A Tutorial on Matrix Approximation by Row Sampling

Rasmus Kyng

June 11, 2018

Contents

1 Fast Linear Algebra Talk

1.1 Matrix Concentration

Consider $\mathbf{A}^\top \in \mathbb{R}^{m\times n}$. For simplicity, assume $m \geq n$ and $\operatorname{rank} \mathbf{A}^\top = n$, and write

$$\mathbf{A}^\top = \begin{pmatrix} \boldsymbol{a}_1^\top \\ \vdots \\ \boldsymbol{a}_m^\top \end{pmatrix}.$$

Definition 1.1 (Spectral ϵ-approximation)**.** *We will say that $\mathbf{B}^\top \in \mathbb{R}^{m\times n}$ is an ϵ-approximation of $\mathbf{A}^\top$ iff for all $\boldsymbol{x}$ in $\mathbb{R}^n$*

$$(1-\epsilon)\left\|\mathbf{A}^\top \boldsymbol{x}\right\|_2^2 \leq \left\|B^\top \boldsymbol{x}\right\|_2^2 \leq (1+\epsilon)\left\|\mathbf{A}^\top \boldsymbol{x}\right\|_2^2.$$

Definition 1.2 (Leverage Score)**.** *The* leverage score *of the ith row of $\mathbf{A}^\top$ is*

$$\tau_i(\mathbf{A}^\top) = \max_{\boldsymbol{x}\in\mathbb{R}^n} \frac{(\boldsymbol{a}_i^\top \boldsymbol{x})^2}{\|\mathbf{A}^\top \boldsymbol{x}\|_2^2}.$$

Theorem 1.3 (Leverage Score Sum)**.** $\sum_{i\in[m]} \tau_i(\mathbf{A}^\top) = \operatorname{rank}(\mathbf{A}^\top)$.

Below is a useful version of a Matrix Chernoff concentration result, this one due to Tropp [Tro12]. Important earlier versions were developed by Rudelson [Rud99] and Ahlswede and Winter [AW02].

Theorem 1.4 (Matrix Chernoff)**.** *Form $S \subset [m]$ by including each i independently with probability $p_i \geq \min(2\epsilon^{-2}\tau_i(\mathbf{A})\log(n/\delta), 1)$. Define the diagonal matrix $\mathbf{D} \in \mathbb{R}^{m\times m}$*

$$\mathbf{D}(i,i) = \begin{cases} \frac{1}{\sqrt{p_i}} & \text{if } i \in S \\ 0 & o.w. \end{cases}$$

Then with probability at least $1-\delta$, $\mathbf{D}\mathbf{A}^\top$ is an ϵ-approximation of $\mathbf{A}^\top$. Note that the expected number of rows in $\mathbf{D}\mathbf{A}^\top$ is at most $2\epsilon^{-2}n\log(n/\delta)$.

1.2 Algorithms for ϵ-Approximation of a matrix

Suppose we are given a matrix $\mathbf{A}^\top$ with $m \gg n$ and we'd like to find $\tilde{\mathbf{A}^\top} \in \mathbb{R}^{m\times n}$ with $m' \leq O(\epsilon^{-2}n\log(n/\delta))$ rows, s.t. $\tilde{\mathbf{A}}$ is an ϵ approximation of $\mathbf{A}$ (whp).

Notice

$$\tau_i(\mathbf{A}^\top) = \max_{\boldsymbol{x}\in\mathbb{R}^n} \frac{(\boldsymbol{a}_i^\top \boldsymbol{x})^2}{\|\mathbf{A}^\top \boldsymbol{x}\|_2^2} = \boldsymbol{a}_i^\top (\mathbf{A}\mathbf{A}^\top)^{-1}\boldsymbol{a}_i$$

So, we can compute leverage scores if we can compute inverses. But that's expensive! To compute the leverage scores naively would require us to

1. Compute $\mathbf{A}\mathbf{A}^\top$. Time $O(n^{\omega-2}m^2)$.
2. Compute $(\mathbf{A}\mathbf{A}^\top)^{-1}$. Time $\widetilde{O}(n^\omega)$.
3. Compute $\mathbf{C} = (\mathbf{A}\mathbf{A}^\top)^{-1}\mathbf{A}$. Time $O(n^{\omega-1}m)$.
4. Compute $\boldsymbol{a}_i^\top \boldsymbol{c}_i$ for all i. Time $O(nm)$.

