

---

# 16 Matrix Sketching

---

The singular value decomposition (SVD) can be interpreted as finding the most dominant *directions* in an  $(n \times d)$  matrix  $A$  (or  $n$  points in  $\mathbb{R}^d$ ). Typically  $n > d$ . It is typically easy to call built in version of the SVD in many programming languages

$$[U, S, V] = \text{svd}(A)$$

where  $U = [u_1, \dots, u_n]$ ,  $S = \text{diag}(\sigma_1, \dots, \sigma_d)$ , and  $V = [v_1, \dots, v_d]$ . Then  $A = USV^T$  and in particular  $A = \sum_{j=1}^d \sigma_j u_j v_j^T$ . To approximate  $A$  we just use the first  $k$  components to find  $A_k = \sum_{j=1}^k \sigma_j u_j v_j^T = U_k S_k V_k^T$  where  $U_k = [u_1, \dots, u_k]$ ,  $S_k = \text{diag}(\sigma_1, \dots, \sigma_k)$ , and  $V_k = [v_1, \dots, v_k]^T$ . Then the vectors  $v_j$  (starting with smaller indexes) provide the best subspace representation of  $A$ .

But, although SVD has been *heavily* optimized on data sets that fit in memory (via LAPACK, found in Matlab, and just about every other language), it can sometimes be improved. The traditional SVD takes  $O(\min\{nd^2, d^2n\})$  time to compute, which can be prohibitive for large  $n$  and/or  $d$ . Here we highlight two of these ways:

- to provide better interpretability of each  $v_j$ .
- to be more efficient on enormous scale, in a stream, or in distributed settings.

## 16.1 Row Sampling

The goal is to approximate  $A$  up to the accuracy of  $A_k$ . But in  $A_k$  the directions  $v_i$  are *linear combinations of features*.

- What is a linear combination of genes?
- What is a linear combination of typical grocery purchases?

Instead our goal is to choose  $V$  so that the columns of  $V$  are also columns of  $A$ .

For each row of  $a_i \in A$ , set  $w_i = \|a_i\|^2$ . Then select  $t = (k/\varepsilon)^2 \cdot \log(1/\delta)$  rows of  $A$ , each proportional to  $w_i$ . Let  $R$  be the “stacking” of these rows.

These  $t$  rows will jointly act in place of  $V_k^T$ . However since  $V$  was orthogonal, then the columns  $v_i, v_j \in V_k$  were orthogonal. This is not the case for  $R$ , we need to orthogonalize  $R$ . Let  $\Pi_R = R^T(RR^T)^{-1}R$  be the projection matrix for  $R$ , so that  $A_R = A\Pi_R$  describes the *projection* of  $A$  onto the subspace of the directions spanned by  $R$ . Now

$$\|A - A\Pi_R\|_F \leq \|A - A_k\|_F + \varepsilon\|A\|_F$$

with probability at least  $1 - \delta$  [4].

- *Why did we not just choose the  $t$  rows of  $A$  with the largest  $w_j$  values?*  
Some may point along the same “direction” and would be repetitive. This should remind you of the choice to run  $k$ -means++ versus the Gonzalez algorithm for greedy point-assignment clustering.
- *Why did we not factor out the directions we already picked?*  
We could, but this allows us to run this in a streaming setting. (See next approach)
- *But  $A\Pi_R$  could be rank  $t$ , can we get it rank  $k \ll t$ ?*  
Yes, you can take its best rank  $k$  approximation  $[\Pi_R A]_k$  and about the same bounds hold, you may need to increase  $t$  slightly.

- Can we get a better error bound?

Yes. First take SVD  $[U, S, V] = \text{svd}(A)$  and let  $U_k$  be the top  $k$  left singular vectors. Let  $U_k(i)$  be the  $i$ th row of  $U_k$ . Now the leverage score of data point  $a_i$  is  $\ell_i = \|U_k(i)\|^2$ . Using the leverage scores as weights  $w_i = \ell_i$  allows one to achieve stronger bounds [2]

$$\|A - A\Pi_R\|_F \leq (1 + \varepsilon)\|A - A_k\|_F.$$

But this requires us to first take the SVD (or other time-consuming procedures), so its is harder to do in a stream; although some new approaches are addressing this [3]. In many cases, this approaches do not seem to provide tangible benefits over the faster  $\|a_i\|^2$ -weighted sampling.

