CS 224 Fall 2023 Scribes: Sara Kangaslahti
9/13

Lecture 3: Iterative Methods for Tensor Decomposition

1 Jennrich's Algorithm

In the previous lecture, we discussed Jennrich's algorithm, which can be used to decompose tensors

$$T = \sum_{i=1}^{k} \lambda_i u_i^{\otimes 3}$$

Where $u_1, ..., u_k$ are linearly independent unit vectors and we assume without loss of generality that $\lambda_1 \geq ... \geq \lambda_k \geq 0$. However, in practice, Jennrich's algorithm has two central issues:

1. It is not very noise robust compared to other algorithms [Jen, SV17]
2. Its runtime is dominated by dense matrix operations. Although the tensor is of dimension d^3, in applications, we often don't need to write down the full tensor, just need to know how it acts on individual vectors. As a result, dense matrix operations significantly constrain the runtime of the algorithm.

   As an example, if $T = \mathbb{E}[x^{\otimes 3}]$, we just need to compute

   $$M_z = T(:, :, z) = \mathbb{E}[\langle x, z\rangle xx^T]$$

   This can be done in $O(d^2)$ operations, so the runtime is bottlenecked by dense matrix operations.

Therefore, in practice, people use heuristics based on the iterative algorithms defined in Section 2

2 Iterative Algorithms

In this section, we assume that $T = \sum_{i=1}^{k} \lambda_i u_i^{\otimes 3}$ for orthonormal vectors $u_1, ..., u_k$. We will justify and remove this orthogonality assumption in Section 2.4

We note from the previous lecture that for any tensor, we can associate the polynomial

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

We note that in the matrix case, we could just compute the eigenvectors of T, but for worst-case tensors, this is NP-hard.

However, since our optimization problem is $p(x) = \sum_i \lambda_i \langle u_i, x \rangle^3$,

- If $x = u'$ s.t. $\langle u_i, u' \rangle \approx 0$, then $p(u') = \sum_i \lambda_i \langle u_i, u' \rangle \approx 0$
- If $x = u_i$, then $p(u_i) = \sum_j \lambda_j \langle u_j, u_i \rangle = \sum_j \lambda_j \mathbb{1}(i = j) = \lambda_i >> 0$

This intuition indicates that for vectors $x = u_i$, the value $p(x)$ is large, so the eigenvalues are likely optimizers for $p(x)$.

We can in fact show that the local maximizers of p are precisely $u_1, ..., u_k$. As a result, the tensor decomposition problem is equivalent to optimizing the associated polynomial.

2.1 Gradient descent

2.1.1 Optimization Problem

We consider the optimization problem for the polynomial associated with T:

$$\max_{\|x\|=1} p(x) = \max_{\|x\|=1} \sum_{a,b,c} T_{abc} x_a x_b x_c$$

Thus, by computing the gradient, we can derive the gradent ascent as follows:

$$\begin{aligned} x^{t+1} &= x^t + \eta \cdot \nabla p(x) \\ &= x^t + 3\eta \cdot T(:, x, x) \end{aligned}$$

However, we note that this gradient descent does not follow the constraint $\|x\| = 1$. Therefore, we need to ensure that x remains on the unit sphere after each gradient ascent step.

2.1.2 Riemannian Gradient Descent

We could solve this problem by directly projecting x onto the unit sphere as follows:

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

However, doing a simple projection onto the unit sphere causes all of the movement of the gradient from x^t in the direction directly away from the center of the circle to be "wasted". Therefore, we instead first project to the tangent space (the tangent line to x^t on the unit circle) and then project to the unit circle. To compute this projection, if we let $\Pi = \text{Id} - x^t(x^t)^T$ denote the projection to the tangent space, we compute:

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

We can simplify x^{t+1} by substituting in for Π:

$$\begin{aligned} x^{t+1} &= \text{proj}(x^t + 3\eta \cdot \Pi \cdot T(:, x, x)) \\ &= \text{proj}(x^t + 3\eta \cdot (\text{Id} - x^t(x^t)^T) \cdot T(:, x, x)) \\ &= \text{proj}(x^t + 3\eta \cdot [T(:, x^t, x^t) - x^t \cdot T(x^t, x^t, x^t)]) \\ &= \text{proj}(x^t + 3\eta \cdot [T(:, x^t, x^t) - x^t \cdot p(x^t)]) \end{aligned}$$

We note that a good choice of step size is $\eta = \frac{1}{3p(x^t)}$. This is because this step size results in step size increasing if the objective decreases and vice versa, and because it leads to a nice cancellation:

$$x^{t+1} = \text{proj}\left(\frac{T(:, x^t, x^t)}{p(x^t)}\right) = \text{proj}(T(:, x^t, x^t))$$

2.2 Tensor Power Method

We know that:

$$x^{t+1} = \text{proj}(T(:, x^t, x^t))$$

We can use this result and generalize the classic matrix power method to tensors.

