Sampling

1 Introducing

Consider a N-order tensor, $\mathcal{X}$ with size $L_1 \times L_2 \times \ldots \times L_N$, the CP decomposition of this tensor is

$$\mathcal{X} = \left[\!\left[\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)} \right]\!\right] = \sum_{r=1}^{R} \boldsymbol{a}_r^{(1)} \circ \cdots \circ \boldsymbol{a}_r^{(N)} \tag{1}$$

where

$$\begin{aligned} \mathbf{A}^{(1)} &= \left[\boldsymbol{a}_1^{(1)}, \boldsymbol{a}_2^{(1)}, \cdots, \boldsymbol{a}_r^{(1)} \right] \in R^{L_1 \times R} \\ \mathbf{A}^{(2)} &= \left[\boldsymbol{a}_1^{(2)}, \boldsymbol{a}_2^{(2)}, \cdots, \boldsymbol{a}_r^{(2)} \right] \in R^{L_2 \times R} \\ &\vdots \\ \mathbf{A}^{(N)} &= \left[\boldsymbol{a}_1^{(N)}, \boldsymbol{a}_2^{(N)}, \cdots, \boldsymbol{a}_r^{(1)} \right] \in R^{L_N \times R} \end{aligned}$$

In equation 1, $\mathbf{A}^{(n)}$ is the factor matrix, and the column size R represents the rank of this tenors, which means that $\mathcal{X}$ can be represented by a sum of these R rank one tensors. The vector $\boldsymbol{a}_r^{(n)}$ is the r-th column of matrix $\mathbf{A}^{(n)}$. In general, let $\boldsymbol{a}_{*r}^{(n)}$ be the k-th column of $\mathbf{A}^{(n)}$, $\boldsymbol{a}_{i_n *}^{(n)}$ be the i_n-th row vector of $\mathbf{A}^{(n)}$, and $a_{i_n k}^{(n)}$ be the element of $\mathbf{A}^{(n)}$.

The element in $\mathcal{X}$ satisfies the following equation:

$$x_{\boldsymbol{i}} = \sum_{r=1}^{R} a_{i_1 r}^{(1)} \cdot a_{i_2 r}^{(2)} \cdots a_{i_N r}^{(N)} \tag{2}$$

The indexes vector $\boldsymbol{i}$ is a shorthand for multi-index $(i_1, i_2, \ldots, i_N)$. We propose two methods, which are wedge sampling and diamond sampling, for estimating the maximum elements $x_{\boldsymbol{i}}$ in tensor when factor matrices $\mathbf{A}^{(n)}, n = 1, 2, \ldots, N$ are given. The wedge sampling is so much like the diamond sampling, and the diamond sampling an improvement of wedge sampling.

2 Diamond Sampling

The diamond sampling is a way to sample the multi-index $\boldsymbol{i} : (i_1, i_2, \ldots, \imath_N)$ proportional to $x_{\boldsymbol{i}}^2$. We consider a matrix A with size $m \times n$ as a weighted bipartite graph G_A. Meanwhile, in the adjacent matrix of the graph we use 0 instead of ∞ to represent the two vertices are not adjacent. So the adjacent matrix of G_A is:

$$\begin{pmatrix} \mathbf{0}_{m \times m} & A \\ A^T & \mathbf{0}_{n \times n} \end{pmatrix}$$

Notation	Explanation
$\mathcal{A}$	tensor
$\mathbf{A}$	matrix
$\mathbf{A}^{(n)}$	n-th factor matrix of tensor
$\boldsymbol{a}_{*r}, \boldsymbol{a}_r$	k-th column of matrix
$\boldsymbol{a}_{i*}$	i-th row of matrix
$\boldsymbol{a}$	vector
a_{ir}	element of matrix

Table 1: Notation

2.1 Graph representation

In equation 1, we have N factor matrices $\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}$. Those N factor matrices are represented by a weighted $(N+1)$-partite graph G_T. And we call those $N+1$ partitions as $\overline{V}, V_1, V_2, \ldots, V_N$. Partition V_n has L_n vertexes and partition $\overline{V}$ has R vertexes which is the same size of tensor rank. We call $v_{i_n}^n$ the i_n-th vertex in partition V_n, and $\overline{v}_r$ the r-th vertex in partition $\overline{V}$. Every two vertexes from different partition classes $V_i, V_j, i, j \in 1, 2, \ldots, N$ are not adjacent. A vertex $\overline{v}_r$ in $\overline{V}$ is only adjacent to vertexes v_i^n in V_n when $a_{i_n r}^{(n)}$ is non-zero with the edge weight $a_{i_n r}^{(n)}$. And the adjacent matrix of G_T is :

$$\begin{pmatrix} \mathbf{0}_{R\times R} & {\mathbf{A}^{(1)}}^T & \ldots & {\mathbf{A}^{(N)}}^T \\ \mathbf{A}^{(1)} & \mathbf{0}_{L_1\times L_1} & \ldots & \mathbf{0}_{L_1\times L_N} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}^{(N)} & \mathbf{0}_{L_N\times L_1} & \ldots & \mathbf{0}_{L_1\times L_N} \end{pmatrix}$$

Under the graph presentation, we define an event $\varepsilon_{\boldsymbol{i},r',r}$: phase 1. Pick an edge $e = (v_{i_1}^1, \overline{v}_r)$; phase 2. Walk $N-1$ times from the $\overline{v}_r$ to other partitions $V_2, \ldots, V_N$, arrive at $v_{i_2}^2, v_{i_3}^3, \ldots, v_{i_N}^N$; phase 3. Walk from $v_{i_1}^1$ to $\overline{V}$ and end in $\overline{v}'_r$. After this event happened, we give a score according to indexes $\boldsymbol{i} : (i_1, i_2, \ldots, i_N)$. We assign each phase a probability so that the final score of $\boldsymbol{i} : (i_1, i_2, \ldots, i_N)$ will be a good estimation of $x_{\boldsymbol{i}}$.

