CS 2243 Fall 2024
09/11/2024

Scribes: Alvan Arulandu
Based on notes by: Itai Shapira and Sara Kangaslahti

Lecture 3: Iterative Methods for Tensor Decomposition

1 Jennrich's Algorithm in Practice

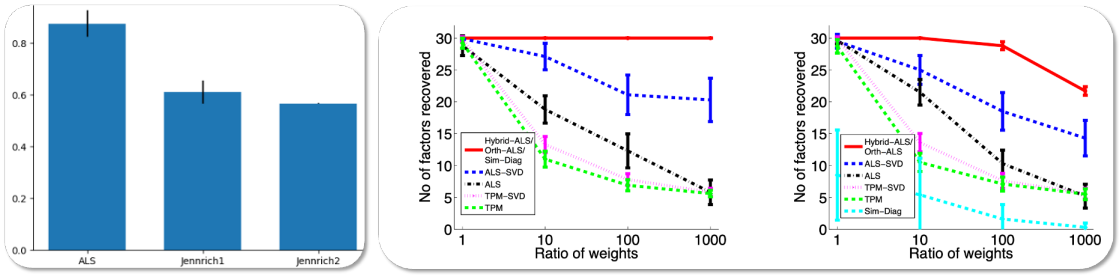
In Lecture 2, we saw tensor decomposition as a powerful tool to reconcile the rotation problem of matrices. Given rank-k tensor $T \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ with

$$T = \sum_{i=1}^{k} \lambda_i \mathbf{u}_i^{\otimes 3} \quad (\{\mathbf{u}_i\}_{i=1}^k \text{ are linearly independent and } \lambda_1 \geq \cdots \geq \lambda_k \geq 0 \text{ WLOG})$$

Jennrich's algorithm recovers $\{\mathbf{u}_i\}_{i=1}^k$, decomposing T into its k components. However, Jennrich's algorithm is wishful thinking in practice.

1.1 Noise Robustness

From Problem Set 1 Problem 2, we see that for the robust eigendecompositions used in Jennrich's, $\sigma_{\min}(U) \geq 1/\text{poly}(d)$. In the ideal noiseless setting, Jennrich's works great, but in practice, poor condition number bounds cause a sharp performance decline for even slight deviations from its low-rank form.



(a) (b)

Figure 1: (a) A comparison between the accuracy in decomposing a $25 \times 25 \times 25$ noised tensor for two implementations of Jennrich's algorithm and alternating least squares [Unk21].(b) A plot of the number of recovered tensor components against a proxy for relative noise strength for various algorithms, Jennrich's being *SimDiag* in cyan [SV17].

We see from Figure 1 that Jennrich's algorithm accurately recovers all tensor components in the noiseless setting, but as noising increasing, its performance declines the fastest. As one can imagine, this isn't great for the real world.

1.2 Computational Complexity

However, even with control on noise strength, Jennrich's is hopeless to compute for practical problems, with time complexity dominated by $O(d^\omega)$ matrix multiplication operations and $O(d^\alpha)$ pseudoinversion calls, where $2 < \alpha, \omega \leq 3$. Of course, T itself takes $O(d^3)$ space, but we can optimize this by only computing contractions $M_z = T(:,:,z) = \mathbb{E}[\langle \mathbf{x}, \mathbf{z} \rangle \mathbf{x}\mathbf{x}^\top]$ in $O(d^2)$ time. However, for extended Jennrich's algorithm on $T = \sum_{i=1}^k u_i \otimes v_i \otimes w_i$, computing w_i requires computing T_{ijk} for various triples which we can do lazily at best in $O(d^3)$.

2 Iterative Methods

Given recent progress in this problem space, practitioners resort to iterative algorithms, primarily Gradient Ascent, Power Iteration, and Alternating Least Squares, to resolve these robustness and computational challenges. Again, we assume that $T = \sum_{i=1}^{k} \lambda_i \mathbf{u}_i^{\otimes 3}$ with $\{u_i\}_{i=1}^{k}$ orthonormal, though this can be waived via Section 2.5. From Lecture 2, we have the following polynomial given by contracting T.

$$p(\mathbf{x}) = T(\mathbf{x}, \mathbf{x}, \mathbf{x}) = \sum_{a,b,c} T_{abc} x_a x_b x_c = \sum_i \lambda_i \langle \mathbf{u}_i, \mathbf{x} \rangle^3 \tag{1}$$

As we've seen before, many tensor problems including computing eigenvectors for worst case T is NP-hard, but as implied by Problem Set 1 Problem 3, local maximizers of $p(x)$ over the unit sphere are exactly $\{\mathbf{u}_i\}_{i=1}^{k}$. We can see some intuition by consider u' such that $\langle \mathbf{u}_i, \mathbf{u}' \rangle \approx 0 \; \forall i \in [k]$. Then,

$$\begin{aligned} \mathbf{x} = \mathbf{u}' &\Rightarrow p(\mathbf{x}) = \sum_{i=1}^{k} \lambda_i \langle \mathbf{u}_i, \mathbf{u}' \rangle \approx 0 \\ \mathbf{x} = \mathbf{u}_j &\Rightarrow p(\mathbf{x}) = \sum_{i=1}^{k} \lambda_i \langle \mathbf{u}_i, \mathbf{u}_j \rangle \Rightarrow \lambda_i \gg 0 \end{aligned} \tag{2}$$

