CS 15-859: Algorithms for Big Data Fall 2017

Lecture 3-2 — September 21, 2017

Prof. David Woodruff *Scribe: Tom Tseng*

1 Affine embeddings

1.1 How to achieve an embedding

Previously, we considered the following problem: we are given matrices A and B and wish to minimize $\|AX - B\|_{\mathrm{F}}^2$ over matrices X. Our technique for the special case of linear regression was to use a subspace embedding and then solve the embedded, lower-dimension problem exactly. However, if B has many columns, then we cannot efficiently use subspace embeddings over the column span of A and B. Instead, we seeked an affine embedding, where we hope to find matrix S such that

$$\|S(AX - B)\|_{\mathrm{F}} = (1 \pm \varepsilon) \|AX - B\|_{\mathrm{F}} \tag{1}$$

for all X. This is indeed possible. Take a matrix S with the following properties:

- Approximate matrix product: for fixed matrices C and D, $\left\|CS^TSD - CD\right\|_F$ is small with large probability.
- Subspace embedding: norms of vectors in the column space of A are approximately preserved by S with large probability.
- Frobenius norm preserving: for a fixed matrix C, $\|SC\|_{\mathrm{F}}^2 = (1 \pm \varepsilon) \|C\|_{\mathrm{F}}^2$ with constant probability.

For example, if S is a CountSketch matrix, then these properties will be satisfied.

Define $B^* = AX^* - B$ where X^* minimizes $\|AX^* - B\|_{\mathrm{F}}^2$. We showed previously that with the first two properties, we get

$$\|S(AX - B)\|_{\mathrm{F}}^2 - \|SB^*\|_{\mathrm{F}}^2 - \left(\|AX - B\|_{\mathrm{F}}^2 - \|B^*\|_{\mathrm{F}}^2\right) = \pm 2\varepsilon \|AX - B\|_{\mathrm{F}}^2 . \tag{2}$$

Then we rearrange and apply the third property to get

$$\|S(AX - B)\|_{\mathrm{F}}^2 = (1 \pm 2\varepsilon) \|AX - B\|_{\mathrm{F}}^2 \pm \varepsilon \|B^*\|_{\mathrm{F}}^2 = (1 \pm 3\varepsilon) \|AX - B\|_{\mathrm{F}}^2 ,$$

where the second equality is true because $\|B^*\|_{\mathrm{F}}^2 \leq \|AX - B\|_{\mathrm{F}}^2$ by how we defined B^*. This shows that with good probability, we've achieved our goal of finding an affine embedding as defined in equation (1) after adjusting ε.

1.2 Missing lemmas

In showing equation (2), we used a few simple lemmas that we deferred the proofs of. We now present the proofs.

Lemma 1. *For $m \times n$ matrices A and B,*

$$\|A + B\|_F^2 = \|A\|_F^2 + \|B\|_F^2 + 2\operatorname{Tr}\left(A^\top B\right).$$

Proof. We can write the squared Frobenius norm of a matrix as the sum of the squared 2-norms of the columns of the matrix. This gives

$$\begin{aligned}\|A+B\|_{\mathrm{F}}^2 &= \sum_{j=1}^{n} \|A_{*,j} + B_{*,j}\|_2^2 = \sum_{j=1}^{n} \langle A_{*,j} + B_{*,j}, A_{*,j} + B_{*,j}\rangle \\ &= \sum_{j=1}^{n} \|A_{*,j}\|_2^2 + \sum_{j=1}^{n} \|B_{*,j}\|_2^2 + 2\sum_{j=1}^{n} \langle A_{*,j}, B_{*,j}\rangle = \|A\|_{\mathrm{F}}^2 + \|B\|_{\mathrm{F}}^2 + 2\operatorname{Tr}\left(A^\top B\right). \quad \blacksquare\end{aligned}$$

Lemma 2. *For an $m \times n$ matrix A and an $n \times m$ matrix B,*

$$\operatorname{Tr}(AB) \le \|A\|_F \|B\|_F.$$

Proof. We have

$$\operatorname{Tr}(AB) = \sum_{i=1}^{m} \langle A_{i,*}, B_{*,i}\rangle \le \sum_{i=1}^{m} \|A_{i,*}\|_2 \|B_{*,i}\|_2 \le \sqrt{\sum_{i=1}^{m} \|A_{i,*}\|_2^2}\sqrt{\sum_{i=1}^{m} \|B_{*,i}\|_2^2} = \|A\|_{\mathrm{F}} \|B\|_{\mathrm{F}}$$

where the first inequality comes from applying Cauchy-Schwarz on each of the terms in the summation, and the second inequality comes from applying Cauchy-Schwarz by treating the whole summation as an inner product. $\blacksquare$

We also need to show that S indeed satisfifes the Frobenius norm preserving property as mentioned in section 1.1.

