Geometry Aware Mappings for High Dimensional Sparse Factors Supplementary Material

Avradeep Bhowmik
University of Texas at Austin, Austin, TX

Nathan Liu
Google, Mountain View, CA

Erheng Zhong
Yahoo! Labs, Sunnyvale, CA

Badri Narayan Bhaskar
Yahoo! Labs, Sunnyvale, CA

Suju Rajan
Yahoo! Labs, Sunnyvale, CA

1 PROOFS

Proof of Lemma 1

Proof. Using standard Euclidean distance for projection of any factor $\mathbf{z}$ on to the tessellating vectors Γ, recall by definition we have

$$\begin{aligned}\arg\min_{\mathbf{a}\in\Gamma} d(\mathbf{a},\mathbf{z}) &= \arg\min_{\mathbf{a}\in\Gamma} 1 - \frac{\mathbf{a}^\top\mathbf{z}}{\|\mathbf{a}\|_2\|\mathbf{z}\|_2} \\ &= \arg\max_{\mathbf{a}\in\Gamma} \mathbf{a}^\top\mathbf{z} \qquad (\because \|\mathbf{a}\|_2 = 1) \\ &= \arg\min_{\tilde{\mathbf{a}}\in\mathcal{A}} 1 - \frac{\tilde{\mathbf{a}}^\top\mathbf{z}}{\|\tilde{\mathbf{a}}\|_2\|\mathbf{z}\|_2}\end{aligned}$$

Suppose $\mathbf{a}$ has t non-zero elements with the corresponding indices $I_\mathbf{a} \subset \{1, 2, \cdots k\}$ with $|I_\mathbf{a}| = t$. Clearly, the corresponding unnormalised $\tilde{\mathbf{a}}$ would also have to have had t non-zero elements, each of them ± 1, and therefore $\|\tilde{\mathbf{a}}\|_2 = \sqrt{t}$.

Therefore, we have

$$\mathbf{a}^\top\mathbf{z} = \frac{\sum_{j\in I_\mathbf{a}} sign(a^j) z^j}{\sqrt{t}}$$

Clearly, for any fixed t, the maximiser of the numerator is an a such that each a^j has the same sign as z^j and a^j is supported (non-zero) at the top t elements (by absolute value) of $\mathbf{z}$. Then, we have

$$\begin{aligned}\max_{\mathbf{a}\in\Gamma} \mathbf{a}^\top\mathbf{z} &= \max_t \max_{\mathbf{a}:|I_\mathbf{a}|=t} \frac{\sum_{j\in I_\mathbf{a}} sign(a^j) z^j}{\sqrt{t}} \\ &= \max_t z_s^t\end{aligned}$$

where $\mathbf{z}_s$ is as defined in Algorithm (2) of the main manuscript. This completes the proof of correctness of the projection operator. □

Proof of Lemma 2

The steps to compute the approximately closest tessellation vector over Γ_D are given in Algorithm (A). The proof given below is not the only one possible, other (possibly tighter) bounds can be obtained by using different proof techniques and different algebraic manipulations of the quantities involved.

Proof. Note that for any scalar s with $|s| \leq 1$, there exists a scalar h with $|h| \leq D$ such that $|s - \frac{h}{D}| < \frac{1}{D}$. Therefore, since each $\tilde{\mathbf{a}}^j$ is a multiple of $\pm\frac{1}{D}$, for any $\mathbf{z} \in \mathbb{S}^k$, there exists $\tilde{\mathbf{a}} \in \mathcal{A}_D = \mathcal{B}_D^k \setminus \{0\}^k$ such that $\|\mathbf{z} - \tilde{\mathbf{a}}\| = \sqrt{\sum_i (\mathbf{z}^i - \tilde{\mathbf{a}}^i)^2} \leq \frac{\sqrt{k}}{D}$.

For any vector $\mathbf{x} \in \mathbb{S}^k$, denote its projection on to $\mathcal{A}_D$ as

$$\mathcal{A}_D(\mathbf{x}) = \arg\min_{\tilde{\mathbf{a}}\in\mathcal{A}_D} \|\tilde{\mathbf{a}} - \mathbf{x}\|_2$$

Clearly, by the preceding discussion,

$$\|\mathbf{x} - \mathcal{A}_D(\mathbf{x})\|_2 \leq \frac{\sqrt{k}}{D} \qquad (1)$$

Moreover, $\mathcal{A}_D(\mathbf{x})$ can be obtained by following steps (2) to (13) in TessVector-$D(\mathbf{x}, D)$ as detailed in Algorithm (A).