For $\mathbf{u} = \mathbf{u}_i$, we see that p is large, and for vectors approximately orthogonal to our component vectors, of which is the majority in high dimensional space, p is small. Formally,

$$\{\mathbf{u}_i\}_{i=1}^{k} \approx \underset{\|\mathbf{x}\|=1}{\operatorname{argmax}} \; p(\mathbf{x}) = \underset{\|\mathbf{x}\|=1}{\operatorname{argmax}} \sum_{i=1}^{k} \lambda_i \langle \mathbf{u}_i, \mathbf{x} \rangle^3 \tag{3}$$

We see polynomial optimization $\Leftrightarrow$ tensor decomposition, motivating our use of iterative optimizers. It follows that all three of these iterative algorithms yield a related analysis. Of course, such methods only give a single local optimum, with our ultimate goal being all local optima.

2.1 Gradient Ascent

Taking our optimization equation in (3), we derive gradient ascent by taking our current $\mathbf{x}$ and taking a η size step in the $\nabla p(\mathbf{x})$ direction. If $\mathbf{x}^{(t)}$ is our vector after t steps, we have the following recurrence which we can simplify as a contraction by (1).

$$\mathbf{x}^{(t)} = \mathbf{x}^{(t-1)} + \eta \cdot \nabla p(\mathbf{x}^{(t-1)}) = \mathbf{x}^{(t-1)} + 3\eta \cdot T(:, \mathbf{x}^{(t-1)}, \mathbf{x}^{(t-1)})$$

A glaring issue is that $\|\mathbf{x}^{(t)}\| \neq 1$ as defined above.

2.1.1 Riemannian Gradient Descent

We solve this we could project to our Reimannian constraint surface, the unit sphere $\|\mathbf{x}\| = 1$.

$$\mathbf{x}^{(t)} = \text{proj}\left(\mathbf{x}^{(t-1)} + 3\eta \cdot T(:, \mathbf{x}^{(t-1)}, \mathbf{x}^{(t-1)})\right)$$

However, this wastes movement in the tangent direction, which is most relevant to the updated position when projected on the circle.

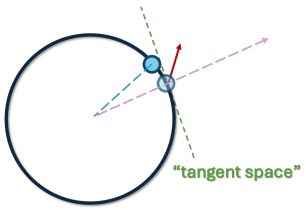


Figure 2: Constrained Gradient Step Projection

Figure 2 demonstrates this. Projecting the red vector as we've done above, skews the effect the tangent component in relation to the perpendicular component. For example, consider an update primarily away from the sphere, with a small tangent component. Since we're constrained to the unit sphere, this component should not affect $\mathbf{x}^{(t)}$, but the tangent component does greatly! We resolve this by first projecting to the tangent space, and then to the circle, to isolate this tangent component before making our final update. Formally,

$$\mathbf{x}^{(t+1)} = \text{proj}\left(\mathbf{x}^{(t)} + 3\eta \cdot \Pi \cdot T(:, \mathbf{x}^{(t)}, \mathbf{x}^{(t)})\right), \quad \Pi = \left(\text{Id} - \mathbf{x}^{(t)}(\mathbf{x}^{(t)})^T\right) \tag{4}$$

where Π is the tangent projection operator. Expanding (4), we have the following.

$$\begin{aligned}
\mathbf{x}^{(t+1)} &= \text{proj}\left(\mathbf{x}^{(t)} + 3\eta \cdot \left(T(:, \mathbf{x}^{(t)}, \mathbf{x}^{(t)}) - \mathbf{x}^{(t)}(\mathbf{x}^{(t)})^T T(:, \mathbf{x}^{(t)}, \mathbf{x}^{(t)})\right)\right) \\
&= \text{proj}\left(\mathbf{x}^{(t)} + 3\eta \cdot \left(T(:, \mathbf{x}^{(t)}, \mathbf{x}^{(t)}) - \mathbf{x}^{(t)} T(\mathbf{x}^{(t)}, \mathbf{x}^{(t)}, \mathbf{x}^{(t)})\right)\right) \\
&= \text{proj}\left(\mathbf{x}^{(t)} + 3\eta \cdot \left(T(:, \mathbf{x}^{(t)}, \mathbf{x}^{(t)}) - \mathbf{x}^{(t)} p(\mathbf{x}^{(t)})\right)\right)
\end{aligned} \tag{5}$$

We observe that $-\mathbf{x}^{(t)} p(\mathbf{x}^{(t)})$ effectively cancels the perpendicular component of the step as intended. This is our final gradient ascent update.

2.2 Tensor Power Method

This form in (5) also motivates our choice of step size η giving the following nice cancellation.

$$\eta = \frac{1}{3p(\mathbf{x}^{(t)})} \Rightarrow \mathbf{x}^{(t+1)} = \text{proj}\left(\frac{T(:, \mathbf{x}^{(t)}, \mathbf{x}^{(t)})}{p(\mathbf{x}^{(t)})}\right) = \text{proj}\left(T(:, \mathbf{x}^{(t)}, \mathbf{x}^{(t)})\right) \tag{6}$$

We call this specific update the tensor power method, a generalization of the matrix power method.