Lemma 3. *If S is a $k \times n$ CountSketch matrix with $k = \Omega\left(1/\varepsilon^2\right)$ and B^* is a $n \times p$ matrix, then $\|SB^*\|_F^2 = (1 \pm \varepsilon)\|B^*\|_F^2$ with constant probability.*

Proof. This is problem 3 in homework 1. The proof uses and is similar to the proof for showing CountSketch satisfies the Johnson–Lindenstrauss property. $\blacksquare$

2 Low-rank approximation

2.1 Motivation

Say we are given an $n \times d$ matrix A, where both dimensions n and d are large. Suppose we had a rank-k matrix B that approximates A. Why might we want such a matrix B?

One reason is that B is easier to store. To store A, we need to store $n \times d$ parameters. As for B, we can decompose B into $B = UV$ where U has dimensions $n \times k$ and V has dimensions $k \times d$. Then we can store B by storing U and V, which takes $nk + kd = k(n + d)$ parameters. If k is small, this gives us large storage savings.

Similarly, multiplying against B is faster than multiplying against A. For a vector $x \in \mathbb{R}^d$, computing Ax takes $O(nd)$ time. On the other hand, computing $Bx = U(Vx)$ takes $O(k(n + d))$ time.

Lastly, in some applications, we might expect that A "truly" has low rank and only has high rank due to noise and corruption. By finding a low-rank approximation, we clean out the noise and perhaps place the data in a more interpretable state.

This inspires the following problem: given a matrix A and a parameter k, find a matrix of B of rank at most k such that B is close to A under some norm, that is, B minimizes $\|B - A\|$ among all rank-k matrices. This problem has applications in data mining, recommendation systems, and other areas.

2.2 Best solution: truncated singular value decomposition

We can compute a best solution through a truncated singular value decomposition (SVD). To approximate A with a rank-k matrix, compute the SVD of A to get $A = U\Sigma V^\top$. Define Σ_k by zeroing out all the singular values in Σ below the k-th row, and define $A_k = U\Sigma_k V^\top$. Note that we can also drop all but the top k rows or $V^\top$ and drop all but the top k columns of U without changing A_k.

It turns out that A_k minimizes $\|A_k - A\|$ among all rank-k matrices for many norms, including the Frobenius norm and the spectral norm. More generally, A_k is the best rank-k approximation under any unitarily invariant norm.

We can compute the SVD of A in $O\left(\min\left(nd^2, n^2 d\right)\right)$ time.

2.3 Faster approximate solution

2.3.1 Background

Like with linear regression, we may find that computing the exact solution to to this problem is too slow. To remedy this, we relax the problem and hope to compute some matrix A' with good relative error, that is,

$$\|A' - A\| \leq (1 + \varepsilon) \|A_k - A\|$$

where A_k is a rank-k matrix minimizing $\|A_k - A\|$. In this lecture, we'll focus on the case where the norm is the Frobenius norm.

This relaxed problem can indeed be solved more quickly in $O(\text{nnz}(A) + (n + d)\,\text{poly}(k/\varepsilon))$ time, where $\text{nnz}(A)$ is the number of non-zero entries in A [1]. We can show that to get reasonable error bounds, we need to spend $\Omega(\text{nnz}(A))$ time. Thus, for small k, this is solving time is close to optimal. Note that even for dense matrices, we can solve the relaxed problem in almost $O(nd)$ time, which is a significant running time improvement over computing the SVD of A and finding an exact solution. For sparse matrices, the improvement is even more dramatic.

2.3.2 Algorithm: setup

We have an $n \times d$ matrix A we wish to find a rank-k approximation to. We will use the same high-level technique we used to solve approximate regression: we reduce the dimension of the problem and then solve the lower-dimensional problem exactly.