Overall, we get $O(n^{\omega-2}m^2)$ time to compute all the leverage scores, and given the leverage scores, we can compute $\tilde{\mathbf{A}} = \mathbf{D}\mathbf{A}$ in time $O(\operatorname{nnz}(\mathbf{A})) \leq O(mn)$, which is a lower order term.

1.2.1 Uniform Sampling for Leverage Score Estimation

Still, our goal is to compute an approximation $\tilde{\mathbf{A}}$ of $\mathbf{A}$ with fewer rows. It turns out that in the above "algorithm sketch", we can replace the use of $\mathbf{A}\mathbf{A}^\top$ in Step 1 and onwards with a very crude approximation of the matrix and still get approximate leverage scores that are good enough for sampling.

The following definition is helpful:

Definition 1.5 (Generalized Leverage Scores)**.** *The generalized leverage score of row i of $AA^\top$ w.r.t. $\mathbf{B}^\top$ is*

$$\tau_i^{\mathbf{B}^\top}(\mathbf{A}^\top) = \max_{\boldsymbol{x}\in\mathbb{R}^n} \frac{(\boldsymbol{a}_i^\top \boldsymbol{x})^2}{\|\mathbf{B}^\top \boldsymbol{x}\|_2^2}.$$

The following theorem is the basis for a simple and clever algorithm for leverage score estimation. It is due to Cohen et al. [CLM+15].

Theorem 1.6 (Uniform Sampling for Leverage Score Approximation)**.** *Let $T \subseteq [m]$ denote a uniform random sample of d rows of $\mathbf{A}^\top$. Define the matrix $\mathbf{T} \in \mathbb{R}^{m\times m}$ to be the diagonal indicator for T, i.e. $\mathbf{T}(i,i) = 1$ if $i \in T$ and all other entries are zero.*

$$\tilde{\tau}_i \overset{\text{def}}{=} \begin{cases} \tau_i^{\mathbf{T}\mathbf{A}^\top}(\mathbf{A}) & \text{if } i \in T, \\ \frac{1}{1+\frac{1}{\tau_i^{\mathbf{T}\mathbf{A}^\top}(\mathbf{A})}} & \text{otherwise.} \end{cases}$$

Then, $\tilde{\tau}_i \geq \tau_i(\mathbf{A})$ for all i and

$$\mathbb{E}\left[\sum_{i=1}^n \tilde{\tau}_i\right] \leq \frac{nm}{d}.$$

Algorithm 1 Repeated Halving Version 1

input: $m \times n$ matrix $\mathbf{A}$, approximation parameter ϵ
output: spectral approximation $\tilde{\mathbf{A}}$ consisting of $O(\epsilon^{-2} n \log n)$ rescaled rows of $\mathbf{A}$

1: **procedure** Repeated Halving
2: Uniformly sample $\frac{m}{2}$ rows of $\mathbf{A}$ to form $\mathbf{A}'$
3: If $\mathbf{A}'$ has $> O(n \log n)$ rows, **recursively** compute a 1/2-spectral approximation $\tilde{\mathbf{A}}'$ of $\mathbf{A}'$
4: Compute generalized leverage scores of $\mathbf{A}$ w.r.t. $\tilde{\mathbf{A}}'$
5: Use these estimates to sample rows of $\mathbf{A}$ to form $\tilde{\mathbf{A}}$
6: **return** $\tilde{\mathbf{A}}$
7: **end procedure**

Remark 1.7. *Not addressing how approximation plays into not using $\mathbf{A}'$ but it's approximation when computing $\tau_i^{\mathbf{T}\mathbf{A}^\top}$. but it looks like using c factor overestimates will at most increase the sum by a factor c, because the $1/(1+1/x)$ transformation is monotone, and grows slower than x, so the effect over approximation is in fact decreased a bit.*

Let's sketch the time required to do this:

1. Compute $\tilde{\mathbf{A}}'$ by recursion. Time O(top level time), b/c $\log n$ levels but geometric decay in time at each level.