2.2.1 Matrix Power Method

The matrix power method efficiently computes the top eigenvector of a matrix, faster than alternative methods via characteristic polynomial root solving. Formally, if $T = \sum_{i=1}^{k} \lambda \mathbf{u}_i \mathbf{u}^\top$, the matrix power method iteratively applies T to x, normalizing the result. In tensor notation, for $x = \sum_{i=1}^{k} c_i \mathbf{u}_i$,

$$T\mathbf{x} = T(:, \mathbf{x}) = \sum_{i=1}^{k} \lambda_i c_i \cdot \mathbf{u}_i \Rightarrow \text{proj}(T(:, \mathbf{x})) = \sum_{i=1}^{k} \frac{\lambda_i c_i}{\left(\sum_j \lambda_j^2 c_j^2\right)^{1/2}} \cdot \mathbf{u}_i$$

Notice this transforms the linear combination coefficients of x from $(c_1, \ldots, c_k)$ to $\text{proj}(\lambda_1 c_1, \ldots, \lambda_k c_k)$. WLOG, we can order $\lambda_1 \geq \ldots \lambda_k$. Then, we see that from the projection, the ratio of the j-th coefficient to the 1st coefficient changes by a factor of $\lambda_j/\lambda_1 < 1$ while the inverse ratio is > 1. Thus, after each projection, the first component is weighted up while the remaining are weighted down, meaning that after iterative projection, these coefficients must converge to $(1, 0, \ldots, 0)$, implying $\mathbf{x} \to \mathbf{u}_1$ which is the top eigenvector.

2.2.2 Tensor Power Method

For tensor power method, we have an analagous analysis with contraction $T(:, \mathbf{x}, \mathbf{x})$.

$$T(:, \mathbf{x}, \mathbf{x}) = \sum_{i=1}^{k} \lambda_i \langle \mathbf{x}, \mathbf{u}_i \rangle^2 \mathbf{u}_i = \sum_{i=1}^{k} \lambda_i c_i^2 \mathbf{u}_i \Rightarrow \text{proj}(T(:, \mathbf{x}, \mathbf{x})) = \text{proj}\left(\sum_{i=1}^{k} \lambda_i c_i^2 \mathbf{u}_i\right) = \sum_{i=1}^{k} \frac{\lambda_i c_i^2}{\sum_{j=1}^{k} \lambda_j^2 c_j^4} \mathbf{u}_i$$

Again, considering the coordinate ratio $r_i = \frac{c_8}{c_1}$,

$$r_i = \frac{c_i}{c_1} \xrightarrow{\text{step}} \frac{\lambda_i c_i^2 / \left(\sum_j \lambda_j^2 c_j^4\right)^{1/2}}{\lambda_1 c_1^2 / \left(\sum_j \lambda_j^2 c_j^4\right)^{1/2}} = \frac{\lambda_i c_i^2}{\lambda_1 c_1^2} = \frac{\lambda_i c_i}{\lambda_1 c_1} r_i \tag{7}$$

Unlike the matrix power method, this does not necessarily decay if $\lambda_i c_i > \lambda_i c_i$. We thus define and suppose $\rho < 1$ where

$$\rho \stackrel{\text{def}}{=} \max_{i \neq 1} \frac{\lambda_i c_i}{\lambda_1 c_1} < 1$$

If $\rho < 1$, r_i must decay by (7) for $i \neq 1$. Further, from the c_i^2, we have the following doubly exponential convergence.

$$r_i \to \rho r_i \to \rho^2 r_i \to \rho^4 r_i \to \rho^4 r_i \to \dots$$

While the tensor power method then has faster convergence that the matrix power method, the convergence of our algorithm depends on $\rho < 1$, which occurs with probability $1/k$ for random initialization. Further, our vector limit depends on the initialization $\mathbf{x}$, where we converge to u_{i^*} where $i^* = \text{argmax}_i \lambda_i c_i$. If this i^* is not unique, then our method has $i^* = \text{argmax}_i c_i$. With this, we have two strategies for computing the remaining components.

2.2.3 Deflation

For finding the rest of the components, we take $\hat{\mathbf{u}} \approx u_i$ to be our converged limit. By (2), we have $\lambda_i = p(\mathbf{u}_i) \approx p(\hat{\mathbf{u}})$. Deflating our tensor of this component,

$$T - p(\hat{\mathbf{u}})\hat{\mathbf{u}}^{\otimes 3} \approx \sum_{j \neq i} \lambda_i \mathbf{u}_i^{\otimes 3}$$

we can apply tensor power method again to get the next top vector. Repeating this process of deflation and approximation yields all the components, with error compounding after each successive deflation.

2.3 Clustering

Alternatively, we can randomly initialize $\mathbf{x}$ and compute many $\hat{\mathbf{u}}$. For the $\mathbf{x}$ such that $i^* = i$, we get a $\hat{\mathbf{u}}$ near $\mathbf{u}_i$. Doing this many times, we build clusters around each $\mathbf{u}_i$ and use a clustering algorithm to estimate each component.