Think of the rows of A as points in $\mathbb{R}^d$. Take some random matrix S with $m \ll n$ and n columns. The rows of SA are random linear combinations of rows of A. Our main idea will be to project the rows of A onto the m-dimensional row space of SA and then use SVD to find the best rank-k approximation within this smaller space.

What sorts of matrices S will make the rank-k approximation in the projected space work well? We can choose S to be an $\Omega(k/\varepsilon) \times n$ matrix of i.i.d. normal random variables. We can also choose S to be an $\Omega(k/\varepsilon) \times n$ fast Johnson–Lindenstrauss matrix [2]. We'll focus on the analysis when S is a $\operatorname{poly}(k/\varepsilon) \times n$ CountSketch matrix, as described in [1]. Although the number of rows is larger, this choice of S lets us compute SA in $O(\operatorname{nnz}(A))$ time.

2.3.3 Algorithm: existence of good solutions in row span of SA

The first thing to check is that there even exists a good rank-k approximation to A living in the row space of SA. Consider the regression problem of minimizing the quantity $\|A_kX - A\|_\mathrm{F}$ over matrices X. Think of this as a sort of a thought experiment — this is not the regression problem we actually care about, and we do not know what A_k is.

Since A_kX has rank at most k and A_k is defined to be the best rank-k approximation to A, the optimal solution is to set X to be the identity matrix I. Thus $\min_X \|A_kX - A\|_\mathrm{F} = \|A_k - A\|_\mathrm{F}$.

Now let's take S to be an affine embedding. In particular, S may be taken a CountSketch matrix with $\operatorname{poly}(k/\varepsilon)$ rows. Notice the dependence on k, the rank of A_k, rather than d, the number of columns of A_k. This is fine because if we let U_k be a $n \times k$ matrix with the same column space as A_k, then for every X there is some Y such that $A_kX = U_kY$ and vice versa. Thus we can replace A_k with U_k without generality, and S can be taken as a CountSketch matrix for the smaller matrix U_k. This explains the dependence on k.

Since S is an affine embedding, we have that for any X that

$$\|SA_kX - SA\|_\mathrm{F} = (1 \pm \varepsilon) \|A_kX - A\|_\mathrm{F}. \tag{3}$$

We can minimize the left-hand side quantity as $\operatorname{argmin}_X \|SA_kX - SA\|_\mathrm{F} = (SA_k)^+SA$. Using this with equation (3) gives

$$\begin{aligned}(1 \pm \varepsilon) \left\|A_k(SA_k)^+SA - A\right\|_\mathrm{F} &= \left\|SA_k(SA_k)^+SA - SA\right\|_\mathrm{F} \\ &\le \|SA_kI - SA\|_\mathrm{F} = (1 \pm \varepsilon) \|A_kI - A\|_\mathrm{F} = (1 \pm \varepsilon) \|A_k - A\|_\mathrm{F}.\end{aligned}$$

Rearranging this, we have that

$$\left\|A_k(SA_k)^+SA - A\right\|_\mathrm{F} \le \frac{1+\varepsilon}{1-\varepsilon} \|A_k - A\|_\mathrm{F} \approx (1 + 2\varepsilon) \|A_k - A\|_\mathrm{F}, \tag{4}$$

and hence $A_k(SA_k)^+SA$ approximates A almost as well as how the best rank-k matrix, A_k, approximates A. The punchline: $A_k(SA_k)^+SA$ is a rank-k matrix with rows in the row space of SA! Therefore, good rank-k approximations to A do live within the row space of SA.

2.3.4 Algorithm: a first attempt at solving within row space of SA

We're not done yet. Although we found that $A_k(SA_k)^+SA$ is a good rank-k approximation A with rows in the row space of SA, we cannot compute it quickly. However, we now know

$$\min_{\text{rank-}k\ X} \|XSA - A\|_{\mathrm{F}} \le \left\|A_k(SA_k)^+SA - A\right\|_{\mathrm{F}} \le (1+\varepsilon)\,\|A_k - A\|_{\mathrm{F}}\,.$$

where XSA with rank-k X captures all rank-k matrices with rows in the row space of SA, and where the second inequality comes from equation (4) after adjusting ε. Thus, if we can minimize the left-hand side quantity, we'll accomplish our goal.

