CS 15-859: Algorithms for Big Data Fall 2022

Lecture 4.1 — 9/29/2022

Prof. David Woodruff *Scribe: Fan Pu Zeng*

1 Affine Embeddings - Recap

Last lecture, we introduced affine embeddings. The setup of the problem is that we have a $n \times d$ matrix A that is tall and thin, and a $n \times m$ matrix B that can have a very large number of columns. The goal is to solve

$$\min_X |AX - B|_F^2. \tag{1}$$

Our previous technique of using subspace embeddings can no longer be applied efficiently in this case, as our subspace embedding will be over the column span of both A and B. As a result, we introduced the notion of affine embeddings, which is a matrix S such that

$$\|S(AX - B)\|_{\mathrm{F}} = (1 \pm \varepsilon)\|AX - B\|_{\mathrm{F}} \tag{2}$$

holds with high probability. We then showed in the previous lecture that CountSketch matrices can work for affine embeddings.

For the rest of the lecture, we will see how to apply affine embeddings to help solve low-rank approximations.

2 Low-Rank Approximation

The low-rank Approximation is also referred to as Principal Component Analysis (PCA). It is one of the most popular ways of performing linear dimensionality reduction, with other more complicated ways being via neural networks.

2.1 Motivation

Suppose you have a $n \times d$ matrix A, where both dimensions are large. This could represent something like a customer-product matrix used in online recommender systems, where each cell $A_{i,j}$ denotes how many times customer i purchased item j. Then it is typically the case that A can be well-approximated by a low-rank matrix. For instance, using the previous example, there might only be a few dominant patterns that describes purchasing behavior in A, and the rest of it is just noise.

Therefore, if we can find such a low-rank approximation, we can achieve significant space savings, and can also help to make the data more interpretable.

We can thus formulate our problem as finding a rank-k matrix A_k such that

$$\arg\min_{\text{rank-}k \text{ matrices } B} \|A - B\|_{\mathrm{F}} \tag{3}$$

2.2 Exact Solution: Truncated Singular Value Decomposition

A natural solution that comes to mind is to use truncated singular value decomposition (SVD), which in fact gives us the best rank-k approximation for A as per Equation 3. To approximate the $n \times d$ matrix 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. The min comes from the fact that we can transpose A if necessary before performing SVD, so that the linear dimension matches the larger dimension.

However, the problem is that performing SVD is too expensive, since n and d are large.

2.3 Approximate Solution: Truncated Singular Value Decomposition

In order to get a better runtime, we relax the problem such that we are satisfied with an approximate solution instead of an exact solution. Our goal is now to find a rank k matrix A' such that

$$\left\|A' - A\right\|_F \leq (1+\varepsilon)\left\|A_k - A\right\|_F \tag{4}$$

with a constant failure probability (say 9/10).

We will now show an algorithm that can do this $O(\mathtt{nnz}(A) + (n+d)\,\mathrm{poly}(k/\varepsilon))$ time, due to Woodruff, Clarkson, and Sarlos [1] [2]. Recall that $\mathtt{nnz}(A)$ is the number of non-zero entries in A. When the input is dense, this is in approximately $O(nd)$ time, which is significantly faster than SVD. When it is sparse, it becomes even faster.

2.4 Intuition

The key idea is to compute SA, such that the number of rows of S is $O(k/\epsilon)$ which is small.

Think of the rows of A as points in $\mathbb{R}^d$. Think of SA as taking a small number of linear combinations of A, where each row of S is picking a random linear combination of all the rows of A, i.e points in $\mathbb{R}^d$:

$$k/\epsilon \ \Big[\ \overset{n}{S}\ \Big] \times n \overset{d}{\begin{bmatrix} \\ A \\ \\ \end{bmatrix}}$$

This allows us to think of SA as a $\frac{k}{\epsilon} \times d$ matrix where we performed this operation k/ϵ times. Then

each row of SA is a linear combination of the rows of A, but now in a subspace of only dimension k/ϵ. If we run SVD on these projected points, we can do it in only $O\left(n\left(\frac{k}{\epsilon}\right)^2\right)$ time!

To summarize, the main idea is to:

1. Project all the rows of A onto a lower dimensional space SA.
2. Find the best rank-k approximation to the points in SA via SVD.