2.4 Alternating Least Squares (ALS)

ALS is a popular technique for simultaneously learning all components, without suffering from deflation issues or requiring many random runs. We iteratively estimate $\{\mathbf{u}_i^{(t)}\}$ where t is the number of steps, using the following step rule.

$$\mathbf{u}_i^{(t+1)} = \text{proj}\left(\arg\min_{\hat{\mathbf{u}}_i} \left\| T - \sum_{i=1}^{k} \hat{\mathbf{u}}_i \otimes \mathbf{u}_i^{(t)} \otimes \mathbf{u}_i^{(t)} \right\|_F^2 \right) \tag{8}$$

Since our objective is linear in optimizing variable $\hat{\mathbf{u}}_i$, this is exactly least-squares regression! While a rigorous analysis is quite hard, ALS has garnered significant attention for robustness and computational efficiency. In fact, we can interpret the tensor power method as a rank-1 version of ALS. As we said at the start, such iterative methods are shockingly related.

Proof. Consider some current iterate $\hat{\mathbf{u}}_i^{(t)}$. Then,

$$\begin{aligned}
\arg\min_{\hat{\mathbf{u}}_i} \left\| T - \sum_{i=1}^{k} \hat{\mathbf{u}}_i \otimes \mathbf{u}_i^{(t)} \otimes \mathbf{u}_i^{(t)} \right\|_F^2 &= \arg\min_{\hat{\mathbf{u}}_i} \sum_{a,b,c} \left(T_{abc} - (\hat{\mathbf{u}}_i)_a \left(\mathbf{u}_i^{(t)}\right)_b \left(\mathbf{u}_i^{(t)}\right)_c \right)^2 \\
&= \arg\min_{\hat{\mathbf{u}}_i} \sum_{a,b,c} \left(T_{abc}^2 - 2T_{abc} \, (\hat{\mathbf{u}}_i)_a \left(\mathbf{u}_i^{(t)}\right)_b \left(\mathbf{u}_i^{(t)}\right)_c + (\hat{\mathbf{u}}_i)_a^2 \left(\mathbf{u}_i^{(t)}\right)_b^2 \left(\mathbf{u}_i^{(t)}\right)_c^2 \right)
\end{aligned}$$

We have that T_{abc}^2 is a constant and $\mathbf{u}_i^{(t)}$ is a previous iterate meaning $\sum_{bc}\left(\mathbf{u}_i^{(t)}\right)_b^2\left(\mathbf{u}_i^{(t)}\right)_c^2 = 1$. Simplifying with contractions,

$$= \arg\min_{\hat{\mathbf{u}}_8}\sum_a\left((\hat{\mathbf{u}}_i)_a^2 - 2\,(\hat{\mathbf{u}}_i)_a\sum_{b,c}T_{abc}\left(\mathbf{u}_i^{(t)}\right)_b\left(\mathbf{u}_i^{(t)}\right)_c\right) = \arg\min_{\hat{\mathbf{u}}_8}\sum_a\left((\hat{\mathbf{u}}_i)_a^2 - T(:,u_i^{(t)},u_i^{(t)})_a\right)$$

We observe that this is minimized for $(\hat{\mathbf{u}}_i)_a = T(:,u_i^{(t)},u_i^{(t)})_a$, meaning

$$\hat{\mathbf{u}}_i = \text{proj}(T(:,u_i^{(t)},u_i^{(t)}))$$

This is equivalent to the tensor power method. *f*

2.5 Orthogonality and Whitening

At first, our assumption that $\{\mathbf{u}_i\}_{i=1}^k$ orthogonality seems quite unsound, but in many applications, we have access to both T and some matrix M of the following form, which allows us to only require linearly independent $\mathbf{u}_i$.

$$M = \sum_{i=1}^k \lambda_i \mathbf{u}_i\mathbf{u}_i^\top$$

We can waive orthogonality because given M, we can modify $\{\mathbf{u}_i\}_{i=1}^k$ such that they are orthonormal via a process called *whitening*. We first eigendecompose $M = VDV^\top$ with eigenvectors in $V \in \mathbb{R}^{d\times k}$ and eigenvalues in diagonal matrix $D \in \mathbb{R}^{k\times k}$. We then let $W = VD^{-1/2} \in \mathbb{R}^{d\times k}$ and $\tilde{\mathbf{u}}_i = \lambda_i^{1/2}W^\top\mathbf{u}_i$ such that W standardizes the data via the following simplification. Since $V^\top V = I$,

$$W^\top MW = D^{-1/2}V^\top VDV^\top VD^{-1/2} = D^{-1/2}DD^{-1/2} = I$$
$$W^\top MW = \sum_{i=1}^k \lambda_i(W^\top\mathbf{u}_i)(W^\top\mathbf{u}_i)^\top = \sum_{i=1}^k \tilde{\mathbf{u}}_i\tilde{\mathbf{u}}_i^\top$$
$$\Rightarrow \sum_{i=1}^k \tilde{\mathbf{u}}_i\tilde{\mathbf{u}}_i^\top = I$$