2.2.1 Matrix Power Method

If T were the matrix $T = \sum_{i=1}^k \lambda_i u_i u_i^T$, then given $x = \sum_i c_i \cdot u_i$, we have

$$T(:, x) = Tx = \sum_i \lambda_i c_i \cdot u_i$$

so

$$\text{proj}(T(:, x)) = \text{proj}\left(\sum_i \lambda_i c_i \cdot u_i\right) = \sum_i \frac{\lambda_i c_i}{(\Sigma_j \lambda_j^2 c_j^2)^{1/2}} \cdot u_i$$

Thus, we went from the coefficients $(c_1, ..., c_k)$ to

$$\text{proj}(\lambda_1 c_1, ..., \lambda_k c_k)$$

We suppose WLOG $\lambda_1 \geq ... \geq \lambda_k$. Then, at each step, the first coordinate gets weighted more than all other coordinates, so the coordinates will converge to $(1, 0, ..., 0)$, yielding the top eigenvector. Formally, this is because the ratio between the ith and 1st coordinate starts at $\frac{c_i}{c_1}$ and is multiplied by λ_i/λ_1 at each round, resulting in exponential "linear" scale.

2.2.2 Generalizing the Power Method to Tensors

For tensors, the analysis is anologous, but we get even faster convergence than in the matrix case. This is because if $x = \sum_i c_i \cdot u_i$, then we have

$$T(:, x, x) = \sum_i \lambda_i \langle x, u_i \rangle^2 u_i = \sum_i \lambda_i c_i^2 \cdot u_i$$

Thus,

$$\text{proj}(T(:, x, x)) = \text{proj}\left(\lambda_i c_i^2 \cdot u_i\right) = \sum_i \frac{\lambda_i c_i^2}{(\Sigma_j \lambda_j^2 c_j^4)^{1/2}} \cdot u_i$$

Thus, we went from the coefficients $(c_1, ..., c_k)$ to

$$\text{proj}(\lambda_1 c_1^2, ..., \lambda_k c_k^2)$$

We suppose WLOG $\lambda_1 \geq ... \geq \lambda_k$. Then, the ratio between the ith and 1st coordinate starts at $\frac{c_i}{c_1}$ and is multiplied by $\lambda_i c_i/\lambda_1 c_1$ at each round. We note that this need not decay if $\lambda_i c_i > \lambda_1 c_1$. However, if our initial $(c_1, ..., c_k)$ are such that

$$\rho = \max_i \frac{\lambda_i c_i}{\lambda_1 c_1} < 1$$

Then in the next step ρ becomes

$$\max_i \frac{\lambda_i(\lambda_i c_i^2)}{\lambda_1(\lambda_1 c_1^2)} = \max_i \left(\frac{\lambda_i c_i}{\lambda_1 c_1}\right)^2 = \rho^2 < 1$$

Thus, the convergence depends on the initialization, but if the c_is are initialized such that $\rho < 1$, then the ratios decay at a doubly exponential rate since ρ is squared at each iteration. Naively, this initialization happens with probability at least $1/k$, but we could also use our derivation to argue that we converge to whichever u_i maximizes $\lambda_i c_i$.

2.2.3 Finding all of the components

We have shown how to converge to the top component, but the remaining components may be computed by one of two strategies:

1. "Deflation", ie take the vector $\hat{u} \approx u_i$ that we converged to, note that $\lambda_i = p(u_i)$, and recurse by finding

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

   However, it is difficult to handle the compounding errors caused by successive deflations in this method.

2. Run tensor power method on many random initializations to get many $\hat{u}$ vectors, cluster them, and get a set of estimates.