2.5 Choice of Sketching Matrix

One might wonder which sketching matrices S would work for this paradigm. In fact, all the sketching matrices that we have covered so far in the course suffices:

1. S as the $\frac{k}{\epsilon} \times n$ matrix of i.i.d normal random variables. We can compute SA in time $O\left(\texttt{nnz}(A)\frac{k}{\epsilon}\right)$ since S is dense. However, we could do a lot better.
2. S as the $\tilde{O}\left(\frac{k}{\epsilon}\right) \times n$ Subsampled Randomized Hadamard Transform (also called the Fast Johnson Lindenstrauss) matrix. SA can be computed in $O(nd \log n)$ time.
3. S as the poly$\left(\frac{k}{\epsilon}\right) \times n$ CountSketch matrix. While the number of rows is larger, SA can be computed in just $\texttt{nnz}(A)$ time.

In this lecture, we will focus on using the CountSketch matrix for low-rank approximation.

2.6 Showing the Existence of a Good Solution in the Row Span of SA

For our proposed algorithm to actually work, we must first guarantee that some good rank-k approximation must live in the row span of SA in the first place. We will use the following thought experiment to show that such a solution indeed exists. We will not actually attempt to recover the solution.

Consider the following hypothetical regression problem:

$$\min_X \|A_k X - A\|_{\mathrm{F}} = \|A_k - A\|_{\mathrm{F}}. \tag{5}$$

Note again that this is only hypothetical: we don't know what A_k is, because if we did, then we are already done!

2.6.1 Best Solution to Hypothetical Regression Problem

The best solution of X to Equation 5 is just the identity matrix I_d, since A_k is already defined to be the best rank-k approximation to A, and we cannot increase the rank by multiplying something in hopes of further decreasing the Frobenius norm. This gives us

$$\min_X \|A_k X - A\|_{\mathrm{F}} = \|A_k - A\|_{\mathrm{F}}. \tag{6}$$

2.6.2 Sketching the Hypothetical Regression Problem with CountSketch

Now take the CountSketch matrix S to be our affine embedding. We claim that just poly $\left(\frac{k}{\epsilon}\right)$ rows suffices, instead of the usual poly $\left(\frac{d}{\epsilon}\right)$ rows. This is because since A_k only has rank k, then we could replace it with some other $n \times k$ rank k matrix U_k that has the same column span as A_k. This works because for every X there is some Y such that $A_k X = U_k Y$ 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.

Recall our affine embedding result:

$$\|SA_kX - SA\|_{\mathrm{F}} = (1 \pm \varepsilon) \|A_kX - A\|_{\mathrm{F}}. \tag{7}$$

To solve for the X that minimizes the left hand side quantity, recall that the normal equation gives us

$$\arg\min_X \|SA_kX - SA\|_{\mathrm{F}} = (SA_k)^- SA. \tag{8}$$

Plug in Equation 8 into the right side of 7 to obtain

$$\|A_k(SA_k)^- SA - A\|_F \leq (1 \pm \epsilon)\|A_k - A\|_F. \tag{9}$$

What is exciting about Equation 9 is that it tells us that $A_k(SA_k)^- SA$ is a $(1 \pm \epsilon)$ approximation for A_k. Furthermore, $A_k(SA_k)^- SA$ is rank-k and is in the row span of SA, which precisely answers our original question of whether a good rank-k approximation in the row span of SA exists in the affirmative!

To belabor the point, we do not know either A_k or $A_k(SA_k)^- SA$; we simply used them to prove that our desired rank-k approximation in SA exists.

2.7 Considering the Optimal Sketched Solution

Our conclusion from the previous section allows us to conclude that

$$\min_{\text{rank-}k X} \|XSA - A\|_{\mathrm{F}}^2 \leq \left\|A_k (SA_k)^- SA - A\right\|_{\mathrm{F}}^2 \leq (1 + \varepsilon) \|A_k - A\|_{\mathrm{F}}^2. \tag{10}$$

However, solving for the left hand side of Equation 10 is different from normal regression, because the rank of our solution for X is constrained. Suppose for a moment that we ignore this rank constraint and proceed to solve for X as per usual using our normal equations. Then by plugging in the normal equations into Equation 10, we obtain

$$\|XSA - A\|_{\mathrm{F}}^2 = \left\|XSA - A(SA)^- SA\right\|_{\mathrm{F}}^2 + \left\|A(SA)^- SA - A\right\|_{\mathrm{F}}^2 \tag{11}$$

by considering the Pythagorean theorem for the elements row-wise: X_iSA is a projection of X_i onto SA in a rank-k space, and $A_i(SA)^- SA$ is the projection of A_i onto SA.