This implies that $\tilde{\mathbf{u}}_i$ are orthogonal. We can then define $T' = T(W,W,W) \in \mathbb{R}^{k\times k\times k}$. Noticing $\langle W\mathbf{x},\mathbf{u}\rangle = \mathbf{x}^\top W^\top\mathbf{u}_i = \lambda_i^{-1/2}\langle\tilde{\mathbf{u}}_i,\mathbf{x}\rangle$,

$$T'(x,y,z) = T(W\mathbf{x},W\mathbf{y},W\mathbf{z}) = \sum_{i=1}^k \lambda_i\langle W\mathbf{x},\mathbf{u}_i\rangle\langle W\mathbf{y},\mathbf{u}_i\rangle\langle W\mathbf{z},\mathbf{u}_i\rangle = \sum_{i=1}^k \lambda_i^{-1/2}\langle\tilde{\mathbf{u}}_i,\mathbf{x}\rangle\langle\tilde{\mathbf{u}}_i,\mathbf{y}\rangle\langle\tilde{\mathbf{u}}_i,\mathbf{z}\rangle$$

$$\Rightarrow T = \sum_{i=1}^k \lambda_i\mathbf{u}_i^{\otimes 3}, \quad T' = \sum_{i=1}^k \lambda_i^{-1/2}\tilde{\mathbf{u}}_i^{\otimes 3}$$

Notice that we've reduced our original tensor problem T to an alternate tensor T' which orthonormal $\tilde{\mathbf{u}}_i$. Applying any method to solve T', we can then map $\tilde{\mathbf{u}}_i \to \mathbf{u}_i$ by applying $\lambda_i^{-1/2}D^{1/2}V^\top$ since:

$$\lambda_i^{-1/2}D^{1/2}V^\top\tilde{\mathbf{u}}_i = \lambda_i^{-1/2}D^{1/2}V^\top\lambda_i^{1/2}W^\top\mathbf{u}_i = D^{1/2}V^\top VD^{-1/2}\mathbf{u}_i = \mathbf{u}_i$$

This reduces any independent $\{\mathbf{u}\}_{i=1}^k$ problem to an equivalent orthonormal one, using M and one additional eigendecomposition.

2.6 No Whitening

If no such M is available, we cannot whiten $\mathbf{u}_i$, making our analysis much harder.

Theorem 1 ([SV17]). *Given $T = \sum_{i=1}^{k} \mathbf{u}_i^{\otimes 3}$ for incoherent unit vectors $\mathbf{u}_1, \ldots, \mathbf{u}_k$, i.e., satisfying*

$$\left|\langle \mathbf{u}_i, \mathbf{u}_j \rangle\right| \leq c_{max} \leq \frac{1}{k^{1+\epsilon}}, \quad (i \neq j)$$

$O(\log k + \log\log d)$ iterations of tensor power method starting from a random initialization yields a vector $\hat{\mathbf{u}}$ that is $O\left(k^{1/2}\max(c_{max}, 1/d)\right)$-close to some $\mathbf{u}_i$, with high probability.

Proof. Let's consider the tensor power method without whitening. Recall our update step.

$$\mathbf{x}^{(t)'} = T(:, \mathbf{x}^{(t-1)}, \mathbf{x}^{(t-1)}) = \sum_{i=1}^{k} \lambda_i \langle \mathbf{x}^{(t-1)}, \mathbf{u}_i \rangle^2 \mathbf{u}_i = \sum_{i=1}^{k} \lambda_i a_{i,t}^2 \Rightarrow \mathbf{x}^t = \text{proj}(\mathbf{x}^{(t)'}) = \frac{\mathbf{x}^{(t)'}}{\|\mathbf{x}^{(t)'}\|}$$

Let $\hat{a}_{i,t} = a_{i,t}/a_{1,t}$. Note that $a, \hat{a}$ were formerly called c, r_i in the orthogonal case. We call the correlation between two of our vector $c_{ij} = \langle \mathbf{u}_i, \mathbf{u}_j \rangle$. Then, since $\|\mathbf{X}^{(t)'}\|$ is independent of j, we can derive a recursive ratio formula.

$$a_{j,t} = \frac{\sum_{i=1}^{k} a_{i,t-1}^2 \langle \mathbf{u}_i, \mathbf{u}_j \rangle}{\|\mathbf{X}^{(t)'}\|} = \frac{a_{1,t-1}^2 \sum_{i=1}^{k} \hat{a}_{i,t-1}^2 c_{ij}}{\|\mathbf{X}^{(t)'}\|} \Rightarrow \hat{a}_{j,t} = \frac{\sum_{i=1}^{k} \hat{a}_{i,t-1}^2 c_{i,j}}{\sum_{i=1}^{k} \hat{a}_{i,t-1}^2 c_{i,1}}$$

Isolating the $i = 1$ term,

$$\hat{a}_{j,t} = \left(c_{1,j} + \sum_{i=2}^{k} \hat{a}_{i,t-1}^2 c_{i,j}\right) \cdot \frac{1}{1 + \sum_{i=2}^{k} \hat{a}_{i,t-1}^2 c_{i,1}}$$