2.2 Probability of picking edge and walking to other partition

- Walking with probability

  In phase 2 we start from a vertex in $\overline{V}$ to V_i, and phase 3 from a vertex in V_1 to $\overline{V}$. We choose the path(edge) that we walk through according to its weight, which is given i_1, pick $r \in \{1, 2, \ldots, R\}$ with probability $|a_{i_1 r'}^{(1)}| / \parallel \boldsymbol{a}_{i_1 *}^{(1)} \parallel_1$ or given r, pick $i_n \in \{1, 2, \ldots, L_n\}$ with probability $|a_{i_n r}^{(n)}| / \parallel \boldsymbol{a}_{*r}^{(n)} \parallel_1$.

- Picking an edge

  When we pick an edge $e = (v_{i_1}^1, \overline{v}_r)$, we are picking the vertexes pair $(v_{i_1}^1, \overline{v}_r)$. Firstly, we assign pair a probability, if $a_{i_1 k}^{(1)} \neq 0$,

  $$p(i_1, r) = \mid a_{i_1 k}^{(1)} \mid \parallel \boldsymbol{a}_{i_1 *}^{(1)} \parallel_1 \parallel \boldsymbol{a}_{*r}^{(1)} \parallel_1 \parallel \boldsymbol{a}_{*r}^{(2)} \parallel_1 \ldots \parallel \boldsymbol{a}_{*r}^{(N)} \parallel_1 \; / \parallel \mathbf{W} \parallel_1$$

  Where

  $$\parallel \mathbf{W} \parallel_1 = \sum_{i_1, r} \mid a_{i_1 k}^{(1)} \mid \parallel \boldsymbol{a}_{i_1 *}^{(1)} \parallel_1 \parallel \boldsymbol{a}_{*r}^{(1)} \parallel_1 \parallel \boldsymbol{a}_{*r}^{(2)} \parallel_1 \ldots \parallel \boldsymbol{a}_{*r}^{(N)} \parallel_1$$

  Then we pick the pair according the probability $p(i_1, r)$.

2.3 Scoring samples

We do s time samples, and for each sample we will get an coordinate $\boldsymbol{i} = (i_1, i_2, \ldots, 1_N)$. If this coordinate has not been sampled previously, let the score in the ℓ-th turn be $\mathbf{X}_{\boldsymbol{i},\ell} = sgn(a^{(1)}_{i_1 r} \cdot a^{(2)}_{i_2 r} \cdots a^{(N)}_{i_N r} \cdot a^{(1)}_{i_1 r'}) a^{(2)}_{i_2 r'} \cdots a^{(N)}_{i_N r'}$, and create a variable $\widehat{x}_{\boldsymbol{i}} = \mathbf{X}_{\boldsymbol{i},\ell}$. Otherwise, increase $\widehat{x}_{\boldsymbol{i}}$ by $\mathbf{X}_{\boldsymbol{i},\ell}$. For $\boldsymbol{i}$ that not be sampled in ℓ-th turn, let $\mathbf{X}_{\boldsymbol{i},\ell} = 0$. So

$$\widehat{x}_{\boldsymbol{i}} = \sum_{\ell} \mathbf{X}_{\boldsymbol{i},\ell}$$

Next, we will show that $\widehat{x}_{\boldsymbol{i}}$ will be a good estimation of $x_{\boldsymbol{i}}$ in scale.

2.4 Correctness and error bounds

As we defined previously, the event $\varepsilon_{\boldsymbol{i},r',r}$ is picking a pair $(v^1_{i_1}, \overline{v}_r)$ then pick paths from $V^1_{i_1}$ in V_1 to $\overline{V}$ and an addition path from $\overline{v}_r$ in $\overline{V}$ to V_1. And we think $\mathbf{X}_{\boldsymbol{i},\ell}, \ell \in \{1, 2 \ldots, s\}$ are independent. Under this assumption, we give two lemmas.

Lemma 2.1. *The expectation of $\widehat{x}_{\boldsymbol{i}}$ equals to $s \cdot x_{\boldsymbol{i}} / \parallel \mathbf{W} \parallel_1$.*

Proof: The final score $\widehat{x}_{\boldsymbol{i}} = \sum_{\ell=1} \mathbf{X}_{\boldsymbol{i},\ell}$. So that

$$\mathbb{E}[\widehat{x}_{\boldsymbol{i}}] = \mathbb{E}[\sum_{\ell=1} \mathbf{X}_{\boldsymbol{i},\ell}] = s\mathbb{E}[\mathbf{X}_{\boldsymbol{i},1}] \tag{3}$$