2. Compute $\tilde{\mathbf{A}}'\tilde{\mathbf{A}}'^{\top}$. Time $\widetilde{O}(n^{\omega})$.

3. Compute $(\tilde{\mathbf{A}}'\tilde{\mathbf{A}}'^{\top})^{-1}$. Time $\widetilde{O}(n^{\omega})$.

4. Compute $\mathbf{C} = (\tilde{\mathbf{A}}'\tilde{\mathbf{A}}'^{\top})^{-1}\mathbf{A}$. Time $O(n^{\omega-1}m)$.

5. Compute $\boldsymbol{a}_i^{\top}\boldsymbol{c}_i$ for all i. Time $O(nm)$.

Overall, we get $O(n^{\omega-1}m)$ time to compute all the leverage scores, and given the leverage scores, we can compute $\tilde{\mathbf{A}} = \mathbf{DA}$ in time $O(\text{nnz}(\mathbf{A})) \leq O(mn)$, which is a lower order term.

1.3 JL Speed-up

But, we can go even faster using, *the Johnson-Lindenstrauss transform* [JL84] and a clever trick from [SS11].

Theorem 1.8 (Johnson-Lindenstrauss Lemma)**.** *Suppose $\mathbf{G}^{\top} \in \mathbb{R}^{r\times d}$ is random matrix whose entries are $N(0,1)$ iid, and $r \geq 8\epsilon^{-2}\log 1/\delta_{JL}$. Then for any fixed vector $\boldsymbol{b} \in R^d$ with probability at least $1-\delta_{JL}$,*

$$(1-\epsilon)\|\boldsymbol{b}\|_2^2 \leq \frac{m}{d}\left\|\mathbf{G}^{\top}\boldsymbol{b}\right\|_2^2 \leq (1+\epsilon)\|\boldsymbol{b}\|_2^2$$

It turns out we can use the JL transform to speed up computation of leverage scores:

Spielman and Srivastava realized that the JL transform can be used to speed up computation of leverage scores:

$$\begin{aligned}
\boldsymbol{a}_i^{\top}(\tilde{\mathbf{A}}'\tilde{\mathbf{A}}'^{\top})^{-1}\boldsymbol{a}_i &= \boldsymbol{a}_i^{\top}(\tilde{\mathbf{A}}'\tilde{\mathbf{A}}'^{\top})^{-1}\tilde{\mathbf{A}}'\tilde{\mathbf{A}}'^{\top}(\tilde{\mathbf{A}}'\tilde{\mathbf{A}}'^{\top})^{-1}\boldsymbol{a}_i \\
&= \left\|\tilde{\mathbf{A}}'^{\top}(\tilde{\mathbf{A}}'\tilde{\mathbf{A}}'^{\top})^{-1}\boldsymbol{a}_i\right\|_2^2 \\
&\approx \frac{m}{n}\left\|\mathbf{G}^{\top}\tilde{\mathbf{A}}'^{\top}(\tilde{\mathbf{A}}'\tilde{\mathbf{A}}'^{\top})^{-1}\boldsymbol{a}_i\right\|_2^2
\end{aligned}$$

Now, we choose $\delta_{\text{JL}} = \delta/m$, and $\epsilon = 1/2$ for generating $\mathbf{G}$ with $r = O(\log(m/\delta))$ s.t. with probability δ, we get estimates of the leverage scores up to constant factors.