Suppose for a factor $\mathbf{z}$, the optimal projection on Γ_D is

$$\begin{aligned}\mathbf{a}_\mathbf{z}^* &= \arg\min\nolimits_{\mathbf{a}\in\Gamma_D} d(\mathbf{a},\mathbf{z}) = \arg\min\nolimits_{\mathbf{a}\in\Gamma_D} \tfrac{1}{2}\|\mathbf{a} - \mathbf{z}\|_2^2 \\ &= \arg\min\nolimits_{\mathbf{a}\in\Gamma_D} \|\mathbf{a} - \mathbf{z}\|_2\end{aligned}$$

Suppose the projection obtained from TessVector-$D(\mathbf{z}, D)$ is $\mathbf{a}_\mathbf{z}$. Then we have,

$$\|\mathbf{z} - \mathbf{a}_\mathbf{z}^*\|_2 \leq \|\mathbf{z} - \mathbf{a}_\mathbf{z}\| \qquad (2)$$

Now, we have

$$\begin{aligned}\|\mathbf{a}_\mathbf{z} - \mathbf{a}_\mathbf{z}^*\| &\leq \|\mathbf{a}_\mathbf{z} - \mathbf{z}\| + \|\mathbf{z} - \mathbf{a}_\mathbf{z}^*\| && [\Delta \text{ ineq}] \quad (3) \\ &\leq 2\|\mathbf{a}_\mathbf{z} - \mathbf{z}\| && [\text{by (2)}] \quad (4) \\ &\leq 2\left(\|\mathbf{a}_\mathbf{z} - \mathcal{A}_D(\mathbf{z})\| + \|\mathcal{A}_D(\mathbf{z}) - \mathbf{z}\|\right) && [\Delta \text{ ineq}] \quad (5)\end{aligned}$$

Note that

$$\mathbf{a_z} = \frac{\mathcal{A}_D(\mathbf{z})}{\|\mathcal{A}_D(\mathbf{z})\|_2} \tag{6}$$

Therefore,

$$\begin{aligned}\|\mathbf{a_z} - \mathcal{A}_D(\mathbf{z})\| &= \|\frac{\mathcal{A}_D(\mathbf{z})}{\|\mathcal{A}_D(\mathbf{z})\|_2} - \mathcal{A}_D(\mathbf{z})\| \\ &= \|\left(1 - \frac{1}{\|\mathcal{A}_D(\mathbf{z})\|_2}\right)\mathcal{A}_D(\mathbf{z})\| \\ &= |\|\mathcal{A}_D(\mathbf{z})\|_2 - 1|\end{aligned}$$

Furthermore, by triangle inequality

$$\|\mathcal{A}_D(\mathbf{z})\|_2 \leq \|\mathcal{A}_D(\mathbf{z}) - \mathbf{z}\|_2 + \|\mathbf{z}\| = \|\mathcal{A}_D(\mathbf{z}) - \mathbf{z}\|_2 + 1$$

Also, by triangle inequality,

$$\|\mathcal{A}_D(\mathbf{z})\|_2 \geq \|\mathbf{z}\| - \|\mathbf{z} - \mathcal{A}_D(\mathbf{z})\|_2 = 1 - \|\mathcal{A}_D(\mathbf{z}) - \mathbf{z}\|_2$$

Therefore,

$$-\|\mathcal{A}_D(\mathbf{z}) - \mathbf{z}\|_2 \leq \|\mathcal{A}_D(\mathbf{z})\|_2 - 1 \leq \|\mathcal{A}_D(\mathbf{z}) - \mathbf{z}\|_2$$

Hence,

$$|\|\mathcal{A}_D(\mathbf{z})\|_2 - 1| \leq \|\mathcal{A}_D(\mathbf{z}) - \mathbf{z}\|_2 \tag{7}$$

Finally, by combining equations (5) and (7) with equation (1), we get that $\|\mathbf{a_z} - \mathbf{a_z^*}\|_2 \sim O(\frac{\sqrt{k}}{D})$.