Equation 11 tells us that the second term on the right hand side does not depend on X, so we can re-formulate Equation 10 as

$$\min_{\text{rank-}k\, X} \|XSA - A\|_{\mathrm{F}}^2 = \left\|A(SA)^- SA - A\right\|_{\mathrm{F}}^2 + \min_{\text{rank}-k\, X} \left\|XSA - A(SA)^- SA\right\|_{\mathrm{F}}^2. \tag{12}$$

2.8 A Simpler Minimization Problem

Equation 12 reduces our original minimization problem to just solving

$$\min_{\text{rank}-k\ X} \left\|XSA - A(SA)^{-}SA\right\|_{\text{F}}^2. \tag{13}$$

To solve this, let's begin by writing $SA = U\Sigma V^\top$ in SVD form. We can do this in $d \cdot \text{poly}\left(\frac{k}{\epsilon}\right)$ time. Then simplify our expression from Equation 12 to obtain

$$\begin{aligned}
\min_{\text{rank-}k\ X} \left\|XSA - A(SA)^{-}SA\right\|_{\text{F}}^2 &= \min_{\text{rank}-k\ X} \left\|XU\Sigma V^\top - A(SA)^{-}U\Sigma V^\top\right\|_{\text{F}}^2 \\
&\qquad \text{(substituting SVD representation of } S\text{)} \\
&= \min_{\text{rank}-k\ X} \left\|\left(XU\Sigma - A(SA)^{-}U\Sigma\right)V^\top\right\|_{\text{F}}^2 \qquad (14) \\
&= \min_{\text{rank}-k\ X} \left\|XU\Sigma - A(SA)^{-}U\Sigma\right\|_{\text{F}}^2 \\
&\qquad (V^\top \text{ is orthonormal and does not change norm}) \\
&= \min_{\text{rank-k}\ Y} \left\|Y - A(SA)^{-}U\Sigma\right\|_{F}^2. \\
&\qquad \text{(change of variables from } X \text{ to } Y\text{, since } U\Sigma \text{ has full rank)} \\
&\qquad (15)
\end{aligned}$$

This shows that it suffices to just compute the SVD of $A(SA)^{-}U\Sigma$ to solve our original problem!

2.9 Issues with Running Time

Unfortunately, we still haven't achieved the running time we want. Recall that our goal was to find the best rank-k approximation in time $O(\texttt{nnz}(A) + (n+d)\,\text{poly}(k/\varepsilon))$. $A(SA)^{-}U\Sigma$ is a $n \times \text{poly}\left(\frac{k}{\epsilon}\right)$ matrix, so one might claim that computing its SVD only takes $O\left(n \cdot \text{poly}\left(\frac{k}{\epsilon}\right)\right)$, which is within our time bounds. We could also compute SA, its pseudoinverse $(SA)^{-}$, and the multiplication $(SA)^{-}U\Sigma$ quickly.

However, the problem is that we cannot multiply on the left by A within the time bounds, because we have no guarantees about the sparsity of $(SA)^{-}U\Sigma$. It may as well be a very dense matrix. This means that computing $A(SA)^{-}U\Sigma$ takes time at least $O\left(\texttt{nnz}(A) \cdot \text{poly}\left(\frac{k}{\epsilon}\right)\right)$, which does not meet our bounds.

2.10 Summary of Current Progress

Now is a good time to stop and review the progress that we have made so far. To recap, our current algorithm works as follows:

1. Compute SA. This is in time $(\texttt{nnz}((A)))$
2. Project each row of A onto SA. We just showed that we couldn't do this in our desired time bound $O\left(\texttt{nnz}(A) + (n+d)\,\text{poly}\left(\frac{k}{\epsilon}\right)\right)$.

3. Find the best rank-k approximation of the projected points inside the rowspace of SA. This is in time $O\left(n \cdot \text{poly}\left(\frac{k}{\epsilon}\right)\right)$.

Therefore Step 2 is our bottleneck that we hope to improve. We will do this by approximating the projection of A onto SA.

2.11 Approximating the Projection

Projection is in fact just least-squares regression in disguise. This inspires us to sketch again to reduce the dimensions of the problem.

Recall that previously we wanted to solve for

$$\min_{\text{rank-}k\, X} \|XSA - A\|_{\text{F}}^2. \tag{16}$$

We can't sketch on the left anymore since X appears on the left, so we try to sketch on the right instead. Let R be an affine embedding matrix, say a transposed CountSketch matrix with $\text{poly} \cdot \left(\frac{k}{\epsilon}\right)$ columns. Then we sketch on the right in Equation 16 to change our target to be

$$\min_{\text{rank-}k\, X} \|X(SA)R - AR\|_{\text{F}}^2. \tag{17}$$

It would be wise to first verify that we can compute all our quantities in the desired bounds. Indeed, computing AR takes $\texttt{nnz}(A)$ time, and computing SAR can be done in $\texttt{nnz}(SA) \leq \texttt{nnz}(A)$ by recalling that SA cannot increase the number of non-zero entries in A.