The probability of $\varepsilon_{\boldsymbol{i},r',r}$ is

$$\begin{aligned}
Pr(\varepsilon_{\boldsymbol{i},r',r}) &= Pr(\text{pick } (v^1_{i_1}, \overline{v}_r) \cdot \prod_{n=2}^{n=N} Pr(\text{pick } (v^n_{i_n}, \overline{v}_r) | \text{given } \overline{v}_r) \cdot Pr(\text{pick } (v^1_{i_1}, \overline{v}'_r) | \text{given } v^1_{i_1}) \\
&= \frac{w_{i_1 r}}{\parallel W \parallel_1} \cdot \frac{|a^{(1)}_{i_1 r'}|}{\parallel \boldsymbol{a}^{(1)}_{i_1 *} \parallel_1} \cdot \prod_{n=2}^{n=N} \frac{|a^{(n)}_{i_n r}|}{\parallel \boldsymbol{a}^{(n)}_{*r} \parallel_1} \\
&= \frac{|a^{(1)}_{i_1 r}| \parallel \boldsymbol{a}^{(1)}_{i_1 *} \parallel_1 \prod_{n=2}^{n=N} \parallel \boldsymbol{a}^{(n)}_{*r} \parallel_1}{\parallel W \parallel_1} \cdot \frac{|a^{(1)}_{i_1 r'}|}{\parallel \boldsymbol{a}^{(1)}_{i_1 *} \parallel_1} \cdot \prod_{n=2}^{n=N} \frac{|a^{(n)}_{i_n r}|}{\parallel \boldsymbol{a}^{(n)}_{*r} \parallel_1} \\
&= \frac{|a^{(1)}_{i_1 r'}| \prod_{n=1}^{n=N} |||a^{(1)}_{i_1 r'}|}{\parallel W \parallel_1}
\end{aligned}$$

We get the probability of one walk.

$$Pr(\varepsilon_{\boldsymbol{i},r',r}) = \frac{|a^{(1)}_{i_1 r'}| \prod_{n=1}^{n=N} |||a^{(1)}_{i_1 r'}|}{\parallel W \parallel_1} \tag{4}$$

Using equation in 3 and 4.

$$\begin{aligned}
\mathbb{E}[x_{\boldsymbol{i}}/s] &= \mathbb{E}[\mathbf{X}_{\boldsymbol{i},1}] \\
&= \sum_{r}\sum_{r'} Pr(\varepsilon_{\boldsymbol{i},r',r}) \cdot sgn(a^{(1)}_{i_1 r} \cdot a^{(2)}_{i_2 r} \cdots a^{(N)}_{i_N r} \cdot a^{(1)}_{i_1 r'}) a^{(2)}_{i_2 r'} \cdots a^{(N)}_{i_N r'} \\
&= \frac{1}{\parallel \mathbf{W} \parallel_1} \sum_{r}\sum_{r'} |a^{(1)}_{i_1 r}| \cdot |a^{(2)}_{i_2 r}| \cdots |a^{(N)}_{i_N r}| \cdot |a^{(1)}_{i_1 r'}| sgn(a^{(1)}_{i_1 r} \cdot a^{(2)}_{i_2 r} \cdots a^{(N)}_{i_N r} \cdot a^{(1)}_{i_1 r'}) a^{(2)}_{i_2 r'} \cdots a^{(N)}_{i_N r'} \\
&= \frac{1}{\parallel \mathbf{W} \parallel_1} \sum_{r}\sum_{r'} a^{(1)}_{i_1 r} \cdot a^{(2)}_{i_2 r} \cdots a^{(N)}_{i_N r} \cdot a^{(1)}_{i_1 r'} a^{(2)}_{i_2 r'} \cdots a^{(N)}_{i_N r'} \\
&= \frac{1}{\parallel \mathbf{W} \parallel_1} \{\sum_{r} a^{(1)}_{i_1 r} \cdot a^{(2)}_{i_2 r} \cdots a^{(N)}_{i_N r}\}^2 \\
&= \frac{x^2_{\boldsymbol{i}}}{\parallel \mathbf{W} \parallel_1}
\end{aligned}$$

□

Lemma 2.2. *Fix $\varepsilon > 0$ and error probability $\sigma \in (0,1)$. Assuming all entries in factor matrices are nonnegative and at most K. If the number of samples*

$$s \leq 3K^N \parallel \mathbf{W} \parallel_1 \log 2/\sigma/(\varepsilon^2 {x_{\boldsymbol{i}}}^2),$$

then

$$Pr[|\widehat{x}_{\boldsymbol{i}} \parallel \mathbf{W} \parallel_1 /s - {x_{\boldsymbol{i}}}^2| \geq \varepsilon {x_{\boldsymbol{i}}}^2] \leq \sigma$$

Theorem 2.1. *Fix some threshold τ and error probability $\sigma \in (0,1)$. Assume all entries in factor matrices are nonnegative and at most K. Suppose $s \geq 12K^N \parallel \mathbf{W} \parallel_1 \log(2L_1 L_2 \cdots L_N/\sigma)/\tau^2$. Then with probability at least $1-\sigma$, the following holds for all indexes $\boldsymbol{i} = (i_1, i_2, \ldots, i_N)$ and $\boldsymbol{i'} = (i'_1, i'_2, \ldots, i'_N)$: if $x_{\boldsymbol{i}} > \tau$ and $x_{\boldsymbol{i'}} < \tau/4$, then $\widehat{x}_{\boldsymbol{i}} > \widehat{x}_{\boldsymbol{i'}}$.*

2.5 Finding top-t largest value

2.6 Finding k-NN for query

At this situation, we use on row of matrix $\mathbf{A}^{(1)}$ each time called $\boldsymbol{u}$. And the tenor of N-1 order called rank tensor for query $\boldsymbol{u}$.

$$\mathcal{X}_{\boldsymbol{u}} = \left[\!\left[\boldsymbol{u}, \mathbf{A}^{(2)} \ldots, \mathbf{A}^{(N)}\right]\!\right] = \sum_{r=1}^{R} u_r \cdot \boldsymbol{a}^{(2)}_r \circ \cdots \circ \boldsymbol{a}^{(N)}_r \tag{5}$$

And find the k most relevant vector sets, each vector set consist N-1 vectors and the

The biggest difference between different queries in sampling processing is the probability for picking vertexes in partition $\overline{V}$, or the frequency sequence $(c_1, c_2, \ldots, c_R)$ of $\overline{v}_r$ to be sampled. Notice that, $\sum_{r=1}^{R} c_r = s$.