Since $d(\mathbf{a_z}, \mathbf{a_z^*}) \propto \|\mathbf{a_z} - \mathbf{a_z^*}\|_2^2$, we have our result

$$d(\mathbf{a_z}, \mathbf{a_z^*}) \sim O\left(\frac{k}{D^2}\right)$$

Moreover, the time complexity of Algorithm (A) is $O(k)$ and each step from (4) to (12) can be computed in parallel for each j and each $\mathbf{z}$. Also, algorithm (A) requires no explicit storage of the tessellating set Γ_D. This completes the proof. □

Algorithm A Region Specification on Γ_D

```
1: procedure TESSVECTOR-$D(\mathbf{z}, D)$
2:     initialise $\tilde{\mathbf{a}}_\mathbf{z}$ to all zeros
3:     for each $\mathbf{z} \in \mathbf{Z}$ do
4:         for each $j \in \{1, 2, 3, \cdots k\}$ do
5:             compute $a_+ = |Dz^j - \lceil Dz^j \rceil|$
6:             compute $a_- = |Dz^j - \lfloor Dz^j \rfloor|$
7:             if $a_+ \leq a_-$ then
8:                 set $\tilde{\mathbf{a}}_z^j = \frac{\lceil Dz^j \rceil}{D}$
9:             else
10:                set $\tilde{\mathbf{a}}_z^j = \frac{\lfloor Dz^j \rfloor}{D}$
11:            end if
12:        end for
13:    end for
14:    normalise to get $\mathbf{a_z} = \frac{\tilde{\mathbf{a}}_\mathbf{z}}{\|\tilde{\mathbf{a}}_\mathbf{z}\|_2}$
15:    return $\mathbf{a_z}$
16: end procedure
```

2 FURTHER DISCUSSION

2.1 Uniform Tessellation

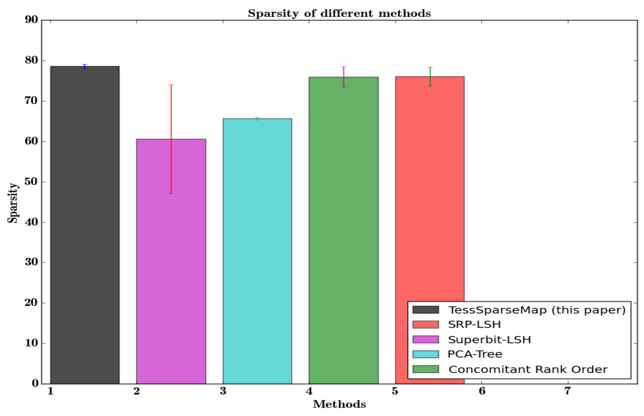
A key consideration while designing a tessellation schema on the unit sphere is whether the tessellation needs to be uniform or non-uniform over the surface of the unit sphere. There is no one way to capture the notion of "uniformity" in the context of a tessellation schema. A few example conditions could be that each tessellating vector should be equidistant from the closest tessellating vector, or by symmetry should have the same number of closest tessellating vectors, or the diameter of each tile (distance between farthest points within the same tile) should be the same for each tile.

But whichever way "uniformity" is defined, as a general scheme, a uniform tessellation would make intuitive sense because it captures the relevant locality properties of any set of factors irrespective of their distribution. However, in many instances a uniform tessellation may be overkill, and especially for clustered data, a non-uniform tessellation might be more appropriate from efficiency considerations. In particular, a uniform tessellation could be made into a non-uniform tessellation simply by dropping some of the tessellating vectors.

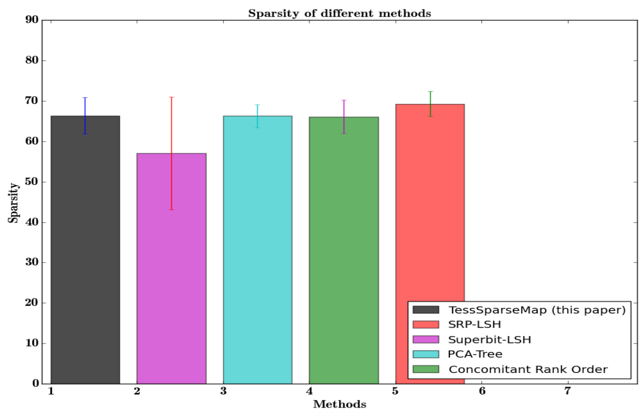
The directional tessellating set $\mathcal{A}$ on a ternary base set $\mathcal{B}$ does not uniformly tessellate the unit sphere. This is because for each tessellating vector in Γ, the distance from the nearest tessellating vector depends on the number of non-zeros in the vector.