Since R is an affine embedding, by Equation 17 we obtain

$$\|X(SA)R - AR\|_{\text{F}}^2. = (1 \pm \varepsilon)\|X(SA) - A\|_{\text{F}}^2. \tag{18}$$

We can rewrite our new target in Equation 17 to obtain

$$\min_{\text{rank-}k\, X} \|XSAR - AR\|_{\text{F}}^2 = \left\|AR(SAR)^{-}SAR - AR\right\|_{\text{F}}^2 + \min_{\text{rank-}k\, X} \left\|XSAR - AR(SAR)^{-}SAR\right\|_{\text{F}}^2 \tag{19}$$

by applying the Pythagorean theorem. Then similarly as before, note that only the second term depends on X, so again we have a smaller minimization problem:

$$\min_{\text{rank-k}\ X} \left\|XSAR - AR(SAR)^{-}(SAR)\right\|_{\text{F}}^2 = \min_{\text{rank-}k\, Y} \left\|Y - AR(SAR)^{-}SAR\right\|_{\text{F}}^2, \tag{20}$$

where we perform a change of variables to Y. We can do this because the optimal Y must live in the row space of SAR and therefore take the form $XSAR$. Suppose if it did not, then since $AR(SAR)^{-}SAR$ is in the row space of SAR, then we could simply remove the orthogonal components to SAR in Y to obtain an even better solution, contradicting its optimality.

2.12 Analyzing the Runtime

This time round, we have truly solved the problem. Previously, we failed because we could not project A onto the column space of SA in the desired time bounds. But this time round, SAR has dimensions

poly $\left(\frac{k}{\epsilon}\right) \times$ poly $\left(\frac{k}{\epsilon}\right)$, and AR is a $n \times \left(\frac{k}{\epsilon}\right)$ matrix that takes $\texttt{nnz}(A)$ time to compute, which means that $AR(SAR)^{-}(SAR)$ can be computed in time $n \cdot \text{poly}\left(\frac{k}{\epsilon}\right) \in O\left(\texttt{nnz}(A) + (n+d)\,\text{poly}\left(\frac{k}{\epsilon}\right)\right)$.

As a result, we can proceed to compute Y in Equation 20 by performing truncated SVD in time $O\left(n \cdot \text{poly}\left(\frac{k}{\epsilon}\right)\right)$.

As noted previously, our solution Y must look like $Y = XSAR$ for some X. We can thus solve for $X = Y(SAR)^{-}$.

Then by Equation 16, the final rank-k approximation that we wish to return is $XSA = Y(SAR)^{-}SA$. A final caveat here is that we must return $Y(SAR)^{-}SA$ in factored form, because if we multiply out the matrices it is a $n \times d$ matrix, which does not fit in our desired running time.

2.13 Returning the Output in Factored Form

We want to output $LR = Y(SAR)^{-}SA$ as L, R where L is $n \times k$ and R is $k \times d$.

To do so, recall that Y is $n \times \text{poly}\left(\frac{k}{\epsilon}\right)$ and so we can recover its SVD representation as $Y = U\Sigma V^{\top}$. Then let

$$L = U\Sigma, \tag{21}$$

$$R = V^{\top}(SAR)^{-}SA, \tag{22}$$

where L is $n \times k$ and R is $k \times d$. One can check that we can perform all the matrix multiplications within the time bounds.

2.14 Bounding the Failure Probability

The sketch by S and R may both fail to be an affine embedding with some constant probability, say $\delta = \frac{1}{100}$. Then we can simply union bound over S and R failing to show that the construction works with small constant failure probability.

2.15 Summary

That was a lot of discussion and analysis, but in the end, the low-rank approximation algorithm is very simple.

1. Compute SA.
2. Compute SAR and AR.
3. Compute $\arg\min_Y \|Y - AR(SAR)^{-}(SAR)\|_{\text{F}}^2$ by truncated SVD.
4. Output $Y(SAR)^{-}SA$ in factored form.

This takes $O\left(\texttt{nnz}(A) + (n+d)\,\text{poly}\left(\frac{k}{\epsilon}\right)\right)$ time overall, 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. *CoRR*, abs/1207.6365, 2012. URL: `http://arxiv.org/abs/1207.6365`, `arXiv:1207.6365`.

[2] Tamas Sarlos. Improved approximation algorithms for large matrices via random projections. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 143–152, 2006. `doi:10.1109/FOCS.2006.37`.

Special thanks to Tom Tseng, who scribed a similar set of notes in Fall 2017. Some explanations in this set of notes were adapted from his sharp exposition.