For effectively implementation, we use the lists $\boldsymbol{g}_1, \boldsymbol{g}_2, \ldots, \boldsymbol{g}_R$ to record the sub-path(walk from a vertex $\overline{v}_r$ in $\overline{V}$ to $V_i, i \in \{2, \ldots, N\}$.

2.6.1 Weight assigning and frequency generating

For each $1 \leq r \leq R$, let $u'_r \leftarrow| u_r \parallel\!| \boldsymbol{a}^{(1)}_{*r} \parallel_1 \| \boldsymbol{a}^{(2)}_{*r} \parallel_1 \ldots \| \boldsymbol{a}^{(N)}_{*r} \parallel_1$. Instead of sampling r directly s times, choosing the c_r randomly such that c_r has the expected value $su'_r/\parallel u' \parallel_1$ will still works.

$$c_r = \begin{cases} \lfloor su'_r/\parallel u' \parallel_1 \rfloor, & \text{with probability } \lceil su'_r/\parallel u' \parallel_1 \rceil - su'_r/\parallel u' \parallel_1 \\ \lceil su'_r/\parallel u' \parallel_1 \rceil, & \text{with probability } \lfloor su'_r/\parallel u' \parallel_1 \rfloor - su'_r/\parallel u' \parallel_1 \end{cases}$$

2.6.2 Sub-walks

When given some r, sampling method need to pick the left indexes $\boldsymbol{i'} = (i'_1, i'_2, \ldots, i'_N)$, which has been stored in the previous query, it saves a lot of computation.

- for $r = 1, 2, \ldots, R$:

  repeat c_r times: sample r' times with the probability $|u_r|/ \parallel u \parallel_1$. If the size of $\boldsymbol{g}_r$ is larger then c_r, use the sub-walk in $\boldsymbol{g}_r$ as the remaining result. Otherwise, sample the remaining.

- for $r = 1, 2, \ldots, R$:

-

3 Two Approaches

In this section, we introduce wedge sampling and the advanced edition diamond sampling.

3.1 Wedge Approach

A random-sampling based algorithm that identifies high-valued entries of nonnegative matrix products, by assigning scores to each entry of $\boldsymbol{q}^T\mathbf{A}$, where $\mathbf{A} \in R^{d\times n} = [\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_n]$ is a matrix whose rows are the instance, and d is the dimension of feature vector. Our task is to find the most relevant instance $\boldsymbol{a}_i$ that maximize dot product $\boldsymbol{q}^T\boldsymbol{a}_i$,

3.2 Graph representation

Consider a product $\mathbf{M} = \mathbf{QA}$, where $\mathbf{Q} \in R^{m\times d}, \mathbf{A} \in R^{d\times n}$ are represented by a layered graph with three layers, and treat matrix $\mathbf{Q}, \mathbf{A}$ as weighted directed bipartite graph. There are m nodes $\{v_1^1, v_2^1, \ldots, v_m^1\}$ in first layer, d nodes $\{v_1^2, v_2^2, \ldots, v_d^2\}$ in second layer , and n nodes $\{v_1^2, v_2^2, \ldots, v_d^2\}$in the third. For a query vector $\boldsymbol{q}^T$, see it as the i-th row of $\mathbf{Q}$, and $\boldsymbol{m} = \boldsymbol{q}^T\mathbf{A}$, then define a trace: random walks from the i-th node, which is the query vector $\boldsymbol{q}^T$ in first layer, and terminates at the j-th node in third layer. In this situation, the first layer has one node.

Walk S times randomly will terminate in third layer S times, and record the time node $j \in \{1, 2, \ldots, n\}$ has been reached to S_j, clearly $\sum_{j=1}^n S_j = S$. If $j \in \{1, 2, \ldots, n\}$ is select with probability $m_j/\sum_{h=1}^n m_h$, then the expected value of S_j equals $m_j S/\sum_{h=1}^n m_h$. Hence, $S_j \sum_{h=1}^n m_h/S$ is an unbiased estimator for m_j.

For every pair of nodes $\{v_k^2, v_j^3\}$, assign the probability to the edge

$$p(e_{kj}) = a_{kj}/\sum_{h=1}^{d} a_{kh}$$

Processing goes in Algorithm 1. Think the trace as first walk to node v_k^2, and then v_j^3. So firstly, compute the times c_k expected to go through node v_k^2, and sample c_k times to arrive v_j^3.

- The expect of c_k.

$$E[c_k] = \frac{Sq'_k}{\sum q'_k}$$

Algorithm 1 Wedge Sampling

Given matrix $\mathbf{A}$ and query$\boldsymbol{q}$

Let S be the number of samples.