From the assumption of the theorem, $|c_{ij}| \leq c_{\max} \leq k^{-(1+\epsilon)}$ for $i \neq j$. Then, $\|\sum_{i=2}^{k} \hat{a}_{i,t-1}^2 c_{i,1}\| \leq k c_{max} \leq k^{-\epsilon} \ll 1$. Recall by taylor expansion that $1/(1+x) = 1 - x + x^2 - \cdots \approx 1 - x$ for small x. Approximating,

$$\hat{a}_{j,t} \approx \left(c_{1,j} + \sum_{i=2}^{k} \hat{a}_{i,t-1}^2 c_{i,j}\right)\left(1 - \sum_{i=2}^{k} \hat{a}_{i,t-1}^2 c_{i,1}\right) \tag{9}$$

Lemma 1. *We show that $\max_{j\neq 1} |\hat{a}_{j,t}| < \beta_t$ for sequence $\{\beta_t\}_{t=0}^{\infty}$ defined recursively:*

$$\beta_t = \begin{cases} \max_{j\neq 1} |\hat{a}_{j,0}| & t = 0 \\ c_{\max} + \beta_{t-1}^2 + 3k c_{\max} \beta_{t-1}^2 \end{cases}$$

Proof. We aim to bound (9) inductively. By the definition of β_0, the base case is clearly true. Assume the inductive claim for $t-1$. For $j \neq 1$, by triangle inequality and the inductive claim,

$$\left|c_{1,j} + \sum_{i=2}^{k} \hat{a}_{i,t-1}^2 c_{i,j}\right| = \left|c_{1,j} + \hat{a}_{j,t-1}^2 + \sum_{i\neq j,1} \hat{a}_{i,t-1}^2 c_{ij}\right| \leq |c_{1,j}| + |\hat{a}_{j,t-1}^2| + \left|\sum_{i\neq j,1} \hat{a}_{i,t-1}^2 c_{ij}\right| \leq c_{\max} + \beta_{t-1}^2 + 2k c_{\max} \beta_{t-1}^2$$

Then, substituting in (9) and using triangle inequality again with the fact that $c_{\max} \leq k^{-(1+\epsilon)} \leq 1$,

$$\begin{aligned}
|\hat{a}_{j,t}| &\leq (c_{\max} + \beta_{t-1}^2 + 2k c_{\max} \beta_{t-1}^2)\left(1 + \left|\sum_{i=2}^{k} \hat{a}_{i,t-1}^2 c_{i,j}\right|\right) \leq (c_{\max} + \beta_{t-1}^2 + 2k c_{\max} \beta_{t-1}^2)(1 + k c_{\max} \beta_{t-1}^2) \\
&= c_{\max} + \beta_{t-1}^2 + k c_{\max}(1 + c_{\max})\beta_{t-1}^2 + k c_{\max}(1 + k c_{\max})\beta_{t-1}^4 \\
&\leq c_{\max} + \beta_{t-1}^2 + 2kc\beta_{t-1}^2 + k(1+k)\beta_{t-1}^2 \beta_{t-1}^2 \\
&= c_{\max} + \beta_{t-1}^2 + (2c_{\max} + (1+k)\beta_{t-1}^2) \cdot k\beta_{t-1}^2 \\
&\leq c_{\max} + \beta_{t-1}^2 + 3k c_{\max} \beta_{t-1}^2 \\
&= \beta_t
\end{aligned}$$

Here the last inequality is due to the fact that β_{t-1}^2 is small such that $(1+k)\beta_{t-1}^2 < c_{\max}$. We can check this inductively by bounding β_0 and then observing that β_t is decreasing in t, so we need to show $(1+k)\beta_0^2 < c_{\max}$. We defer this shortly and continue having proven the claim. *f*

It then suffices to analyze the recursive decay of β_t, reducing an analysis of $k-1$ quantities to a sole bound. In the orthogonal case, $c_{\max} = 0$ meaning $\beta_t = \beta_{t-1}^2 = \beta_0^{2^t} \to 0$ at a doubly exponential case if $\beta_0 < 1$, which corroborates what we previously found. If $c_{\max} > 0$, as discussed in the assumption of the proof of Lemma 1, we need $(1+k)\beta_0^2 < c_{\max}$ which holds if β_0 is sufficiently smaller than 1, $1 - \beta_0 \gg kc_{\max}$. If $c_{\max} \ll k^{-2}$, it turns out that this is true with high probability via random initialization [SV17]. Now, to analyze the β_t recursion, we break our evolution in three stages: $\beta_t \geq 0.1$, $0.1 \geq \beta_t \geq \sqrt{m}$, and $\beta_t \leq \sqrt{m}$ where $m = \max(c_{\max}, 1/d)$. If this is true, then for some constant C and $k > C$,

$$\beta_t \geq 0.1 \Rightarrow k\beta_{t-1}^2 \geq 0.1k \geq 1 \Rightarrow c_{\max} \leq kc_{\max}\beta_{t-1}^2 \Rightarrow \beta_t \leq (1+4kc_{\max})\beta_{t-1}^2$$

Unrolling this bound,

$$\beta_t \leq (1+4kc_{\max})^{\sum_{\vartheta=0}^{t-1} 2^\vartheta}\beta_0^{2^t} = (1+4kc_{\max})^{2^t-1}\beta_0^{2^t} \leq (\beta_0(1+4kc_{\max}))^{2^t}$$