By the normal equations and the Pythagorean theorem, we get

$$\|XSA - A\|_{\mathrm{F}}^2 = \left\|A(SA)^+SA - A\right\|_{\mathrm{F}}^2 + \left\|XSA - A(SA)^+SA\right\|_{\mathrm{F}}^2$$

Intuitively, right-multiplying A by $(SA)^+SA$ projects the rows of A onto the row space of SA. The term $\|A(SA)^+SA - A\|_{\mathrm{F}}^2$ has no dependence on X and is the cost we're forced to pay by working in the row space of SA. Now we have

$$\min_{\text{rank-}k\ X} \|XSA - A\|_{\mathrm{F}}^2 = \left\|A(SA)^+SA - A\right\|_{\mathrm{F}}^2 + \min_{\text{rank-}k\ X} \left\|XSA - A(SA)^+SA\right\|_{\mathrm{F}}^2,$$

and we just seek to minimize $\|XSA - A(SA)^+SA\|_{\mathrm{F}}^2$.

Write $SA = U\Sigma V^\top$ in SVD form. This we can do in $d\,\mathrm{poly}(k/\varepsilon)$ time. Also note that U and Σ are small, square matrices of dimension $\mathrm{poly}(k/\varepsilon) \times \mathrm{poly}(k/\varepsilon)$. We set this into our minimization problem to get

$$\begin{aligned}\min_{\text{rank-}k\ X} \left\|XSA - A(SA)^+SA\right\|_{\mathrm{F}}^2 &= \min_{\text{rank-}k\ X} \left\|XU\Sigma V^\top - A(SA)^+U\Sigma V^\top\right\|_{\mathrm{F}}^2 \\ &= \min_{\text{rank-}k\ X} \left\|\left(XU\Sigma - A(SA)^+U\Sigma\right)V^\top\right\|_{\mathrm{F}}^2.\end{aligned}$$

Since $V^\top$ has orthonormal rows, we can remove $V^\top$ on the right-hand side without affecting the norm, and hence

$$\min_{\text{rank-}k\ X} \left\|XSA - A(SA)^+SA\right\|_{\mathrm{F}}^2 = \min_{\text{rank-}k\ X} \left\|XU\Sigma - A(SA)^+U\Sigma\right\|_{\mathrm{F}}^2.$$

Dropping $V^\top$ corresponds to switching to working within a coordinate representation of the row space of SA. This is nice because it decreases the number of columns we're concerned about.

Next, consider that if A has high rank, $U\Sigma$ will be invertible. (If A has low rank, then we can forget about S and operate within A's row space.) Therefore, minimizing over $XU\Sigma$ for rank-k X is the same as minimizing over Y for rank-k X, and

$$\min_{\text{rank-}k\ X} \left\|XSA - A(SA)^+SA\right\|_{\mathrm{F}}^2 = \min_{\text{rank-}k\ Y} \left\|Y - A(SA)^+U\Sigma\right\|_{\mathrm{F}}^2.$$

To solve the minimization problem on the right-hand side, we can compute the truncated SVD of $A(SA)^+U\Sigma$.

Still, we haven't achieved the running time we want. We can compute the SVD of SA quickly and then subsequently find $(SA)^+$ and $U\Sigma$. However, multiplying $(SA)^+U\Sigma$ on the left by A is too costly to afford.

2.3.5 Algorithm: getting faster by sketching the sketch

Roughly speaking, the bottleneck we're hitting here is that projecting the rows of A onto the row space of SA is costly. We have issues with the term $A(SA)^+U\Sigma$, and before we dropped the $V^\top$ factor, this term was $A(SA)^+U\Sigma V^\top = A(SA)^+SA$, which is the projection of A onto the row space of SA. The rows of $A(SA)^+U\Sigma$ correspond to a coordinate representation of the projection of the rows of A. If we knew $A(SA)^+U\Sigma$, we could compute its SVD and be done.

What if we could instead quickly approximately project the rows of A? In fact, we already know how to do that. Projection is just least-squares regression, which we can do approximately by sketching. This inspires us to sketch again to reduce the dimensions of the problem.

In the previous section, we wanted to solve

$$\min_{\text{rank-}k\ X} \|XSA - A\|_{\mathrm{F}}^2 .$$

To avoid our previous issues, we use another affine embedding. Let R be an affine embedding matrix, say a transposed CountSketch matrix with $\operatorname{poly}(k/\varepsilon)$ columns, so that

$$\|XSAR - AR\|_{\mathrm{F}}^2 = (1 \pm \varepsilon)\,\|XSA - A\|_{\mathrm{F}}^2$$

for all X. (We remark that R needs to be a larger CountSketch matrix than S, that is, the number of columns of R is greater than the number of rows of S.)