1: Preprocessing stage
2: **for** k=1,...,d **do**
3: compute $a_k = \sum_{h=1}^{n} a_{kh}$ and $q'_k = q_k a_k$
4: **end for**
5: **for** k=1,...,d **do** determine the number of samples c_i needed for
6:

$$c_i = \begin{cases} \lfloor \frac{Sq'_k}{\sum q'_k} \rfloor, & \text{w/prob. } \lceil \frac{Sq'_k}{\sum q'_k} \rceil - \frac{Sq'_k}{\sum q'_k} \\ \lceil \frac{Sq'_k}{\sum q'_k} \rceil, & \text{w/prob. } \lfloor \frac{Sq'_k}{\sum q'_k} \rfloor - \frac{Sq'_k}{\sum q'_k} \end{cases}$$

7: **end for**
8: **for** $k = 1, \ldots, d$ **do**
9: Sample $j \in \{1, \ldots, n\}$ c_k times with probability $p(e_{kj}) = a_{kj} / \sum_{h=1}^{n} a_{kh}$.
10: Increase the counter of S_j.
11: **end for**
12: Postprocessing

- The expect of S_j.

$$\begin{aligned} E[S_j] &= \sum_{k=1}^{d} c_k p(e_{kj}) \\ &= \sum_{k=1}^{d} \frac{S q_k a_k}{\sum q_k a_k} \frac{a_{kj}}{a_k} \\ &= \frac{S}{\sum q_k a_k} \sum_{k=1}^{d} q_k a_{kj} \\ &= \frac{S m_j}{\sum q_k a_k} = \frac{S m_j}{\sum m_h} \end{aligned}$$

As we can see, $S_j \sum_{h=1}^{n} m_h / S$ is an unbiased estimator for m_j.

3.3 Implementing details for the wedge sampling

To find the maximum of $\mathbf{M} = \mathbf{QA}$, see $\mathbf{Q}$ as a bunch of query vector $\boldsymbol{q_i}$. Replace the matrix $\mathbf{A}$ by lists $L_1, L_2, \ldots, L_d$ of indices $\{1, 2, \ldots, n\}$. For each input query $\boldsymbol{q}$, the algorithm first processes $\boldsymbol{q}$ to determine how many samples are needed for each of the d lists L_k ,or the times c_k expected to go through node v_k^2. It then retrieves these samples from the appropriate stored lists and counts how many times each of $\{1, 2, \ldots, n\}$ was sampled. It saves the time for repeating sampling of different query in line 9 of Algorithm 1.

3.4 Diamond Approach

For $\boldsymbol{m} = \boldsymbol{q}^T \mathbf{A}$, the diamond sampling aim to find two intersecting wedges, so that the probability to terminate at $j = \{1, 2, \ldots, n\}$ is proportional to m_j^2.

4 Sampling in Tensor

We can consider the problem as finding some pair of vectors that from two different bunch of vector set, in previous, to have the the top-t maximum dot products.

Now, we extend the problem from two vector set to N vector sets. And find a list of vectors $\{\boldsymbol{a}_{i_1}^{(1)}, \ldots, \boldsymbol{a}_{i_N}^{(N)}\}$, where $\boldsymbol{a}_{i_n}^{(n)}$ is a vector from the n-th vector set $\mathbf{A}^{(n)}$ and $i_n \in \{1, \ldots, L^{(n)}\}$, according to the value of

$$\sum_{k=1}^{R} a_{i_1 k}^{(1)} a_{i_2 k}^{(2)} \cdots a_{i_N k}^{(N)}$$

A more mathematic interpretation is: given factor matrixes $\mathbf{A}^{(n)} \in R^{L^{(n)} \times R}, n = \{1, 2, \ldots, N\}$, which is the CP decomposition of tensor $\boldsymbol{\mathcal{X}}$. Find the top-t value in tensor $\boldsymbol{\mathcal{X}}$.

4.1 Wedge Sampling in Tensor

We separated those matrixes into three lapsers, the first layer has L^1 nodes, indicated by the first factor matrix $\mathbf{A}^{(1)}$. And the second layer called the middle layer has R nodes, which picture the feature dimension or the rank of CP decomposition. The layer has $N-1$ parallel layers, each layer has L^n nodes.Under this assumption, we define a trace: walk from the first layers at node $v_{i_1}^1$ to middle layer v_k^m , then terminates at each parallel layers with nodes $\{v_{i_2}^2, v_{i_3}^3, \ldots, v_{i_N}^N\}$.

As similar, define the probability of $\{v_k^m, v_{i_n}^n\}$:

$$P(e_{k i_n}) = a_{i_n k}^{(n)} / \sum_{h=1}^{R} a_{i_n h}^{(n)}$$

4.1.1 Find the most relevant vector lists of query q

Given a query $\boldsymbol{q}$, find the most relevant vector lists. It is the same when $\mathbf{A}^{(1)}$ is a vector or see $\boldsymbol{q}$ as one column of $\mathbf{A}^{(1)}$, since we define the row of factor matrix is the number of instances.

- Preprocessing
  $q'_k = q_k \parallel \boldsymbol{a}_{*r}^{(2)} \parallel_1 \ldots \parallel \boldsymbol{a}_{*r}^{(N)} \parallel_1$, where $\parallel \boldsymbol{a}_{*r}^{(n)} \parallel_1 = \sum_{h=1}^{R} a_{i_n h}^{(n)}$.
- Compute the times c_k expected to go through node v_k^m

$$c_i = \begin{cases} \lfloor \frac{S q'_k}{\sum q'_k} \rfloor, & \text{w/prob. } \lceil \frac{S q'_k}{\sum q'_k} \rceil - \frac{S q'_k}{\sum q'_k} \\ \lceil \frac{S q'_k}{\sum q'_k} \rceil, & \text{w/prob. } \lfloor \frac{S q'_k}{\sum q'_k} \rfloor - \frac{S q'_k}{\sum q'_k} \end{cases}$$

  As we can see

$$E[c_k] = \frac{S q'_k}{\sum q'_k}$$

- Sample each node in parallel layers c_k times with probability $P(e_{k i_n})$, and increase the counter of $S_{i_2, \ldots, i_N}$

The expect of $S_{i_2,\ldots,i_N}$.

$$\begin{aligned} E[S_{i_2,\ldots,i_N}] &= \sum_{k=1}^{R} c_k p(e_{ki_2}) p(e_{ki_3}) \cdots p(e_{ki_N}) \\ &= \frac{S}{\sum q'_k} \sum_{k=1}^{R} q_k \cdot a^{(2)}_{i_2 k} \cdots a^{(N)}_{i_N k} \\ &= \frac{S m_{i_2,\ldots,i_N}}{\sum q'_k} \\ &= \frac{S m_{i_2,\ldots,i_N}}{\sum m_{i_2,\ldots,i_N}} \end{aligned}$$

So the $S_{i_2,\ldots,i_N} \sum m_{i_2,\ldots,i_N} / S$ is an unbiased estimator for $m_{i_2,\ldots,i_N}$.