With high probability, it turns out that $\beta_0 \leq 1 - 5kc_{\max}$. Then,

$$\leq (1 - kc_{\max} - 20k^2c_{\max}^2)^{2^t} \leq (1-kc_{\max})^{2^t} \Rightarrow t \leq \log_2(\log(\beta_0)/\log(1-kc_{\max})) = \log(\text{poly}(k)) = O(\log(k))$$

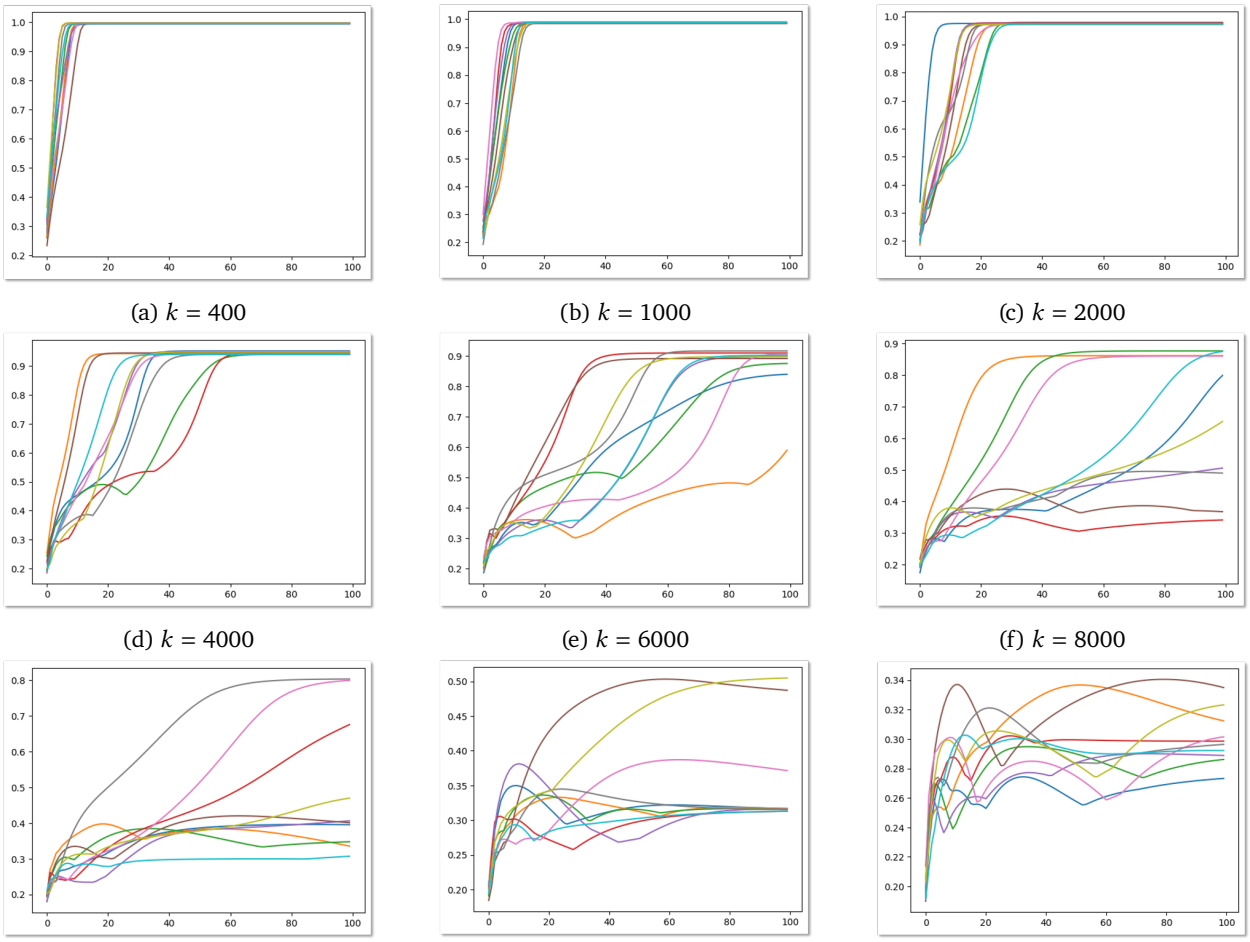
This implies that we stay in this regime for $\log k$ iterations. For the next regime, we simply re-index β_0 to when the first regime ends. Since $\beta_t \geq \sqrt{m}$, we have $\beta_t \geq \sqrt{m} \geq \sqrt{c_{\max}}$. Then,

$$\beta_t = (1+3kc_{\max})\beta_{t-1}^2 + c_{\max} \leq (2+3kc_{\max})\beta_{t-1}^2 \leq 3\beta_{t-1}^2 \leq 3^{2^t-1}\beta_0^{2^t} \leq (3\beta_0)^{2^t} \leq (0.3)^{2^t}$$

This implies that $t \leq O(\log\log\beta_t) = O(\log\log d)$. In the final regime, $\beta_t \leq \sqrt{m}$ implies the doubly exponential decay rate as seen before. Putting this all together, $t = O(\log k + \log\log d)$ iterations are needed. *f*

3 Empirical Mysteries

Our theoretical understanding of tensor decomposition algorithms is still limited for large k, underscoring the inherent difficulty of non-convex optimization problems such as those in neural networks. Deep learning aside, we're still trying to understand tensor power method! Figure 3 shows the behavior of tensor power method for varying k. We see that for low k, tensor power method does very well, as pictured in the high correlation in the first row of plots. However, at a lower threshold of $k \approx d$ non-monotonic behavior emerges, suggesting that the algorithm is switching its convergence target between tensor components, but past a larger critical threshold of $k \approx d^{3/2}$, the performance completely deteriorates.