There exist more complicated and slower approaches which achieve the same bound with smaller  $k$  [1].

- Can we also sample columns this way?

Yes. All tricks can be run on  $A^T$  the same way (in fact most of the literature talks about sampling columns instead of rows). And, both approaches can be combined. This is known as the CUR-decomposition of  $A$ .

A significant downside of these row sampling approaches is that the  $(1/\varepsilon^2)$  coefficient can be quite large for a small error tolerance. If  $\varepsilon = 0.01$ , meaning 1% error, then this part of the coefficient alone is 10,000. In practice, the results may be better, but for guarantees, this may only work on very enormous matrices.

## 16.2 Frequent Directions

Another efficient solution is provided by using a Misra-Gries trick. It is called Frequent Directions [8, 6].

We will consider a matrix  $A$  one row (one point  $a_i$ ) at a time. We will maintain a matrix  $B$  that is  $\ell \times d$ , that is it only has  $\ell$  rows (directions). We maintain that one row is always empty (has all 0s) at the end of each round (this will always be the last row  $B_\ell$ ).

We initialize with the first  $2\ell - 1$  rows  $a_i$  of  $A$  as  $B$ , again with the last row  $B_\ell$  left as all zeros. Then on each new row, we put  $a_i$  in the empty row of  $B$ . We set  $[U, S, V] = \text{svd}(B)$ . Now examine  $S = \text{diag}(\sigma_1, \dots, \sigma_\ell)$ , which is a length  $\ell$  diagonal matrix. If  $\sigma_\ell = 0$  (then  $a_i$  is in the subspace of  $B$ ), do nothing. Otherwise subtract  $\delta = \sigma_\ell^2$  from each (squared) entry in  $S$ , that is  $\sigma'_j = \sqrt{\max\{0, \sigma_j^2 - \delta\}}$  and in general  $S' = \text{diag}(\sqrt{\sigma_1^2 - \delta}, \sqrt{\sigma_2^2 - \delta}, \dots, \sqrt{\sigma_{\ell-1}^2 - \delta}, 0, \dots, 0)$ .

Now we set  $B = S'V^T$ . Notice, that since  $S'$  only has non-zero elements in the first  $\ell - 1$  entries on the diagonal, then  $B$  is at most rank  $\ell - 1$  and we can then treat  $V$  and  $B$  as if the  $\ell$ th row does not exist.

---

### Algorithm 16.2.1 Frequent Directions

---

Set  $B$  all zeros ( $2\ell \times d$ ) matrix.

**for** rows (i.e. points)  $a_i \in A$  **do**

Insert  $a_i$  into a zero-valued row of  $B$

**if** ( $B$  has no zero-valued rows) **then**

$[U, S, V] = \text{svd}(B)$

Set  $\delta_i = \sigma_\ell^2$

# the  $\ell$ th entry of  $S$

Set  $S' = \text{diag}(\sqrt{\sigma_1^2 - \delta}, \sqrt{\sigma_2^2 - \delta}, \dots, \sqrt{\sigma_{\ell-1}^2 - \delta}, 0, \dots, 0)$ .

Set  $B = S'V^T$

# the last rows of  $B$  will again be all zeros

**return**  $B$

---

The result of Algorithm 16.2.1 is a matrix  $B$  such that for any (direction) unit vector  $x \in \mathbb{R}^d$

$$0 \leq \|Ax\|^2 - \|Bx\|^2 \leq \|A - A_k\|_F^2 / (\ell - k)$$

and [7, 6]

$$\|A - A\Pi_{B_k}\|_F^2 \leq \frac{\ell}{\ell - k} \|A - A_k\|_F^2,$$

for any  $k < \ell$ , including when  $k = 0$ . So setting  $\ell = 1/\varepsilon$ , then in any direction in  $\mathbb{R}^d$ , the squared mass in that direction is preserved up to  $\varepsilon\|A\|_F^2$  (that is,  $\varepsilon$  times the total squared mass) using the first bound. And in the second bound if we set  $\ell = \lceil k/\varepsilon + k \rceil$  then we have  $\|A - A\Pi_{B_k}\|_F^2 \leq (1 + \varepsilon)\|A - A_k\|_F^2$ . Recall that  $\|A\|_F = \sqrt{\sum_{a_i \in A} \|a_i\|^2}$ .