4.1.2 Find the maximum value in tensor

The counter can only indicate the relatively relation of one vector list to query, which can not be used to compare different query. So we need to consider the value overall. Query a single vector , however, is still useful in some application like recommendation system, for at most time we only need to find the most relevant item for on user(query), and the global maximum is useless and time consuming. The trick of replace data instance by a list of indices used in pervious is also helpful when we consider a number of user at a time.

To find the maximum is tensor $\mathcal{X}$, the diamond sampling supply referenced method.

- ***Preprocessing***
  For every i_1, k compute the weight
  $w_{i_1 k} = \mid a^{(1)}_{i_1 k} \mid \mid\mid \boldsymbol{a}^{(2)}_{*r} \parallel_1 \ldots \parallel \boldsymbol{a}^{(N)}_{*r} \parallel_1$, where $\parallel \boldsymbol{a}^{(n)}_{*r} \parallel_1 = \sum_{h=1}^{R} a^{(n)}_{i_n h}$.
- ***Random walk to v^m_k.***
  Sample S times with the probability $w_{i_1 k} / \parallel \mathbf{W} \parallel_1$
- Sample each node in parallel layers
  Given k sample $i_2, \ldots, i_N$ with probability $P(e_{ki_n})$, and increase the value of $c_{i_1,i_2,\ldots,i_N}$ by $a^{(2)}_{i_2 k} \cdots a^{(N)}_{i_N k}$
  The expect of $x_{i_1,i_2,\ldots,i_N}$.

$$\begin{aligned} E[x_{i_1,i_2,\ldots,i_N}] &= \sum_{k=1}^{R} p(i_1, k) p(e_{ki_2}) p(e_{ki_3}) \cdots p(e_{ki_N}) \\ &= \frac{S}{\parallel \mathbf{W} \parallel_1} \sum_{k=1}^{R} a^{(1)}_{i_1 k} \cdot a^{(2)}_{i_2 k} \cdots a^{(N)}_{i_N k} \\ &= \frac{S}{\parallel \mathbf{W} \parallel_1} x_{i_1,i_2,\ldots,i_N} \end{aligned}$$

So the $c_{i_1,i_2,\ldots,i_N} \frac{S}{\|\mathbf{W}\|_1}$ is an unbiased estimator for $x_{i_2,\ldots,i_N}$.

4.2 Diamond Sampling in Tensor

Diamond sampling is the advanced edition of wedge sampling, which can deal with the negative value.

4.2.1 Find the maximum value in tensor

A direct extension for diamond sampling in tensor is stated in Algorithm 2.

Algorithm 2 Diamond Sampling with factor matrixes

Given factor matrix $\mathbf{A}^{(n)}, n = 1, 2, \ldots, N$.
Let s be the number of samples.

for all $a^{(1)}_{i_1 k} \neq 0$ **do**
  $w_{i_1 k} \leftarrow\mid a^{(1)}_{i_1 k} \mid\mid\mid \boldsymbol{a}^{(1)}_{i_1 *} \parallel_1 \parallel \boldsymbol{a}^{(1)}_{*r} \parallel_1 \parallel \boldsymbol{a}^{(2)}_{*r} \parallel_1 \ldots \parallel \boldsymbol{a}^{(N)}_{*r} \parallel_1$
end for
$\mathcal{X} \leftarrow$ all-zeros tensor of size $L^{(1)} \times L^{(2)} \ldots \times L^{(N)}$
for $\ell = 1, \ldots, s$ **do**
  Sample (i_1, k) with probability $w_{i_1 k} / \parallel \mathbf{W} \parallel_1$
  for $n = 2, ..., N$ **do**
    Sample i_n with probability $|a^{(n)}_{i_n k}| / \parallel \boldsymbol{a}^{(n)}_{*r} \parallel_1$
  end for
  Sample k' with probability $|a^{(1)}_{i_1 k'}| / \parallel \boldsymbol{a}^{(1)}_{i_1 *} \parallel_1$
  $x_{i_1, i_2, \cdots, i_N} \leftarrow x_{i_1, i_2, \cdots, i_N} + sgn(a^{(1)}_{i_1 k} \cdot a^{(2)}_{i_2 k} \cdots a^{(N)}_{i_N k} \cdot a^{(1)}_{i_1 k'}) a^{(2)}_{i_2 k'} \cdots a^{(N)}_{i_N k'}$
end for
Postprocessing

4.2.2 Find the most relevant vector lists of query u

Consider query $\boldsymbol{u}$ is one column of $\mathbf{A}^{(1)}$, and the retrieve tensor

$$\mathcal{X}_u = \left[\!\left[\boldsymbol{u}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)} \right]\!\right] = \sum_{r=1}^{R} \boldsymbol{a}^{(2)}_r \circ \cdots \circ \boldsymbol{a}^{(N)}_r \circ u_r$$

We can use the counter to indicate the value, but is will not deal with the negative value, so a degenerate method of diamond sampling was proposed. The way to find the maximum of $x_{u,\boldsymbol{i}}$ may be more concise when sampling starts with $\boldsymbol{u}$ as we can see in line 2 of Algorithm 3.