In particular, the nearest neighbour to a vector $\mathbf{a}_i$ is every vector $\mathbf{a}_j$ such that the unnormalised vectors $\tilde{\mathbf{a}}_i$ and $\tilde{\mathbf{a}}_j$ differ by a Euclidian distance of 1 in the unnormalised space. That is, every nearest neighbour to a vector $\mathbf{a}_i$ can be found by replacing a single element in the unnormalised version of the vector $\tilde{\mathbf{a}}_i$ in the following way. First, obtain $\tilde{\mathbf{a}}_j$ by replacing a single 1 or -1 in $\tilde{\mathbf{a}}_i$ by a 0, or replace a single 0 by either a 1 or a -1. This is then re-normalised to get the corresponding $\mathbf{a}_j$.

The proof of the above statement is the following. First, clearly the nearest neighbour to every $\mathbf{a}_i$ must belong to the same orthant as $\mathbf{a}_i$. Suppose $\mathbf{a}_j$ belongs to the set of nearest neighbours. Therefore, $sign(\mathbf{a}_i^\iota)sign(\mathbf{a}_j^\iota) \geq 0$ for every $\iota = 1, 2, \cdots k$. Therefore, without loss of generality, assume that $\mathbf{a}_i$ lies in

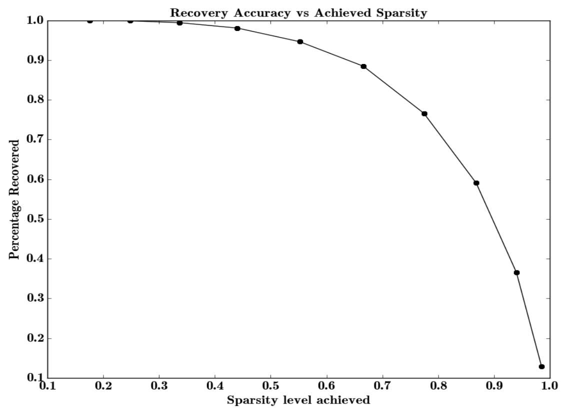


(a) average percentage of discarded items for Synthetic Data

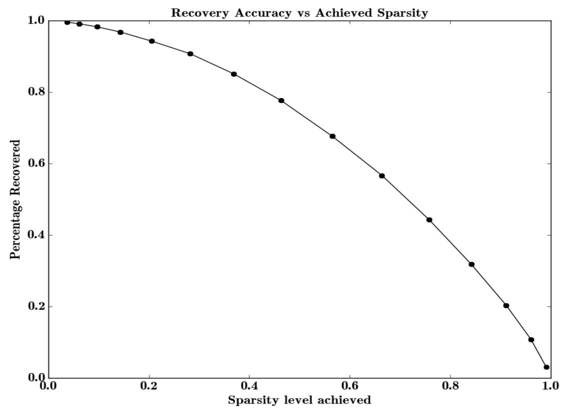


(b) average percentage of discarded items for MovieLens

Figure 1: Mean percentage of discarded items across users for (a) Synthetic Data (b) MovieLens Data



(a) Recovery Accuracy



(b) Recovery Accuracy versus Sparsity

Figure 2: Plot for Recovery Accuracy versus Sparsity Achieved (a) Synthetic Data (b) MovieLens Data

the non-negative orthant, $\mathbf{a}_j^\iota \geq 0 \; \forall \; \iota$.

Suppose $\mathbf{a}_i$ has t non-zero elements, and $\mathbf{a}_j$ has $t + s$ non-zero elements, with $t > 0, s \neq 0$. Then, the angular distance between $\mathbf{a}_i$ and $\mathbf{a}_j$ is

$$d(\mathbf{a}_i, \mathbf{a}_j) = \begin{cases} 1 - \sqrt{\frac{t}{t+s}} & \text{if } s > 0 \\ 1 - \sqrt{\frac{t-s}{t}} & \text{if } s < 0 \end{cases}$$

Clearly, the minimum distance is attained for any t by setting $s = 1$.