2.3 Alternating Least Squares (ALS)

ALS is a popular algorithm that can learn all of the components at once. In ALS, given the current iterates $\{u_i^t\}$, we consider the optimization problem

$$u^{t+1} = \min_{\hat{u}_i} \left\| T - \sum_{i=1}^{k} \hat{u}_i \otimes u_i^t \otimes u_i^t \right\|_F^2$$

We note that ALS is just a least-squares regression problem. It is quite hard to analyze rigorously, but very powerful in practice. Furthermore, the tensor power method can be interpreted as a "rank-1" version of ALS.

2.4 Analysis of the Orthogonality Assumption

In this section, we show why the initial orthogonality assumption that we made is reasonable.

2.4.1 Whitening

We can show that the orthogonality assumption is reasonable using whitening. We can do this in many applications of tensor decomposition in which we get access to not only $T = \sum_i \lambda_i u_i^{\otimes 3}$, but also to

$$M = \sum_i \lambda_i u_i u_i^T$$

We assume that the u_is are linearly independent, but not necessarily orthogonal. Then, we can use M to whiten the data so that $u_1, ..., u_k$ become orthogonal.

We note that we can write M as $M = VDV^T$, where $V \in \mathbb{R}^{d \times k}$ and $D \in \mathbb{R}^{k \times k}$ is a diagonal matrix with the eigenvalues on the diagonal. Then, we let $W = VD^{-1/2} \in \mathbb{R}^{d \times k}$ and $\tilde{u}_i = \lambda_i W^T u_i$. Then, we show that W standardizes the data as follows:

$$\begin{aligned}
W^T M W &= D^{-1/2} V^T V D V^T V D^{-1/2} \\
&= D^{-1/2} D D^{-1/2} \\
&= \mathrm{Id}_k \\
&= \sum_i \lambda_i (W^T u_i)(W^T u_i)^T \\
&= \sum_i \tilde{u}_i \tilde{u}_i^T
\end{aligned}$$

Thus, the $\tilde{u}_i$ values are orthogonal because $\sum_i \tilde{u}_i \tilde{u}_i^T = \mathrm{Id}_k$. Then, if we let $T' = T(W, W, W) \in \mathbb{R}^{k \times k \times k}$,

$$\begin{aligned}
T'(x, y, z) &= T(Wx, Wy, Wz) \\
&= \sum_i \lambda_i \langle Wx, u_i \rangle \langle Wy, u_i \rangle \langle Wz, u_i \rangle \\
&= \sum_i \lambda_i^{-1/2} \langle \tilde{u}_i, x \rangle \langle \tilde{u}_i, y \rangle \langle \tilde{u}_i, z \rangle
\end{aligned}$$

Thus, $T' = \sum_i \lambda_i^{-1/2} \tilde{u}_i^{\otimes 3}$. As a result, we have reduced linearly independent u_is to the orthogonal case, and so we can assume that the u_is are orthogonal in the iterative methods above.

2.4.2 Case with no Whitening

If we cannot whiten the u_is, then they are only linearly independent and analyzing the iterative tensor methods becomes significantly more challenging:

Theorem 1. *[SV17] Given $T = \sum_{i=1}^k u_i^{\otimes 3}$ for "incoherent" unit vectors $u_1, ..., u_k$, ie satisfying*

$$|\langle u_i, u_j \rangle| \leq c_{\max} \leq \frac{1}{k^{1+\epsilon}}$$

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

Some additional results are the following:

- Conjecture [SV17]: If $u_1, ..., u_k$ are random unit vectors and $k \leq O(d^{3/2})$, then tensor power method/gradient descent/ALS converges from random initialization to one of the components with high probability.

- In the overcomplete tensor decomposition case (when $k >> d$), if you initialize at a point slightly better than random initialization, then the optimization landscape is benign [GM17].

There have been numerous works that show that the theory is still very far from explaining the empirical behavior of the tensor power method [SV17, WZ22].

References

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

[Jen] Will the real jennrich's algorithm please stand up? Accessed: 2023-09-18.

[SV17] Vatsal Sharan and Gregory Valiant. Orthogonalized als: A theoretically principled tensor decomposition algorithm for practical use, 2017.

[WZ22] Yuchen Wu and Kangjie Zhou. Lower bounds for the convergence of tensor power iteration on random overcomplete models, 2022.