Algorithm 3 Diamond Sampling with a query vector

Given factor matrix $\mathbf{A}^{(n)} \in R^{L^{(n)} \times R}, n = 2, \ldots, N, \boldsymbol{u} \in R^{1 \times R}$.
Let s be the number of samples.

1: **for** all $u_k \neq 0$ **do**
2: $w_k \leftarrow\mid u_k \mid\mid\mid \boldsymbol{u} \parallel_1 \parallel \boldsymbol{a}^{(2)}_{*r} \parallel_1 \parallel \boldsymbol{a}^{(3)}_{*r} \parallel_1 \ldots \parallel \boldsymbol{a}^{(N)}_{*r} \parallel_1$
3: **end for**
4: $\mathcal{X}_u \leftarrow$ all-zeros tensor of size $L^{(2)} \times L^{(3)} \ldots \times L^{(N)}$
5: **for** $\ell = 1, \ldots, s$ **do**
6: Sample k with probability $w_k / \parallel \boldsymbol{w} \parallel_1$
7: **for** $n = 2, ..., N$ **do**
8: Sample i_n with probability $|a^{(n)}_{i_n k}| / \parallel \boldsymbol{a}^{(n)}_{*r} \parallel_1$
9: **end for**
10: Sample k' with probability $\mid u_{k'} \mid / \parallel \boldsymbol{u} \parallel_1$
11: $x_{i_2, i_3, \cdots, i_N} \leftarrow x_{i_2, i_3, \cdots, i_N} + sgn(a^{(2)}_{i_2 k} \cdot a^{(3)}_{i_3 k} \cdots a^{(N)}_{i_N k} \cdot u_k \cdot u_{k'}) a^{(2)}_{i_2 k'} a^{(3)}_{i_3 k'} \cdots a^{(N)}_{i_N k'}$
12: **end for**
13: Postprocessing

5 Implementation issues

****

5.1 Determined Sample Numbers Online

Pre-sample a number of instance, and it can be used with the accuracy bound to determine a reasonable number of samples.

5.2 Dealing with query

When we want to find the maximus relevant vectors for a number of queries. It is effective to save the the sampled instances into lists $L_1, L_2, \ldots, L_d$. When processes a query, sampled the nodes in middle layer. It then retrieves these samples form the stored lists.

6 Experiments

****

---

7 Sampling in Fashion Recommendation

7.1 Structure of U and $A^{(n)}$

Using sampling to solve a fashion recommendation problem.

$$r_{u,\mathfrak{O}_t} = \sum_{n=1}^{N} \boldsymbol{u}^{(n)} \cdot \boldsymbol{h}^{(n,n)}(\boldsymbol{x}_{i_n}) + \sum_{n=1}^{N} \sum_{m=n+1}^{N} \boldsymbol{h}^{(n,m)}(\boldsymbol{x}_{i_n}) \cdot \boldsymbol{h}^{(m,n)}(\boldsymbol{x}_{i_m})$$

Given a user $\boldsymbol{u}^{(n)} \in R^D$, and the latent space vector $\boldsymbol{h}^{(n,m)}(\boldsymbol{x}_{i_n}) \in R^D$, we need to find the maximum value in tensor $r_{u,\mathfrak{O}_t}$, which is the outfit most be like by this user. To do so, we need to reform the vectors into factor matrixes before hand.

We introduce a matrix called $Matrix(\boldsymbol{a}^{(n)}_{i_n*})$ with size $N \times ND$, which is the reshaped matrix of vector $\boldsymbol{a}^{(n)}_{i_n*} \in R^{N^2D}$.

$$Matrix(\boldsymbol{a}_{i_n*}^{(n)}) = \begin{matrix} \\ i_n\text{-th row} \end{matrix} \begin{matrix} \overset{\qquad\qquad\qquad\qquad i_n\text{-th column}\qquad\qquad\qquad\qquad}{} \\ \begin{pmatrix} \boldsymbol{e} & \boldsymbol{e} & \cdots & \frac{1}{2}\boldsymbol{h}^{(n,1)}(\boldsymbol{x}_{i_n}) & \cdots & \boldsymbol{e} \\ \boldsymbol{e} & \boldsymbol{e} & \cdots & \frac{1}{2}\boldsymbol{h}^{(n,2)}(\boldsymbol{x}_{i_n}) & \cdots & \boldsymbol{e} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{1}{2}\boldsymbol{h}^{(n,1)}(\boldsymbol{x}_{i_n}) & \frac{1}{2}\boldsymbol{h}^{(n,2)}(\boldsymbol{x}_{i_n}) & \cdots & \boldsymbol{h}^{(n,n)}(\boldsymbol{x}_{i_n}) & \cdots & \frac{1}{2}\boldsymbol{h}^{(n,N)}(\boldsymbol{x}_{i_n}) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{e} & \boldsymbol{e} & \cdots & \frac{1}{2}\boldsymbol{h}^{(n,N)}(\boldsymbol{x}_{i_n}) & \cdots & \boldsymbol{e} \end{pmatrix} \end{matrix} \tag{6}$$

Where $\boldsymbol{x}_{\boldsymbol{i_n}}$ is the i_n-th item in n-th category, $\boldsymbol{h}^{(n,m)}(\boldsymbol{x}_{i_n})$, a row vector, is the function to map the n-th category's items to the latent space for matching with the m-th category's items, $\boldsymbol{e}$ a row vector with all ones. And every vector can be see as a block of this matrix. Use $\boldsymbol{a}_{i_n*}^{(n)}$ to form the factor matrix $\mathbf{A}^{(n)} \in R^{L^{(n)} \times N^2 D}$.