Following this, we see that the distance between closest neighbours $\mathbf{a}_i, \mathbf{a}_j$ is $d(\mathbf{a}_i, \mathbf{a}_j) = 1 - \sqrt{\frac{t}{t+1}}$. Therefore, the distance between closest neighbours for a tessellating vector depends on the number of non-zero elements in the vector. In particular, the set $\mathcal{A}$ is more densely packed with vectors oriented towards the "centre" of each orthant as opposed to vectors along the axes or along any lower dimensional subspaces formed from subsets of the axes.

As mentioned in the main manuscript, obtaining uniform tessellations deterministically is a challenging task and heuristics [2, 1, 3] must be resorted to.

2.2 Parse Tree Constructions

In this section we describe some examples of parse-tree constructions for the permutation mapping step, in particular the parse tree used in the experiments in the main manuscript. Note that computing the permutation map proceeds in two steps- (i) reading $\mathbf{a_z}$ at time j as a sequence of δ characters at a time as $\tilde{\mathbf{a}}_\delta^j = [\tilde{\mathbf{a}}^{j-\delta}, \cdots \tilde{\mathbf{a}}^j]$, and (ii) marking the next non-zero index via a counter τ_t on $\boldsymbol{\phi}(\cdot)$ as a function of τ_{j-1} and $\tilde{\mathbf{a}}_j^\delta$ as $\tau_j = f(\tau_{j-1}; \tilde{\mathbf{a}}_\delta^j)$.

A key desideratum for our mapping scheme is that for any two $\mathbf{a}, \mathbf{a}'$ at any step $\tau_j = \tau_j'$ if and only if $[\mathbf{a}^{j-t}, \cdots \mathbf{a}^j] = [\mathbf{a}'^{j-t}, \cdots \mathbf{a}'^j]$ for some $t_0 \geq \delta$. This is useful in preventing "accidental" overlapping sparsity, so that the same sparsity pattern is not obtained accidentally via two entirely different set of sliding window characters read on $\mathbf{a}$ and $\mathbf{a}'$.

It is immediately clear that the one-hot encoding satisfies this property with $t_0 = \delta = 1$. Another simple scheme (and one that we used in our experiments) is the following.

Consider a sliding window of size $\delta = 1$. Suppose after $j-1$ steps, the counter is at position τ_{j-1}. Shift the counter to position τ_j depending on the value of currently read $\mathbf{a}^j$ as follows-

$$\tau_j = \begin{cases} kj & \text{if } \mathbf{a}^j = 1 \\ \tau_{j-1} + 1 & \text{if } \mathbf{a}^j = 0 \\ k(k+j) & \text{if } \mathbf{a}^j = -1 \end{cases}$$

The dimensionality increase required is $p \sim O(k^2)$, however, with the inverted index representation, we only require $O(klogk)$ storage space complexity.

Many other parse-tree methods are possible. In particular, a straightforward generalisation of the one-hot scheme described in the manuscript would obtain a class of methods that involve a one-hot encoding on a D-ary tessellation with a δ-parse-tree which has D^δ leaf nodes. For this schema, for any two $\mathbf{a}, \mathbf{a}'$ we shall have the corresponding counters τ_j and τ_l' at time j and l respectively to be equal $\tau_j = \tau_l'$ if and only if $j = l$ and $\tilde{\mathbf{a}}_j^\delta = \tilde{\mathbf{a}}_l^{'\delta}$.

3 ADDITIONAL PLOTS

We show some additional plots to augment the experimental results given in the main manuscript. Figure (1a) and (1b) respectively show the average sparsity levels obtained across all users for different methods mentioned in this mansucript. We also show error bars to give an idea of the variance.

Figure (2a) and (2b) show a plot of recovery accuracy plotted against average sparsity achieved across all users for our method.

References

[1] E. L. Altschuler, T. J. Williams, E. R. Ratner, R. Tipton, R. Stong, F. Dowla, and F. Wooten. Possible global minimum lattice configurations for thomson's problem of charges on a sphere. *Physical Review Letters*, 78(14):2681, 1997.

[2] A. Katanforoush and M. Shahshahani. Distributing points on the sphere, i. *Experimental Mathematics*, 12(2):199–209, 2003.

[3] M. Tegmark. An icosahedron-based method for pixelizing the celestial sphere. *The Astrophysical Journal*, 470:L81, 1996.