Let's plug this into our previous algorithm:

Algorithm 2 Repeated Halving Version 2

input: $m \times n$ matrix $\mathbf{A}$, approximation parameter ϵ
output: spectral approximation $\tilde{\mathbf{A}}$ consisting of $O(\epsilon^{-2}n\log n)$ rescaled rows of $\mathbf{A}$

1: **procedure** Repeated Halving
2: Uniformly sample $\frac{m}{2}$ rows of $\mathbf{A}$ to form $\mathbf{A}'$
3: If $\mathbf{A}'$ has $> O(n\log n)$ rows, **recursively** compute a 1/2-spectral approximation $\tilde{\mathbf{A}}'$ of $\mathbf{A}'$
4: Compute **approximate (JL-based)** generalized leverage scores of $\mathbf{A}$ w.r.t. $\tilde{\mathbf{A}}'$
5: Use these estimates to sample rows of $\mathbf{A}$ to form $\tilde{\mathbf{A}}$
6: **return** $\tilde{\mathbf{A}}$
7: **end procedure**

What's different? Let's again sketch the time required to do this:

1. Compute $\tilde{\mathbf{A}}'$ by recursion. Time $O(\text{top level time})$, b/c $\log n$ levels but geometric decay in time at each level.
2. Compute $\tilde{\mathbf{A}}'\tilde{\mathbf{A}}'^\top$. Time $\widetilde{O}(n^\omega)$.
3. Compute $(\tilde{\mathbf{A}}'\tilde{\mathbf{A}}'^\top)^{-1}$. Time $\widetilde{O}(n^\omega)$.
4. Compute $\mathbf{C} = \mathbf{G}^\top \tilde{\mathbf{A}}'^\top (\tilde{\mathbf{A}}'\tilde{\mathbf{A}}'^\top)^{-1}$. Time $O(r \operatorname{nnz}(\tilde{\mathbf{A}}')) \leq O(r \operatorname{nnz}(\mathbf{A}))$ for the $\mathbf{G}^\top \tilde{\mathbf{A}}'^\top$ product, plus $O(rn^2)$ time to mutiply the result by $(\tilde{\mathbf{A}}'\tilde{\mathbf{A}}'^\top)^{-1}$. The entire result is an $r \times n$ matrix.
5. Compute $\mathbf{C}\boldsymbol{a}_i$ for all i. Time $O(r \operatorname{nnz}(\mathbf{A}))$.

The dominating terms are $\widetilde{O}(\operatorname{nnz}(A) + n^\omega)$.

1.4 BSS-sparsification

The following theorem, due to Batson, Spielman, and Srivastava [BSST13] show that in fact $O(n/\epsilon^2)$ rows is enough to produce an ϵ-sparsifier.

Theorem 1.9 (BSS Sparsifiers)**.** *Given* $\mathbf{A}$ *and* $\gamma \in (0,1)$*, the algorithm* BSSSPARSIFY *selects a set of* n/γ^2 *rescaled rows of* $\mathbf{A}^\top$ *to form* $\tilde{\mathbf{A}}^\top$ *which is a* $2\gamma + \gamma^2$*-approximation of* $\mathbf{A}^\top$*. The algorithm is deterministic and can be implemented to run in* $O(\gamma^{-2}n^3m)$ *time.*

Algorithm 3 BSS SPARSIFICATION

input: $m \times n$ matrix $\mathbf{A}$, approximation parameter γ
output: spectral approximation $\tilde{\mathbf{A}}$ consisting of $O(\epsilon^{-2}n\log n)$ rescaled rows of $\mathbf{A}$