And similarly, we define the $Matrix(\mathbf{U})$, as the reshaped matrix of row vector $\boldsymbol{u}$.

$$Matrix(\boldsymbol{u}) = \begin{pmatrix} \boldsymbol{u}^{(1)} & \boldsymbol{e} & \cdots & \boldsymbol{e} \\ \boldsymbol{e} & \boldsymbol{u}^{(2)} & \cdots & \boldsymbol{e} \\ \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{e} & \boldsymbol{e} & \cdots & \boldsymbol{u}^{(N)} \end{pmatrix} \tag{7}$$

With the definition of equation 6 and equation 7, we can conclude that:

$$\begin{aligned} r_{u,\mathfrak{O}_t} =& \sum_{n=1}^{N} \boldsymbol{u}^{(n)} \cdot \boldsymbol{h}^{(n,n)}(\boldsymbol{x}_{i_n}) + \\ & \sum_{n=1}^{N} \sum_{m=n+1}^{N} \boldsymbol{h}^{(n,m)}(\boldsymbol{x}_{i_n}) \cdot \boldsymbol{h}^{(m,n)}(\boldsymbol{x}_{i_m}) \\ =& \sum_{k=1}^{N^2 D} u_k a_{i_1 k}^{(1)} a_{i_2 k}^{(2)} \cdots a_{i_N k}^{(N)} \\ =& x_{\boldsymbol{i}} \end{aligned}$$

Suppose $k-1 = xND + yD + z$, then $(x+1, y+1)$ represents the block vector's position in $Matrix(\boldsymbol{a}_{i_n*}^{(n)})$ or $Matrix(\boldsymbol{u})$, and $z+1$ represents the element's index in $(x+1, y+1)$-th block. i.e

$$a_{i_n k}^{(n)} = \begin{cases} 1, & x+1 \neq n, y+1 \neq n; \\ (1/2)h_{z+1}^{(n,y+1)}(\boldsymbol{x}_{i_n}), & x+1 = n, y+1 \neq n; \\ (1/2)h_{z+1}^{(n,x+1)}(\boldsymbol{x}_{i_n}), & x+1 \neq n, y+1 = n; \\ h_{z+1}^{(n,n)}(\boldsymbol{x}_{i_n}), & x+1 = y+1 = n. \end{cases} \quad u_k = \begin{cases} 1, & x+1 \neq n \text{ or } y+1 \neq n; \\ u_{z+1}^{(n)}, & x+1 = y+1 = n. \end{cases}$$

Base on the special structure of $\boldsymbol{u}$ and $\mathbf{A}^{(n)}$, the Algorithm 3 can have more concise formation.

7.2 Compute the weight u_k

See the line 2 of Algorithm 3

- $\| \mathbf{U} \|_1$ is redundant.
- Only two of $\{| u_k |, \| \boldsymbol{a}_{*r}^{(1)} \|_1, \| \boldsymbol{a}_{*r}^{(2)} \|_1, \ldots, \| \boldsymbol{a}_{*r}^{(N)} \|_1\}$ need to be computed. For robustness, the weight w_k is replaced by $| u_k | \| \boldsymbol{a}_{*r}^{(1)} \|_1 \| \boldsymbol{a}_{*r}^{(2)} \|_1 \ldots \| \boldsymbol{a}_{*r}^{(N)} \|_1 / (L^{(1)} \times L^{(2)} \ldots \times L^{(N)})$. In each turn, two components is calculated, and divided by the corresponding length.

7.3 Compute the $x_{i_1,i_2,\cdots,i_N}$

- In each round of line 8 of Algorithm 3, only two indexes of $\{i_1, i_2, ..., i_N\}$ need to be sampled with particular distribution ($\boldsymbol{p}_n = \boldsymbol{a}_{*r}^{(n)} / \parallel \boldsymbol{a}_{*r}^{(n)} \parallel_1$).
- Update values Δx

$$\Delta x = sgn(a_{i_1k}^{(1)} a_{i_2k}^{(2)} \cdot a_{i_3k}^{(3)} \cdots a_{i_Nk}^{(N)} \cdot u_k \cdot u_{k'}) a_{i_1k'}^{(1)} a_{i_2k'}^{(2)} a_{i_3k'}^{(3)} \cdots a_{i_Nk'}^{(N)}$$

And only four of elements need to be calculated.

- Suppose $k-1 = xND+yD+z$ and $x+1 = y+1 = n$, then only i_n need to be sampled with specific distribution. After sampled i_n, k',we update $x_{i_1,i_2,i_3,\cdots,i_N}$ for $i_m = \{1, \ldots, L^{(m)}\}, m \neq n$

$$x_{i_1,i_2,i_3,\cdots,i_N} \leftarrow x_{i_1,i_2,i_3,\cdots,i_N} + \frac{1}{\prod_{m \neq n} L^{(m)}} \Delta x$$

- Or only i_n, i_m need to be sampled with specific distribution. After sampled i_n, i_m, k', then we update $x_{i_1,i_2,i_3,\cdots,i_N}$ for $i_h = \{1, \ldots, L^{(h)}\}, h \neq m, n$

$$x_{i_1,i_2,i_3,\cdots,i_N} \leftarrow x_{i_1,i_2,i_3,\cdots,i_N} + \frac{1}{\prod_{h \neq m,n} L^{(h)}} \Delta x$$