- *Why does this work?*

Just like with Misra-Greis [9], when some mass is deleted from one counter it is deleted from all  $\ell$  counters, and none can be negative. So here when one direction has its (squared) mass decreased, at least  $\ell$  directions (with non-zero squared mass) are decreased by the same amount. So no direction can have more than  $1/\ell$  fraction of the total squared mass  $\|A\|_F^2$  decreased from it.

Finally, since squared mass can be summed independently along any set of **orthogonal** directions, we can subtract each of them without affecting others; see [6] for more details, spelled out in a few lines of linear algebra.

- *Why do we use the svd?*

The SVD defines the true axis of the ellipse associated with the norm of  $B$  at each step. If we shrink along an basis (or even a set of non-orthogonal vectors) we will warp the ball, and we will not be able to ensure that each direction of  $B$  shrinks in squared norm by at most  $\delta_i$ .

- *Did we **need** to use the svd? (its expensive, right)?*

The cost is amortized. We only call the svd once every  $\ell$  steps, so at most  $O(n/\ell)$  times. Since each call takes  $O(d\ell^2)$  time, the total cost is  $O(nd\ell)$ , or only  $\ell$  times as long as reading the matrix.

It is also possible to call approximate versions of the SVD [5]. This allows versions which have runtime depending on the number of non-zeros in the input matrix. This makes a big difference for very sparse word count or recommendation system matrices.

- *What happened to  $U$  in the svd output?*

The matrix  $U$  just related the main directions to each of the  $n$  points (rows) in  $A$ . But we don't want to keep around the space for this. In this application, we only care about the directions or subspace that best represents the points; e.g. PCA only cares about the right singular vectors.



---

# Bibliography

---

- [1] Joshua Batson, Daniel A. Spielman, and Nikhil Srivastava. Twice-ramanujan sparsifiers. *SIAM Review*, (315–334), 56. arXiv:0808.0163.
- [2] Christos Boutsidis, Michael W Mahoney, and Petros Drineas. An improved approximation algorithm for the column subset selection problem. In *Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 968–977. Society for Industrial and Applied Mathematics, 2009.
- [3] Michael B. Cohen, Cameron Musco, and Christopher Musco. Ridge leverage scores for low-rank approximation. Technical report, arXiv:1511.07263, 2015.
- [4] Alan Frieze, Ravi Kannan, and Santosh Vempala. Fast monte-carlo algorithms for finding low-rank approximations. In *Foundations of Computer Science, 1998. Proceedings. 39th Annual Symposium on*. IEEE, 1998.
- [5] Mina Ghashami, Edo Liberty, and Jeff M. Phillips. Efficient frequent directions algorithm for sparse matrices. Technical report, arXiv:1602.00412, 2016.
- [6] Mina Ghashami, Edo Liberty, Jeff M. Phillips, and David P. Woodruff. Frequent directions: Simple and deterministic matrix sketching. Technical report, arXiv:1501.01711, 2015.
- [7] Mina Ghashami and Jeff M. Phillips. Relative errors for deterministic low-rank matrix approximations. In *ACM-SIAM 25th Symposium on Discrete Algorithms*, 2014.
- [8] Edo Liberty. Simple and deterministic matrix sketching. In *Proceedings 19th ACM Conference on Knowledge Discovery and Data Mining*, 2013.
- [9] J. Misra and D. Gries. Finding repeated elements. *Sc. Comp. Prog.*, 2:143–152, 1982.