = \left\|(XU_A)^T Xb^\perp\right\|_2^2. \quad (10)$$

Using condition (6) we observe that

$$\|z_{opt}\|_2^2 \leq \epsilon \mathcal{Z}^2. \quad (11)$$

Let us rewrite the norm of the residual vector as

$$\begin{aligned}
\|b - A\tilde{x}_{opt}\|_2^2 &= \|b - Ax_{opt} + Ax_{opt} - A\tilde{x}_{opt}\|_2^2 \\
&= \|b - Ax_{opt}\|_2^2 + \|Ax_{opt} - A\tilde{x}_{opt}\|_2^2 \quad (12)\\
&= \mathcal{Z}^2 + \|U_A z_{opt}\|_2^2 \quad (13)\\
&\leq \mathcal{Z}^2 + \epsilon\mathcal{Z}^2, \quad (14)
\end{aligned}$$

where (12) follows by Pythagoras, since $b - Ax_{opt} = b^\perp$, which is orthogonal to A, and consequently to $A(x_{opt} - \tilde{x}_{opt})$; (13) follows by the definition of z_{opt} and $\mathcal{Z}$; and (14) follows by (11) and the orthogonality of U_A. The first claim of the lemma follows since $\sqrt{1+\epsilon} \leq 1 + \epsilon$.

$\diamond$

Next time, we will start by proving that the vector achieving the optimum in the subsampled problem is a very good approximation to the vector achieving the optimum in the original problem.