Note that computing AR and SAR takes only $\operatorname{nnz}(A)$ time, so we can indeed set up this new regression problem of solving

$$\min_{\text{rank-}k\ X} \|XSAR - AR\|_{\mathrm{F}}^2 .$$

We saw previously that outputting $\operatorname{argmin}_{\text{rank-}k\ X} \|XSA - A\|_{\mathrm{F}}^2$ gives an approximately optimal rank-k approximation to A, and therefore now we have that outputting $\operatorname{argmin}_{\text{rank-}k\ X} \|XSAR - AR\|_{\mathrm{F}}^2$ will give an approximately optimal approximation as well. Similar to in the previous section, by the normal equations and the Pythogrean theorem,

$$\min_{\text{rank-}k\ X} \|XSAR - AR\|_{\mathrm{F}}^2 = \left\|AR(SAR)^+SAR - AR\right\|_{\mathrm{F}}^2 + \min_{\text{rank-}k\ X} \left\|XSAR - AR(SAR)^+SAR\right\|_{\mathrm{F}}^2 .$$

Now we just want to solve $\min_{\text{rank-}k\ X} \left\|XSAR - AR(SAR)^+(SAR)\right\|_{\mathrm{F}}^2$.

Let's relax this problem by instead solving $\min_{\text{rank-}k\ X} \|Y - (AR)(SAR)^+SAR\|_{\mathrm{F}}^2$ exactly through truncated SVD. This is the point at which we were having issues in last section, but this time, we can compute the SVD quickly. The dimensions are now small enough that computing $(AR)(SAR)^+SAR$ and its SVD takes only $O((n+d)\operatorname{poly}(k/\varepsilon))$ time.

This gives us some rank-k Y. This Y actually must take on the form $XSAR$ for some X, so we have exactly what we want. To see this, note that if Y didn't take on the form $XSAR$, then some row of Y doesn't lie in the row space of SAR. Then define $Y' = Y(SAR)^+SAR$ by projecting Y's rows onto the row space of SAR. We have

$$\left\|Y' - AR(SAR)^+SAR\right\|_{\mathrm{F}}^2 = \left\|\left(Y - AR(SAR)^+SAR\right)(SAR)^+SAR\right\|_{\mathrm{F}}^2 . \qquad (5)$$

For any projection matrix P and matrix C, we have by the Pythagorean theorem that $\|CP\|_{\mathrm{F}}^2 \le \|CP\|_{\mathrm{F}}^2 + \|C(I-P)\|_{\mathrm{F}}^2 = \|C\|_{\mathrm{F}}^2$ with equality only if C already lies in P's space. In particular, taking $P = (SAR)^+SAR$ and $C = Y - AR(SAR)^+SAR$ and combining with equation (5) gives $\|Y' - AR(SAR)^+SAR\|_F^2 < \|Y - AR(SAR)^+SAR\|_F^2$, which contradicts the optimality of Y.

Thus we can recover $X = Y(SAR)^+$ and output XSA. Actually, XSA might be too large to output. Instead, we give the output $X \cdot (SA)$ in factored form by outputting the $n \times \mathrm{poly}(k/\varepsilon)$ matrix X and the $\mathrm{poly}(k/\varepsilon) \times d$ matrix SA.

2.3.6 Algorithm: summary

That was a lot of discussion and analysis, but in the end, the algorithm is easy to describe. Pick CountSketch matrices S and R. Compute SA, SAR, and AR. Compute Y to be the rank-k minimizer of $\|Y - AR(SAR)^+(SAR)\|_F^2$ by truncated SVD. Then output $(Y(SAR)^+)(SA)$ in factored form.

This all takes $O(\mathrm{nnz}(A) + (n+d)\,\mathrm{poly}(k/\varepsilon))$ time, which is much faster than applying truncated SVD to A directly.

References

[1] Kenneth L Clarkson and David P Woodruff. Low rank approximation and regression in input sparsity time. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 81–90. ACM, 2013.

[2] Tamás Sarlós. Improved approximation algorithms for large matrices via random projections. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*, pages 143–152, 2006.