procedure BSSSPARSIFY
  Compute $\mathbf{H} = (\mathbf{A}\mathbf{A}^\top)^{-1/2}$
  For each $i \in [m]$, let $\boldsymbol{v}_i = \mathbf{H}\boldsymbol{a}_i$.
  For convienience, let $d = 1/\gamma^2$
  Let $\epsilon_{\mathrm{U}} = \frac{\sqrt{d}-1}{d+\sqrt{d}}$, $\epsilon_{\mathrm{L}} = \frac{1}{\sqrt{d}}$, $\delta_{\mathrm{U}} = \frac{\sqrt{d}+1}{\sqrt{d}-1}$, and $\delta_{\mathrm{L}} = 1$
  Let $u_0 \leftarrow n/\epsilon_{\mathrm{U}}$, $l_0 \leftarrow -n/\epsilon_{\mathrm{L}}$, $\mathbf{V}_0 = \mathbf{0}$, and $S \leftarrow \emptyset$.
  for $t = 1$ to dn **do**
    $u_t \leftarrow u_{t-1} + \delta_{\mathrm{U}}$, $l_t \leftarrow l_{t-1} + \delta_{\mathrm{L}}$
    $\mathbf{M}_t \leftarrow (u_{t-1}\mathbf{I} - \mathbf{V}_{t-1})^{-1}$, $\mathbf{M}_{t,+} \leftarrow (u_t\mathbf{I} - \mathbf{V}_{t-1})^{-1}$,
    $\mathbf{N}_t \leftarrow (\mathbf{V}_{t-1} - l_{t-1}\mathbf{I})^{-1}$, $\mathbf{N}_{t,+} \leftarrow (\mathbf{V}_{t-1} - l_t\mathbf{I})^{-1}$
    Find i s.t. $w_i \leftarrow \frac{\boldsymbol{v}_i^\top \mathbf{M}_{t,+}^2 \boldsymbol{v}_i}{\operatorname{Tr}(\mathbf{M}_t) - \operatorname{Tr}(\mathbf{M}_{t,+})} + \boldsymbol{v}_i^\top \mathbf{M}_{t,+} \boldsymbol{v}_i \leq \frac{\boldsymbol{v}_i^\top \mathbf{N}_{t,+}^2 \boldsymbol{v}_i}{\operatorname{Tr}(\mathbf{N}_{t,+}) - \operatorname{Tr}(\mathbf{N}_t)} - \boldsymbol{v}_i^\top \mathbf{N}_{t,+} \boldsymbol{v}_i$
    Let $\mathbf{V}_t \leftarrow \mathbf{V}_t + \frac{1}{w_i}\boldsymbol{v}_i\boldsymbol{v}_i^\top$, and $S \leftarrow S \cup \{i\}$.
  end for
  return $\tilde{\mathbf{A}} \leftarrow \sqrt{\frac{1-1/\sqrt{d}}{dn}} \begin{pmatrix} w_i^{-1/2}\boldsymbol{a}_{i_1} \\ \vdots \\ w_i^{-1/2}\boldsymbol{a}_{i_T} \end{pmatrix}$ where $S = \{i_1, \ldots, i_T\}$.
end procedure

References

[AW02] Rudolf Ahlswede and Andreas Winter. Strong converse for identification via quantum channels. *IEEE Transactions on Information Theory*, 48(3):569–579, 2002.

[BSST13] Joshua Batson, Daniel A Spielman, Nikhil Srivastava, and Shang-Hua Teng. Spectral sparsification of graphs: theory and algorithms. *Communications of the ACM*, 56(8):87–94, 2013.

[CLM+15] Michael B. Cohen, Yin Tat Lee, Cameron Musco, Christopher Musco, Richard Peng, and Aaron Sidford. Uniform sampling for matrix approximation. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, ITCS '15, pages 181–190, New York, NY, USA, 2015. ACM.

[JL84] William Johnson and Joram Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. In *Conference in modern analysis and probability (New Haven, Conn., 1982)*, volume 26 of *Contemporary Mathematics*, pages 189–206. American Mathematical Society, 1984.

[Rud99] Mark Rudelson. Random vectors in the isotropic position. *Journal of Functional Analysis*, 164(1):60–72, 1999.

[SS11] Daniel A Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. *SIAM Journal on Computing*, 40(6):1913–1926, 2011.

[Tro12] Joel A Tropp. User-friendly tail bounds for sums of random matrices. *Foundations of computational mathematics*, 12(4):389–434, 2012.