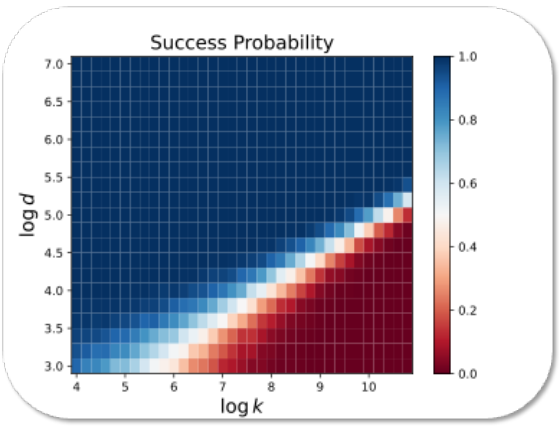
(a) $k = 400$ (b) $k = 1000$ (c) $k = 2000$

(d) $k = 4000$ (e) $k = 6000$ (f) $k = 8000$

(g) $k = 10000$ (h) $k = 12000$ (i) $k = 15000$

Figure 3: A graph of average correlation between algorithm output and the closest original tensor component against the number of tensor power method iterations. Each plot describes a $d = 400$ setting with varying ranks k. Randomly initialized runs are shown in various colors.

Since each matrix in $\mathbb{R}^{d\times d}$ has rank at most d, every order-3 tensor in $\mathbb{R}^{d\times d\times d}$ has at most rank d^2. This means that as long as $k \leq d^2$, a unique decomposition exists. However, it is conjectured that for $k \leq O(d^{3/2})$ a random initialization converges to one component with high probability, but past this threshold, it is strongly believed that there is a computational polynomial bottleneck [SV17].



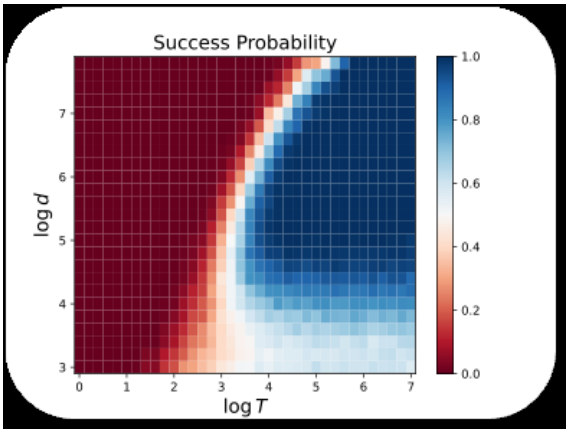


Figure 4: Success probability of tensor power iteration for varying k and d, with blue regions being high success, red regions low success, and white regions marking a $k = d^{\ell/2}$ threshold.

Figure 4 demonstrates an empirical phase transition of $k = d^{\ell/2}$ between high probability convergence and failure. The authors [WZ23] analyze the $\ell = 4$ case in particular, interpreting the white slope of $\log k = 2 \log d$ to suggest success for $\log k \ll 2 \log d$ and failure when $\log k \gg 2 \log d$. We also observe that polynomial steps are required for tensor power method convergence since $\log t$ scales linearly with $\log d$ near the success/failure boundary. There is also some work regarding overcomplete tensor decomposition when $k \gg d$ which claims that for slightly better than random initialization, the optimization landscape is benign [GM17].

4 Summary

We see that while Jennrich's algorithm is theoretically viable, it is generally ill-suited for practical applications, with iterative methods posing more robust alternatives. There is a strong connection between the behavior of such local maximizers and tensor decomposition as well as close relations between the iterative methods themselves. Ultimately, while our existing theory falls short of fully describing empirical behavior, it does set the stage for our following lecture on provable overcomplete decomposition using a smoothed analysis.

References

[GM17] Rong Ge and Tengyu Ma. On the optimization landscape of tensor decompositions, 2017.

[SV17] Vatsal Sharan and Gregory Valiant. Orthogonalized als: A theoretically principled tensor decomposition algorithm for practical use. In *International Conference on Machine Learning*, pages 3095–3104. PMLR, 2017.

[Unk21] Unknown. Will the real jennrich's algorithm please stand up?, 2021. https://www.mathsci.ai/post/jennrich/.

[WZ23] Yuchen Wu and Kangjie Zhou. Lower bounds for the convergence of tensor power iteration on random overcomplete models. In *The Thirty Sixth Annual Conference on Learning Theory*, pages 3783–3820. PMLR, 2023.